Ve203 Discrete Mathematics

Dr. Horst Hohberger

University of Michigan - Shanghai Jiaotong University
Joint Institute

Summer Term 2011

Office Hours, Email, TAs

My office is currently Room 218 in the Law School Building. My email is horst@sjtu.edu.cn and I'll try to answer email queries within 24 hours.

Office hours will be announced on SAKAI and I'll answer questions on the SAKAI course site. Please use the chat room and the forum for asking questions or discussing problems.

This is a fast-paced course and it is very easy for you to fall behind - please do not hesitate to contact me or the TA to discuss any problems you might have.

You may also email your TA with your questions and make use of the TA's office hours and recitation classes. Please respect your TA. It is not polite to phone your TA at 10 pm in the evening and request help for the exercise set due the next day!! Your TA is there to help you, but is not a 24/7 help line. The contact details and office hours of the TA will be announced on the SAKAI course site.

Coursework Policy

I expect you to complete your coursework carefully, punctually and originally. That means:

- Hand in your coursework on time, by the date given on each set of course work. Late work will not be accepted unless you come to me personally and I find your explanation for the lateness acceptable.
- You can be deducted up to 10% of your marks if you fail to write neatly and legibly. Messiness will be penalized!
- Cooperate in finding solutions, but write your own answers! **Do not copy!** It is acceptable to discuss the problems orally, but you may not look at each others' written notes. **Do not show your written solutions to any other student!** This will be considered a violation of the **Honor Code** and punished accordingly.

Use of Wikipedia and Other Sources

When faced with a particularly difficult problem, you may want to refer to other textbooks or online sources such as Wikipedia. Here are a few guidelines:

1. Outside sources may treat a similar sounding subject matter at a much more advanced or a much simpler level than this course. This means that explanations you find are much more complicated or far too simple to help you. For example, when looking up the "induction axiom" you may find many high-school level explanations that are not sufficient for our problems; on the other hand, wikipedia contains a lot of information relating to formal logic that is far beyond what we are discussing here.

Use of Wikipedia and Other Sources

2. If you do use any outside sources to help you solve a homework problem, you are not allowed to just copy the solution; this is an Honor Code violation! The correct way of using outside sources is to understand the contents of your source and then to write *in your own words and without referring back to the source* the solution of the problem. Essentially, the information needs to be filtered through your brain before you may use it.

Grade Policy

Please note that the grading policy for this course has been updated. Please read the following passage carefully.

The final course grade will be expressed as a letter grade or as grade points on the following scale:

Course Grade	F	D	C−	C	C+	B−	B	B+	A−	A	A+
Grade Points	0	1	1.7	2	2.3	2.7	3	3.3	3.7	4	4

The course contains four *grade components*:

- Three examinations,
- Course work.

Each of the grade components is assigned a letter grade / grade points according to the above scale.

Grade Policy

Basic Criterion. A student is said to satisfy the basic criterion if

(i) The sum of the grade points obtained for the three exams is at least 3.0.

(ii) At least one grade point is obtained for the course work.

The course grade is calculated as follows:

1. A student who does not satisfy the basic criterion will receive an F (0 grade points).
2. A student who does satisfy the basic criterion will receive a Course GPA calculated as the weighted mean of the grades of the course components, with the following weights used:
   - First midterm exam: 20%
   - Second midterm exam: 25%
   - Final exam: 30%
   - Course work: 25%

Grade Policy

The Course GPA is then translated into a letter grade as follows:

Course GPA	< 0.83	< 1.5	< 1.83	< 2.17	< 2.5	
Course Grade	F	D	C−	C	C+	
Grade Points	0	1	1.7	2	2.3	
Course GPA	< 2.83	< 3.17	< 3.5	< 3.83	< 4.00	4.00
Course Grade	B−	B	B+	A−	A	A+
Grade Points	2.7	3	3.3	3.7	4	4

Example. Suppose that a student has obtained the following grades:

1^{st} Exam	2^{nd} Exam	3^{rd} Exam	Course Work
1	0	1	3.7

Since the student does not satisfy the basic criterion, the course grade is an F.

Grade Policy

Example. Suppose that a student has obtained the following grades:

1st Exam	2nd Exam	3rd Exam	Course Work
1	2	1	3.7

The student satisfies the basic criterion. The Course GPA is calculated as follows:

$$0.20 \cdot 1 + 0.25 \cdot 2 + 0.30 \cdot 1 + 0.25 \cdot 3.7 = 1.93$$

The course grade is then a C according to the previous table.

Note that due to this process it is "difficult" to get a grade lower than a C or higher than a B+.

It is expected that the median grade for students of this course will be a B.

LaTeX Policy and Textbook

As engineers, you are strongly encouraged to familiarize yourselves with a mathematical typesetting program called LaTeX. This is open-source software, and there are various implementations available. I suggest that you use Baidu or Google to find a suitable implementation for your computer and OS.

While the use of LaTeX is **optional**, there will be a **10% bonus to the awarded marks** for those assignments handed in as typed LaTeX manuscripts.

The main textbook for this course is

- Rosen, K. H., *Discrete Mathematics and its Applications*, 6th Ed., McGraw-Hill International Edition 2007.

Part I

Basic Concepts in Discrete Mathematics

Basic Concepts in Logic

Propositional Logic, Statements

A *statement* (also called a *proposition*) is anything we can regard as being either *true* or *false*. We do not define here what the words "statement", "true" or "false" mean. This is beyond the purview of mathematics and falls into the realm of philosophy. Instead, we apply the principle that "we know it when we see it."

Contrary to the textbook, we will generally not use examples from the "real world" as statements. The reason is that in general objects in the real world are much to loosely define for the application of strict logic to make any sense. For example, the statement "It is raining." may be considered true by some people ("Yes, raindrops are falling out of the sky.") while at the same time false by others ("No, it is merely drizzling.") Furthermore, important information is missing (Where is it raining? When is it raining?). Some people may consider this information to be implicit in the statement (It is raining *here* and *now*.) but others may not, and this causes all sorts of problems. Generally, applying strict logic to colloquial expressions is pointless.

The Natural Numbers

Instead, our examples will be based on numbers. For now, we assume that the set of natural numbers

$$\mathbb{N} := \{0, 1, 2, 3, \ldots\}$$

has been constructed. In particular, we assume that we know what a *set* is! If n is a natural number, we write $n \in \mathbb{N}$. (We will later discuss naive set theory and give a formal construction of the natural numbers.) We also assume that on $\mathbb{N}$ we have defined the operations of addition $+ : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ and multiplication $\cdot : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ and that their various properties (commutativity, associativity, distributivity) hold.

The Natural Numbers

1.1.1. Definition. Let $m, n \in \mathbb{N}$ be natural numbers.

(i) We say that n is *greater than or equal to* m, writing $n \geq m$, if there exists some $k \in \mathbb{N}$ such that $n = m + k$. If we can choose $k \neq 0$, we say n is *greater than* m and write $n > m$.

(ii) We say that m *divides* n, writing $m \mid n$, if there exists some $k \in \mathbb{N}$ such that $n = m \cdot k$.

(iii) If $2 \mid n$, we say that n is even.

(iv) If there exists some $k \in \mathbb{N}$ such that $n = 2k + 1$, we say that n is odd.

(v) Suppose that $n > 1$. If there does not exist any $k \in \mathbb{N}$ with $1 < k < n$ such that $k \mid n$, we say that n is *prime*.

It can be proven that every number is either even or odd and not both. We also assume this for the purposes of our examples.

Statements

1.1.2. Examples.

- "$3 > 2$" is a *true statement*.
- "$x^3 > 10$" is not a statement, because we can not decide whether it is true or not.
- "the cube of any natural number is greater than 10" is a *false statement*.

The last example can be written using a *statement variable* n:

- "For any natural number n, $n^3 > 10$"

The first part of the statement is a *quantifier* ("for any natural number n"), while the second part is called a *statement form* or *predicate* ("$n^3 > 10$").

A statement form becomes a statement (which can then be either true or false) when the variable takes on a specific value; for example, $3^3 > 10$ is a true statement and $1^3 > 10$ is a false statement.

Working with Statements

We will denote statements by capital letters such as $A, B, C, \ldots$ and statement forms by symbols such as $A(x)$ or $B(x, y, z)$ etc.

1.1.3. Examples.

- A: 4 is an even number.
- B: $2 > 3$.
- $A(n)$: $1 + 2 + 3 + \ldots + n = n(n+1)/2$.

We will now introduce logical operations on statements. The simplest possible type of operation is a *unary operation*, i.e., it takes a statement A and returns a statement B.

1.1.4. Definition. Let A be a statement. Then we define the *negation of* A, written as $\neg A$, to be the statement that is true if A is false and false if A is true.

Negation

1.1.5. Example. If A is the statement A: $2 > 3$, then the negation of A is $\neg A$: $2 \not> 3$.

We can describe the action of the unary operation $\neg$ through the following table:

A	$\neg A$
T	F
F	T

If A is true (T), then $\neg A$ is false (F) and vice-versa. Such a table is called a *truth table*.

We will use truth tables to define all our operations on statements.

Conjunction

The next simplest type of operations on statements are *binary operations*. The have two statements as arguments and return a single statement, called a *compound statement*, whose truth or falsehood depends on the truth or falsehood of the original two statements.

1.1.6. Definition. Let A and B be two statements. Then we define the *conjunction* of A and B, written $A \wedge B$, by the following truth table:

A	B	$A \wedge B$
T	T	T
T	F	F
F	T	F
F	F	F

The conjunction $A \wedge B$ is spoken "A and B." It is true only if both A and B are true, false otherwise.

Disjunction

1.1.7. Definition. Let A and B be two statements. Then we define the *disjunction* of A and B, written $A \vee B$, by the following truth table:

A	B	$A \vee B$
T	T	T
T	F	T
F	T	T
F	F	F

The conjunction $A \vee B$ is spoken "A or B." It is true only if either A or B is true, false otherwise.

1.1.8. Example.

- Let A: $2 > 0$ and B: $1 + 1 = 1$. Then $A \wedge B$ is false and $A \vee B$ is true.
- Let A be a statement. Then the compound statement "$A \vee (\neg A)$" is always true, and "$A \wedge (\neg A)$" is always false.

Proofs using Truth Tables

How do we prove that "$A \vee (\neg A)$" is an always true statement? We are claiming that $A \vee (\neg A)$ will be a true statement, regardless of whether the statement A is true or not. To prove this, we go through all possibilities using a truth table:

A	$\neg A$	$A \vee (\neg A)$
T	F	T
F	T	T

Since the column for $A \vee (\neg A)$ only lists T for "true," we see that $A \vee (\neg A)$ is always true. A compound statement that is always true is called a *tautology*.

Correspondingly, the truth table for $A \wedge (\neg A)$ is

A	$\neg A$	$A \wedge (\neg A)$
T	F	F
F	T	F

so $A \wedge (\neg A)$ is always false. A compound statement that is always false is called a *contradiction*.

Implication

1.1.9. Definition. Let A and B be two statements. Then we define the *implication* of B and A, written $A \Rightarrow B$, by the following truth table:

A	B	$A \Rightarrow B$
T	T	T
T	F	F
F	T	T
F	F	T

We read "$A \Rightarrow B$" as "A implies B," "if A, then B" or "A only if B". (The last formulation refers to the fact that A can not be true unless B is true.)

To illustrate why the implication is defined the way it is, it is useful to look at a specific implication of predicates: we expect the predicate

$$A(n)\colon n \text{ is prime} \Rightarrow n \text{ is odd}, \qquad n \in \mathbb{N}, \tag{1.1.1}$$

to be false if and only if we can find a prime number n that is not odd.

Implication

By selecting different values of n we obtain the following types of statements

- $n = 3$. Then n is prime and n is odd, so we have T $\Rightarrow$ T.
- $n = 4$. Then n is not prime and n is not odd, so we have F $\Rightarrow$ F.
- $n = 9$. Then n is not prime, but n is odd. We have F $\Rightarrow$ T.

None of these values of n would cause us to designate (1.1.1) as generating false statements. Therefore, we should assign the truth value "T" to each of these three cases.

However, let us take

- $n = 2$. Then n is prime, but n is not odd. We have T $\Rightarrow$ F.

This is clearly a value of n for which (1.1.1) should be false. Hence, we we should assign the truth value "F" to the implication T $\Rightarrow$ F.

Equivalence

1.1.10. Definition. Let A and B be two statements. Then we define the *equivalence* of A and B, written $A \Leftrightarrow B$, by the following truth table:

A	B	$A \Leftrightarrow B$
T	T	T
T	F	F
F	T	F
F	F	T

We read "$A \Leftrightarrow B$" as "A is equivalent to B" or "A if and only if B". Some textbooks abbreviate "if and only if" by "iff."

If A and B are both true or both false, then they are equivalent. Otherwise, they are not equivalent. In propositional logic, "equivalence" is the closest thing to the "equality" of arithmetic.

Equivalence

On the one hand, logical equivalence is strange; two statements A and B do not need to have anything to do with each other to be equivalent. For example, the statements "$2 > 0$" and "$100 = 99 + 1$" are both true, so they are equivalent.

On the other hand, we use equivalence to manipulate compound statements.

1.1.11. Definition. Two compound statements A and B are called *logically equivalent* if $A \Leftrightarrow B$ is a tautology. We then write $A \equiv B$.

1.1.12. Example. The two *de Morgan rules* are the tautologies

$$\neg(A \vee B) \Leftrightarrow (\neg A) \wedge (\neg B), \qquad \neg(A \wedge B) \Leftrightarrow (\neg A) \vee (\neg B).$$

In other words, they state that $\neg(A \vee B)$ is logically equivalent to $(\neg A) \wedge (\neg B)$ and $\neg(A \wedge B)$ is logically equivalent to $(\neg A) \vee (\neg B)$.

Contraposition

An important tautology is the *contrapositive* of the compound statement $A \Rightarrow B$.

$$(A \Rightarrow B) \Leftrightarrow (\neg B \Rightarrow \neg A).$$

For example, for any natural number n, the statement "$n > 0 \Rightarrow n^3 > 0$" is equivalent to "$n^3 \not> 0 \Rightarrow n \not> 0$." This principle is used in proofs by contradiction.

We prove the contrapositive using a truth table:

A	B	$\neg A$	$\neg B$	$\neg B \Rightarrow \neg A$	$A \Rightarrow B$	$(A \Rightarrow B) \Leftrightarrow (\neg B \Rightarrow \neg A)$
T	T	F	F	T	T	T
T	F	F	T	F	F	T
F	T	T	F	T	T	T
F	F	T	T	T	T	T

Rye Whiskey

The following song is an old Western-style song, called "Rye Whiskey" and performed by Tex Ritter in the 1930's and 1940's.

> *If the ocean was whiskey and I was a duck,*
> *I'd swim to the bottom and never come up.*
>
> *But the ocean ain't whiskey, and I ain't no duck,*
> *So I'll play jack-of-diamonds and trust to my luck.*
>
> *For it's whiskey, rye whiskey, rye whiskey I cry.*
> *If I don't get rye whiskey I surely will die.*

The lyrics make sense (at least as much as song lyrics generally do).

Rye Whiskey

We can use de Morgan's rules and the contrapositive to re-write the song lyrics as follows

> *If I never reach bottom or sometimes come up,*
> *Then the ocean's not whiskey, or I'm not a duck.*
>
> *But my luck can't be trusted, or the cards I'll not buck,*
> *So the ocean is whiskey or I am a duck.*
>
> *For it's whiskey, rye whiskey, rye whiskey I cry.*
> *If my death is uncertain, then I get whiskey (rye).*

These lyrics seem to be logically equivalent to the original song, but are just humorous nonsense. This again illustrates clearly why it is futile to apply mathematical logic to everyday language.

Some Logical Equivalencies

The following logical equivalencies can be established using truth tables or by using previously proven equivalencies. Here T is the compound statement that is always true, T: $A \vee (\neg A)$ and F is the compound statement that is always false, F: $A \wedge (\neg A)$

Equivalence	Name
$A \wedge T \equiv A$	Identity for $\wedge$
$A \vee F \equiv A$	Identity for $\vee$
$A \wedge F \equiv F$	Dominator for $\wedge$
$A \vee T \equiv T$	Dominator for $\vee$
$A \wedge A \equiv A$	Idempotency of $\wedge$
$A \vee A \equiv A$	Idempotency of $\vee$
$\neg(\neg A) \equiv A$	Double negation

Some Logical Equivalencies

Equivalence	Name
$A \wedge B \equiv B \wedge A$	Commutativity of $\wedge$
$A \vee B \equiv B \vee A$	Commutativity of $\vee$
$(A \wedge B) \wedge C \equiv A \wedge (B \wedge C)$	Associativity of $\wedge$
$(A \vee B) \vee C \equiv A \vee (B \vee C)$	Associativity of $\vee$
$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$	Distributivity
$A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$	Distributivity
$A \vee (A \wedge B) \equiv A$	Absorption
$A \wedge (A \vee B) \equiv A$	Absorption

These laws include all that are necessary for a *boolean algebra generated by* $\wedge$ *and* $\vee$ (identity element, commutativity, associativity, distributivity). Hence the name *boolean logic* for this calculus of logical statements.

Some Logical Equivalencies

We omitted de Morgan's laws from the previous table. We now list some equivalences involving conditional statements.

Equivalence
$A \Rightarrow B \equiv \neg A \vee B \equiv \neg B \Rightarrow \neg A$
$(A \Rightarrow B) \wedge (A \Rightarrow C) \equiv A \Rightarrow (B \wedge C)$ $(A \Rightarrow B) \vee (A \Rightarrow C) \equiv A \Rightarrow (B \vee C)$
$(A \Rightarrow C) \wedge (B \Rightarrow C) \equiv (A \vee B) \Rightarrow C$ $(A \Rightarrow C) \vee (B \Rightarrow C) \equiv (A \wedge B) \Rightarrow C$
$(A \Leftrightarrow B) \equiv ((\neg A) \Leftrightarrow (\neg B))$ $(A \Leftrightarrow B) \equiv (A \Rightarrow B) \wedge (B \Rightarrow A)$ $(A \Leftrightarrow B) \equiv (A \wedge B) \vee ((\neg A) \wedge (\neg B))$ $\neg(A \Leftrightarrow B) \equiv A \Leftrightarrow (\neg B)$

Logical Quantifiers

In the previous examples we have used predicates $A(x)$ with the words "for all x." This is an instance of a *logical quantifier* that indicates for which x a predicate $A(x)$ is to be evaluated to a statement.

In order to use quantifiers properly, we clearly need a universe of objects x which we can insert into $A(x)$ (a *domain* for $A(x)$). This leads us immediately to the definition of a *set*. We will discuss set theory in detail later. For the moment it is sufficient for us to view a set as a "collection of objects" and assume that the following sets are known:

- the set of natural numbers $\mathbb{N}$ (which includes the number 0),
- the set of integers $\mathbb{Z}$,
- the set of real numbers $\mathbb{R}$,
- the empty set $\emptyset$ (also written $\varnothing$ or $\{\}$) that does not contain any objects.

If M is a set containing x, we write $x \in M$ and call x an *element* of M.

Logical Quantifiers

There are two types of quantifiers:

- the *universal quantifier*, denoted by the symbol $\forall$, read as "for all" and
- the *existential quantifier*, denoted by $\exists$, read as "there exists."

1.1.13. Definition. Let M be a set and $A(x)$ be a predicate. Then we define the quantifier $\forall$ by

$$\underset{x\in M}{\forall} A(x) \quad \Leftrightarrow \quad A(x) \text{ is true for all } x \in M$$

We define the quantifier $\exists$ by

$$\underset{x\in M}{\exists} A(x) \quad \Leftrightarrow \quad A(x) \text{ is true for at least one } x \in M$$

We may also write $\forall x \in M$: $A(x)$ instead of $\underset{x\in M}{\forall} A(x)$ and similarly for $\exists$.

Logical Quantifiers

We may also state the domain before making the statements, as in the following example.

1.1.14. Examples. Let x be a real number. Then

- $\forall x$: $x > 0 \Rightarrow x^3 > 0$ is a true statement;
- $\forall x$: $x > 0 \Leftrightarrow x^2 > 0$ is a false statement;
- $\exists x$: $x > 0 \Leftrightarrow x^2 > 0$ is a true statement.

Sometimes mathematicians put a quantifier at the end of a statement form; this is known as a *hanging quantifier*. Such a hanging quantifier will be interpreted as being located just before the statement form:

$$\exists y\text{: } y + x^2 > 0 \qquad\qquad \forall x$$

is equivalent to $\exists y \forall x$: $y + x^2 > 0$.

Contraposition and Negation of Quantifiers

We do not actually need the quantifier $\exists$ since

$$\begin{aligned} \underset{x\in M}{\exists} A(x) \quad &\Leftrightarrow \quad A(x) \text{ is true for at least one } x \in M \\ &\Leftrightarrow \quad A(x) \text{ is not false for all } x \in M \\ &\Leftrightarrow \quad \neg \underset{x\in M}{\forall} (\neg A(x)) \end{aligned} \tag{1.1.2}$$

The equivalence (1.1.2) is called *contraposition of quanifiers*. It implies that the negation of $\exists x \in M$: $A(x)$ is equivalent to $\forall x \in M$: $\neg A(x)$. For example,

$$\neg \left(\exists x \in \mathbb{R} \colon x^2 < 0\right) \qquad \Leftrightarrow \qquad \forall x \in \mathbb{R} \colon x^2 \not< 0.$$

Conversely,

$$\neg \left(\forall x \in M \colon A(x)\right) \qquad \Leftrightarrow \qquad \exists x \in M \colon \neg A(x).$$

Vacuous Truth

If the domain of the universal quantifier $\forall$ is the empty set $M = \varnothing$, then the statement $\forall x \in M$: $A(x)$ is defined to be true regardless of the predicate $A(x)$. It is then said that $A(x)$ is *vacuously true*.

1.1.15. Example. Let M be the set of real numbers x such that $x = x + 1$. Then the statement

$$\forall_{x \in M} \; x > x$$

is true.

This convention reflects the philosophy that a universal statement is true unless there is a counterexample to prove it false. While this may seem a strange point of view, it proves useful in practice.

This is similar to saying that "All pink elephants can fly." is a true statement, because it is impossible to find a pink elephant that can't fly.

Nesting Quantifiers

We can also treat predicates with more than one variable as shown in the following example.

1.1.16. Examples. In the following examples, x, y are taken from the real numbers.

- $\forall x \forall y$: $x^2 + y^2 - 2xy \geq 0$ is equivalent to $\forall y \forall x$: $x^2 + y^2 - 2xy \geq 0$. Therefore, one often writes $\forall x, y$: $x^2 + y^2 - 2xy \geq 0$.
- $\exists x \exists y$: $x + y > 0$ is equivalent to $\exists y \exists x$: $x + y > 0$, often abbreviated to $\exists x, y$: $x + y > 0$.
- $\forall x \exists y$: $x + y > 0$ is a true statement.
- $\exists x \forall y$: $x + y > 0$ is a false statement.

As is clear from these examples, the order of the quantifiers is important if they are different.

Examples from Calculus

Let I be an interval in $\mathbb{R}$. Then a function $f\colon I \to \mathbb{R}$ is said to be *continuous* on I if and only if

$$\forall_{\varepsilon>0} \forall_{x\in I} \exists_{\delta>0} \forall_{y\in I} \quad |x-y| < \delta \Rightarrow |f(x)-f(y)| < \varepsilon.$$

The function f is *uniformly continuous* on I if and only if

$$\forall_{\varepsilon>0} \exists_{\delta>0} \forall_{x\in I} \forall_{y\in I} \quad |x-y| < \delta \Rightarrow |f(x)-f(y)| < \varepsilon.$$

It is easy to see that a function that is uniformly continuous on I must also be continuous on I.

If I is a closed interval, $I = [a, b]$, it can also be shown that a continuous function is also uniformly continuous. However, that requires techniques from calculus and is not obvious just by looking at the logical structure of the definitions.

Examples from Calculus

Negating complicated expressions can be done step-by-step. For example, the statement that f is not continuous on I is equivalent to

$$\neg\Big(\underset{\varepsilon>0}{\forall}\ \underset{x\in I}{\forall}\ \underset{\delta>0}{\exists}\ \underset{y\in I}{\forall}\quad |x-y|<\delta \Rightarrow |f(x)-f(y)|<\varepsilon\Big)$$

$$\Leftrightarrow \Big(\underset{\varepsilon>0}{\exists}\ \neg\ \underset{x\in I}{\forall}\ \underset{\delta>0}{\exists}\ \underset{y\in I}{\forall}\quad |x-y|<\delta \Rightarrow |f(x)-f(y)|<\varepsilon\Big)$$

$$\Leftrightarrow \Big(\underset{\varepsilon>0}{\exists}\ \underset{x\in I}{\exists}\ \neg\ \underset{\delta>0}{\exists}\ \underset{y\in I}{\forall}\quad |x-y|<\delta \Rightarrow |f(x)-f(y)|<\varepsilon\Big)$$

$$\Leftrightarrow \Big(\underset{\varepsilon>0}{\exists}\ \underset{x\in I}{\exists}\ \underset{\delta>0}{\forall}\ \neg\ \underset{y\in I}{\forall}\quad |x-y|<\delta \Rightarrow |f(x)-f(y)|<\varepsilon\Big)$$

$$\Leftrightarrow \Big(\underset{\varepsilon>0}{\exists}\ \underset{x\in I}{\exists}\ \underset{\delta>0}{\forall}\ \underset{y\in I}{\exists}\quad (|x-y|<\delta)\wedge\neg(|f(x)-f(y)|<\varepsilon)\Big)$$

$$\Leftrightarrow \Big(\underset{\varepsilon>0}{\exists}\ \underset{x\in I}{\exists}\ \underset{\delta>0}{\forall}\ \underset{y\in I}{\exists}\quad (|x-y|<\delta)\wedge(|f(x)-f(y)|\geq\varepsilon)\Big)$$

Examples from Calculus

1.1.17. Example. The *Heaviside function* H: $\mathbb{R} \to \mathbb{R}$,

$$H(x) := \begin{cases} 1 & \text{if } x \geq 0, \\ 0 & \text{if } x < 0, \end{cases}$$

is not continuous on $I = \mathbb{R}$. To see this, we need to show that there exists an $\varepsilon > 0$ (take $\varepsilon = 1/2$) and an $x \in \mathbb{R}$ (take $x = 0$) such that for any $\delta > 0$ there exists a $y \in \mathbb{R}$ such that

$$|x - y| = |y| < \delta \qquad \text{and} \qquad |H(x) - H(y)| = |1 - H(y)| \geq \varepsilon = \frac{1}{2}.$$

Given any $\delta > 0$ we can choose $y = -\delta/2$. Then $|y| = \delta/2 < \delta$ and $|1 - H(y)| = 1 > 1/2$. This proves that H is not continuous on $\mathbb{R}$.

Arguments in Mathematics

The previous example contains a *mathematical argument* to show that the Heaviside function is not continuous on its domain. The argument boils down to the following:

(i) We know that

$$A\colon \underset{\varepsilon>0}{\exists}\ \underset{x\in\mathbb{R}}{\exists}\ \underset{\delta>0}{\forall}\ \underset{y\in\mathbb{R}}{\exists}\quad (|x-y|<\delta)\wedge(|H(x)-H(y)|\geq\varepsilon)$$

implies

B: H is not continuous on its domain.

(ii) We show that A is true.

(iii) Therefore, B is true.

Logically, we can express this argument as

$$\big(A\wedge(A\Rightarrow B)\big)\Rightarrow B.$$

Arguments and Argument Forms

1.1.18. Definition.

(i) An *argument* is a finite sequence of statements. All statements except for the final statement are called *premises* while the final statement is called the *conclusion*. We say that an argument is *valid* if the truth of all premises implies the truth of the conclusion.

(ii) An *argument form* is a finite sequence of predicates (statement forms). An argument form is *valid* if it yields a valid argument whenever statements are substituted for the predicates.

From the definition of an argument it is clear that an argument consisting of a sequence of premises $P_1, \ldots, P_n$ and a conclusion C is valid of and only if

$$(P_1 \wedge P_2 \wedge \cdots \wedge P_n) \Rightarrow C \tag{1.1.3}$$

is a tautology, i.e., a true statement for any values of the premises an the conclusion.

Arguments and Argument Forms

An argument is a finite list of premises $P_1, \ldots, P_n$ followed by a conclusion C. We usually write this list as

$$\begin{array}{l} P_1 \\ P_2 \\ \vdots \\ P_n \\ \hline \therefore \quad C \end{array}$$

where the symbol $\therefore$ is pronounced "therefore". You may only use this symbol when constructing a logical argument in the notation above. Do not use it as a general-purpose abbreviation of "therefore".

Certain basic valid arguments in mathematics are given latin names and called *rules of inference*.

Modus Ponendo Ponens

1.1.19. Example. The rule of inference

$$\begin{array}{l} A \\ A \Rightarrow B \\ \hline \therefore \quad B \end{array}$$

is called *modus ponendo ponens* (latin for "mode that affirms by affriming"); it is often abbreviated simply "modus ponens". The associated tautology is

$$\big(A \wedge (A \Rightarrow B)\big) \Rightarrow B \tag{1.1.4}$$

We verify that (1.1.4) actually is a tautology using the truth table:

A	B	$A \Rightarrow B$	$A \wedge (A \Rightarrow B)$	$\big(A \wedge (A \Rightarrow B)\big) \Rightarrow B$
T	T	T	T	T
T	F	F	F	T
F	T	T	T	T
F	F	T	T	T

Hypothetical Syllogisms

A *syllogism* is an argument that has exactly two premises. We first give three *hypothetical syllogisms*, i.e., syllogisms involving the implication "$\Rightarrow$".

Rule of Inference	Name
$\begin{array}{l} A \\ A \Rightarrow B \\ \hline \therefore \quad B \end{array}$	Modus (Ponendo) Ponens *Mode that affirms (by affirming)*
$\begin{array}{l} \neg B \\ A \Rightarrow B \\ \hline \therefore \quad \neg A \end{array}$	Modus (Tollendo) Tollens *Mode that denies (by denying)*
$\begin{array}{l} A \Rightarrow B \\ B \Rightarrow C \\ \hline \therefore \quad A \Rightarrow C \end{array}$	Transitive Hypothetical Syllogism

Hypothetical Syllogisms

1.1.20. Examples.

(i) Modus ponendo ponens:

3 is both prime and greater than 2
If 3 is both prime and greater than 2, then 3 is odd

$\therefore$ 3 is odd.

(ii) Modus tollendo tollens:

4 is not odd
If 4 is both prime and greater than 2, then 4 is odd

$\therefore$ 4 is not both prime and greater than 2.

(iii) Transitive hypothetical syllogism:

If 5 is greater than 4, then 5 is greater than 3
If 5 is greater than 3, then 5 is greater than 2

$\therefore$ If 5 is greater than 4, then 5 is greater than 2.

Disjunctive and Conjunctive Syllogisms

There are two important syllogisms involving the disjunction "$\vee$" and the conjunction "$\wedge$":

Rule of Inference	Name
$\begin{array}{l} A \vee B \\ \neg A \\ \hline \therefore \quad B \end{array}$	Modus Tollendo Ponens *Mode that affirms by denying*
$\begin{array}{l} \neg(A \wedge B) \\ A \\ \hline \therefore \quad \neg B \end{array}$	Modus Ponendo Tollens *Mode that denies by affirming*
$\begin{array}{l} A \vee B \\ \neg A \vee C \\ \hline \therefore \quad B \vee C \end{array}$	Resolution

Disjunctive and Conjunctive Syllogisms

1.1.21. Examples.

(i) Modus tollendo ponens:

$$\begin{array}{ll} & 4 \text{ is odd or even} \\ & 4 \text{ is not odd} \\ \hline \therefore & 4 \text{ is even.} \end{array}$$

(ii) Modus ponendo tollens:

$$\begin{array}{ll} & 4 \text{ is not both even and odd} \\ & 4 \text{ is even} \\ \hline \therefore & 4 \text{ is not odd.} \end{array}$$

(iii) Resolution:

$$\begin{array}{ll} & 4 \text{ is even or } 4 \text{ is greater than } 2 \\ & 4 \text{ is odd or } 4 \text{ is prime} \\ \hline \therefore & 4 \text{ is greater than } 2 \text{ or } 4 \text{ is prime.} \end{array}$$

Some Simple Arguments

Finally, we give some seemingly obvious, but nevertheless useful, arguments:

Rule of Inference	Name
$\begin{array}{l} A \\ B \\ \hline \therefore\ A \wedge B \end{array}$	Conjunction
$\begin{array}{l} A \wedge B \\ \hline \therefore\ A \end{array}$	Simplification
$\begin{array}{l} A \\ \hline \therefore\ A \vee B \end{array}$	Addition

Examples for these are left to the reader!

Validity and Soundness

The previous rules of inference are all *valid arguments*. In the examples we gave, the arguments always led to a correct conclusion. This was, however, only because all the premises were true statements. It is possible for a valid argument to lead to a wrong conclusion if one or more of its premises are false.

If, in addition to being valid, an argument has only true premises, we say that the argument is *sound*. In that case, its conclusion is true.

1.1.22. Example. The following argument is valid (it is based on the rule of resolution), but not sound:

$$\begin{array}{ll} & 4 \text{ is even or } 4 \text{ is prime} \\ & 4 \text{ is odd or } 4 \text{ is prime} \\ \hline \therefore & 4 \text{ is prime.} \end{array}$$

(The second premise is false, so the conclusion doesn't have to be true.)

Non Sequitur

The term *non sequitur* (latin for "it does not follow") is often used to describe logical fallacies, i.e., inferences that invalid because they are not based on tautologies. Some common fallacies are listed below:

Rule of Inference	Name
$\begin{array}{l} B \\ A \Rightarrow B \\ \hline \therefore \quad A \end{array}$	Affirming the Consequent
$\begin{array}{l} \neg A \\ A \Rightarrow B \\ \hline \therefore \quad \neg B \end{array}$	Denying the Antecedent
$\begin{array}{l} A \vee B \\ A \\ \hline \therefore \quad \neg B \end{array}$	Affirming a Disjunct

Non Sequitur

1.1.23. Examples.

(i) Affirming the consequent:

$$\begin{array}{ll} & \text{If 9 is prime, then it is odd} \\ & \text{9 is odd} \\ \hline \therefore & \text{9 is prime.} \end{array}$$

(ii) Denying the antecedent

$$\begin{array}{ll} & \text{If 9 is prime, then it is odd} \\ & \text{9 is not prime} \\ \hline \therefore & \text{9 is not odd.} \end{array}$$

(iii) Affirming a disjunct:

$$\begin{array}{ll} & \text{2 is even or 2 is prime} \\ & \text{2 is even} \\ \hline \therefore & \text{2 is not prime.} \end{array}$$

Rules of Inference for Quantified Statements

Without proof or justification, we give the following rules of inference for quantified statements. They are often assumed as axioms in abstract logic systems.

Rule of Inference	Name
$\dfrac{\underset{x\in M}{\forall} P(x)}{\therefore \quad P(x_0) \text{ for any } x_0 \in M}$	Universal Instantiation
$\dfrac{P(x) \text{ for any arbitrarily chosen } x \in M}{\therefore \quad \underset{x\in M}{\forall} P(x)}$	Universal Generalization
$\dfrac{\underset{x\in M}{\exists} P(x)}{\therefore \quad P(x_0) \text{ for a certain (unknown) } x_0 \in M}$	Existential Instantiation
$\dfrac{P(x_0) \text{ for some (known) } x_0 \in M}{\therefore \quad \underset{x\in M}{\exists} P(x)}$	Existential Generalization

Constructing Arguments

Often, complex arguments can be broken down into syllogisms. As an example, we give a logical proof of the following theorem:

1.1.24. Theorem. Let $n \in \mathbb{N}$ be a natural number and suppose that n^2 is even. Then n is even.

Proof.

We use the following premises:

$$P_1: \underset{n\in\mathbb{N}}{\forall} \neg(n \text{ even} \wedge n \text{ odd}),$$

$$P_2: n \text{ odd} \Rightarrow n^2 \text{ odd},$$

$$P_3: n^2 \text{ even} \wedge (n \text{ even} \vee n \text{ odd})$$

and we wish to arrive at the conclusion

$$C: n \text{ even}.$$

Constructing Arguments

Proof (continued).

Premise P_2 can be easily checked: if n is odd, there exists some k such that $n = 2k + 1$, so

$$n^2 = (2k+1)^2 = 2(2k^2 + 2k) + 1 = 2k' + 1$$

where $k' = 2k^2 + 2k$. Hence n^2 is also odd. We have

$$\frac{P_3 \colon n^2 \text{ even} \wedge (n \text{ even} \vee n \text{ odd})}{\therefore \quad P_4 \colon n^2 \text{ even}.}$$

by the Rule of Simplification. By Universal Instantiation, we obtain

$$\frac{P_1 \colon \underset{n \in \mathbb{N}}{\forall} \neg(n \text{ even} \wedge n \text{ odd})}{\therefore \quad P_5 \colon \neg(n^2 \text{ even} \wedge n^2 \text{ odd}).}$$

Constructing Arguments

Proof (continued).

Furthermore, by Modus Ponendo Tollens,

$$\begin{array}{ll} & P_4 : n^2 \text{ even} \\ & P_5 : \neg(n^2 \text{ even} \wedge n^2 \text{ odd}) \\ \hline \therefore \quad & P_6 : \neg(n^2 \text{ odd}). \end{array}$$

Using Modus Tollendo Tollens,

$$\begin{array}{ll} & P_6 : \neg(n^2 \text{ odd}) \\ & P_2 : n \text{ odd} \Rightarrow n^2 \text{ odd} \\ \hline \therefore \quad & P_7 : \neg(n \text{ odd}). \end{array}$$

Simplification yields

$$\begin{array}{ll} & P_3 : n^2 \text{ even} \wedge (n \text{ even} \vee n \text{ odd}) \\ \hline \therefore \quad & P_8 : n \text{ even} \vee n \text{ odd}. \end{array}$$

Constructing Arguments

Proof (continued).

Finally, Modus Tollendo Ponens gives

$$\begin{array}{ll} & P_7 : \neg(n \text{ odd}) \\ & P_8 : n \text{ even} \vee n \text{ odd} \\ \hline \therefore & C : n \text{ even}. \end{array}$$

This completes the proof. □

1.1.25. Remark. Of course, this proof could have been shortened and simplified if we had replaced "odd" with "not even" throughout, and we might have formulated premise P_3 slightly differently (as two separate premises) to avoid using the rule of simplification. However, our goal was to illustrate the usage of a wide variety of rules of inference and that writing down a logically valid proof is in most cases extremely tedious; in most mathematics, many of the mentioned rules of inference are used implicitly without being stated.

Naive Set Theory: Sets via Predicates

We want to be able to talk about "collections of objects"; however, we will be unable to strictly define what an "object" or a "collection" is (except that we also want any collection to qualify as an "object"). The problem with "naive" set theory is that any attempt to make a formal definition will lead to a contradiction - we will see an example of this later. However, for our practical purposes we can live with this, as we won't generally encounter these contradictions.

We indicate that an object (called an *element*) x is part of a collection (called a *set*) X by writing $x \in X$. We characterize the elements of a set X by some predicate P:

$$x \in X \qquad \Leftrightarrow \qquad P(x).$$

We write such a set X in the form $X = \{x\colon P(x)\}$.

Notation for Sets

We define the empty set $\varnothing := \{x\colon x \neq x\}$. The empty set has no elements, because the predicate $x \neq x$ is never true.

We may also use the notation $X = \{x_1, x_2, \ldots, x_n\}$ to denote a set. In this case, X is understood to be the set

$$X = \{x\colon (x = x_1) \vee (x = x_2) \vee \cdots \vee (x = x_n)\}.$$

We will frequently use the convention

$$\{x \in A\colon P(x)\} = \{x\colon x \in A \wedge P(x)\}$$

1.2.1. Example. The set of even positive integers is

$$\{n \in \mathbb{N}\colon \exists_{k \in \mathbb{Z}}\, n = 2k\}$$

Subsets and Equality of Sets

If every object $x \in X$ is also an element of a set Y, we say that X is a subset of Y, writing $X \subset Y$; in other words,

$$X \subset Y \qquad \Leftrightarrow \qquad \forall x \in X\colon x \in Y.$$

We say that $X = Y$ if and only if $X \subset Y$ and $Y \subset X$.

We say that X is a *proper subset* of Y if $X \subset Y$ but $X \neq Y$. In that case we write $X \subsetneq Y$.

Some authors write $\subseteq$ for $\subset$ and $\subset$ for $\subsetneq$. Pay attention to the convention used when referring to literature.

Examples of Sets and Subsets

1.2.2. Examples.

1. For any set X, $\varnothing \subset X$. Since $\varnothing$ does not contain any elements, the domain of the statement $\forall x \in X$: $x \in Y$ is empty. Therefore, it is vacuously true and hence $\varnothing \subset X$.
2. Consider the set $A = \{a, b, c\}$ where a, b, c are arbitrary objects, for example, numbers. The set

$$B = \{a, b, a, b, c, c\}$$

is equal to A, because it satisfies $A \subset B$ and $B \subset A$ as follows:

$$x \in A \quad \Leftrightarrow \quad (x = a) \vee (x = b) \vee (x = c) \quad \Leftrightarrow \quad x \in B.$$

Therefore, neither order nor repetition of the elements affects the contents of a set.

If $C = \{a, b\}$, then $C \subset A$ and in fact $C \subsetneq A$. Setting $D = \{b, c\}$ we have $D \subsetneq A$ but $C \not\subset D$ and $D \not\subset C$.

Power Set and Cardinality

If a set X has a finite number of elements, we define the *cardinality* of X to be this number, denoted by $\#X$, $|X|$ or $\operatorname{card} X$.

We define the *power set*

$$\mathcal{P}(M) := \{A \colon A \subset M\}.$$

Here the elements of the set $\mathcal{P}(M)$ are themselves sets; $\mathcal{P}(M)$ is the "set of all subsets of M." Therefore, the statements

$$A \subset M \qquad \text{and} \qquad A \in \mathcal{P}(M)$$

are equivalent.

1.2.3. Example. The power set of $\{a, b, c\}$ is

$$\mathcal{P}(\{a, b, c\}) = \big\{\varnothing, \{a\}, \{b\}, \{c\}, \{a, b\}, \{b, c\}, \{a, c\}, \{a, b, c\}\big\}.$$

The cardinality of $\{a, b, c\}$ is 3, the cardinality of the power set is $|\mathcal{P}(\{a, b, c\})| = 8$.

Operations on Sets

If $A = \{x\colon P_1(x)\}$, $B = \{x\colon P_2(x)\}$ we define the *union*, *intersection* and *difference* of A and B by

$$A \cup B := \{x\colon P_1(x) \vee P_2(x)\}, \qquad A \cap B := \{x\colon P_1(x) \wedge P_2(x)\},$$
$$A \setminus B := \{x\colon P_1(x) \wedge (\neg P_2(x))\}.$$

Let $A \subset M$. We then define the *complement* of A by

$$A^{\mathsf{c}} := M \setminus A.$$

If $A \cap B = \emptyset$, we say that the sets A and B are *disjoint*.

Occasionally, the notation $A - B$ is used for $A \setminus B$ and A^{c} is sometimes denoted by $\overline{A}$.

1.2.4. Example. Let $A = \{a, b, c\}$ and $B = \{c, d\}$. Then

$$A \cup B = \{a, b, c, d\}, \qquad A \cap B = \{c\}, \qquad A \setminus B = \{a, b\}.$$

Operations on Sets

The laws for logical equivalencies immediately lead to several rules for set operations. For example, the distributive laws for $\wedge$ and $\vee$ imply

- $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
- $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$

Other such rules are, for example,

- $(A \cup B) \setminus C = (A \setminus C) \cup (B \setminus C)$
- $(A \cap B) \setminus C = (A \setminus C) \cap (B \setminus C)$
- $A \setminus (B \cup C) = (A \setminus B) \cap (A \setminus C)$
- $A \setminus (B \cap C) = (A \setminus B) \cup (A \setminus C)$

Some of these will be proved in the recitation class and the exercises.

Operations on Sets

Occasionally we will need the following notation for the union and intersection of a finite number $n \in \mathbb{N}$ of sets:

$$\bigcup_{k=0}^{n} A_k := A_0 \cup A_1 \cup A_2 \cup \cdots \cup A_n,$$

$$\bigcap_{k=0}^{n} A_k := A_0 \cap A_1 \cap A_2 \cap \cdots \cap A_n.$$

This notation even extends to $n = \infty$, but needs to be properly defined:

$$x \in \bigcup_{k=0}^{\infty} A_k \quad :\Leftrightarrow \quad \exists_{k \in \mathbb{N}} \; x \in A_k,$$

$$x \in \bigcap_{k=0}^{\infty} A_k \quad :\Leftrightarrow \quad \forall_{k \in \mathbb{N}} \; x \in A_k.$$

Operations on Sets

In particular,

$$\bigcap_{k=0}^{\infty} A_k \subset \bigcup_{k=0}^{\infty} A_k.$$

1.2.5. Example. Let $A_k = \{0, 1, 2, \ldots, k\}$ for $k \in \mathbb{N}$. Then

$$\bigcup_{k=0}^{\infty} A_k = \mathbb{N}, \qquad \bigcap_{k=0}^{\infty} A_k = \{0\}.$$

To see the first statement, note that $\mathbb{N} \subset \bigcup_{k=0}^{\infty} A_k$ since $x \in \mathbb{N}$ implies $x \in A_x$ implies $x \in \bigcup_{k=0}^{\infty} A_k$. Furthermore, $\bigcup_{k=0}^{\infty} A_k \subset \mathbb{N}$ since $x \in \bigcup_{k=0}^{\infty} A_k$ implies $x \in A_k$ for some $k \in \mathbb{N}$ implies $x \in \mathbb{N}$.

For the second statement, note that $\bigcap_{k=0}^{\infty} A_k \subset \mathbb{N}$. Now $0 \in A_k$ for all $k \in \mathbb{N}$. Thus $\{0\} \subset \bigcap_{k=0}^{\infty} A_k$. On the other hand, for any $x \in \mathbb{N} \setminus \{0\}$ we have $x \notin A_{x-1}$ whence $x \notin \bigcap_{k=0}^{\infty} A_k$.

Ordered Pairs

A set does not contain any information about the order of its elements, e.g.,

$$\{a, b\} = \{b, a\}.$$

Thus, there is no such a thing as the "first element of a set". However, sometimes it is convenient or necessary to have such an ordering. This is achieved by defining an *ordered pair*, denoted by

$$(a, b)$$

and having the property that

$$(a, b) = (c, d) \quad \Leftrightarrow \quad (a = c) \wedge (b = d).$$

There are two ways of defining an ordered pair as a set:

1. $(a, b) := \{\{a\}, \{a, b\}\}$ or
2. $(a, b) := \{\{1, a\}, \{2, b\}\}$.

The first definition does not need the natural numbers and uses only set theory, but both are of course equivalent.

Cartesian Product of Sets

If A, B are sets and $a \in A,\ b \in B$, then we denote the set of all ordered pairs by

$$A \times B := \{(a, b) : a \in A,\ b \in B\}.$$

$A \times B$ is called the *cartesian product* of A and B.

In this manner we can define an *ordered triple* (a, b, c) or, more generally, an ordered *n-tuple* $(a_1, \ldots, a_n)$ and the n-fold cartesian product $A_1 \times \cdots \times A_n$ of sets $A_k,\ k = 1, \ldots, n$.

If we take the cartesian product of a set with itself, we may abbreviate it using exponents, e.g.,

$$\mathbb{N}^2 := \mathbb{N} \times \mathbb{N}.$$

The Russel Antinomy

We have previously mentioned that naive set theory leads to inherent contradictions; this is exemplified by the *Russel antinomy:*

1.2.6. Lemma. The predicate $P(x)\colon x \notin x$ does not define a set $A = \{x\colon P(x)\}$.

Proof.

1. Assume $A \in A$. Then $P(A)$ is true, therefore $A \notin A$. ↯
2. Assume $A \notin A$. Then $P(A)$ is false, therefore $A \in A$. ↯

Since we cannot decide whether $A \in A$ or $A \notin A$, A can not be a set. □

The Russel Antinomy

There are several examples in classical literature and philosophy of the Russel antimony:

1. A person says: "This sentence is a lie." Is he lying or telling the truth?
2. A library contains two separate catalogues of its books: The first catalogue lists the names of all books that reference themselves. It may or may not list itself without contradiction. The second catalogue lists the names of all books that do not reference themselves. Should it also list itself or not?

Russel Antinomy

We will simply ignore the existence of such contradictions and build on naive set theory. There are further paradoxes (antinomies) in naive set theory, such as *Cantor's paradox* and the *Burali-Forti paradox*. All of these are resolved if naive set theory is replaced by a *modern axiomatic set theory* such as *Zermelo-Fraenkel set theory.*

Further Information:

Set Theory, Stanford Encyclopedia of Philosophy,
HTTP://PLATO.STANFORD.EDU/ENTRIES/SET-THEORY/

T. Jech, *Set Theory: The Third Millennium Edition, Revised and Expanded*, Reprinted by World Publishing Corporation Beijing, ¥78.00元

The Peano Axioms

The "counting numbers" $0, 1, 2, 3, \ldots$ are the basis of discrete mathematics. We refer to their totality as the set of *natural numbers* and denote it by $\mathbb{N}$. We have up to now used them to supply examples for our introduction to logic and to enumerate sets. It is time we briefly discuss how they can be formally defined.

We will represent the natural numbers as set of objects (denoted by $\mathbb{N}$) together with a relation called "succession": If n is a natural numbers, the "successor" of n, $\mathrm{succ}(n)$, is defined and also in $\mathbb{N}$. In elementary terms, 1 is the successor to 0, 2 is the successor to 1, etc.

There is no unique set of natural numbers in the sense that we can exhibit "the" set $\mathbb{N}$. Rather, any pair $(\mathbb{N}, \mathrm{succ})$, can qualify as a *realization of the natural numbers* if it satisfies certain axioms.

The Peano Axioms

1.3.1. Definition. Let $\mathbb{N}$ be any set and suppose that the successor of any element of $\mathbb{N}$ has been defined. The *Peano axioms* are

1. $\mathbb{N}$ contains at least one object, called zero.
2. If n is in $\mathbb{N}$, the successor of n is in $\mathbb{N}$.
3. Zero is not the successor of a number.
4. Two numbers of which the successors are equal are themselves equal.
5. (*Induction axiom*) If a set $S \subset \mathbb{N}$ contains zero and also the successor of every number in S, then $S = \mathbb{N}$.

Any set with a successor relation satisfying these axioms is called a realization of the natural numbers.

The von Neumann Realization

The Hungarian mathematician John von Neumann gave a very elegant realization of the natural numbers that is based on set theory:

Assume the empty set $\emptyset$ exists. Our idea is to define

$$\begin{aligned} 0 &:= \emptyset, \\ 1 &:= \{0\} = \{\emptyset\}, \\ 2 &:= \{0, 1\} = \{\emptyset, \{\emptyset\}\}, \\ 3 &:= \{0, 1, 2\} = \{\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}\} \end{aligned}$$

and so on. It is clear that we want 1 to be the successor of 0, 2 the successor of 1 and so on. We therefore define

$$\operatorname{succ}(n) := n \cup \{n\}, \tag{1.3.1a}$$

$$\mathbb{N} := \{\emptyset\} \cup \Big\{n\colon \exists_{m \in \mathbb{N}}\, n = \operatorname{succ}(m)\Big\}. \tag{1.3.1b}$$

The von Neumann Realization

One interesting feature of this realization is that

$$n \in \text{succ}(n) \qquad \text{and} \qquad n \subsetneqq \text{succ}(n).$$

Furthermore, we have a natural definition of the ordering relation "$<$" that does not require any concept of addition:

$$m < n \qquad :\Leftrightarrow \qquad m \subsetneqq n.$$

Of course, we still need to check that the Peano axioms are satisfied:

1. $0 = \emptyset \in \mathbb{N}$ by definition.
2. For any $n \in \mathbb{N}$, $\text{succ}(n) \in \mathbb{N}$ by definition.
3. $0 = \emptyset$ and for any n, $\text{succ}(n) = n \cup \{n\} \neq \emptyset$, so 0 is not the successor of any number.
5. Any set S with the properties given is equal to $\mathbb{N}$ by the definition of $\mathbb{N}$.

The von Neumann Realization

We have skipped over the fourth Peano axiom: We need to show $\text{succ}(m) = \text{succ}(n)$ implies $m = n$. This is a little complicated and requires two further lemmas, which we discuss below.

1.3.2. Lemma. Let n be a natural number in the von Neumann realization (1.3.1). Then

$$m \in n \quad \Rightarrow \quad n \not\subset m, \tag{1.3.2}$$

or, equivalently,

$$n \subset m \quad \Rightarrow \quad m \notin n$$

Proof.

Let S be the set of natural numbers such that (1.3.2) holds for all $n \in S$. Our goal is to show that $S = \mathbb{N}$.

First, $n = 0 \in S$, since $0 = \emptyset$ and there are no elements $m \in 0$. (This is another example of a a statement being vacuously true.)

The von Neumann Realization

Proof (continued).

Next, let $n \in S$. Since $n \subset n$, we find that $n \notin n$. Then

$$\text{succ}(n) = n \cup \{n\} \not\subset n.$$

Furthermore, suppose that $\text{succ}(n) \subset m$ for some m. Then $n \subset m$ and we can deduce $m \notin n$.Hence, $\text{succ}(n) \not\subset n$ and $\text{succ}(n) \not\subset m$ for all $m \in n$.

Since the elements of $\text{succ}(n)$ consist of n and the elements of n, it follows that $\text{succ}(n) \not\subset m$ for all $m \in \text{succ}(n)$ and, therefore, $\text{succ}(n) \in S$.

We have shown that the set S contains 0 and the successor of every element of S. By the induction axiom (which we have already established for the von Neumann construction) it follows that $S = \mathbb{N}$. □

The above is a prototypical "proof by induction". In the next section, we will analyze develop this important method further.

The von Neumann Realization

1.3.3. Definition. A set A is called transitive if

$$y \in x \wedge x \in A \quad \Rightarrow \quad y \in A.$$

In particular, a set A is transitive if and only if $x \subset A$ for all $x \in A$.

1.3.4. Lemma. Let n be a natural number in the von Neumann realization (1.3.1). Then n is transitive.

Proof.
We again proceed by induction. Let S be the set of transitive natural numbers. Then $0 \in S$ is vacuously true. Now let $n \in S$. If $x \in \mathrm{succ}(n)$, then either $x \in n$ or $x = n$. If $x \in n$, then $x \subset n \subset \mathrm{succ}(n)$, because $n \in S$. If $x = n$, then $x \subset n \cup \{n\} = \mathrm{succ}(n)$. Hence, $\mathrm{succ}(n) \in S$ and we again deduce $S = \mathbb{N}$ by the induction axiom. □

The von Neumann Realization

Finally, we can prove that the von Neumann numbers satisfy the fourth Peano axiom:

1.3.5. Lemma. Let m and n be natural numbers in the von Neumann realization (1.3.1). Then n

$$\text{succ}(m) = \text{succ}(n) \quad \Rightarrow \quad m = n$$

Proof.
Let $n \cup \{n\} = m \cup \{m\}$. Then $n \in m$ or $n = m$. Similarly, $m = n$ or $m \in n$. Suppose that $m \neq n$. Then $n \in m$ and $m \in n$. By Lemma 1.3.4, $n \in n$. However, Lemma 1.3.2 then implies $n \notin n$, which is a contradiction. □

Addition

Once the set of natural numbers is established (e.g., through the Peano axioms), it is possible to define the operation of addition on $\mathbb{N}$. A detailed procedure is given in E. Landau's book *Foundations of Analysis*. Here, we restrict ourselves to listing the properties that this operation then has.

For any two natural numbers $a, b \in \mathbb{N}$ we can define the natural number $c = a + b \in \mathbb{N}$ called the *sum* of a and b. This addition has the following properties (here $a, b, c \in \mathbb{N}$):

1. $a + (b + c) = (a + b) + c$ (*Associativity*)
2. $a + 0 = 0 + a = a$ (*Existence of a neutral element*)
3. $a + b = b + a$ (*Commutativity*)

Multiplication

Similarly, we can define *multiplication*, where $a \cdot b \in \mathbb{N}$ is called the *product* of a and b. We have the following properties (here $a, b, c \in \mathbb{N}$):

1. $a \cdot (b \cdot c) = (a \cdot b) \cdot c$ (*Associativity*)
2. $a \cdot 1 = 1 \cdot a = a$ (*Existence of a neutral element*)
3. $a \cdot b = b \cdot a$ (*Commutativity*)

We also have a property that essentially states that addition and multiplication are *consistent*,

$$a \cdot (b + c) = a \cdot b + a \cdot c. \qquad \text{(Distributivity)}$$

Note that we are not able to define subtraction or division for all natural numbers.

Notation for Addition and Multiplication

For numbers $a_1, a_2, \ldots, a_n$ we define the notation

$$a_1 + a_2 + \cdots + a_n =: \sum_{j=1}^{n} a_j =: \sum_{1 \le j \le n} a_j$$

and

$$a_1 \cdot a_2 \cdots a_n =: \prod_{j=1}^{n} a_j =: \prod_{1 \le j \le n} a_j.$$

For $n \in \mathbb{N}$ we define

$$0! := 1 \qquad \text{and} \qquad n! := n \cdot (n-1)! \quad \text{for } n > 1.$$

This is an example of a *recursive definition*.

Powers and Divisors

We are also able to define the exponential

$$a^b = \underbrace{a \cdot a \cdot \ldots \cdot a}_{b \text{ times}} \qquad \text{for } a, b \in \mathbb{N}$$

by setting $a^0 := 1$ and $a^n := a \cdot a^{n-1}$. (This is another recursive definition.)
We note that

$$a^{b+c} = a^b \cdot a^c \qquad \text{and} \qquad (a^b)^c = a^{b \cdot c}.$$

For $a, b \in \mathbb{N}$ define the statement

$$a \mid b \qquad \Leftrightarrow \qquad \exists c \in \mathbb{N} \colon c \cdot a = b,$$

read as "a divides b." If $a \mid b$, then a is called a *divisor* of b.

Relations

We will now work to introduce further sets of numbers, such as integers and quotients. This requires some further set-theoretic background. For this, we will preview some of the material of Chapter 8 of the textbook.

Given two sets, the elements of these sets may be paired together in a certain way. A more precise formulation is given by the concept of a *relation*, which we now introduce. First, here are some examples to illustrate what we mean:

1.3.6. Examples.

- For $n \in \mathbb{N}$, $n \mapsto n^2$ associates to every number n its square n^2.
- $n^3 \mapsto n$ associates to some numbers $n^3 = p \in \mathbb{N}$ the number n.

We implement this by defining a relation R to be a set of ordered pairs. In the above examples, we might define

$$R = \{(a, b) \in \mathbb{N}^2 : b = a^2\},$$
$$R = \{(a, b) \in \mathbb{N}^2 : a = b^3\},$$

Relations

In general, we define a *relation R between two sets M and N* as

$$R = \{(m, n) \in M \times N \colon P(m, n)\} \tag{1.3.3}$$

where P is a statement frame (predicate). If $M = N$, we say that R is a *relation on M*.

Thereby the *active* concept of a relation (we associate one number to another; we *do* something) is defined by the *static* concept of a set (no action takes place; R is just an object). This is a fairly modern idea in mathematics.

Instead of writing $(a, b) \in R$, we often write $a \sim_R b$ (or $a \sim b$ if the relation R is clear from context).

We define the *domain* of a relation R by

$$\operatorname{dom} R = \{a \colon \exists b \colon (a, b) \in R\}$$

and the *range* by

$$\operatorname{ran} R = \{b \colon \exists a \colon (a, b) \in R\}.$$

Equivalence Relations

1.3.7. Definition. We say that a relation R on a set M is

1. *reflexive* if $(a, a) \in R$ for all $a \in M$;
2. *symmetric* if $(a, b) \in R$ implies $(b, a) \in R$ for all $a, b \in M$;
3. *transitive* if $(a, b) \in R$ and $(b, c) \in R$ implies $(a, c) \in R$ for all $a, b, c \in M$.

A reflexive, symmetric and transitive relation on M is called an *equivalence relation on* M. In more intuitive notation

- reflexivity means $a \sim a$ for all $a \in M$,
- symmetry means $a \sim b \Rightarrow b \sim a$,
- transitivity means $a \sim b \wedge b \sim c \Rightarrow a \sim c$.

Relations

1.3.8. Examples.

- The relation $R = \{(a, b) \in \mathbb{N}^2 : a > b\}$ includes the pairs $(1, 0), (2, 1), (2, 0)$ but not the pair $(0, 1)$. We write $(a, b) \in R \Leftrightarrow a \sim b \Leftrightarrow a > b$ and notice that $\sim$ is transitive (since $a > b$ and $b > c$ implies $a > c$), but not symmetric ($a > b \not\Rightarrow b > a$) or reflexive ($a \not> a$).
- For $n \in \mathbb{N}$ we define the *integer sum* $I(n)$ as the sum of all integers that compose the number, e.g. $I(125) = 1 + 2 + 5 = 8$, $I(78) = 7 + 8 = 15$.

  Then the relation $R = \{(a, b) \in \mathbb{N}^2 : I(a) = I(b)\}$ includes the pairs $(22, 4), (14, 5), (3, 30)$ but not the pair $(4, 1)$. We note that R is reflexive ($I(a) = I(a)$), symmetric (if $I(a) = I(b)$, then also $I(b) = I(a)$) and transitive if $I(a) = I(b)$ and $I(b) = I(c)$, then $I(a) = I(c)$), so R is an equivalence relation. We may then call all numbers with equal integer sums *equivalent* in this sense.

Equivalence Classes

A *partition* of a set A is a family $\mathcal{F}$ of disjoint subsets of A such that their union is A. An element of a partition is called a *fiber* or an *equivalence class*. An element of such an equivalence class is called a *representative* of the class.

1.3.9. Example. Denote by $2\mathbb{N} = \{0, 2, 4, 6, \ldots\} \subset \mathbb{N}$ the set of all even natural numbers and by $2\mathbb{N}+1 = \{1, 3, 5, 7, \ldots\} \subset \mathbb{N}$ the set of all odd natural numbers. Since

$$2\mathbb{N} \cap (2\mathbb{N}+1) = \emptyset \qquad \text{and} \qquad 2\mathbb{N} \cup (2\mathbb{N}+1) = \mathbb{N}$$

it follows that $\mathcal{F} = \{2\mathbb{N}, 2\mathbb{N}+1\}$ is a partition of $\mathbb{N}$ and $2\mathbb{N}$ and $2\mathbb{N}+1$ are the two equivalence classes of this partition. The number 4 is a representative of $2\mathbb{N}$, as are 0, 128 and 456, and the numbers 1, 7, 457 are representatives of $2\mathbb{N}+1$.

Equivalence Classes

We often denote equivalence classes by one of their representatives enclosed in square brackets, so we might write

$$2\mathbb{N} = [0] \qquad \text{and} \qquad 2\mathbb{N}+1 = [1].$$

1.3.10. Theorem. Every partition $\mathcal{F}$ of M induces an equivalence relation $\sim$ on a set M by

$$a \sim b \quad :\Leftrightarrow \quad a, b \in M \text{ are in the same equivalence class.} \tag{1.3.4}$$

Proof.
It is easily seen that the relation $\sim$ defined by (1.3.4) is reflexive, symmetric and transitive. □

Fundamental Theorem on Partitions and Equivalence Relations

1.3.11. Theorem. Every equivalence relation $\sim$ on a set M induces a partition $\mathcal{F} = \{[a] : a \in M\}$ of M by

$$a \in [b] \qquad :\Leftrightarrow \qquad a \sim b. \tag{1.3.5}$$

We write $\mathcal{F} = M/\sim$.

Proof.

Since $\sim$ is **reflexive**, it follows that $a \in [a]$ for every $a \in M$. Thus, the union of all classes is M. We need to show that the classes are disjoint. Let $[a], [b]$ be two classes. Assume that there is an element $c \in M$ such that $c \in [a]$ and $c \in [b]$. Then $c \sim b$ and $c \sim a$ for every $a \in [a]$. By **symmetry**, $a \sim c$ for every $a \in [a]$ and by **transitivity** $a \sim b$, i.e., $a \in [b]$ for every $a \in [a]$. It follows that $[a] \subset [b]$. Changing the roles of a and b, we have $[b] \subset [a]$, hence $[a] = [b]$. □

The Integers

We will now introduce the negative numbers. One big deficiency in the natural numbers is that there is no *inverse element* for addition, i.e., for every $n \in \mathbb{N}$ we would like to have an element $-n$ such that

$$n + (-n) = 0$$

Such an element does not exist in $\mathbb{N}$. We therefore consider the set of ordered pairs

$$\mathbb{N}^2 = \{(n, m) : m, n \in \mathbb{N}\}.$$

We can consider $\mathbb{N}$ as a natural subset of $\mathbb{N}^2$ by replacing $n \in \mathbb{N}$ with $(n, 0) \in \mathbb{N}^2$. Furthermore, we define the following equivalence relation on $\mathbb{N}^2$:

$$(n, m) \sim (p, q) \qquad :\Leftrightarrow \qquad n + q = m + p. \tag{1.3.6}$$

Therefore, the pair $(5, 0)$ (which corresponds to $5 \in \mathbb{N}$) is equivalent to $(6, 1)$, because $5 + 1 = 0 + 6$.

The Integers

We have the following facts:

- (1.3.6) defines an equivalence relation, which induces a partition $\mathbb{Z} = \mathbb{N}^2/\sim$ on $\mathbb{N}^2$.
- Every pair of the form $(n, 0) \in \mathbb{N}^2$, $n \in \mathbb{N}$, is in a different equivalence class of this partition. We denote these equivalence classes by $[+n] \ni (n, 0)$.
- Every pair of the form $(0, n) \in \mathbb{N}^2$, $n \in \mathbb{N}$, $n \geq 1$, is in yet another equivalence class, denoted by $[-n] \ni (0, n)$.
- Any other pair is in a class $[+n]$ or a class $[-n]$ for some $n \in \mathbb{N}$.
- It follows that

$$\mathbb{Z} = \{[+n] \colon n \in \mathbb{N}\} \cup \{[-n] \colon n \in \mathbb{N} \setminus \{0\}\}.$$

The Integers

We now want to define addition for elements of $\mathbb{N}^2$ by

$$(n, m) + (p, q) = (n + p, m + q). \tag{1.3.7}$$

If $(n, m) \sim (\widetilde{n}, \widetilde{m})$ and $(p, q) \sim (\widetilde{p}, \widetilde{q})$, then

$$(n, m) + (p, q) \sim (\widetilde{n}, \widetilde{m}) + (\widetilde{p}, \widetilde{q}). \tag{1.3.8}$$

This means that we can define the sum of two equivalence classes by (1.3.7): let $[\pm n], [\pm m] \in \mathbb{Z}$. Then we define $[\pm n] + [\pm m]$ as the class whose representatives are obtained by adding any representative of $[\pm n]$ to any representative of $[\pm m]$ according to (1.3.7). The fact that the result does not depend on which representative we choose is expressed by (1.3.8).

We say that the so defined addition on $\mathbb{Z}$ is *independent of the chosen representative* and therefore *well-defined*.

The Integers

Note that

$$(n, m) + (0, 0) = (n, m), \qquad (n, m) + (p, q) = (p, q) + (n, m)$$

and we also have the associative law of addition.Therefore, the addition on $\mathbb{Z}$ also has these properties and $[0] \in \mathbb{Z}$ is the neutral element of addition.

Furthermore, let $(n, m) \in \mathbb{N}^2$. Then

$$(n, m) + (m, n) = (n + m, n + m) \sim (0, 0).$$

In particular, $[n] + [-n] = [0]$, so we now have an inverse element for the addition defined on $\mathbb{Z}$.

The set $\mathbb{Z}$ is known as the set of *integers*. We write n instead of $[n]$ and $-n$ instead of $[-n]$. Furthermore, we abbreviate $n + (-m)$ by $n - m$ and call the "operation" $-$ *subtraction*. The letter $\mathbb{Z}$ is used for historical reasons: it stands for the German word for numbers, *Zahlen*.

The Integers

Let us review our strategy:

- The set of natural numbers was too small, because it didn't include an inverse element of addition.
- We introduce pairs of natural numbers and identify the natural numbers with certain pairs ($n \leftrightarrow (n, 0)$).
- We introduce a suitable equivalence relation on $\mathbb{N}^2$ and take $\mathbb{Z} = \mathbb{N}^2/\sim$ to be the induced partition.
- We characterize the partition $\mathbb{Z} = \{[n]\} \cup \{[-n]\}$.
- We define addition on $\mathbb{N}^2$ so that it is compatible with the previously defined addition on $\mathbb{N}$, i.e., we have $n \leftrightarrow (n, 0)$, $m \leftrightarrow (m, 0)$ and $(n, 0) + (m, 0) = (n + m, 0) \leftrightarrow n + m$.
- We show that the addition on $\mathbb{N}^2$ is compatible with the equivalence classes and can hence be used to define the sum of two classes in $\mathbb{Z}$.
- We see that for the addition in $\mathbb{Z}$ we now have an inverse element for every element of $\mathbb{Z}$.

Rational Numbers

We now repeat this strategy for the operation of multiplication: we still do not have an inverse element of multiplication in $\mathbb{Z}$.

We consider $\mathbb{Z}^2$, the set of pairs of integers. We define the equivalence relation

$$(n, m) \sim (p, q) \qquad :\Leftrightarrow \qquad n \cdot q = m \cdot p. \tag{1.3.9}$$

for $(n, m), (p, q) \in \mathbb{Z}^2$.

Furthermore, we define the product of two pairs of integers by

$$(n, m) \cdot (p, q) = (n \cdot p, m \cdot q) \tag{1.3.10}$$

and we consider $\mathbb{Z}$ as a subset of $\mathbb{Z}^2$ by associating $n \leftrightarrow (n, 1)$.

Rational Numbers

In the same way as previously it can now be shown that the multiplication (1.3.9) can be used to define a multiplication on $\mathbb{Q} := \mathbb{Z}^2/\sim$ and that this multiplication is well-defined, commutative, associative, has neutral element $[(1,1)]$ and every element $[(n,m)] \in \mathbb{Q}$ has an inverse element $[(n,m)^{-1}] = [(m,n)]$.

We further define addition on $\mathbb{Q}$ by

$$(m,n) + (p,q) = (q \cdot m + p \cdot n, nq).$$

We will ordinarily identify a representative (n,m) with its class $[(n,m)]$ and further write

$$(n,m) =: \frac{n}{m} \in \mathbb{Q}.$$

Comments on the Literature

The book by Landau is quite old (1929) and therefore very classical in its approach. Our constructions are essentially the same as those in that book, but there are some differences:

- Landau introduces natural numbers from the Peano axioms, without the set-theoretic construction we used.
- Landau starts with the positive natural numbers, then introduces fractions and then integers and rational numbers as equivalence classes of fractions. He then uses ordering relations $<$, $>$, $=$ to introduce negative numbers.
- While Landau uses "classes" and "equivalence", he does not introduce them as formally as we do.

Basic Properties of Numbers - Addition

From now on we assume that the natural numbers are constructed as in (1.3.1) and that from them we have obtained the integers and the rational numbers.

We briefly recapitulate the basic properties of the rational numbers. For addition we have

associativity	$\forall_{a,b,c\in\mathbb{Q}} \quad a+(b+c)=(a+b)+c$	**(P1)**
neutral element	$\exists_{0\in\mathbb{Q}} \forall_{a\in\mathbb{Q}} \quad a+0=0+a=a$	**(P2)**
inverse element	$\forall_{a\in\mathbb{Q}} \exists_{-a\in\mathbb{Q}} \quad (-a)+a=a+(-a)=0$	**(P3)**
commutativity	$\forall_{a,b\in\mathbb{Q}} \quad a+b=b+a.$	**(P4)**

Multiplication

For multiplication, we have a similar set of properties

associativity	$\forall_{a,b,c\in\mathbb{Q}} \quad a\cdot(b\cdot c)=(a\cdot b)\cdot c,$	**(P5)**
neutral element	$\exists_{\substack{1\in\mathbb{Q}\\1\neq 0}} \forall_{a\in\mathbb{Q}} \quad a\cdot 1=1\cdot a=a,$	**(P6)**
inverse element	$\forall_{a\in\mathbb{Q}} \exists_{a^{-1}\in\mathbb{Q}} \quad a\cdot a^{-1}=a^{-1}\cdot a=1,$	**(P7)**
commutativity	$\forall_{a,b\in\mathbb{Q}} \quad a\cdot b=b\cdot a.$	**(P8)**

With these properties we can prove that if $a\neq 0$ and $a\cdot b=a\cdot c$, then $b=c$.

Combining Addition and Multiplication

These twice four properties for addition and multiplication are basically independent of each other (the only connection was that $0 \neq 1$). The following property ensures that the two operations "work well together",

$$\text{distributivity} \qquad \forall_{a,b,c \in \mathbb{Q}} \quad a \cdot (b + c) = a \cdot b + a \cdot c. \qquad \textbf{(P9)}$$

It is this relation that allows us to prove that $a - b = b - a$ only if $a = b$, and it is also the law of distributivity that we employ to do elementary multiplications. Furthermore, we can now prove that if $a \cdot b = 0$, then either $a = 0$ or $b = 0$.

Size Comparisons

We will discuss general *ordering relations* in detail at a later time. For now, we define

$$\begin{aligned} m < n &\quad :\Leftrightarrow \quad m \subsetneq n && \text{for } m, n \in \mathbb{N}, \\ [(0,m)] < [(n,0)] & && \text{for all } [(0,m)] \in \mathbb{Z} \setminus \mathbb{N},\ n \in \mathbb{N}. \\ [(0,m)] < [(0,n)] &\quad :\Leftrightarrow \quad n < m && \text{for } [(0,m)], [(0,n)] \in \mathbb{Z} \setminus \mathbb{N}. \end{aligned}$$

For rational numbers $m \in \mathbb{Q}$, we first define

$$0 < m = \frac{p}{q} \quad :\Leftrightarrow \quad (p < 0 \wedge q < 0) \vee (0 < p \wedge 0 < q) \quad \text{for } p, q \in \mathbb{Z},$$

and then set

$$m < n \quad :\Leftrightarrow \quad 0 < n - m \quad \text{for } m, n \in \mathbb{Q}.$$

We also define

$$m \leq n \quad :\Leftrightarrow \quad (m < n) \vee (m = n) \quad \text{for } m, n \in \mathbb{Q}.$$

Lastly, we write $m > n$ if $n < m$ and $m \geq n$ if $n \leq m$.

Further Perspectives

The set of rational numbers is complete with respect to addition and multiplication. However, it is not so with respect to operations such as squaring: there is no inverse element for operation of squaring. Given $b \in \mathbb{Q}$, it may not be possible to find $a \in \mathbb{Q}$ such that $a^2 = b$.

To remedy this, the rational numbers can be extended further to the set of *real numbers*. As we will discuss later, while the set of rational numbers is countable, this extension will be uncountable. In particular, the real numbers do not fall within the purview of discrete mathematics, but rather in that of calculus and analysis. We will therefore not treat this further extension (and the subsequent extension to *complex numbers*) here. We do note that one way of defining real numbers involves equivalence classes of convergent and Cauchy sequences.

Mathematical Induction

Typically one wants to show that some statement frame $A(n)$ is true for all $n \in \mathbb{N}$ with $n \geq n_0$ for some $n_0 \in \mathbb{N}$. Mathematical induction works by establishing two statements:

(I) $A(n_0)$ is true.

(II) $A(n+1)$ is true whenever $A(n)$ is true for $n \geq n_0$, i.e.,

$$\forall_{\substack{n \in \mathbb{N} \\ n \geq n_0}} \big(A(n) \Rightarrow A(n+1)\big)$$

Note that (II) does not make a statement on the situation when $A(n)$ is false; it is permitted for $A(n+1)$ to be true even if $A(n)$ is false.

The principle of mathematical induction now claims that $A(n)$ is true for all $n \geq n_0$ if (I) and (II) are true. This follows from the fifth Peano axiom (the induction axiom).

Introductory Example

1.4.1. Example. Consider the statement

$$\sum_{k=1}^{n}(2k-1) = n^2 \qquad \text{for all } n \in \mathbb{N} \setminus \{0\}.$$

This is a typical example, in that $A(n)\colon \sum_{k=1}^{n}(2k-1) = n^2$ is a predicate which is to be shown to hold for all natural numbers $n > 0$.

We first establish that $A(1)$ is true:

$$\sum_{k=1}^{1}(2k-1) = 2 \cdot 1 - 1 = 1 \qquad \text{and} \qquad 1^2 = 1,$$

so $A(1)\colon 1 = 1$ is true.

Introductory Example

We next show that $A(n) \Rightarrow A(n+1)$ for all $n \in \mathbb{N} \setminus \{0\}$. This means we show that $\sum_{k=1}^{n+1}(2k-1) = (n+1)^2$ if $\sum_{k=1}^{n}(2k-1) = n^2$. Let n now be any n for which $A(n)$ is true. We then write

$$\sum_{k=1}^{n+1}(2k-1) = \sum_{k=1}^{n}(2k-1) + 2(n+1) - 1$$

If $A(n)$ is true for this specific n, we can replace the sum on the right by n^2, yielding

$$\sum_{k=1}^{n+1}(2k-1) = n^2 + 2n + 1 = (n+1)^2$$

But this is just the statement $A(n+1)$. Therefore, if $A(n)$ is true, then $A(n+1)$ will also be true. We have shown that $A(n) \Rightarrow A(n+1)$.

Foundations of Induction

Essentially, mathematical induction claims that for all $n_0 \in \mathbb{N}$,

$$\Big(A(n_0) \wedge \underset{\substack{n\in\mathbb{N}\\ n\geq n_0}}{\forall} (A(n) \Rightarrow A(n+1))\Big) \Rightarrow \underset{\substack{n\in\mathbb{N}\\ n\geq n_0}}{\forall} A(n). \tag{1.4.1}$$

To simplify the discussion, let us consider first the case $n_0 = 0$. Why is the conclusion (1.4.1) justified? Recall the 5th Peano axiom for the natural numbers:

> Induction axiom: *If a set $S \subset \mathbb{N}$ contains zero and also the successor of every number in S, then $S = \mathbb{N}$.*

Let us take $S = \{n \in \mathbb{N} \colon A(n)\}$. Then we first show that $A(0)$ is true, so $0 \in S$. Next we show that $A(n) \Rightarrow A(n+1)$ for all $n \in \mathbb{N}$. This means that if $n \in S$, then $n+1$ (the successor of n) is also in S. By the induction axiom, this means that $S = \mathbb{N}$. Thus, by showing the hypothesis in (1.4.1) we arrive at the conclusion, $A(n)$ is true for all $n \in \mathbb{N}$.

Foundations of Induction

To adapt the previous argument to the case where $n_0 > 0$, we can just take

$$S = \{n \in \mathbb{N} \colon n < n_0 \vee ((n \geq n_0) \wedge A(n))\}.$$

The details are left to you!

We observe that the 5th Peano axiom is basically tailor-made to ensure the validity of induction. We could paraphrase the axiom as "induction works!" In fact, there are alternative choices for the 5th axiom, such as the so-called

> Well-Ordering Principle: *Every non-empty set $S \subset \mathbb{N}$ has a least element.*

This axiom presupposes that we know what "least", or, more precisely, "less than" means for the natural numbers. While we will discuss ordering relations in detail later, for now we make a provisional definition as follows.

Foundations of Induction

Let S_m be the set containing m as well as the successor of any element of S_m. Then

$$m < n \quad :\Leftrightarrow \quad (n \in S_m) \wedge (n \neq m).$$

Of course, if we use the set-theoretic construction of the natural numbers (1.3.1) then we simply have $m \leq n \Leftrightarrow m \subset n$.

We now take a "least element of a set M" to be an element $m_0 \in M$ such that $M \subset S_{m_0}$, or in other words, $m_0 \leq m$ for all $m \in M$. We then have the Well-Ordering Principle can then be written as

$$\forall_{M \subset \mathbb{N}} \; \exists_{m \in M} \; M \subset S_m.$$

(The Well-Ordering Principle is sometimes also called the Least Element Principle.)

Foundations of Induction

1.4.2. Theorem. Assume that a system of numbers satisfies the first four Peano axioms and the Well-Ordering Principle. Then the Induction Axiom holds.

Proof.
Let $S \subset \mathbb{N}$ satisfy the property that $0 \in S$ and that if $n \in S$, then succ($n_{\in}S$. We need to show that $S = \mathbb{N}$. Suppose that there exists an $n_0 \in \mathbb{N}$ such that $n_0 \notin S$. Then the set $M = \{n \in \mathbb{N} : n \notin S\}$ is non-empty. By the well-ordering principle, M must have a least element m_0. Since $0 \in S$, $m_0 \neq 0$. Since $m_0 > 0$ and $m_0 \in \mathbb{N}$, there exists a number $m_0 - 1$ preceding m_0. This number is not in M, because m_0 is the least element of M. Therefore, $m_0 - 1 \in S$. However, by the definition of S, the successor of $m_0 - 1$ must also be in S. Since the successor of every number is unique, $m_0 \in S$ and hence $m_0 \notin M$. We arrive at a contradiction. □

Foundations of Induction

In fact, the Induction Axiom and the Well-Ordering Principle are equivalent: you will prove in the assignments that the Induction Axiom also implies the Well-Ordering Principle. This means that if we take one of the two as an axiom, the other becomes a theorem that can be proven. It is a common situation in mathematics that we have several equivalent choices for a system of axioms.

Further Examples of Induction

Mathematical induction can be used to prove any sort of statement on the natural numbers in a variety of contexts. Some examples follow:

1.4.3. Examples.

1. $\forall_{n\in\mathbb{N}} (1+\frac{1}{2})^n \geq 1+n/2.$
2. $\forall_{n\in\mathbb{N}} \forall_{a,b\in\mathbb{Q}} (a+b)^n = \sum_{k=0}^{n} \frac{n!}{(n-k)!k!} a^n b^{n-k}.$
3. $\forall_{n\in\mathbb{N}} \forall_{\substack{r\in\mathbb{Q} \\ r=p/q \\ q^2>p^2}} \sum_{k=0}^{n} r^k = \frac{r^{k+1}-1}{r-1}.$
4. Let $H_k = \sum_{k=1}^{n} \frac{1}{k}$ for $n \in \mathbb{N}\setminus\{0\}$. Then $\forall_{n\in\mathbb{N}} H_{2^n} \geq 1+n/2.$
5. Let M be a set and $A_1, \ldots, A_n \subset M$. Then

$$\left(\bigcap_{i=1}^{n} A_i\right)^{\mathrm{c}} = \bigcup_{i=1}^{n} A_i^{\mathrm{c}}.$$

Further Examples of Induction

6. Let M be a set with cardinality $\operatorname{card} M = n$, $n \in \mathbb{N}$. Then $\operatorname{card} \mathcal{P}(M) = 2^n$.
7. Shootout at the O.K. Corral: an odd number of lawless individuals, standing at mutually distinct distances to each other, fire pistols at each other in exactly the same instant. Every person fires at their nearest neighbor, hitting and killing this person. Then there is at least one survivor.

The Shootout

Let us discuss the last example in more detail: Let $P(n)$ be the statement that there is at least one survivor whenever $2n+1$ people, fire pistols at each other in the same instant. Each person fires at his nearest neighbor, and all people stand at mutually distinct distances to each other.

For $n = 1$, there are three people, A, B and C. Suppose that the distance between A and B is the smallest distance of any two of them. Then A fires at B and vice-versa. C fires at either A or B and will not be fired at, so C survives.

Suppose $P(n)$. Let $2n+3$ people participate in the shootout. Suppose that A and B are the closest pair of people. The A and B fire at each other.

- If at least one other person fires at A or B, then there remain $2n$ shots fired among the remaining $2n+1$ people, so there is at least one survivor.
- If no-one else fires at A or B, then there are $2n+1$ shots fired among the $2n+1$ people and by $P(n)$, there is at least one survivor.

Pitfalls in Induction

While mathematical induction is an extremely powerful technique, it must be executed most carefully. In proceeding through an induction proof it can happen quite easily that implicit assumptions are made that are not justified, thereby invalidating the result.

1.4.4. Example. Let us use mathematical induction to argue that every set of $n \geq 2$ lines in the plane, no two of which are parallel, meet in a common point.

The statement is true $n = 2$, since two lines are not parallel if and only if they meet at some point. Since these are the only lines under considerations, this is the common meeting point of the lines.

We next assume that the statement is true for n lines, i.e., any n non-parallel lines meet in a common point. Let us now consider $n+1$ lines, which we number 1 through $n+1$. Take the set of lines 1 through n; by the induction hypothesis, they meet in a common point. The same is true of the lines $2, \ldots, n+1$. We will now show that these points must be identical.

Pitfalls in Induction

Assume that the points are distinct. Then all lines $2, \ldots, n$ must be the same line, because any two points determine a line completely. Since we can choose our original lines in such a way that we consider distinct lines, we arrive at a contradiction. Therefore, the points must be identical, so all $n+1$ lines meet in a common point. This completes the induction proof.

Where is the mistake in the above "proof" of our (obviously false) supposition?

Strong (Complete) Induction

The method of induction can be strengthened. We can replace

(I) $A(n_0)$ is true.

(II) $A(n+1)$ is true whenever $A(n)$ is true for $n \geq n_0$.

with

(I) $A(n_0)$ is true.

(II') $A(n+1)$ is true whenever all the statements $A(n_0), A(n_0+1), \ldots, A(n)$ are true.

1.4.5. Example. We will show the following statement: *Every natural number $n \geq 2$ is a prime number or the product of primes.*

Clearly the statement is true for $n = 2$, which is prime. Next assume that $2, 3, \ldots, n$ are all prime or the product of prime numbers. Then $n+1$ is either prime or not prime. If it is prime, we are finished. If it is not prime, it is the product of two numbers $a, b < n+1$. However, a and b are themselves products of prime numbers by our assumption, and hence so is $n+1 = a \cdot b$.

Induction vs. Strong Induction

While induction is the principle that

$$\Big(A(n_0) \wedge \underset{\substack{n\in\mathbb{N}\\ n\geq n_0}}{\forall} (A(n) \Rightarrow A(n+1))\Big) \Rightarrow \underset{\substack{n\in\mathbb{N}\\ n\geq n_0}}{\forall} A(n) \qquad (1.4.2)$$

strong induction states

$$\Big(A(n_0) \wedge \underset{\substack{n\in\mathbb{N}\\ n\geq n_0}}{\forall} \big((A(n_0) \wedge \cdots \wedge A(n)) \Rightarrow A(n+1)\big)\Big) \Rightarrow \underset{\substack{n\in\mathbb{N}\\ n\geq n_0}}{\forall} A(n). \quad (1.4.3)$$

It is clear that (1.4.3) implies (1.4.2), since of course

$$\big((A(n_0) \wedge \cdots \wedge A(n)) \Rightarrow A(n+1)\big) \Rightarrow \big(A(n) \Rightarrow A(n+1)\big)$$

Thus the "usual" induction is a special case of strong induction. However, the converse is also true, as we shall see.

Induction vs. Strong Induction

We now show that (1.4.2) implies (1.4.3), i.e., strong induction follows from induction.

We fix $n_0 \in \mathbb{N}$ and define $B(n) : A(n_0) \wedge \cdots \wedge A(n)$ for $n \geq n_0$. Then $A(n_0) = B(n_0)$ and we can write strong induction as

$$\Big(B(n_0) \wedge \underset{\substack{n\in\mathbb{N}\\ n\geq n_0}}{\forall} \big(B(n) \Rightarrow A(n+1)\big)\Big) \Rightarrow \underset{\substack{n\in\mathbb{N}\\ n\geq n_0}}{\forall} A(n).$$

We can write

$$\big(B(n) \Rightarrow A(n+1)\big) \equiv \big(B(n) \Rightarrow (A(n+1) \wedge B(n))\big) \equiv \big(B(n) \Rightarrow B(n+1)\big)$$

so strong induction becomes

$$\Big(B(n_0) \wedge \underset{\substack{n\in\mathbb{N}\\ n\geq n_0}}{\forall} \big(B(n) \Rightarrow B(n+1)\big)\Big) \Rightarrow \underset{\substack{n\in\mathbb{N}\\ n\geq n_0}}{\forall} A(n). \qquad (1.4.4)$$

Since $\underset{\substack{n\in\mathbb{N}\\ n\geq n_0}}{\forall} A(n) \equiv \underset{\substack{n\in\mathbb{N}\\ n\geq n_0}}{\forall} B(n)$ we see that (1.4.4) is just induction in B.

Recursive Definitions

The induction axiom also allows us to make *recursive definitions*. For example, instead of defining $0! := 1$ and then setting

$$n! := \prod_{k=1}^{n} k, \qquad n \in \mathbb{N} \setminus \{0\}, \tag{1.4.5}$$

we could define

$$0! := 1, \qquad n! := n \cdot (n-1)!, \qquad n \in \mathbb{N} \setminus \{0\}. \tag{1.4.6}$$

The definition (1.4.6) is called an *inductive* or *recursive formula* for $n!$, while (1.4.5) is called a *closed formula* for $n!$. Recursive definitions often occur naturally in the formulation of a problem, and finding a closed formula can be extremely difficult. In some situations, a closed formula is highly desirable, while at other times, important properties are best expressed through recursive expressions.

Recursive Definitions

For example, there exists a continuous extension of the factorial, given by the *Euler gamma function*, defined for $t > 0$,

$$\Gamma(t) := \int_0^\infty z^{t-1} e^{-z}\, dz, \qquad t > 0.$$

It is possible to show that $\Gamma(1) = 1$ and that

$$\Gamma(t+1) = t\Gamma(t) = t\Gamma(t+1-1) \qquad \text{for } t > 0.$$

Comparing with (1.4.6), we see that

$$\Gamma(n+1) = n! \qquad \text{for } n \in \mathbb{N}.$$

Since the gamma function is defined for all positive numbers, we have a "continuous extension" of the factorial.

Justification of Recursive Definitions

The justification for recursive definitions again lies in the induction axiom: we define an expression for $n = 0$ and also for $n + 1$ given its value for $n \in \mathbb{N}$. In other words, it is defined on a set $S \subset \mathbb{N}$ including zero and all successors of elements in S. By the induction axiom, this defines the expression for all natural numbers $n \in \mathbb{N}$.

A slight modification allows for recursive definitions "starting" at $n = n_0$ instead of $n = 0$. Furthermore, we can define functions not just based on their preceding value, but on several such values.

1.4.6. Example. The *Fibonacci sequence* is defined through

$$f_0 := 0, \qquad f_1 := 1, \qquad f_n := f_{n-1} + f_{n-2}, \qquad n \in \mathbb{N} \setminus \{0, 1\}.$$

This type of recursive definition also follows from the induction axiom, much as strong induction does.

Recursive Definitions of Sets

In the same manner, we can define sets recursively. For example, consider the set $S \subset \mathbb{N}$ such that

$$3 \in S \qquad \text{and} \qquad x, y \in S \Rightarrow x + y \in S.$$

We know that $3 \in S$, so $3 + 3 = 6 \in S$, $3 + 6 = 9 \in S$, $6 + 6 = 12 \in S$ and so on. In fact, we will prove later that

$$S = \{n \in \mathbb{N} \colon \exists k \in \mathbb{N} \setminus \{0\} \colon n = 3k\}$$

This type of definition is also based on the induction axiom; in fact, stating that the set S contains 0 and the successor of any element of S is in itself making a recursive definition. The axiom states that this particular definition should be taken to give $\mathbb{N}$.

Alphabets and Strings

We introduce an example from the theory of formal languages:

1.4.7. Definition. An *alphabet* Σ is a finite, non-empty set of elements called *symbols*. We define the set Σ^* of *strings* (or *words*) over Σ as follows:

1. $\lambda \in \Sigma^*$, where λ is the *empty string* (*null string*) containing no symbols.
2. If $w \in \Sigma^*$ and $x \in \Sigma$, then $wx \in \Sigma^*$.

1.4.8. Example. Let $\Sigma = \{0, 1\}$. The elements of Σ are called *bits* and the words over Σ are called *bit strings*. λ is a word, so also $\lambda 0 = 0$ is a word, as is $\lambda 1 = 1$. Since $\{0, 1\} \subset \Sigma^*$, the two-symbol words $01, 10, 11, 00 \in \Sigma^*$ and, continuing, $000, 001, 010, 011, 100, 101, 110, 111 \in \Sigma^*$. In this way, we inductively find all words over Σ.

If we like, we can interpret strings as tuples or finite sequences, but that is not necessary.

Concatenation of Strings

To denote a non-empty word $w \neq \lambda$, we often write wx, where $w \in \Sigma^*$ and $x \in \Sigma$.

1.4.9. Definition. We define *concatenation of strings* as follows:

1. If $w \in \Sigma^*$, then $w \cdot \lambda = w$, where λ is the empty string.
2. If $w_1, w_2 \in \Sigma^*$, $x \in \Sigma$, then

$$w_1 \cdot (w_2 x) = (w_1 \cdot w_2)x.$$

This recursive definition is a bit difficult to read at first, so we give an example.

1.4.10. Example. Let $\Sigma = \{0, 1\}$. The concatenation of $110, 101 \in \Sigma^*$ according to Definition 1.4.9 is computed as follows:

$$\underset{w_1}{110} \cdot (\underset{w_2}{10}\,\underset{x}{1}) = (110 \cdot 10)1 = \big((110 \cdot 1)0\big)1$$
$$= \big(\big((110)1\big)0\big)1 = 110101.$$

Length of Strings

Definition 1.4.9 thus reduces the concatenation of strings to the concatenation of strings with symbols, which we know how to do from Definition 1.4.7. In other words, we concatenate words by appending one symbol at a time.

1.4.11. Definition. We define the *length* $l(w)$ *of a string* $w \in \Sigma^*$ as follows:

1. $l(\lambda) = 0$,
2. $l(wx) = l(w) + 1$, where $x \in \Sigma$.

Bit Strings in Logic

Bit strings can be used in logic programming by replacing the values T and F by 1 and 0, respectively. Logical operations then become analogously defined *bit operations*. In particular, $\neg$ becomes NOT, $\wedge$ becomes AND, $\vee$ becomes OR, where (for example)

b	NOT b
1	0
0	1

a	b	a AND b
1	1	1
1	0	0
0	1	0
0	0	0

a	b	a OR b
1	1	1
1	0	1
0	1	1
0	0	0

Given a bit string we then define corresponding *bitwise operations* recursively. For example,

$$\mathrm{NOT}(\lambda) := \lambda, \quad \mathrm{NOT}(wx) := \mathrm{NOT}(w)\,\mathrm{NOT}(x), \quad w \in \Sigma^*,\ x \in \Sigma.$$

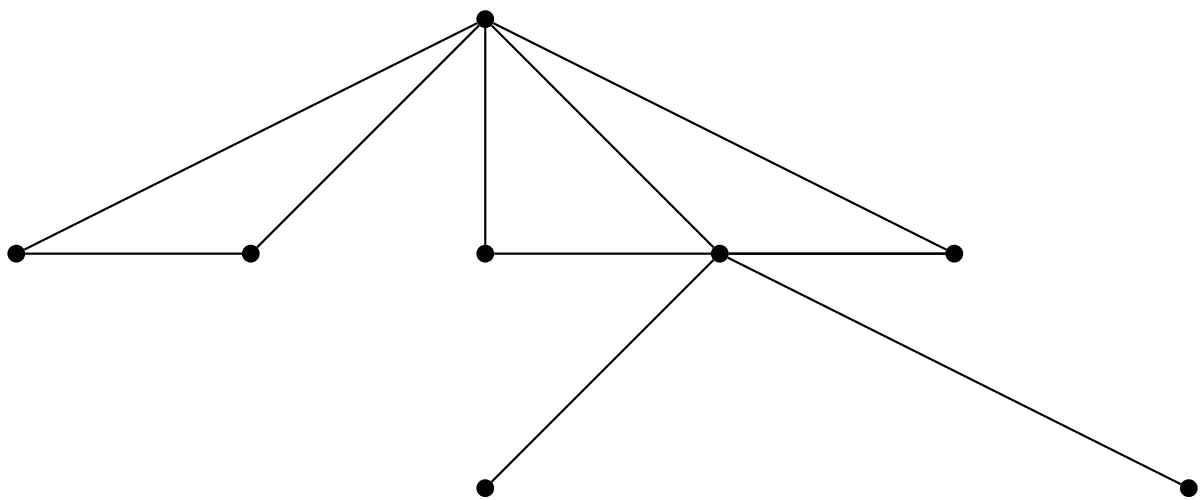
Thus

$$\mathrm{NOT}(011) = \mathrm{NOT}(01)\,\mathrm{NOT}(1) = \mathrm{NOT}(0)\,\mathrm{NOT}(1)\,\mathrm{NOT}(1) = 100.$$

Graphs

We will discuss the theory of trees and graphs extensively in a later section. For now, the type of graphs known as *rooted trees* serve as another example of an inductive definition.

A *graph* consists of *vertices* and *edges* that join two vertices. We will not define here what a vertex is, what an edge is or what "join" means in a mathematical sense. We visualize vertices as points and edges as straight lines joining these points. For example, the following is a graph:



Rooted Trees

We now define rooted trees inductively:

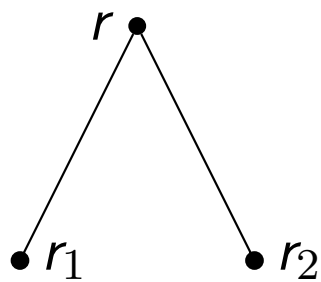
1.4.12. Definition.

1. A single vertex r is a rooted tree.
2. Let $T_1, \ldots, T_n$, $n \in \mathbb{N} \setminus \{0, 1\}$ be disjoint rooted trees with roots $r_1, \ldots, r_n$. Then the graph formed by starting with a root r, which is not a vertex of any of $T_1, \ldots, T_n$, and adding an edge from r to each of the vertices $r_1, \ldots, r_n$ is also a rooted tree.

1.4.13. Example. A rooted tree obtained from a single vertex r_1 by adding a vertex r and an edge joining the vertices is shown below:

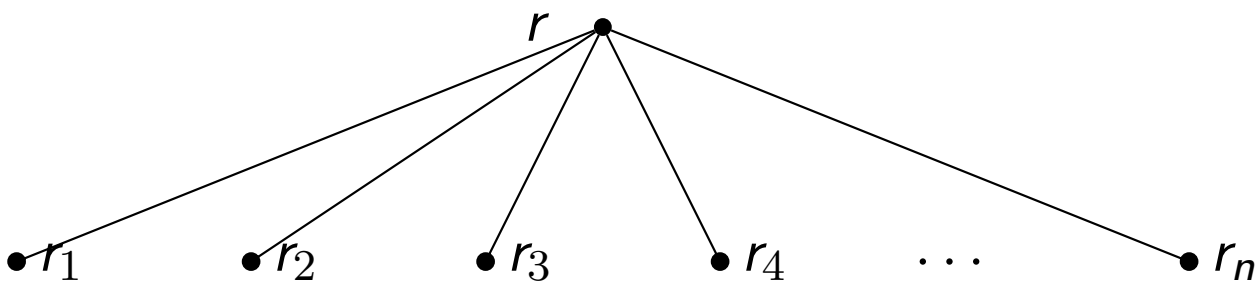
Rooted Trees

A rooted tree obtained from single vertices r_1, r_2 by adding a vertex r and edges joining the vertices to r is shown below:



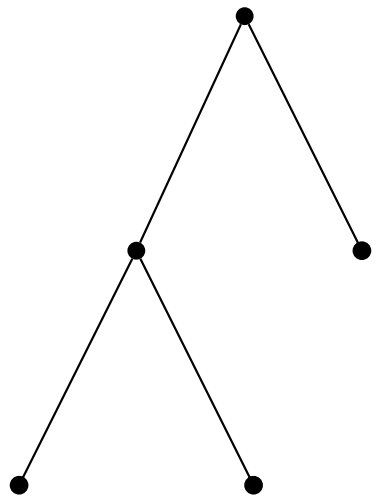
(1.4.7)

From n single vertices we obtain



Rooted Trees

If we add a single vertex to the tree (1.4.7) we obtain



It should now be clear how rooted trees are constructed inductively. More examples can be found in the textbook.

Extended Binary Trees

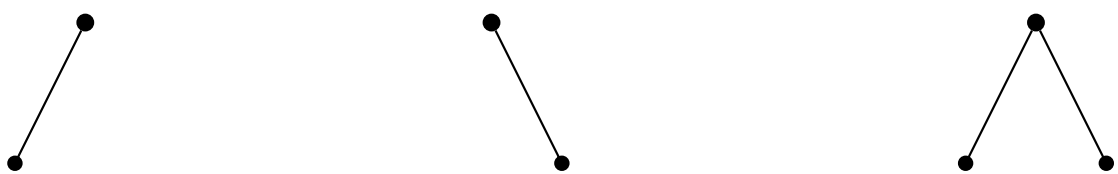
We now turn to a specific type of rooted tree, called a binary tree. We will distinguish two types:

1.4.14. Definition.

1. The empty set is an *extended binary tree*.
2. If T_1 and T_2 are extended binary trees, there is an extended binary tree, denoted $T_1 \cdot T_2$, consisting of a root r together with edges connecting the root to each of the roots of the trees T_1, T_2, whenever these trees are non-empty.

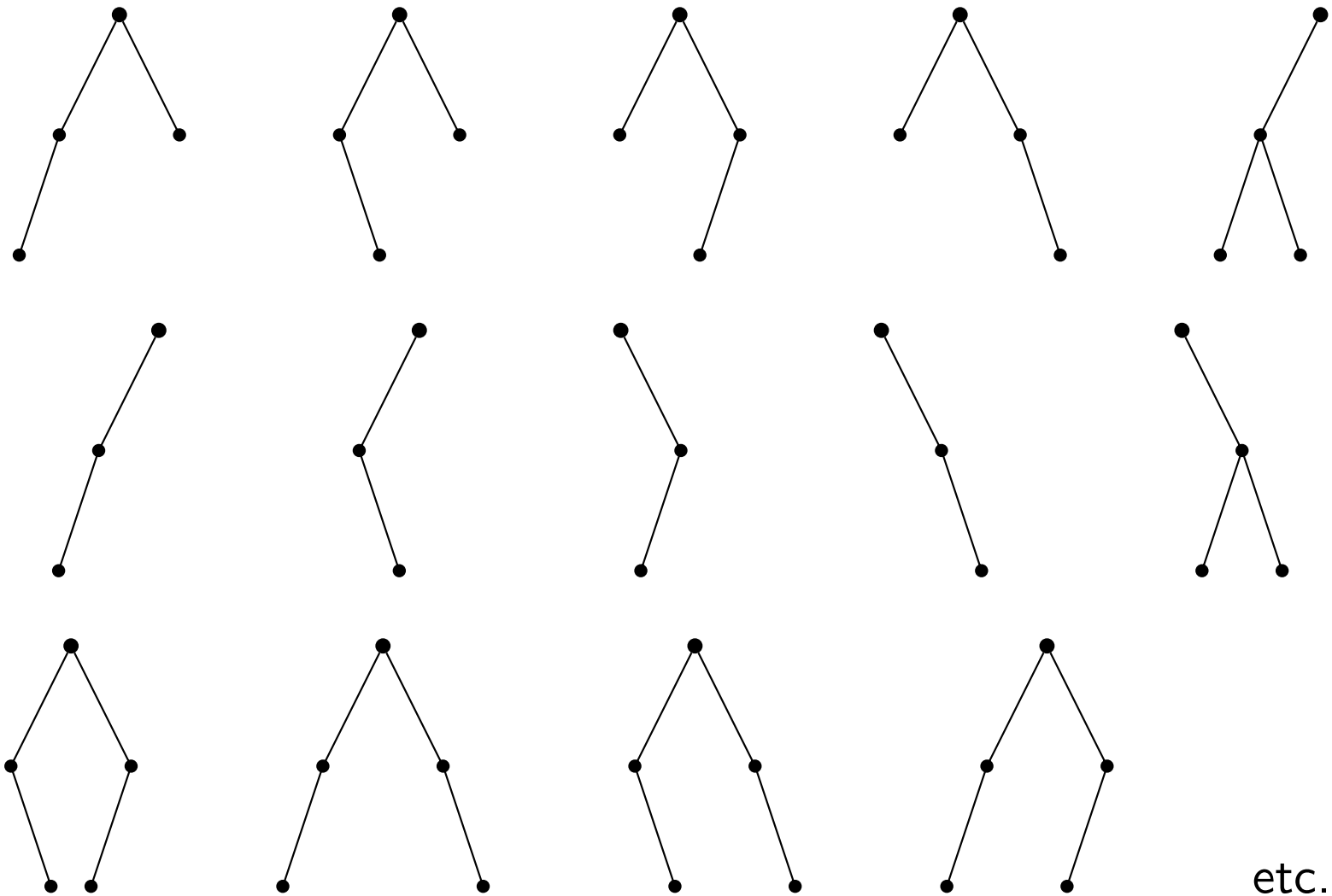
1.4.15. Example. Since the empty set $\emptyset$ is an extended binary tree (EBT), the first non-empty EBT is $\emptyset \cdot \emptyset$, consisting of a root r and no edges.

We construct further EBTs by taking $T_1 = \emptyset$ and $T_2 = r$, $T_1 = r$ and $T_2 = \emptyset$, and $T_1 = T_2 = r$:



Extended Binary Trees

Further extended binary tress are, for example,



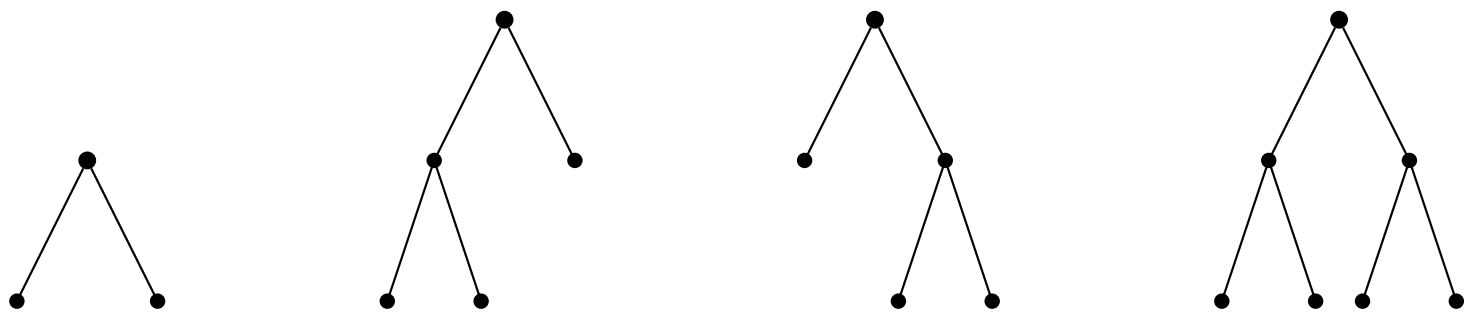
Full Binary Trees

1.4.16. Definition.

1. The tree consisting of a single vertex r is a *full binary tree*.
2. If T_1 and T_2 are full binary trees, there is an full binary tree, denoted $T_1 \cdot T_2$, consisting of a root r together with edges connecting the root to each of the roots of the trees T_1, T_2.

Note that the only difference to the definition of the extended binary tree is that we do not allow the empty set to be a full binary tree. This turns out to drastically effect the shapes of the trees.

1.4.17. Example. The first few full binary trees are



Structural Induction

Structural induction is a useful variant of induction that allows us to prove properties for recursively defined objects, such as the strings or trees we have just introduced.

Structural induction establishes a statement on a recursively defined set in two steps. We call those elements specifically included in the set (e.g., the empty string in Σ^* or the empty set in the set of extended binary trees) the basis elements of the set.

1. Establish the statement for the basis elements.
2. Show that if the statement is true for each of the elements used to construct new elements in the recursive step of the definition, the statement holds for these new elements.

Structural Induction

As a first example, consider a proposition $P(w)$, where $w \in \Sigma^*$ is a string. In order to prove that $P(w)$ is true for all $w \in \Sigma^*$, we need to show

1. $P(\lambda)$, where λ is the empty string.
2. $\forall_{w \in \Sigma^*} \forall_{x \in \Sigma} P(w) \Rightarrow P(wx)$.

1.4.18. Example. We prove that $l(xy) = l(x) + l(y)$ for $x, y \in \Sigma^*$, where $l(w)$ is the length of a string $w \in \Sigma^*$, cf. Definition 1.4.11.

We need to formulate the statement in such a way that we can imply induction. Let us write it as

$$P(y): \forall_{x \in \Sigma^*} l(xy) = l(x) + l(y).$$

We first establish $P(\lambda)$. This is the statement

$$P(\lambda): \forall_{x \in \Sigma^*} l(x\lambda) = l(x) + l(\lambda).$$

Since $x\lambda = x$ and $l(\lambda) = 0$, $P(\lambda)$ is true.

Structural Induction

Next, assume that $P(y)$ is true. We must now show that $P(ya)$ is true for all $a \in \Sigma$, i.e.,

$$P(y) \Rightarrow \underset{a\in\Sigma}{\forall} P(ya)$$

or

$$\left(\underset{x\in\Sigma^*}{\forall} l(xy) = l(x) + l(y)\right) \Rightarrow \left(\underset{a\in\Sigma}{\forall}\ \underset{x\in\Sigma^*}{\forall} l(xya) = l(x) + l(ya)\right)$$

Since $l(xya) = l(xy) + 1$ and $l(ya) = l(y) + 1$ by Definition 1.4.11, the implication follows and the proof is complete.

The justification for structural induction lies in the usual induction, applied to the statement

> $P(n)$: The claim is true for all elements of the set generated with n or fewer applications of the recursive rules for the set.

Structural induction first establishes $P(0)$ and then $P(n) \Rightarrow P(n+1)$.

Height and Vertices of a Full Binary Tree

1.4.19. Definition.

1. The height of the full binary tree T consisting of only a root r is $h(T) = 0$.
2. If T_1, T_2 are full binary trees, then the full binary tree $T = T_1 \cdot T_2$ has height $h(T) = 1 + \max(h(T_1), h(T_2))$.

We also define $n(T)$ to denote the number of vertices of a full binary tree. Note that $n(T) = 1$ if T is the full binary tree consisting of only a root r and $n(T_1 \cdot T_2) = 1 + n(T_1) + n(T_2)$. We now prove a result that relates $h(t)$ and $n(t)$.

1.4.20. Theorem. If T is a full binary tree, then $n(T) \leq 2^{h(T)+1} - 1$.

Height and Vertices of a Full Binary Tree

Proof.

We first show that the statement is true for the full binary tree T consisting of just one vertex. Since $n(T) = 1$ and $h(T) = 0$, we have

$$n(T) \leq 2^{h(T)+1} + 1 \qquad \Leftrightarrow \qquad 1 \leq 2^{0+1} - 1,$$

which is clearly true. Next, we need to show that

$$\underset{\substack{\text{full binary trees} \\ T_1, T_2}}{\forall} \Big(\big((n(T_1) \leq 2^{h(T_1)+1} - 1) \wedge (n(T_2) \leq 2^{h(T_2)+1} - 1) \big) \Rightarrow (n(T_1 \cdot T_2) \leq 2^{h(T_1 \cdot T_2)+1} - 1) \Big)$$

Height and Vertices of a Full Binary Tree

Proof (continued).

We have

$$\begin{aligned} n(T_1 \cdot T_2) &= 1 + n(T_1) + n(T_2) \\ &\leq 1 + 2^{h(T_1)+1} - 1 + 2^{h(T_2)+1} - 1 \\ &\leq 2 \max(2^{h(T_1)+1}, 2^{h(T_2)+1}) - 1 \\ &= 2 \cdot 2^{\max(h(T_1),h(T_2))+1} - 1 \\ &= 2^{h(T_1 \cdot T_2)+1} - 1, \end{aligned}$$

completing the proof. □

The Rational Numbers are Insufficient

While the rational numbers are closed with respect to addition and multiplication, and furthermore are large enough to include "inverse elements" $-a$ and a^{-1} for these operations, they are still "incomplete" in more than one sense:

1. Algebraically: there is no inverse element for operation of squaring. Given $b \in \mathbb{Q}$, it may not be possible to find $a \in \mathbb{Q}$ such that $a^2 = b$.
2. Analytically: Given a sequence of rational numbers that come closer and closer to each other, there may not be a "limiting value" in $\mathbb{Q}$. (We will discuss this later.)
3. Set theoretically: Given a bounded subset of $\mathbb{Q}$, there may not be a least upper or least lower bound for this set.

We will discuss i) and iii) in more detail now, and come back to ii) later, when we construct a suitable expansion of the rational numbers.

The Square Root Problem

1.5.1. Theorem. There exists no number $a \in \mathbb{Q}$ such that $a^2 = 2$.

Proof.
Assume that $a = p/q \in \mathbb{Q}$ has the property that $(p/q)^2 = 2$, where $p, q \in \mathbb{N}$. We further assume that p and q have no common divisor. Then

$$p^2 = 2q^2,$$

so p^2 is even. This implies that p is even, so $p = 2k$ for some $k \in \mathbb{N}$. But then

$$2q^2 = 4k^2 \quad \Rightarrow \quad q^2 = 2k^2 \text{ is even} \quad \Rightarrow \quad q \text{ is even.}$$

Thus $2 \mid p$ and $2 \mid q$. ↯ □

Bounded Subsets of $\mathbb{Q}$

1.5.2. Definition. We say that a set $U \subset \mathbb{Q}$ is *bounded* if there exists a constant $c \in \mathbb{Q}$ such that

$$|x| \leq c \qquad \text{for all } x \in U.$$

If this is not the case, we say that U is *unbounded*.

Numbers c_1 and c_2 such that

$$c_1 \leq x \leq c_2 \qquad \text{for all } x \in U$$

are called *lower* and *upper bounds* for U, respectively.

We say that a set $U \in \mathbb{Q}$ is *bounded above* if there exists an upper bound for U, and *bounded below* if there exists a lower bound for U.

Bounded Subsets of $\mathbb{Q}$

1.5.3. Examples.

- The set $U = \{1/n \colon n \in \mathbb{N}^*\}$ is bounded since $|x| \leq 1$ for all $x \in U$. Furthermore, the number -1 is a lower bound and 1 is an upper bound. Of course, there are many more upper and lower bounds.
- The set $\mathbb{N} \subset \mathbb{Q}$ is bounded below (lower bound: 0), but not bounded above. It is unbounded.

Maxima and Minima of Sets

1.5.4. Definition. Let $U \subset \mathbb{Q}$ be a subset of the rational numbers. We say that a number $x_1 \in U$ is the *minimum* of U if

$$x_1 \leq x, \qquad \text{for all } x \in U$$

and we write $x_1 =: \min U$.

Similarly, we say that $x_2 \in U$ is the *maximum* of U if

$$x_2 \geq x, \qquad \text{for all } x \in U$$

and write $x_2 =: \max U$.

A set that is not bounded below has no minimum, and a set that is not bounded above has no maximum. However, even a bounded set does not need to have a maximum or a minimum.

Greatest Lower and Least Upper Bounds of Sets

1.5.5. Definition. We say that an upper bound $c_2 \in \mathbb{Q}$ of a set $U \subset \mathbb{Q}$ is the *least upper bound* of U if no $c \in \mathbb{Q}$ with $c < c_2$ is an upper bound. We then write $c_2 =: \sup U$.

We say that a lower bound $c_1 \in \mathbb{Q}$ is the *greatest lower bound* of U if no $c \in \mathbb{Q}$ with $c > c_1$ is a lower bound. We then write $c_1 =: \inf U$.

1.5.6. Example. The set $U = \{1/n \colon n \in \mathbb{N}^*\}$ has

$$\inf U = 0, \qquad \sup U = 1,$$
$$\min U \text{ does not exist}, \qquad \max U = 1.$$

In general, if $\max U$ exists, $\sup U = \max U$, and if $\min U$ exists, $\inf U = \min U$.

Existence of Suprema and Infima

Not every bounded set in $\mathbb{Q}$ has an infimum or supremum: consider

$$U = \{x \in \mathbb{Q} : (x > 0) \wedge (2 < x^2 \leq 4)\}$$

It is clear that $\sup U = \max U = 2$, but the minimum and even the infimum do not exist (you will prove this in the exercises).

On the other hand, if we add a postulate to our previous axioms, we can widen the rational numbers to include solutions of $x^2 = 2$.

The Real Numbers

We define the set of real numbers $\mathbb{R}$ as the smallest extension of the rational numbers $\mathbb{Q}$ such that the following property holds:

> ***(P13)*** *If $A \subset \mathbb{R}$, $A \neq \emptyset$ is bounded above, then there exists a least upper bound for A in $\mathbb{R}$.*

We will not construct the such a set in this course. Instead, we stipulate that such a set exists, and we call it the set of *real numbers* $\mathbb{R}$. Furthermore, we can add and multiply elements of $\mathbb{R}$ and all relevant properties of the operations (associativity, commutativity, distributivity etc.) remain valid for elements of $\mathbb{R}$.

Note that (P13) is sufficient to guarantee the existence of greatest lower bounds; we need but consider $-A = \{x \in \mathbb{R} : \ -x \in A\}$ instead of A. The least upper bound of $-A$ will yield the negative of the greatest lower bound of A.

The Square Root Problem

We can now solve the square root problem:

1.5.7. Theorem. For every $x > 0$, the set $\mathbb{R}$ contains exactly one strictly positive solution to the equation $y^2 = x$.

Proof.

The proof is in two parts: uniqueness and existence, i.e., we show separately that i) there exists a solution and ii) the solution is unique. Often, it is easier to show uniqueness than existence. Also, the uniqueness proof may yield helpful results that can be used for the (generally more difficult) existence proof. Therefore, one often starts with proving the uniqueness.

Assume that there are two numbers $y_1 > y_2 > 0$ such that $y_1^2 = y_2^2 = x$. Then

$$0 = y_1^2 - y_2^2 = \underbrace{(y_1 - y_2)}_{>0}\underbrace{(y_1 + y_2)}_{>0} > 0$$ ↯

The Square Root Problem

Proof (continued).

We now show the existence of a solution to the equation $y^2 = x$. Let $M := \{t \in \mathbb{R} \colon t > 0 \wedge t^2 > x\}$. Then M is bounded from below and by (P13) it has a greatest lower bound $y = \inf M$. We will establish that $y^2 = x$ by showing that $y^2 > x$ and $y^2 < x$ lead to contradictions.

Assume that $y^2 > x$, so $y > \frac{x}{y}$. Then

$$0 < \left(y - \frac{x}{y}\right)^2 = y^2 - 2y\frac{x}{y} + \left(\frac{x}{y}\right)^2,$$

so that

$$y^2 + \left(\frac{x}{y}\right)^2 > 2y\frac{x}{y} = 2x.$$

The Square Root Problem

Proof (continued).

Set $s := \frac{1}{2}(y + \frac{x}{y})$. Then

$$s^2 = \frac{1}{4}\left(y^2 + 2\frac{x}{y}y + \left(\frac{x}{y}\right)^2\right) > \frac{1}{4}(2x + 2x) = x$$

so $s \in M$. However,

$$s = \frac{1}{2}(y + \frac{x}{y}) < \frac{1}{2}(y + y) = y = \inf M \lightning$$

Similarly, it can be shown that $y^2 < x$ leads to a contradiction. □

Therefore the real numbers include all square roots of positive numbers; we denote the *positive* solution to $y^2 = x$ by $y = \sqrt{x}$. Similarly, we write the positive solution to $y^n = x$ as $y = \sqrt[n]{x}$.

We call all real numbers that are not rational *irrational numbers*.

Subsets of the Real Numbers

We introduce some notation we will have occasion to use often later.

1.5.8. Definition. Let $a, b \in \mathbb{R}$ with $a < b$. Then we define the following special subsets of $\mathbb{R}$, which we call *intervals*:

$$
\begin{aligned}
[a, b] &:= \{x \in \mathbb{R} \colon (a \leq x) \wedge (x \leq b)\} = \{x \in \mathbb{R} \colon a \leq x \leq b\},\\
[a, b) &:= \{x \in \mathbb{R} \colon (a \leq x) \wedge (x < b)\} = \{x \in \mathbb{R} \colon a \leq x < b\},\\
(a, b] &:= \{x \in \mathbb{R} \colon (a < x) \wedge (x \leq b)\} = \{x \in \mathbb{R} \colon a < x \leq b\},\\
(a, b) &:= \{x \in \mathbb{R} \colon (a < x) \wedge (x < b)\} = \{x \in \mathbb{R} \colon a < x < b\}.
\end{aligned}
$$

Furthermore, for any $a \in \mathbb{R}$ we set

$$
\begin{aligned}
[a, \infty) &:= \{x \in \mathbb{R} \colon x \geq a\}, & (a, \infty) &:= \{x \in \mathbb{R} \colon x > a\},\\
(-\infty, a] &:= \{x \in \mathbb{R} \colon x \leq a)\}, & (-\infty, a) &:= \{x \in \mathbb{R} \colon x < a\}
\end{aligned}
$$

Finally, we set $(-\infty, \infty) := \mathbb{R}$.

Subsets of the Real Numbers

1.5.9. Definition. We call an interval I

- *open* if $I = (a, b)$, $a \in \mathbb{R}$ or $a = -\infty$, $b \in \mathbb{R}$ or $b = \infty$;
- *closed* if $I = [a, b]$, $a, b \in \mathbb{R}$;
- *half-open* if is neither open or closed.

We often denote intervals by capital letters I or J. If we say "Let I be an open interval..." we mean "Let $I \subset \mathbb{R}$ be a set such that there exist numbers $a, b \in \mathbb{R}$ such that $I = (a, b)$..."

Subsets of the Real Numbers

More generally, we have the following classification of points with respect to a set:

1.5.10. Definition. Let $A \subset \mathbb{R}$.

1. We call $x \in \mathbb{R}$ an *interior point of A* if there exists some $\varepsilon > 0$ such that the interval $(x-\varepsilon, x+\varepsilon) \subset A$.
2. We call $x \in \mathbb{R}$ an *exterior point of A* if there exists some $\varepsilon > 0$ such that the interval $(x-\varepsilon, x+\varepsilon) \cap A = \emptyset$.
3. We call $x \in \mathbb{R}$ a *boundary point of A* if for every $\varepsilon > 0$ $(x-\varepsilon, x+\varepsilon) \cap A \neq \emptyset$ and $(x-\varepsilon, x+\varepsilon) \cap A^c \neq \emptyset$.

Subsets of the Real Numbers

We can hence define general open and closed sets:

1.5.11. Definition.

1. A set $A \subset \mathbb{R}$ is called open if all points of A are interior points.
2. A set $A \subset \mathbb{R}$ is called closed if it contains all its boundary points.
3. Let $A \subset \mathbb{R}$ be any set and $B \subset \mathbb{R}$ the set of boundary points of A. Then $\overline{A} := A \cup B$ is called the *closure of* A.

Functions and Maps - Definitions

Recall that we have defined a relation R between two sets X, Y as

$$R = \{(x, y) \colon P(x, y),\ x \in X,\ y \in Y\}$$

where P is a predicate.

1.6.1. Definition. A *function* or *map* is defined as a relation R with the additional property that

$$\forall (x_1, y_1) \in R\, \forall (x_2, y_2) \in R \colon (x_1 = x_2 \ \Rightarrow\ y_1 = y_2).$$

The definitions of domain and range for relations of course carry over to functions,

$$\operatorname{dom} R = \{x \in X \colon \exists y \colon (x, y) \in R\},$$
$$\operatorname{ran} R = \{y \in Y \colon \exists x \colon (x, y) \in R\}.$$

Functions and Maps - Definitions

This means that for every $x \in \operatorname{dom} R$, there is only a single pair $(x, y) \in R$. In other words, the function R associates to every $x \in \operatorname{dom} R$ precisely one $y \in \operatorname{ran} R$. We denote this element by $y = R(x)$ or $y = R_x$, writing

$$R = \{(x, R(x)) : x \in \operatorname{dom} R\}.$$

The association $x \mapsto R(x)$ is determined through the statement frame $P(x, R(x))$, which must be true for $x \in \operatorname{dom} R$.

Let R be a function, $\operatorname{dom} R =: \Omega \subset X$, $\operatorname{ran} R \subset Y$. We say that R *maps* Ω *into* Y and write

$$R\colon \Omega \to Y, \qquad\qquad R\colon x \mapsto R(x).$$

Instead of R we sometimes write $R(\,\cdot\,)$, where the dot indicates where the variable x is inserted. This is often a useful notation for functions which are not represented by letters.

Definition and Graph

In order to specify a unique function we thus need to give

- the domain Ω and the space Y containing the range,
- the association ("mapping") $x \mapsto R(x)$.

Although a function is actually defined through a set of pairs of elements, sometimes one prefers to think of a function as being given by i) and ii) above, and denotes by

$$\Gamma(f) = \{(x, f(x)) \in X \times Y \colon x \in \operatorname{dom} f\}$$

the *graph* of the function f.

If $\Omega \subset \mathbb{R}$ and $Y = \mathbb{R}$, we visualize the graph through the drawing of two perpendicular axes, the horizontal of which represents the real numbers that encompass the domain, the vertical representing the reals including the range. In this way, a point $(x, y) \in \mathbb{R}^2$ is represented by a point in the "coordinate system" of these graphs.

Compositions, Sums, Products

If X, Y, Z are sets, $\Sigma \subset Y$, and two maps $f\colon X \to Y$, $g\colon \Sigma \to Z$ are given such that $\operatorname{ran} f \subset \Sigma = \operatorname{dom} g$, then we define the *composition*

$$g \circ f\colon X \to Z, \qquad g \circ f\colon x \mapsto g(f(x)).$$

If $f, g\colon X \to Y$, we define the *pointwise sum* and *pointwise product* of f and g by

$$\begin{aligned} f+g\colon X &\to Y, & (f+g)(x) &:= f(x) + g(x), \\ fg\colon X &\to Y, & (fg)(x) &:= f(x) \cdot g(x). \end{aligned}$$

Injections, Surjections, Bijections

A function $f\colon X \to Y$ is called

1. *injective* if $f(x_1) = f(x_2) \Rightarrow x_1 = x_2$ for all $x_1, x_2 \in X$;
2. *surjective* if $\operatorname{ran} f = Y$;
3. *bijective* if it is injective and surjective.

If f is bijective, then there exists an inverse function $f^{-1}\colon Y \to X$ such that $f^{-1}(f(x)) = x$ for all $x \in X$ and $f(f^{-1}(y)) = y$ for all $y \in Y$.

Important Functions in Discrete Mathematics

We now introduce the *floor* and *ceiling* functions.

1.6.2. Definition. The function

$$\lfloor \cdot \rfloor : \mathbb{R} \to \mathbb{Z}, \qquad \lfloor x \rfloor = \max\{z \in \mathbb{Z} : z \leq x\}$$

is called the *floor function*. Sometimes it is also called the *greatest integer function* and denoted by $[\cdot]$ instead of $\lfloor \cdot \rfloor$.

The function

$$\lceil \cdot \rceil : \mathbb{R} \to \mathbb{Z}, \qquad \lceil x \rceil = \min\{z \in \mathbb{Z} : z \geq x\}$$

is called the *ceiling function*.

We denote by log the logarithm to base 2, i.e., $\log x := \log_2 x$.

We will occasionally use other functions which we assume to be known from calculus, such as the absolute value function $|\cdot|$.

Sequences

We assume that sequences have been studied to a large extent in calculus. For reference, a sequence is considered simply to be a function $a\colon \mathbb{N} \to \mathbb{R}$, so that each $n \in \mathbb{N}$ is mapped to some real number.

We often write $a_n = a(n)$. We sometimes list a sequence by giving its values for $n = 0, 1, 2, 3, \ldots$. In this spirit, the sequence a is also often denoted $(a_n)_{n\in\mathbb{N}}$, $(a_n)_{n=0}^{\infty}$ or simply (a_n).

1.6.3. Example. The sequence $a\colon n \mapsto n^2$ can be written as

$$(a_n) = (0, 1, 4, 9, 16, 25, \ldots),$$

$(n^2)_{n\in\mathbb{N}}$ or as "the sequence with values $a_n = n^2$."

The above generalizes easily to sequences defined on a subset $\{n \in \mathbb{N}\colon n \geq n_0,\ n_0 \in \mathbb{N}\}$ of the natural numbers. Sequences whose values are integers are called *integer sequences*.

Limits of Functions and Sequences

In the following, we will assume some knowledge from calculus about limits of functions. Although we would be able to formulate many of the following statements and definitions without using limits, it certainly makes our life much easier if we use the corresponding techniques. We briefly recall the following definition.

1.6.4. Definition. Let f be a real-valued function defined either

- on an interval (a, ∞) for some $a \in \mathbb{R}$ or
- on a subset $\{n \in \mathbb{N} \colon n \geq n_0,\ n_0 \in \mathbb{N}\}$ of the natural numbers (i.e., f is a sequence).

Then we say that *the limit of f as $x \to \infty$ is equal to $C \in \mathbb{R}$*, written

$$\lim_{x\to\infty} f(x) = C \qquad :\Leftrightarrow \qquad \forall_{\varepsilon>0}\ \exists_{L>0}\ \forall_{x\in\operatorname{dom} f}\ x > L \Rightarrow |f(x) - C| < \varepsilon.$$

We sometimes write "$f(x) \to L$ as $x \to \infty$" instead of "$\lim_{x\to\infty} f(x) = L$".

Summability of Sequences and Series

The *partial sum* of a sequence $a\colon \mathbb{N} \to \mathbb{R}$ is defined as

$$S_n = \sum_{k \leq n} a_k.$$

We thus obtain a sequence (S_n). If this sequence converges to the limit $C \in \mathbb{R}$ as $n \to \infty$ we say that a is *summable* and the *sum of a is C*. We express this by writing

$$\sum_{n=n_0}^{\infty} a_n = C,$$

assuming that a is defined for $n \in \mathbb{N}$ with $n \geq n_0$.

1.6.5. Example. Fix $x \in \mathbb{R}$ and consider the sequence given by $a_n = x^n$, $n \in \mathbb{N}$. Then (a_n) is summable if and only if $|x| < 1$. In that case,

$$\sum_{n=0}^{\infty} x^n = \frac{1}{1-x}$$

Landau Symbols

In some sense, we would expect the functions $f, g\colon \mathbb{R} \to \mathbb{R}$

$$f(x) = \sqrt{x^2 + 1} \qquad \text{and} \qquad g(x) = x$$

to behave similarly as $x \to \infty$, even if the individual limits do not exist.

In the same way, $1/x^2$ should be "much smaller" than $1/x$ as $x \to \infty$, even though both limits vanish.

The main tools for characterizing such behavior are known as *Landau symbols*. There are two different "kinds": big-Oh and little-Oh. We introduce big-Oh in the lecture and leave little-Oh to the exercises.

Landau Symbols

1.6.6. Definition. Let f be a real-valued function defined either

- on an interval (a, ∞) for some $a \in \mathbb{R}$ or
- on a subset $\{n \in \mathbb{N}\colon n \geq n_0,\ n_0 \in \mathbb{N}\}$ of the natural numbers.

Then we say that

$$f(x) = O(\phi(x)) \qquad \text{as } x \to \infty$$

if there exist constants $C, L \geq 0$ such that

$$|f(x)| \leq C|\phi(x)| \qquad \text{whenever } x > L. \tag{1.6.1}$$

In this course, we will often leave out the words "as $x \to \infty$". In calculus, one is often interested in the behavior of functions near arbitrary points in $\mathbb{R}$, not just "near infinity", so limits and Landau symbols are defined for these points, too, and the the clarification "as $x \to \infty$" is necessary. Here, we will only look at this case, so we may omit it.

Landau Symbols

1.6.7. Examples.

1. $x + x^2 = O(x^2)$ as $x \to \infty$,
2. $\dfrac{1}{x^2 + x} = O\left(\dfrac{1}{x}\right)$ as $x \to \infty$,
3. $\dfrac{1}{x^2} = O\left(\dfrac{1}{x}\right)$ as $x \to \infty$.

Landau Symbols

More properly, "$O(\phi(x))$" denotes a set: the set of all functions f such that for some $C \geq 0$ and some $L > 0$, $|f(x)| \leq C|\phi(x)|$ for all $x > L$. A better notation for this set might be $O_{x_0}(\phi)$ and we could (perhaps more correctly) write

$$f \in O(\phi) \qquad \text{instead of} \qquad f(x) = O(\phi(x)) \quad \text{as } x \to \infty.$$

However, the second notation has become standard among physicists and mathematicians, and we write expressions like

$$(x+1)^4 = x^4 + O(x^3) \qquad \text{meaning} \qquad (x+1)^4 - x^4 = O(x^3).$$

This is often more intuitive than writing

$$(x+1)^4 - x^4 \in O(x^3) \qquad \text{or} \qquad (x+1)^4 = x^4 \bmod (x^3).$$

Some Remarks on Landau Symbols

We do sacrifice the symmetry of the "=" sign, however. For example,

$$O(x^2) = O(x^3) \quad \text{as } x \to \infty$$

means "a function f such that $|f(x)| \leq C|x|^2$ for all $x > L$ for some L, C also satisfies $|f(x)| \leq C'|x|^3$ for all $x > L'$ for some L', C'." Clearly,

$$O(x^3) = O(x^2) \quad \text{as } x \to \infty \qquad \text{is false.}$$

Landau symbols can be combined with each other and with functions. For example, as $x \to \infty$,

$$c \cdot O(x^n) = O(x^n), \qquad x^p O(x^q) = O(x^{p+q}),$$
$$O(x^n) + O(x^m) = O(x^{\max(n,m)}), \qquad O(x^p)O(x^q) = O(x^{p+q})$$

for $m, n > 0$, $p, q \in \mathbb{Z}$ and $c \in \mathbb{R}$.

Landau Symbols as Equivalence Classes

Essentially, we can regard $O(\phi(x))$ as an equivalence class induced by the equivalence relation

$$f \sim g \qquad f(x) - g(x) = O(\phi(x)) \text{ as } x \to \infty. \tag{1.6.2}$$

For example,

$$\sqrt{x^4 + 1} = x^2 + O\left(\frac{1}{x^2}\right)$$

is the same as saying that $\sqrt{x^4 + 1} \sim x^2$ with respect to the equivalence relation (1.6.2) with $\phi(x) = x^{-2}$.

Landau Symbols using Limits

In many cases, the Landau symbols can be calculated using limits.

1.6.8. Theorem. Let f be a real-valued function defined either

- on an interval (a, ∞) for some $a \in \mathbb{R}$ or
- on a subset $\{n \in \mathbb{N} \colon n \geq n_0,\ n_0 \in \mathbb{N}\}$ of the natural numbers.

If there exists some $C \geq 0$ such that

$$\lim_{x \to \infty} \frac{|f(x)|}{|\phi(x)|} = C,$$

then $f(x) = O(\phi(x))$ as $x \to \infty$.

Landau Symbols using Limits

Proof.

Assume that for some $C \geq 0$

$$\lim_{x \to \infty} \frac{|f(x)|}{|\phi(x)|} = C.$$

Then

$$\forall_{\varepsilon > 0} \exists_{L > 0} \forall_{x \in \mathbb{R}} \; x > L \Rightarrow \left| \frac{|f(x)|}{|\phi(x)|} - C \right| < \varepsilon.$$

Choose $\varepsilon = 1$. Then there exists some $L > 0$ such that $x > L$ implies

$$\big| |f(x)| - C|\phi(x)| \big| < |\phi(x)|,$$

i.e.,

$$|f(x)| \leq (C+1)|\phi(x)|.$$

This is just the definition of $f(x) = O(\phi(x))$ as $x \to x_0$. □

Variants of Landau Symbols

The definition of Landau symbols we have used is the "standard" definition. If $f(x) = O(\phi(x))$, then this means that the values of $f(x)$ are not larger than a constant (independent of x) times the values of $\phi(x)$ when x is large enough. However, $f(x)$ can actually be much smaller than $\phi(x)$, e.g., we could write

$$\frac{1}{x} = O(x^{100})$$

which would be correct, but pretty meaningless. In many practical applications, one wants to express that not only is f not much larger than ϕ, but also it is not much smaller. Physicists have often done this by defining special variants of Landau symbols, which look identical but according to the article in which they occur are defined differently. This can of course lead to some confusion.

To clarify the situation, the computer scientist Donald E. Knuth has introduced common variants using different symbols. To my knowledge, they are, however, not widely used in the physics and mathematics literature.

Variants of Landau Symbols

1.6.9. Definition. Let f be a real-valued function defined either

- on an interval (a, ∞) for some $a \in \mathbb{R}$ or
- on a subset $\{n \in \mathbb{N}\colon n \geq n_0,\ n_0 \in \mathbb{N}\}$ of the natural numbers.

Then we say that

$$f(x) = \Omega(\phi(x)) \qquad \text{as } x \to \infty$$

if there exist constants $C, L \geq 0$ such that

$$|f(x)| \geq C|\phi(x)| \qquad \text{whenever } x > L. \tag{1.6.3}$$

Clearly, $f(x) = \Omega(\phi(x))$ if and only if $\phi = O(f(x))$.

Variants of Landau Symbols

1.6.10. Definition. Let f be a real-valued function defined either

- on an interval (a, ∞) for some $a \in \mathbb{R}$ or
- on a subset $\{n \in \mathbb{N}\colon n \geq n_0,\ n_0 \in \mathbb{N}\}$ of the natural numbers.

Then we say that

$$f(x) = \Theta(\phi(x)) \qquad \text{as } x \to \infty$$

if there exist constants $C, L > 0$ such that

$$\frac{1}{C}|\phi(x)| \leq |f(x)| \leq C|\phi(x)| \qquad \text{whenever } x > L. \tag{1.6.4}$$

Clearly, $f(x) = \Theta(\phi(x))$ if and only if $f(x) = O(\phi(x))$ and $f(x) = \Omega(\phi(x))$.

Variants of Landau Symbols

1.6.11. Example. The sum

$$a_n = \sum_{k=1}^{n} k = \frac{n(n+1)}{2}$$

satisfies $a_n = \Theta(n^2)$, since, for $n \in \mathbb{N}$

$$a_n = \frac{n^2}{2} + \frac{n}{2} \leq 2\frac{n^2}{2} = n^2$$

and

$$a_n = \frac{n^2}{2} + \frac{n}{2} \geq \frac{n^2}{2}.$$

Thus, (1.6.4) is satisfied with $C = 2$ and $L = 0$.

Algorithms

We will now study some elements of algorithms and computer programs. Whether this constitutes an application of mathematics to a non-mathematical subject (like physics) or counts as a branch of math in itself is disputed. In any case, we will make some definitions that may not be completely rigorous, starting with the following.

1.7.1. Definition. An *algorithm* is a finite set of precise instructions for performing a computation of for solving a problem.

This "definition" is more a "dictionary definition" than a mathematical one. Nevertheless, we will be able to work with it.

Algorithms

1.7.2. Example. One possible algorithm for finding the maximum of a finite sequence of integers is as follows:

1. Set the temporary maximum equal to the first integer in the sequence.
2. Compare the next integer in the sequence to the temporary maximum, and if it is large than the temporary maximum, set the temporary maximum equal to this integer.
3. Repeat the previous step if there are more integers in the sequence.
4. Stop when there are no integers left in the sequence. The temporary maximum at this point is the largest integer in the sequence.

Algorithms in Pseudocode

The description of the algorithm in Example 1.7.2 is quite cumbersome. In fact, most algorithms are intended to be realized as programs on computers. There are several programming languages that can implement such algorithms, such as Pascal or C++. It is assumed that you have at least a passing familiarity with such programming languages.

Instead of using a specific language, we will use express algorithms using *pseudocode*. This code is loosely based on Pascal and can easily be modified to work in several languages. The advantage of pseudocode is that it is as succinct as a computer program but general enough to be universally understood. The properties and conventions of our pseudocode are given in Appendix 3 of the textbook. Please read through this appendix thoroughly.

Algorithms in Pseudocode

The algorithm of Example 1.7.2 can be expressed in pseudocode as follows:

1.7.3. Algorithm. Find the maximum element in a finite sequence

procedure: $max(a_1, a_2, \ldots, a_n$: integers)
  $max := a_1$
  for $i = 2$ **to** n
    if $max < a_i$ **then**
      $max := a_i$
  {max is the largest element}

Properties of Algorithms

There are several properties that algorithms generally share. These include:

- *Input.* An algorithm has input values from a specified set;
- *Output.* For given input, the algorithm produces output values from a specified set;
- *Definiteness.* The steps of the algorithm are defined precisely;
- *Correctness.* For each input, the algorithm produces the correct output values;
- *Finiteness.* For given input, the algorithm produces output after a finite number of steps;
- *Effectiveness.* Each step of the algorithm can be performed exactly;
- *Generality.* The algorithm is generally applicable, not just for certain input values.

Properties of Algorithms

1.7.4. Example. Algorithm 1.7.3 has all of these properties:

- *Input.* The input is a finite sequence of integers;
- *Output.* The output is the largest integer of the set;
- *Definiteness.* Only assignments, a finite loop and conditional statements occur; these are well-defined;
- *Finiteness.* The algorithm uses a finite number of steps because it terminates after all integers have been examined;
- *Effectiveness.* Each step of the algorithm can be performed exactly;
- *Generality.* The algorithm is generally applicable, because it can be used to find the maximum of any finite sequence of numbers;
- *Correctness.* The algorithm is correct, because it does return the maximum of the integers in the sequence. The proof of correctness is based on induction and requires a bit more background; we will study this in more detail and return to this example later.

Search Algorithms

Search algorithms are extremely important in many applications. The problem is to find an element x within a tuple $(a_1, \ldots, a_n)$ of elements, if it is present. We consider x to be found if we know which $a_k = x$, i.e., if we know the value of k. The following algorithm is quite simple, comparing each element in order with x and terminating when a match is found.

1.7.5. Algorithm. Linear Search

procedure: *linear search*(x: integer, $a_1, a_2, \ldots, a_n$: distinct integers)
  $i := 1$
  while $i < n$ and $x \neq a_i$
    $i := i + 1$
  if $i \leq n$ **then**
    location $:= i$
  else
    location $:= 0$

Binary Search

If the tuple $(a_1, \ldots, a_n)$ consists of objects in ascending order (e.g., numbers sorted from smallest to largest) then another strategy is to successively split the list in half and only search with the sublists whose elements might contain x.

1.7.6. Example. To search for the number 7 in the list $(2, 3, 7, 8, 10, 11, 13, 16)$ we first split the list into two smaller lists, $(2, 3, 7, 8)$ and $(10, 11, 13, 16)$. Since $7 \not> 8$ we do not search the second list. We next split the first list in two, $(2, 3)$ and $(7, 8)$. Since $7 > 3$, we search the second list, which we split in two, (7) and (8). Since $7 \not> 7$, we now search the first list. Since it contains only element, we make a comparison and find 7 in this list.

If we had been searching for the number 6, we would have completed the same steps, but the final comparison would not have yielded a match.

Binary Search

1.7.7. Algorithm. Binary Search

procedure: *binary search*(x: integer, $a_1, a_2, \ldots, a_n$: increasing integers)
  $i := 1$ {i is left endpoint of search interval}
  $j := n$ {j is right endpoint of search interval}
  while $i < j$
    begin
      $m := \lfloor (i + j)/2 \rfloor$
      if $x > a_m$ **then** $i := m + 1$
      else $j := m$
    end
  if $x = a_i$ **then** *location* $:= i$
  else *location* $:= 0$
  {*location* is the subscript of the term that equals x, or is 0 if x is not found}

Sorting Algorithms

Another very important problem is the sorting of unsorted lists (tuples). We assume here that we are sorting numbers, but if we specify an ordering for any set of objects (e.g., lexicographic ordering for words) the procedures we discuss here are generally applicable.

We first introduce a very simple, but not usually very fast algorithm called the *bubble sort*. It essentially works by pairwise comparisons and transpositions.

1.7.8. Example. To put the list $(3, 2, 1, 4)$ in ascending order, we first compare the first and the second element. Since $3 > 2$, we interchange the elements, yielding $(2, 3, 1, 4)$. Then we compare the second and the third element of this list. Since $3 > 1$, we interchange and get $(2, 1, 3, 4)$. Comparing the third and the fourth elements, we obtain $3 \not> 4$, so we do not interchange any elements. This completes the first pass.

In the second pass, we again compare the first and the second elements and interchange them. The list is now in order.

Bubble Sort

So how many passes are necessary in general? After the first pass, the largest number will always be in the correct position, after the second pass the second-largest number will be in the correct position and so on. Therefore, the bubble sort algorithm will consist of $n-1$ passes. If the list contains n elements, the ith pass can then stop comparing after the $(n-i)$th position. In its simplest form, bubble sort does not check if it is finished after fewer passes, simply performing $n-1$ passes.

1.7.9. Algorithm. Bubble Sort

procedure: *bubblesort*($a_1, a_2, \ldots, a_n$: real numbers with $n \geq 2$)
  for $i := 1$ **to** $n-1$
    for $j := 1$ **to** $n-i$
      if $a_j > a_{j+1}$ **then** interchange a_j and a_{j+1}
  $\{(a_1, \ldots, a_n)$ is in increasing order$\}$

Insertion Sort

Another algorithm is called *insertion sort*. This algorithm looks at the jth element of an n element list and inserts it in the correct position in the first $j-1$ elements.

1.7.10. Algorithm. Insertion Sort

procedure: *insertionsort*$(a_1, a_2, \ldots, a_n$: real numbers with$n \geq 2)$

```
for j := 2 to n
  begin
    i := 1
    while a_j > a_i
      i := i + 1
    m := a_j
    for k := 0 to j − i − 1
      a_{j−k} := a_{j−k−1}
    a_i := m
  end {(a_1, ..., a_n) is in increasing order}
```

Optimization and Greedy Algorithms

One of the main interests in the study of algorithms is to find those that solve problems in the most efficient way possible. Problems seeking an optimal solution are known as *optimization problems*.

In general, a problem is solved in different steps. E.g., the bubble sort algorithm compares two numbers and then interchanges them or leaves them as is. This is one step of the entire algorithm. Of course, taking the optimal course at every step does not automatically mean that the sequence of optimal steps actually leads to an optimal solution. Nevertheless, this is often the case and such algorithms are very important. Algorithms that find an optimal solution at every step are called *greedy algorithms*.

Optimization and Greedy Algorithms

1.7.11. Example. Consider the problem of making change for an amount of less than 1 US$, using the standard american coins: penny (1 cent), nickel (5 cent), dime (10 cent) and quarter (25 cent)

We consider the optimization problem, whereby the change should be made with the minimum number of coins possible. For example, if 27 cent change is to be made, this can be accomplished with three coins (a quarter and two pennies) 27 pennies or various other numbers of coins.

Greedy Change-Making

We can implement a greedy algorithm by making change using the following procedure:

If n is the amount of change required, we take the largest denomination smaller than n, then consider the remaining amount. We again take the largest denomination smaller than this amount and consider the remainder, continuing until the original amount has been realized.

1.7.12. Algorithm. Greedy Change-making

procedure: *change*($c_1, c_2, \ldots, c_r$: values of denominations of coins, where $c_1 > c_2 > \cdots > c_r$; n: a positive integer)
  for $i := 1$ **to** r
    while $n \geq c_i$
      begin
        add a coin with value c_i to the change
        $n := n - c_i$
      end

Greedy Change-Making

Algorithm 1.7.12 is greedy, since at each step it takes the greatest allowable denomination and adds it to the change. However, it is not necessarily optimal, depending on the denominations available. For example, if only pennies, dimes and quarters are available (no nickels) then change for 30 cents is made with one quarter and 5 pennies (6 coins) while it would be optimal to make change using 3 dimes.

It is thus not clear at all that the change-making algorithm is optimal assuming available denominations of pennies, nickels, dimes and quarters. As an illustration of the proof of optimality for algorithms, we will now show that this is in fact the case. We first need a preliminary result.

Greedy Change-Making

1.7.13. Lemma. If n is a positive integer, then n cents in change using quarters, dimes, nickels and pennies using the fewest coins possible has at most two dimes, at most one nickel, at most four pennies and cannot have two dimes and a nickel. The amount of change in dimes, nickels and pennies cannot exceed 24 cents.

Proof.
The proof is a simple argument by contradiction. For example, if the optimal solution had more than two dimes, we could replace them by a quarter and a nickel, yielding a contradiction. The details are left to the reader. □

1.7.14. Theorem. Given coin denominations of pennies, nickels, dimes and quarters, Algorithm 1.7.12 produces change using the fewest coins possible.

Greedy Change-Making

Proof.

Suppose that there exists a number n so that there is a way to make change for n using fewer coins than the greedy algorithm produces. First we note that if q' is the number of quarters in this optimal way and q is the number of quarters in the greedy algorithm, then $q = q'$. This follows, because the greedy algorithm uses the most quarters possible ($q \geq q'$) and by Lemma 1.7.13 $q' \not< q$ (otherwise there would be ≥ 25 cents in dimes, nickels and pennies).

Now both ways of making change must contain the same value of pennies, nickels and dimes, since they contain the same number of quarters. There must be the same number of dimes, since the greedy algorithm uses the greatest number of dimes and in an optimal algorithm there are t most one nickel and four pennies. Similarly, we also have the same number of nickels and then also of pennies. □

Computational Complexity

Implementing an algorithm generally requires resources, both of a physical and a temporal type. That is, if we want to run an algorithm such as Algorithm 1.7.3 for finding the maximum element of a set, we will need computer hardware, in particular memory, as well as the time to run the algorithm. Different algorithms may have different requirements; one algorithm may require less time but more hardware resources than another.

These requirements can be compared by considering the *space complexity* and the *time complexity* of a given algorithm. The memory requirements (space complexity) depend on the data structures used. We will not cover this aspect of computer science in our course, and instead restrict our analysis to the time complexity of algorithms.

Time complexity is generally analyzed by considering the number of operations an algorithm requires for completion instead of actual time. This allows a comparison independent of specific computer characteristics (processor speeds, parallel vs. serial processing etc.)

Time Complexity

1.7.15. Example. Consider the time complexity of Algorithm 1.7.3 for finding the maximum element of a set. At each step, two comparisons are made:

- one comparison to determine whether the end of the list has been reached and
- one comparison of the temporary maximum with the current element.

If the list has n elements, these comparisons are performed $n-1$ times, with an additional comparison to exit the loop. Therefore, a total of $2n-1$ comparisons are made. It is not essential how many computer operations are necessary to implement a single "comparison" – we formulate the time complexity of the algorithm by stating that $\Theta(n)$ operations are necessary for the algorithm.

In this example, we see that the number of operations grows linearly with the size of the list. That means that if the list size is doubled, the number of operations required to determine the maximum also doubles. This is a fairly benign behavior.

Worst-Case Complexity

In our calculation, we have counted the comparisons made to determine whether the end of the list has been reached. These are usually not included in the calculation, since they are $\Theta(n)$ and most algorithms use at least this number of operations anyway. For example, if an algorithm uses $\Theta(n^2)$ operations without comparisons, it will still use $\Theta(n^2) + \Theta(n) = \Theta(n^2)$ operations when taking these comparisons into account.

Example 1.7.15 demonstrates a calculation of *worst-case* complexity. This means that in the given number of operations, the algorithm is guaranteed to complete. Sometimes, it is more realistic to consider *average-case* complexity. This gives a "typical" completion time which can (but doesn't have to be) be much shorter than the "worst-case" time. Average case complexity is usually much more difficult to calculate than worst-case complexity and we will omit examples. Instead, we consider a sorting algorithm.

Worst-Case Complexity

1.7.16. Example. Consider the time complexity of bubble sort, Algorithm 1.7.9. During the ith pass, $n-i$ comparisons are used and $n-1$ passes are made. The total number of operations is then

$$\sum_{i=1}^{n-1}(n-i) = n(n-1) - \frac{n(n-1)}{2} = \frac{n(n-1)}{2} = \Theta(n^2)$$

In this example, we see that the number of operations grows quadratically with the size of the list. That means that if the list size is doubled, the number of operations required to sort the list grows fourfold.

Categorization of Complexity

In general, we distinguish between the following complexities for algorithms (in increasing order of complexity):

Complexity	Terminology
$\Theta(1)$	constant complexity
$\Theta(\log n)$	logarithmic complexity
$\Theta(n)$	linear complexity
$\Theta(n \log n)$	$n \log n$ complexity
$\Theta(n^b),\ b > 0$	polynomial complexity
$\Theta(b^n),\ b > 1$	exponential complexity
$\Theta(n!)$	factorial complexity

Problems where a solution algorithm with polynomial or lower complexity is known to exist are called *tractable*. Those that can not be solved using an algorithm with polynomial (worst-case) complexity are called *intractable*.

Complexity Classes

In practice, however, many intractable problems have a much lower average-case complexity and can be solved in practical applications.

Problems for which a solution algorithm (perhaps with specified complexity) exists are called *solvable*, otherwise they are *unsolvable*. Thus intractable problems are those that are unsolvable in polynomial time.

However, many intractable problems have the property that a solution, once present, can be checked in polynomial time. For example, the factorization problem (finding the prime factors of a number) has this property. Such problems are said to belong to the *class NP*, where NP stands for *nondeterministic polynomial* time. Tractable problems are said to belong to *clqss P*.

Complexity Classes

An important class of problems are the *NP-complete* problems: they are all equivalent in the sense that if a polynomial worst-case complexity algorithm can be found for any one of these problems, all others also become tractable. To this day, it is not known whether such an algorithm exists, and the question "$P = NP$-complete?" is one of the important unanswered problems in the theory of complexity of algorithms.

Recursive and Iterative Algorithms

We can divide algorithms into two main types: *iterative* and *recursive algorithms*. Iterative algorithms solve repeat a key step until a problem is solved. Recursive algorithms solve a problem by reducing it to an instance of the same problem with smaller input.

An iterative algorithm to calculate the value of $n!$ is as follows:

1.7.17. Algorithm. Iterative calculation of $n!$, $n \in \mathbb{N} \setminus \{0\}$

procedure: *factorial*(n: a positive integer)
  $factorial := 1$
  for $i := 1$ **to** n
    $factorial := i \cdot factorial$

Recursive and Iterative Algorithms

An recursive algorithm to calculate the value of $n!$ is as follows:

1.7.18. Algorithm. Recursive calculation of $n!$, $n \in \mathbb{N} \setminus \{0\}$

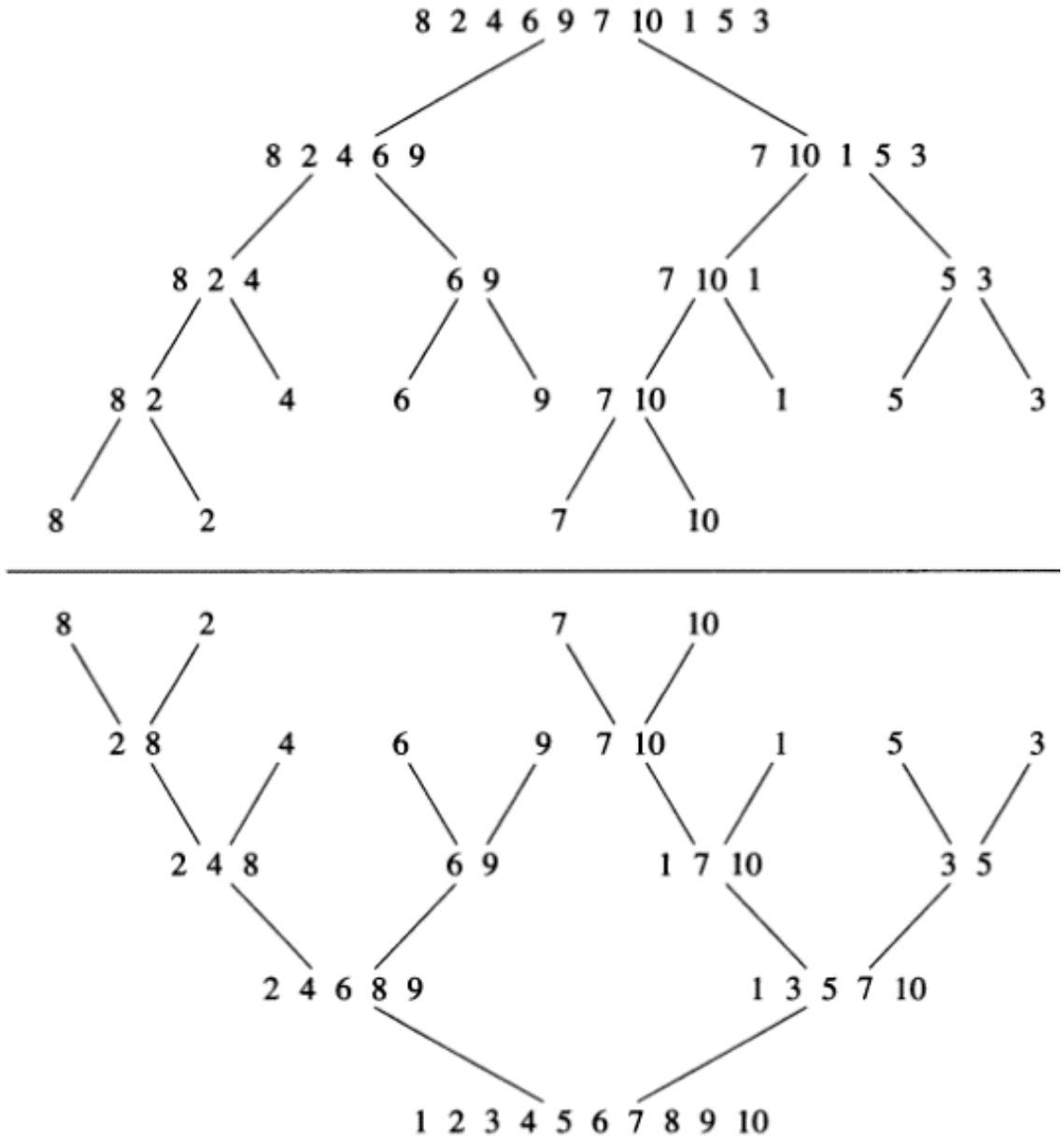
procedure: *factorial*(n: a positive integer)
  if $n = 1$ **then**
    $factorial(n) := 1$
    else $factorial(n) := n \cdot factorial(n-1)$

In general, an algorithm can be expressed much more elegantly and efficiently in recursive form. The disadvantage of recursion is that it may take many more operations to run than an iterative algorithm. On the other hand, if several instances of the algorithm need to be evaluated and the previous instances are stored in memory, a recursive algorithm may be more effective.
For example, if a list of the values of $100!$, $101!$ through $200!$ needs to be calculated, Algorithm 1.7.17 will need far more computations than Algorithm 1.7.18.

Merge Sort

An efficient sorting algorithm that can be formulated recursively is the *merge sort*. It works by successively splitting a list in half until only lists of length 1 remain. These are then successively joined to ordered lists.

We illustrate by showing how the merge sort orders the list 8,2,4,6,9,7,10,1,5,3 at right.



Merge Sort

The recursive pseudocode for merge sort is quite simple; it depends on having an algorithm for merging two ordered lists into a single ordered list:

1.7.19. Algorithm. Recursive Merge Sort

procedure: $mergesort(L = a_1, \ldots, a_n)$
  if $n > 1$ **then**
    begin
    $m := \lfloor n/2 \rfloor$
    $L_1 := a_1, \ldots, a_m$
    $L_2 := a_{m+1}, \ldots, a_n$
    $L := merge(mergesort(L_1), mergesort(L_2))$
    end $\{L$ is now sorted into elements in nondecreasing order$\}$

We will present an algorithm for merging two sorted lists in pseudocode.

Merge Sort

1.7.20. Algorithm. Merging two lists

procedure: *merge*(L_1, L_2: sorted lists)
  $L :=$ empty list
  while L_1 and L_2 are both nonempty
    begin
    remove smaller of first element of L_1 and L_2 from the list it is in
        and put it at the right end of L
    if removal of this element makes one list empty
    then remove all elements from the other list and append to L
    end $\{L$ is the merged list with elements in increasing order$\}$

Merge Sort

1.7.21. Lemma. Two sorted lists with m and n elements, respectively, can be merged into a sorted list using no more than $n + m - 1$ comparisons.

Proof.
The least efficient way for Algorithm 1.7.20 to run is for none of the lists to become empty while the other has more than one element in it. Then $m - 1 + n - 1$ comparisons are made. The final comparison is then made between the remaining two elements. □

We use this result to analyze the efficiency of merge sort. Recall that bubble sort needed $O(n^2)$ operations to sort a list with n elements. We will show that merge sort is rather more efficient:

1.7.22. Theorem. The number of comparisons needed to merge sort a list with n elements is $O(n \log n)$.

This is actually the optimal behavior for a sorting algorithm based on comparisons,as we shall see later.

Merge Sort

Proof of Theorem 1.7.22.

We assume that $n = 2^m$ for simplicity. If this is not the case, the following arguments can be modified and will yield the same result.

At the first stage of the splitting procedure, the list is split into two sublists with 2^{m-1} elements each. This procedure is repeated, and after k splittings, there are 2^k sublists with 2^{m-k} elements each. After m steps, there are 2^m lists of length 1.

Then the 2^m lists are merged into 2^{m-1} lists of length 2, requiring 2^{m-1} comparisons. This procedure continues, so at level k $(k = m, m-1, \ldots, 2, 1)$ 2^k lists each with 2^{m-k} elements are merged into 2^{k-1} lists. By Lemma 1.7.21, each of the 2^{k-1} mergers requires at most $2^{m-k} + 2^{m-k} - 1 = 2^{m-k+1} - 1$ comparisons.

Merge Sort

Proof of Theorem 1.7.22 (continued).

The total number of comparisons is required is then

$$\begin{aligned}\sum_{k=1}^{m} 2^{k-1}(2^{m-k+1}-1) &= \sum_{k=1}^{m} 2^m - \sum_{k=1}^{m} 2^{k-1} \\ &= m2^m - (2^m - 1) \\ &= n\log n - n + 1.\end{aligned}$$

□

Efficiency of Sorting

1.7.23. Theorem. A sorting algorithm based on comparisons of elements needs $O(n \log n)$ comparisons to sort a list of n elements.

Proof.
Consider a list of n distinct numbers (the worst case). There are $n!$ possible ways of arranging such a list and one arrangement will be the sorted list. A comparison sort that completes in $f(n)$ steps will be able to compare at most $2^{f(n)}$ of these arrangements since each comparison has only two possible outcomes. In order to find the sorted arrangement starting from an arbitrary arrangement, all $n!$ arrangements may need to be studied, so

$$2^{f(n)} \geq n!.$$

This immediately gives $f(n) \geq \log n!$. From Stirling's formula it follows that $\log n! = \Omega(n \log n)$, providing the lower bound. □

Program Correctness

Recall that an algorithm is called *correct* if it actually produces the output it was designed to produce. Proving that an algorithm actually is correct is usually quite complex. Proving that a recursive algorithm is correct can be done using (strong) induction.

1.7.24. Example. We show that Algorithm 1.7.18 for recursively calculating $n!$ is correct. First, note that $\mathit{factorial}(1) = 1$, which is equal to $1!$. This is the induction basis. Next, assuming that $\mathit{factorial}(n-1) = (n-1)!$, we will show that $\mathit{factorial}(n) = n!$. In fact,

$$\mathit{factorial}(n) = n \cdot \mathit{factorial}(n-1) = n \cdot (n-1)! = n!,$$

so the algorithm is correct.

In general, proving that a program is correct involves logical inferences. There are several methods of proving the correctness of programs. As an introduction, we will discuss one approach here.

Program Correctness

Showing the correctness of a program involves two steps:

- If the program terminates, the output gives the correct answer (*partial correctness*) and
- The program terminates (*finiteness*).

1.7.25. Definition. A program, or program segment, S is said to be *partially correct with respect to the initial assertion p and the final assertion q* if whenever p is true for the input values of S and S terminates, then q is true for the output values of S.

The notation $p\{S\}q$ (called a *Hoare triple*) indicates that S is partially correct with respect to the initial assertion p and the final assertion q.

1.7.26. Example. The program segment

$y := 2$
$z := x + y$

is correct with respect to the initial assertion p: $x = 1$ and q: $z = 3$.

Composition Rule

Suppose that a program is split into two subprograms S_1 and S_2 carried out one after another. We indicate this by writing $S = S_1; S_2$. Then

$$\big(p\{S_1\}q \wedge q\{S_2\}r\big) \Rightarrow p\{S_1; S_2\}r.$$

This allows us to analyze the correctness of subprograms individually. We now give logical procedures to analyze the correctness of conditional statements and loops.

Suppose a program segment has the form

if *condition* **then**
  S

where S is a block of statements. To analyze the correctness of this segment, we need to check two things:

- If p is true and *condition* is true, then q is true after S terminates;
- If p is true and *condition* is false, then q is true.

Conditional Statements

We can formulate this as

$$\big[\big((p \wedge condition)\{S\}q\big) \wedge \big((p \wedge \neg condition) \Rightarrow q\big)\big] \Rightarrow p\{\textbf{if } condition \textbf{ then } S\}q$$

1.7.27. Example. We want to verify that the program segment

if $x > y$ **then**
  $y := x$

is correct for the initial assertion p: $\mathbf{T}$ (the tautology) and the final assertion q: $y \geq x$.

- If p is true and $x > y$, then S is carried out. After S terminates (if it does) $y = x$, so $y \geq x$.
- If p is true and $x \not> y$, then $y \geq x$ and the final assertion is true.

This establishes the partial correctness of the segment. Since the code block S is just an assignment ($y := x$), it will terminate, so the segment is correct.

Conditional Statements

Similarly, the correctness of

if *condition* **then**
  S_1
else
  S_2

is established through

$$\big[\big((p \wedge \mathit{condition})\{S_1\}q\big) \wedge \big((p \wedge \neg \mathit{condition})\{S_2\}q\big)\big] \Rightarrow p\{\textbf{if } \mathit{condition} \textbf{ then } S_1 \textbf{ else } S_2\}q$$

Loop Invariants

To verify the correctness of the code segment

while *condition*
  S

an assertion that remains true each time that S is executed must be chosen. Such an assertion is called a *loop invariant*. In other words, p is a loop invariant if

$$(p \wedge \mathit{condition})\{S\}p$$

is true. Now suppose that p is a loop invariant. If p is true before the program segment is executed, p and $\neg \mathit{condition}$ are true after termination, if it occurs. Thus,

$$\big((p \wedge \mathit{condition})\{S\}p\big) \quad \Rightarrow \quad p\{\textbf{while} \;\; \mathit{condition} \;\; S\}(\neg \mathit{condition} \wedge p)$$

Before giving an example for illustration, we note that and **for** loop can always be expressed as a **while** loop, so we can apply this method to verify the partial correctness of arbitrary loops.

Loop Invariants

1.7.28. Example. Consider Algorithm 1.7.17 for calculating $n!$ iteratively. Rewriting the **for** loop as a **while** loop, we need to check that

$i := 1$
$factorial := 1$
while $i < n$
  begin
  $i := i + 1$
  $factorial := i \cdot factorial$
  end

terminates with $factorial = n!$. Let p: $(factorial = i! \wedge i \leq n)$. We claim that p is a loop invariant.

Suppose that at the beginning of one execution of the **while** loop, p is true and the loop condition is true. This means that $factorial = i!$ and $i < n$.

Loop Invariants

Then after the loop completes, i will have become $i_{\text{new}} = i + 1$, which satisfies $i \leq n$ and factorial will have become $i_{\text{new}} \cdot \mathit{factorial} = (i+1) \cdot \mathit{factorial} = (i+1)! = i_{\text{new}}!$. Thus, p remains true and we have shown that $(p \wedge i < n)\{S\}p$. Therefore, p is a loop invariant.

Now we discuss the program correctness. The assignments before the **while** loop guarantee that $i = 1 \leq n$ and $\mathit{factorial} = 1 = 1! = i!$ both hold, so p is true. When (and if) the loop terminates, p will be true and $i < n$ will be false, so $i \geq n$. But

$$\begin{aligned} \big(p \wedge (i \geq n)\big) \quad &\Leftrightarrow \quad \big(\mathit{factorial} = i! \wedge i \leq n \wedge i \geq n\big) \\ &\Leftrightarrow \quad \big(\mathit{factorial} = i! \wedge i = n\big) \\ &\Leftrightarrow \quad \mathit{factorial} = n! \end{aligned}$$

as desired. Lastly, the **while** loop will terminate since after $n - 1$ traversals, $i = n$.

First Midterm Exam

This concludes the material that will be covered in the first midterm exam. The material is based on the following textbook chapters:

- Chapter 1;
- Chapter 2;
- Chapter 3, Sections 1-3;
- Chapter 4;
- Chapter 8, Section 5.

Some of the material (such as the construction of the integers) may not appear in this form in the textbook; it is nevertheless part of the exam. The above textbook chapters are for reference only. The lecture slides and Exercise Sets represent the definitive exam material.

For the exam, you may not bring a calculator or any other electronic device, such as a cellphone or electronic dictionary. A monolingual dictionary in the form of a paper book is allowed.

Part II

Applications of Number Theory and Counting

Introduction to Number Theory

Introduction to Counting

Introduction to Discrete Probability

More Counting

Introduction to Number Theory

Introduction to Counting

Introduction to Discrete Probability

More Counting

Division

Recall that we defined $a \mid b$ ("a divides b") for natural numbers a and b to mean the statement "there exists a $c \in \mathbb{N}$ such that $a \cdot c = b$". We can extend this definition to the integers:

2.1.1. Definition. For $a, b \in \mathbb{Z}$ we define

$$a \mid b \qquad :\Leftrightarrow \qquad \exists_{c \in \mathbb{Z}} \; a \cdot c = b.$$

If $a \mid b$, we say a is a *factor* of b and b is a *multiple* of a. If a does not divide b, we write $a \nmid b$.

2.1.2. Theorem. For all $a, b, c \in \mathbb{Z}$ the following statements hold:

1. $a \mid b$ and $a \mid c$ implies $a \mid (b + c)$.
2. $a \mid b$ implies $a \mid (bc)$.
3. $a \mid b$ and $b \mid c$ implies $a \mid c$.

Division

2.1.3. Corollary. $a \mid b$ and $b \mid c$ implies $a \mid (mb + nc)$ for all $a, b, c, m, n \in \mathbb{Z}$.

The following theorem is of course not an algorithm, but we use its traditional designation:

2.1.4. Division Algorithm. Let $a \in \mathbb{Z}$ and $d \in \mathbb{Z}_+ := \mathbb{N} \setminus \{0\}$. then there exist unique $q, r \in \mathbb{Z}$ with $0 \leq r < d$ such that

$$a = dq + r. \tag{2.1.1}$$

Proof.
We first establish the uniqueness: assume that there exist $r, \widetilde{r}$ and $q, \widetilde{q}$ such that

$$dq + r = a = d\widetilde{q} + \widetilde{r}.$$

Then $d(q - \widetilde{q}) = \widetilde{r} - r$, so $d \mid (\widetilde{r} - r)$. However, since $-d \leq \widetilde{r} - r \leq d$, this is only possible if $\widetilde{r} - r = 0$. This means that $dq = d\widetilde{q}$. Since $d \neq 0$, we obtain $q = \widetilde{q}$.

Division

Proof (continued).

To prove the existence, we will apply the well-ordering principle directly. Let $S(a,d) := \{n \in \mathbb{N}\colon n = a - dq, q \in \mathbb{Z}\}$. This set is non-empty, because for $q = -\lceil a/d \rceil$ we obtain an element of the set. By the well-ordering principle, $S(a,d)$ has a least element, which we denote by $r = a - dq_0 \geq 0$. Furthermore, $r < d$, because otherwise we could find a smaller element $a - d(q_0 + 1)$ which would also be in $S(a,d)$. This establishes that there exists a representation $a = r + dq_0$ where $q_0 \in \mathbb{Z}$ and $0 \leq r < d$. □

2.1.5. Definition. In the unique representation (2.1.1), d is called the *divisor*, a is called the *dividend*, q is called the *quotient* and r is called the *remainder*. We write

$$q = a \operatorname{div} d, \qquad r = a \bmod b.$$

Modular Arithmetic

2.1.6. Definition. For $a, b \in \mathbb{Z}$ and $m \in \mathbb{Z}_+$ we say that *a is congruent to b modulo m* if $m \mid (a - b)$. We then write

$$a \equiv b \mod m.$$

The following easy result is left as an exercise:

2.1.7. Theorem. Let $a, b \in \mathbb{Z}$ and $m \in \mathbb{Z}_+$. Then

$$a \equiv b \mod m \quad \Leftrightarrow \quad a \bmod m = b \bmod m$$

It is easily checked that congruence mod m, $m \in \mathbb{Z}_+$, induces an equivalence relation on $\mathbb{Z}$:

$$a \sim b \quad \Leftrightarrow \quad a \equiv b \mod m \tag{2.1.2}$$

is reflexive, symmetric and transitive. For example, if $m = 2$, the relation (2.1.2) induces the partition of Example 1.3.9. The fibers (equivalence classes) of this partition are called *congruence classes*.

Modular Arithmetic

We now have several results that clarify the relation between two congruent numbers.

2.1.8. Theorem. Let $a, b \in \mathbb{Z}$ and $m \in \mathbb{Z}_+$. Then

$$a \equiv b \mod m \quad \Leftrightarrow \quad \exists_{k \in \mathbb{Z}} \; a = b + km.$$

Proof.

$$a \equiv b \mod m \quad \Leftrightarrow \quad m \mid (a - b) \quad \Leftrightarrow \quad \exists_{k \in \mathbb{Z}} \; a - b = km$$

$$\Leftrightarrow \quad \exists_{k \in \mathbb{Z}} \; a = b + km$$

□

2.1.9. Theorem. Let $a, b, c, d \in \mathbb{Z}$ and $m \in \mathbb{Z}_+$. If $a \equiv b \mod m$ and $c \equiv d \mod m$ then

$$a + c \equiv b + d \mod m \quad \text{and} \quad ac \equiv bd \mod m.$$

Modular Arithmetic

Proof.

By Theorem 2.1.8 there exist $s, t \in \mathbb{Z}$ such that $b = a + sm$, $d = c + tm$. Then

$$b + d = a + c + m(s + t) \qquad \Leftrightarrow \qquad a + c \equiv b + d \mod m$$

and

$$bd = ac + m(at + cs + stm) \qquad \Leftrightarrow \qquad ac \equiv bd \mod m. \qquad \square$$

2.1.10. Corollary. Let $a, b \in \mathbb{Z}$ and $m \in \mathbb{Z}_+$. Then

$$(a + b) \bmod m = \big(a \bmod m + b \bmod m\big) \bmod m$$
$$ab \bmod m = \big((a \bmod m)(b \bmod m)\big) \bmod m.$$

Hashing Functions

One application of modular arithmetic lies in simple *hashing functions*. Suppose that records are to be stored on a computer system, e.g., the JI student grade transcripts. Each record is identified using a *key*, e.g., the student ID number. There are far less students than possible key combinations, so it makes sense to transform the keys into a number with fewer digits. (Suppose that this number is to designate the storage location of the record on the computer system.) This number is known as a *hash*. A hash is generated from a key using a hashing function. A simple hashing function can be implemented using modular arithmetic, setting

$$h(k) = k \mod m$$

for some key k. Here m is chosen according to the hash length. Supposing that we have 543 records, we might take $m = 1000$, so that each record is associated to some number between 1 and 1000.

Hashing Functions

Of course, it is always possible that h is not injective, i.e., that for different keys k_1 and k_2 we obtain $h(k_1) = h(k_2)$. This is known as a *hash collision*. In such cases, a work around is usually implemented, e.g., by adding $j \geq 1$ to the hash value to prevent the collision.

2.1.11. Example. Suppose that there are records with keys 064212848, 037149212, 107405723 and we use the hash function k mod 111. Then $h(064212848) = 14$, $h(037149212) = 65$ and $h(107405723) = 14$. To prevent the hash collision, we would then simply set $h(107405723) = 15$, assuming that no other key has this hash value.

Hashing functions are extremely important in encryption algorithms and verification of digital signatures. For example, the source code of a program can be converted into a single (extremely long) number and a hash value generated and publicized. (E.g., the SHA1 hash). When source code is downloaded that purports to be the code of the program, its hash can be checked against the public hash - it will only be identical if no surreptitious changes have been made (unless there is a hash collision).

Pseudorandom Numbers

Another application concerns the generation of *pseudorandom numbers*, i.e., numbers that appear to be random but are actually generated using a mathematical deterministic function. The *linear congruential method* generates a sequence of numbers through

$$x_{n+1} = (ax_n + c) \bmod m,$$

where the starting value x_0 (called the *seed*) is given, a is called the *multiplier*, c the increment and m the modulus. We require

$$0 \leq x_0 < m, \qquad 2 \leq a < m \qquad 0 \leq c < m.$$

2.1.12. Example. Choosing $m = 9$, $a = 7$, $c = 4$, $x_0 = 3$ we obtain the sequence

$$7, 8, 6, 1, 2, 0, 4, 5, 3, 7, 8, 6, 1, 2, 0, 4, 5, 3, \ldots$$

which repeats after 9 terms.

Pseudorandom Numbers

If $c = 0$, we say that the sequence is a *pure multiplicative generator*. Many computers generate pseudorandom number in this way, using $m = 2^{31} - 1$ and $a = 7^5 = 16807$. It can be shown that $2^{31} - 2$ numbers are generated before repetition.

In order to generate a number between 0 and 1, we simply take x_n/m.

Caesar's Cipher

A very simple encryption scheme is based on coding letters of the alphabet into numbers between 1 and 26 and then adding an integer modulus 26. Thus, we set

$$A \mapsto 1, \qquad B \mapsto 2, \qquad \ldots \qquad Z \mapsto 26.$$

Then we add an integer k to these numbers modulo 26 and convert back to letters. This method was used by the Roman emperor Julius Caesar in the first century B.C. and is still used in some newsgroups on the internet, where it is known as *ROT 13* (where $k = 13$).

Cryptology is the science of obscuring messages (or, generally, information). A message that is to be obscured is called the *plain text* while the obscured method is called the *cipher text*. The process of generating the cipher text from the plain text is called *encryption*, while the reverse process is called *decryption*.

Caesar's Cipher

2.1.13. Example. The plain text MEET YOU IN THE PARK is to be encrypted using Caesar's cipher with $k = 3$. We first convert the text into numbers,

$$12\ 4\ 4\ 19 \qquad 24\ 14\ 20\ 8 \qquad 13\ 19\ 7\ 4 \qquad 15\ 0\ 17\ 10$$

Then we add 3 modulo 26 to these numbers,

$$15\ 7\ 7\ 22 \qquad 1\ 17\ 23\ 11 \qquad 16\ 22\ 10\ 7 \qquad 18\ 3\ 20\ 13$$

and translate back into letters, obtaining the cipher text PHHW BRXL QWKH. Decryption is performed by reversing the process.

Note that if $k = 13$, then $x + 13 \bmod 26 = x - 13 \bmod 26$ and decryption and encryption use the same procedure.

Prime Numbers

Although we have previously mentioned the concept of a *prime number*, we give a formal definition here:

2.1.14. Definition. A number $p \in \mathbb{N} \setminus \{0, 1\}$ is called prime if the only positive factors of p are 1 and p. If p is not prime, we say it is *composite*.

The following basic result shows that the prime numbers are the "building blocks of the integers":

2.1.15. Fundamental Theorem of Arithmetic. Every $p \in \mathbb{N} \setminus \{0, 1\}$ is prime or the product of primes. The prime numbers whose product is p are called *prime divisor* of p and they are uniquely determined by p.

The first part of this theorem has already been proven (see Example 1.4.5). We will establish the uniqueness of the decomposition later.

We call the representation of p as a product of non-decreasing primes the *prime factorization* of p.

Prime Numbers

2.1.16. Theorem. If n is a composite integer, then n has a prime divisor less than $\sqrt{n}$.

Proof.
If n is composite, we can write $n = ab$ with $a, b \in \mathbb{N}$, $1 < a < b$ and $b > 1$. Suppose that $a > \sqrt{n}$ and $b > \sqrt{n}$. Then $a \cdot b > n$. Thus either a or b must be less than $\sqrt{n}$. This divisor is either prime or the product of primes. In any case, n has a prime divisor less than $\sqrt{n}$. □

2.1.17. Example. We can quickly see that 101 is prime by checking all primes less than $\sqrt{101}$ for whether they divide 101. Since 101 is not divisible by $2, 3, 5$ or 7 it follows that 101 is prime.

Number of Primes

2.1.18. Theorem. There are infinitely many primes.

Proof.

Assume that there are only a finite number of primes, which we list as $p_1, \ldots, p_n$. Set $Q = p_1 \cdot p_2 \cdots p_n + 1$. By the Fundamental Theorem of Arithmetic, Q is either prime or a product of primes. If Q is prime, we have found a prime not in the list $p_1, \ldots, p_n$ and arrive at a contradiction. If it is not prime, then it has a prime factor. However, none of $p_1, \ldots, p_n$ divides Q, since otherwise it would also divide $Q - p_1 p_2 \ldots p_n = 1$, which is impossible. Therefore, there must be a prime not in the list. □

2.1.19. Prime Number Theorem. Let $\theta(x)$ denote the number of primes less than or equal to $x \in \mathbb{N}$. Then

$$\lim_{x \to \infty} \frac{\theta(x)}{x/\ln x} = 1,$$

where ln denotes the natural logarithm (to base e).

Conjectures about Primes

1. Does there exist a polynomial f with integer coefficients such that $f(n)$ is prime for every $n \in \mathbb{N}$?
2. Are there infinitely many primes p such that $p = n^2 + 1$ for some $n \in \mathbb{N}$?
3. Are there infinitely many twin primes (those differing by 2)?
4. Is every even integer $n > 2$ the sum of two prime numbers?

1. No, as you will prove in the exercises. The polynomial $f(n) = n^2 - n + 41$ gives a prime number for $n \leq 40$, but $f(41)$ is composite.
2. Unknown. There are infinitely many n such that $n^2 + 1$ is prime or the product of two primes.
3. Unknown. The largest known twin primes are $16,869,987,339,975 \cdot 2^{171,960} \pm 1$.
4. Unknown.

Greatest Common Divisors

We will write $\mathbb{N}_+$ for $\mathbb{N} \setminus \{0\}$.

2.1.20. Definition. Let $a, b \in \mathbb{N}_+$. The largest $d \in \mathbb{N}_+$ such that $d \mid a$ and $d \mid b$ is called the *greatest common divisor* of a and b and is denoted by $\gcd(a, b)$.

If $\gcd(a, b) = 1$ we say that a and b are *relatively prime*.

We can calculate $\gcd(a, b)$ through

$$\gcd(a, b) = \max\{d \in \mathbb{N}_+ : (d \mid a) \wedge (d \mid b)\}$$

Since the $d \leq a, b$ the above sets are finite and the maximum exists.

2.1.21. Example.

- $\gcd(24, 36) = \max\{1, 2, 3, 4, 6, 12\} = 12$.
- $\gcd(17, 22) = 1$, so 17 and 22 are relatively prime.

Greatest Common Divisors

We say that the elements of a finite set $\{a_1, \ldots, a_n\} \subset \mathbb{N}_+$ are pairwise relatively prime if $\gcd(a_i, a_j) = 1$ whenever $i \neq j$.

2.1.22. Lemma. Let $a, b \in \mathbb{N}_+$ and $p_1, \ldots, p_n$ be prime numbers such that

$$a = p_1^{a_1} \cdot p_2^{a_2} \cdots p_n^{a_n}, \qquad b = p_1^{b_1} \cdot p_2^{b_2} \cdots p_n^{b_n}$$

with $a_1, \ldots, a_n, b_1, \ldots, b_n \in \mathbb{N}$. Then

$$\gcd(a, b) = p_1^{\min(a_1, b_1)} \cdot p_2^{\min(a_2, b_2)} \cdots p_n^{\min(a_n, b_n)}.$$

Proof.

It is clear that $p_1^{\min(a_1, b_1)} \cdot p_2^{\min(a_2, b_2)} \cdots p_n^{\min(a_n, b_n)}$ divides both a and b because their prime factorizations can be arranged so that they include this number as a common factor. No large number can divide both a and b because no other primes can be added to the product and the exponents can not be increased. We use here that if $d \mid a$, then the number d occurs in the (unique!) prime factorization of a. □

Least Common Multiple

2.1.23. Definition. Let $a, b \in \mathbb{N}_+$. The smallest $d \in \mathbb{N}_+$ such that $a \mid d$ and $b \mid d$ is called the *least common multiple* of a and b and is denoted by $\operatorname{lcm}(a, b)$.

The existence of the least common multiple is guaranteed by the well-ordering principle: The set $\{n \in \mathbb{N} \colon (a \mid n) \wedge (b \mid n)\}$ is non-empty (it contains $ab \in \mathbb{N}$) and therefore has a least element, $\operatorname{lcm}(a, b)$.

Similarly to Lemma 2.1.22 we can prove the following result:

2.1.24. Lemma. Let $a, b \in \mathbb{N}_+$ and $p_1, \ldots, p_n$ be prime numbers such that

$$a = p_1^{a_1} \cdot p_2^{a_2} \cdots p_n^{a_n}, \qquad b = p_1^{b_1} \cdot p_2^{b_2} \cdots p_n^{b_n}$$

with $a_1, \ldots, a_n, b_1, \ldots, b_n \in \mathbb{N}$. Then

$$\operatorname{lcm}(a, b) = p_1^{\max(a_1, b_1)} \cdot p_2^{\max(a_2, b_2)} \cdots p_n^{\max(a_n, b_n)}.$$

Greatest Common Divisor and Least Common Multiple

Combining Lemmas 2.1.22 and 2.1.24 we obtain the following result. We leave the details of the proof to the reader.

2.1.25. Theorem. Let $a, b \in \mathbb{N}_+$. Then

$$a \cdot b = \gcd(a, b) \cdot \operatorname{lcm}(a, b).$$

Representations of Integers

We are interested in representing integers using not only the common notation based on multiples of 10, but alternatively using multiples of other numbers, such as 2, 8 or 16. For example,

$$124 = 1 \cdot 10^2 + 2 \cdot 10^1 + 4 \cdot 10^0$$

can also be written as

$$124 = 1 \cdot 8^2 + 7 \cdot 8^1 + 4 \cdot 8^0.$$

That this is possible for all natural numbers greater than zero is formulated below.

2.1.26. Definition and Theorem. Let $b \in \mathbb{N} \setminus \{0, 1\}$ and $n \in \mathbb{N}_+$. Then there exist unique numbers $a_0, \ldots, a_k < b$ with $a_k \neq 0$ such that

$$n = a_k b^k + a_{k-1} b^{k-1} + \cdots + a_1 b + a_0. \tag{2.1.3}$$

The representation (2.1.3) is called the *base b expansion of n*. The numbers a_k are called the *digits* of n in base b.

Binary and Hexadecimal Expansions

The proof of Theorem 2.1.26 can be performed using induction. We will omit it here. Several bases are in common use:

- $b = 2$ (binary expansion)
- $b = 8$ (octal expansion)
- $b = 10$ (decimal expansion)
- $b = 16$ (hexadecimal expansion)

Analogously to the case $b = 10$ we will give a number in a base b simply by listing its digits in this base. We indicate the base of the expansion by a subscript as follows:

$$124 = (124)_{10} = (174)_8.$$

For the hexadecimal expansion we represent the digits a_j in (2.1.3) through

$$0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.$$

For example, the number $(2AE0B)_{16}$ is given by

$$2 \cdot 16^4 + 10 \cdot 16^3 + 14 \cdot 16^2 + 0 \cdot 16 + 11 = (175627)_{10}.$$

Base Conversion

Different bases for integers are used to store information in computers. For example, a bit string represents a number in binary expansion. A bit string of length 8 (called a *byte*) then stores a number of size between 0 and $2^8 - 1$. One byte can also be represented by two hexadecimal digits.

The following procedure allows the generation of base b representations of numbers:

- Write $n = bq_0 + a_0$ with $0 \leq a_0 < b$, i.e., set

$$q_0 = n \operatorname{div} b, \qquad a_0 = n \mod b.$$

  The number a_0 is the rightmost digit in the base b expansion of n.
- Next, calculate

$$q_1 = q_0 \operatorname{div} b, \qquad a_1 = q_0 \mod b$$

  to obtain the next digit a_1.

Continue this process until $q_k \operatorname{div} b = 0$ for some $k \in \mathbb{N}$.

Base Conversion

2.1.27. Example. To find the octal expansion of $(12345)_{10}$ we first divide by 8 to obtain $12345 = 8 \cdot 1543 + 1$, so

$$\begin{aligned} 1543 &= 12345 \text{ div } 8, & 1 &= 12345 \text{ mod } 8, \\ 192 &= 1543 \text{ div } 8, & 7 &= 1543 \text{ mod } 8, \\ 24 &= 192 \text{ div } 8, & 0 &= 192 \text{ mod } 8, \\ 3 &= 24 \text{ div } 8, & 0 &= 24 \text{ mod } 8, \\ 0 &= 3 \text{ div } 8, & 3 &= 3 \text{ mod } 8. \end{aligned}$$

We hence obtain $(12345)_{10} = (30071)_8$.

Pseudocode for Base Conversion

2.1.28. Algorithm. Constructing Base b expansions

procedure: *base b expansion*(n: positive integer)

$q := n$

$k := 0$

while $q \neq 0$

begin

$a_k := q \bmod b$

$q := \lfloor q/b \rfloor$

$k := k + 1$

end

{The base b expansion of n is $(a_{k-1} \ldots a_1 a_0)_b$}

Base Conversion

The table below gives the numbers 0 through 15 in different base expansions.

Decimal	0	1	2	3	4	5	6	7
Hexadecimal	0	1	2	3	4	5	6	7
Octal	0	1	2	3	4	5	6	7
Binary	0	1	10	11	100	101	110	111
Decimal	8	9	10	11	12	13	14	15
Hexadecimal	8	9	A	B	C	D	E	F
Octal	10	11	12	13	14	15	16	17
Binary	1000	1001	1010	1011	1100	1101	1110	1111

It is very easy to convert between binary and hexadecimal expansions because each hexadecimal digit corresponds to a block of four binary digits (as listed above with leading zeroes omitted). For example,

$$(11\ 1110\ 1011\ 1100)_2 = (\mathrm{A8D})_2.$$

Addition of Integers

In computer science it is very important to have procedures for performing algebraic manipulations with integers in their binary expansions. We first look at addition. The sum $a + b$ of two integers with binary expansions is calculated just as you have learned in the school for the addition of multi-digit numbers:

If $a = (a_{n-1}a_{n-2}\ldots a_0)_2$ and $b = (b_{n-1}b_{n-2}\ldots b_0)_2$ we add these numbers by the right-most digits and carrying as follows:

$$a_0 + b_0 = c_0 \cdot 2 + s_0$$

so that s_0 is the right-most digit of $a + b$, $s_0 = (a + b)_0$. We then add

$$a_1 + b_1 + c_0 = c_1 \cdot 2 + s_1,$$

so that $(a + b)_1 = s_1$. We continue this procedure until we reach the step $a_{n-1} + b_{n-1} + c_{n-2} = c_{n-1}s + s_{n-1}$. Then, $s_n = c_{n-1}$ is the leading digit of the sum of $a + b$.

Pseudocode for the Addition of Integers

2.1.29. Algorithm. Adding Integers in their binary representation

procedure: $add(a, bs$: positive integer$)$
$\{a = (a_{n-1}a_{n-2}\dots a_0)_2$ and $b = (b_{n-1}b_{n-2}\dots b_0)_2\}$
$c := 0$
for $j := 0$ **to** $n-1$
begin
$d := \lfloor (a_j + b_j + c)/2 \rfloor$
$s_1 := a_j + b_j + c - 2d$
$c := d$
end
$s_n := c$ $\{$The binary expansion of the sum is $(s_n s_{n-1}\dots s_1 s_0)_2\}$

2.1.30. Remark. Algorithm 2.1.29 uses $O(n)$ additions of bits.

Pseudocode for the Multiplication of Integers

2.1.31. Algorithm. Multiplying integers in their binary representation

procedure: $add(a, b$: positive integer$)$
$\{a = (a_{n-1}a_{n-2}\dots a_0)_2$ and $b = (b_{n-1}b_{n-2}\dots b_0)_2\}$
for $j := 0$ **to** $n-1$
begin
if $b_j = 1$ **then** $c_j = a$ shifted j places
else $c_j = 0$
end
$\{c_0, \dots, c_{n-1}$ are the partial products$\}$
$p_0 := 1$
for $j := 0$ **to** $n-1$
$p := p + c_j$
$\{p$ is the value of $ab\}$

Efficiency of Multiplication of Integers

2.1.32. Remark. Algorithm 2.1.31 uses $O(n^2)$ additions and shifts of bits. There exist more efficient algorithms for multiplication. One, which we will introduce later in Example 2.4.19 iv), uses $O(n^{\log 3}) = O(n^{1.584})$ operations.

Pseudocode for the Division Algorithm

2.1.33. Algorithm. The Division Algorithm for computing div and mod

procedure: *divisionalgorithm*(a: integer, d: positive integer)
  $q := 0$
  $r := |a|$
  while $r \geq d$
    begin
      $r := r - d$
      $q := q + 1$
    end
  if $a < 0$ and $r > 0$ **then**
    begin
      $r := d - r$
      $q := -(q + 1)$
    end
  $\{q = a \operatorname{div} d$ is the quotient, $r = a \bmod d$ the remainder$\}$

Efficiency of the Division Algorithm

2.1.34. Remark. Algorithm 2.1.33 uses $O(q \log a)$ bit operations. There exist more efficient algorithms that require only $O(\log a \cdot \log d)$ operations.

Modular Exponentiation

In cryptology, it is very important to be able to calculate b^n mod m efficiently, where $n \in \mathbb{N}$, $b \in \mathbb{Z}$ and $m \in \mathbb{N} \setminus \{0, 1\}$. In general, b^n will be very large, so it is not practical to calculate b^n first and then compute the modulus.

We first develop an efficient strategy for multiplication (more efficient than Algorithm 2.1.31) and combine it with finding the remainder. We use the binary expansion of n to compute b^n:

$$b^n = b^{a_{k-1}2^{k-1}+\cdots+a_1\cdot 2+a_0} = \prod_{j=0}^{k-1} b^{a_j 2^j}, \qquad a_0, \ldots, a_{k-1} \in \{0, 1\}.$$

We first find all powers b^{2^j} recursively, by noting that $b^{2^{j+1}} = (b^{2^j})^2$. Next we calculate their values mod m and finally compute their product.

In the following procedure we will assume that a procedure for computing the modulus is available. In practice, this will be a more efficient procedure than Algorithm 2.1.33.

Pseudocode for Modular Exponentiation

2.1.35. Algorithm. Modular Exponentiation

procedure: *modularexponentiation*(b: integer, $n = (a_{k-1} \dots a_0)_2$, m: positive integers)

$x := 1$
$power := b \bmod m$
for $i := 0$ **to** $k - 1$
  begin
    if $a_i = 1$ **then** $x := (x \cdot power) \bmod m$
    $power := (power \cdot power) \bmod m$
  end
{x equals $b^n \bmod m$}

2.1.36. Remark. Algorithm 2.1.35 uses $O((\log m)^2 \log n)$ bit operations to find $b^n \bmod m$.

The Euclidean Algorithm

We now turn to the problem of calculating the greatest common divisor of two integers, $\gcd(a, b)$. Using the method of Lemma 2.1.22 is quite inefficient, because finding the prime factorization of a and b can not be done efficiently. The method we present here is ancient and was already known to the greek mathematician Euclid. It is based on the following fact:

2.1.37. Lemma. Let $a, b, q, r \in \mathbb{Z}$ and $a = bq + r$. Then $\gcd(a, b) = \gcd(b, r)$.

Proof.
We will show that all common divisors of a and b are also common divisors of b and r, and vice-versa.

Suppose that $d \mid a$ and $d \mid b$. Applying Theorem 2.1.2, $d \mid (a - bq)$, so $d \mid r$.

Suppose that $d \mid b$ and $d \mid r$. Then $d \mid (r + bq)$, so $d \mid a$. □

Pseudocode for the Euclidean Algorithm

2.1.38. Algorithm. The Euclidean Algorithm

procedure: $gcd(a, b$: positive integers$)$

$x := a$

$y := b$

while $y \neq 0$

begin

$r := x \bmod y$

$x := y$

$y := r$

end $\{\gcd(a, b)$ is $x\}$

Efficiency of the Euclidean Algorithm

2.1.39. Lamé's Theorem. Let $a, b \in \mathbb{Z}_+$ with $a \geq b$. Then the number of divisions used by the Euclidean Algorithm to find $\gcd(a, b)$ is less than or equal to five times the number of decimal digits in b.

Proof.

Setting $a = r_0$ and $b = r_1$, each step of the algorithm produces an equivalent equation as follows:

$$\begin{aligned}
r_0 &= r_1 q_1 + r_2, & 0 \leq r_2 < r_1,\\
r_1 &= r_2 q_2 + r_3, & 0 \leq r_3 < r_2,\\
&\vdots\\
r_{n-2} &= r_{n-1} q_{n-1} + r_n & 0 \leq r_n < r_{n-1}\\
r_{n-1} &= r_n q_n,
\end{aligned}$$

where $r_n = \gcd(a, b)$. Note that $q_1, \ldots q_{n-1} \geq 1$ and $q_n \geq 2$, since $r_{n-1} > r_n$.

Efficiency of the Euclidean Algorithm

Proof (continued).

If we denote by $(f_n)_{n\in\mathbb{N}}$ the sequence of Fibonacci numbers $(1, 1, 2, 3, 5, \ldots)$ we have

$$\begin{aligned} r_n &\geq 1, & r_n &\geq f_1, \\ r_{n-1} &\geq 2r_n \geq 2, & r_{n-1} &\geq f_2, \\ r_{n-2} &\geq r_{n-1} + r_n, & r_{n-2} &\geq f_2 + f_1 = f_3, \\ &\vdots & & \\ r_2 &\geq r_3 + r_4 & r_2 &\geq f_{n-1} + f_{n-2} = f_n, \\ b = r_1 &\geq r_2 + r_3, & b &\geq f_n + f_{n-1} = f_{n+1}. \end{aligned}$$

It follows that if n steps are needed to complete the Euclidean Algorithm, then $b \geq f_{n+1}$. It can be shown by induction that $f_n \geq \alpha^{n-1}$ for $n > 2$, where $\alpha = (1 + \sqrt{5})/2$.

Efficiency of the Euclidean Algorithm

Proof (continued).

Using $\log_{10} \alpha > 1/5$ we find

$$\log_{10} b > (n-1) \log_{10} \alpha > \frac{n-1}{5}.$$

Hence $n - 1 > \log_{10} b$. Assuming that b has k decimal digits. Then $b < 10^k$ and $n < 5k + 1$. Since n is an integer, this implies $n \leq 5k$.

Lamé's Theorem implies that the Euclidean algorithm uses $O(\log b)$ divisions to find $\gcd(a, b)$ when $a \geq b$.

The GCD as a Linear Combination

It turns out that $\gcd(a, b)$ can be expressed as a linear combination of a and b with integer coefficients:

2.1.40. Theorem. Let $a, b \in \mathbb{Z}_+$. Then there exist $s, t \in \mathbb{Z}$ such that $\gcd(a, b) = sa + tb$.

The proof will be part of the exercises.

2.1.41. Lemma. Let $a, b, c \in \mathbb{Z}_+$. If $\gcd(a, b) = 1$ and $a \mid bc$, then $a \mid c$.

Proof.
By Theorem 2.1.40 there exist $s, t \in \mathbb{Z}$ such that $sa + tb = 1$, so

$$sac = c - tbc. \tag{2.1.4}$$

By Theorem 2.1.2, (2.1.4) implies $a \mid c$. □

Some Useful Results

2.1.42. Corollary. Let $m \in \mathbb{Z}_+$ and $a, b, c \in \mathbb{Z}$. If $ac \equiv bc \mod m$ and $\gcd(c, m) = 1$, then $a \equiv b \mod m$.

Proof.

$$ac \equiv bc \mod m \quad \Rightarrow \quad m \mid c(a-b) \quad \underset{\gcd(c,m)=1}{\Rightarrow} \quad m \mid a-b$$
$$\Rightarrow \quad a \equiv b \mod m. \qquad \square$$

The following result will prepare the proof of the uniqueness of the prime factorization.

2.1.43. Lemma. Let $p \in \mathbb{N} \setminus \{0, 1\}$ be a prime number and $a_1, \ldots, a_n \in \mathbb{Z}$. If $p \mid a_1 a_2 \ldots a_n$ then $p \mid a_i$ for some a_i.

The proof will be part of the exercises.

Uniqueness of the Prime Factorization

Proof of the Fundamental Theorem of Arithmetic 2.1.15.

The existence of the prime factorization has already been proven in Example 1.4.5. We will now prove the uniqueness. Assume there exist primes $p_1, \ldots, p_s$ and $q_1, \ldots, q_t$ such that

$$n = p_1 p_2 \ldots p_s = q_1 q_2 \ldots q_t.$$

From the equality of the product of primes we remove all common primes, giving

$$p_{i_1} \ldots p_{i_u} = q_{j_1} \ldots q_{j_v}$$

where $u, v \in \mathbb{Z}_+$ and no prime occurs on both sides of this equation. By Lemma 2.1.43 it follows that $p_{i_1} \mid q_{j_k}$ for some k. But this is a contradiction. □

Linear Congruences

Let $a, b \in \mathbb{Z}$ and $m \in \mathbb{Z}_+$. A *linear congruence* is an equation

$$ax \equiv b \mod m \tag{2.1.5}$$

for $x \in \mathbb{Z}$. We want to find all x that satisfy (2.1.5). One method involves finding an integer $\widetilde{a}$ such that $a\widetilde{a} \equiv 1 \mod m$. Such an integer is called an *inverse of a modulo m*.

2.1.44. Theorem. Let $a \in \mathbb{Z}_+$ and $m \in \mathbb{N} \setminus \{0, 1\}$. If $\gcd(a, m) = 1$, the inverse of a modulo m exists. This inverse is unique modulo m.

Proof.

By Theorem 2.1.40 there exist $s, t \in \mathbb{Z}$ such that $sa + tm = 1$, in particular, $sa + tm \equiv 1 \mod m$. But then

$$sa \equiv 1 \mod m \tag{2.1.6}$$

and s is the inverse of a modulo m. The proof of the uniqueness of the inverse uses Corollary 2.1.42 and will be part of the exercises. □

Linear Congruences

The proof of Theorem 2.1.44 also yields a method for finding an inverse of a modulo m: find a linear combination of a and m that equals 1. The coefficient of a will be an inverse of a modulo m.

Once we have found the inverse, we can easily solve the congruence (2.1.5) by multiplying both sides with the inverse.

2.1.45. Example. We want to solve the congruence $3x \equiv 4 \mod 7$. Since $\gcd(3, 7) = 1$, we know that the inverse of 3 modulo 7 exists. Now $7 + (-2) \cdot 3 = 1$, so -2 is an inverse of 3 modulo 7. We then multiply both sides of $3x \equiv 4 \mod 7$ with -2, yielding

$$-6x \equiv -8 \mod 7 \qquad \Leftrightarrow \qquad x \equiv 6 \mod 7,$$

where we have used $-6 \equiv 1 \mod 7$ and $-8 \equiv 6 \mod 7$.

The Chinese Remainder Theorem

The Chinese Remainder theorem is based on a problem posed by the Chinese mathematician Sunzi (孫子 / 孙子) and published in his book "孫子算經" (perhaps translated as "Master Sun's Mathematical Manual") in the first or third century AD (dates by: Rosen's book and wikipedia, respectively.)

Sunzi asks:

> *There are certain things whose number is unknown. When divided by 3, the remainder is 2; when divided by 5, the remainder is 3; and when divided by 7, the remainder is 2. What will be the number of things?*

今有物不知其数，三三数之剩二，五五数之剩三，
七七数之剩二，问物几何?

The Chinese Remainder Theorem

Sunzi's problem can be written in modern mathematical notation as a system of congruences,

$$\begin{aligned} x &\equiv 2 \mod 3, \\ x &\equiv 3 \mod 5, \\ x &\equiv 2 \mod 7. \end{aligned}$$

2.1.46. Chinese Remainder Theorem. Let $m_1, \ldots, m_n \in \mathbb{Z}_+$ be pairwise relatively prime and $a_1, \ldots, a_n \in \mathbb{Z}$. Then the system of congruences

$$\begin{aligned} x &\equiv a_1 \mod m_1, \\ x &\equiv a_2 \mod m_2, \\ &\vdots \\ x &\equiv a_n \mod m_n. \end{aligned} \tag{2.1.7}$$

has a unique solution modulo $m = m_1 m_2 \ldots m_n$.

The Chinese Remainder Theorem

Proof.

We will show the existence only. The proof of uniqueness will be part of the exercises. Let

$$M_k := \frac{m}{m_k} = \prod_{i \neq k} m_i.$$

Then $\gcd(m_k, M_k) = 1$ and by Theorem 2.1.44 there exists an inverse y_k to M_k modulo m_k, i.e.,

$$M_k y_k \equiv 1 \mod m_k.$$

Then it is easily verified that

$$x = \sum_{k=1}^{n} a_k M_k y_k$$

is a solution to (2.1.7) (noting that $M_j \equiv 0 \mod m_i$ for $i \neq j$).

The Chinese Remainder Theorem

To find the answer to Sunzi's original question, set $m = m_1 m_2 m_3 = 3 \cdot 5 \cdot 7 = 105$. Then

$$M_1 = m/3 = 35, \qquad M_2 = m/5 = 21, \qquad M_3 = m/7 = 15.$$

We then find inverse of $M_k \mod m_k$,

$$y_1 = 2, \qquad y_2 = 1, \qquad y_3 = 1.$$

The solution to the system is then

$$x = 2 \cdot 35 \cdot 2 + 3 \cdot 21 \cdot 1 + 2 \cdot 15 \cdot 1 = 233 = 23 \bmod 105.$$

Thus 23 is the smallest integer that satisfies Sunzi's problem, but the solution is only unique modulo 105.

Computer Arithmetic with Large Integers

Let $m_1, \ldots, m_n \in \mathbb{N} \setminus \{0, 1\}$ be pairwise relatively prime, $m = \prod_{i=1}^{n} m_i$ and $a \in \mathbb{Z}$ with $0 \leq a < m$. Then it follows from the Chinese Remainder Theorem (see exercises) that a is uniquely determined by the n-tuple

$$(a \bmod m_1, \ldots, a \bmod m_n).$$

2.1.47. Example. Take $m_1 = 3$, $m_2 = 4$. Then all integers less than $3 \cdot 4 = 12$ can be represented using their remainders upon division by 3 and 4 as follows:

n	0	1	2	3	4	5
$(n \bmod 3, n \bmod 4)$	(0,0)	(1,1)	(2,2)	(0,3)	(1,0)	(2,1)
n	6	7	8	9	10	11
$(n \bmod 3, n \bmod 4)$	(0,2)	(1,3)	(2,0)	(0,1)	(1,2)	(2,3)

Computer Arithmetic with Large Integers

We can now perform arithmetic operations on these tuples of integers instead of the original number.

2.1.48. Example. The numbers 95,97,98,99 are pairwise relatively prime. Then we can represent any integer less than $99 \cdot 98 \cdot 97 \cdot 96 = 89\,403\,930$ as a quadruple of the remainders with respect to these numbers.

To calculate the sum $123684 + 413456$ we first represent the summands as quadruples (n mod 99, n mod 98, n mod 97, n mod 95) and then add them:

$$\begin{aligned} & (33, 8, 9, 89) + (32, 92, 42, 16) \\ = {} & (65 \bmod 99, 100 \bmod 98, 51 \bmod 97, 105 \bmod 95) \\ = {} & (65, 2, 51, 10) \end{aligned}$$

To convert the result back to a number, we solve the system of congruences

$$x \equiv 65 \mod 99, \quad x \equiv 2 \mod 98, \quad x \equiv 52 \mod 97, \quad x \equiv 10 \mod 95$$

and find $x = 537\,140$.

Pseudoprimes

According to some sources, ancient Chinese mathematicians thought that a number n was prime if and only if

$$2^{n-1} \equiv 1 \mod n.$$

This is, however, not quite correct. the congruence holds for all primes, but also for some numbers that are not prime. Such numbers are called *pseudoprimes*. The fact that the congruence does hold for all primes is formulated below.

2.1.49. Fermat's Little Theorem. If p is prime and $a \in \mathbb{Z}$ is not divisible by p, then

$$a^{p-1} \equiv 1 \mod p. \tag{2.1.8}$$

Furthermore, for any $a \in \mathbb{Z}$ and p prime

$$a^p \equiv a \mod p.$$

The proof is relegated to the exercises.

The RSA Cryptosystem

The RSA cryptosystem (named after its (public) inventors, Rivest, Shamir and Adleman) is a method to encrypt messages based on modular exponentiation. For the encryption/decryption of messages, two (large) prime numbers p and q are needed. We suppose that the message has the form of an integer M. For example, we can convert letters into characters by setting $A \mapsto 01, B \mapsto 02, \ldots, Z \mapsto 26$ and then simply concatenate all the numbers into a single (extremely large) integer.

We furthermore need a number $e \in \mathbb{Z}_+$ that is relatively prime to $(p-1)(q-1)$. For convenience, we take e as a prime number, then we must only check that $e \nmid (p-1)(q-1)$.

We assume that p and q are relatively prime to M. (This will nearly always be the case in practice, because p and q are extremely large and prime, so that it is very unlikely that the "message integer" M will have p or q as divisors.)

The RSA Cryptosystem

The integer M (the plaintext) is then encrypted to an integer C (the ciphertext) by the formula

$$C = M^e \mod n,$$

where $n = pq$ is the product of p and q.

Now suppose the integer C is given. Then how can M be regained? Suppose that $d \in \mathbb{Z}_+$ is an inverse to e modulo $(p-1)(q-1)$. (d exists by Theorem 2.1.44.) Then

$$\exists_{k \in \mathbb{Z}} \; de = 1 + k(p-1)(q-1)$$

and

$$C^d \equiv (M^e)^d = M^{de} = M^{1+k(p-1)(q-1)} \mod n$$

From this we obtain

$$C^d \equiv M \cdot (M^{p-1})^{k(q-1)} \mod p$$

$$C^d \equiv M \cdot (M^{q-1})^{k(p-1)} \mod q.$$

The RSA Cryptosystem

By Fermat's Little Theorem, $M^{p-1} \equiv 1 \mod p$ and $M^{q-1} \equiv 1 \mod q$. Hence,

$$C^d \equiv M \mod p \qquad \text{and} \qquad C^d \equiv M \mod q.$$

By the Chinese Remainder Theorem,

$$C^d \equiv M \mod n.$$

Thus, to encrypt a message we need to know $n = pq$ and e. To decrypt a message, we need n and d. Therefore, if Bob wants to receive a message from Alice, he tells her n and e. In fact, he can make n and e available to anyone. The pair (n, e) constitutes Bob's *public key*. In order to decode the message, Bob uses his *private key* (n, d). Therefore, anyone with the encryption key (n, e) can send messages to Bob, but only Bob will be able to decipher them. This type of system is known as *asymmetric encryption*.

Security of the RSA Cryptosystem

The security of this procedure lies in the fact that it is extremely difficult to obtain d from knowing just n and e. The straightforward way to obtain d is to first find p and q, then calculate d from $de \equiv 1 \mod (p-1)(q-1)$. However, finding p and q from n is a very time-consuming task. For example, if n is a 400-digit number, it will take computers millions of years or longer to find p and q. Furthermore, it seems that no method of finding d from e exists that does not amount to factorizing n. Therefore, the system is a secure encryption method when nothing is known about M.

In practice, there are some weaknesses in the pure algorithm which need to be compensated. For example, if the ciphertext for a known plaintext is available, then this may be used to attack the ciphertext of an unknown message. Various measures (such as "padding" of messages) are performed by software to eliminate these weaknesses. More information can be found at HTTP://EN.WIKIPEDIA.ORG/WIKI/RSA.

Introduction to Number Theory

Introduction to Counting

Introduction to Discrete Probability

More Counting

Counting and Cardinality

In mathematics, counting refers to finding a bijection of an arbitrary set M to a subset $S \subset \mathbb{N}$ of the natural numbers.

2.2.1. Definition. Let M be a non-empty set. We say that M is *finite* if there exists a number n and bijective map $\varphi\colon M \to \{1, \ldots, n\}$. In that case, we say that M has n elements, writing $\operatorname{card} M = n$.

The empty set $\emptyset$ is also considered finite and has cardinality $\operatorname{card} \emptyset := 0$.

If M is not finite, we say that M is infinite. If M is infinite and there exists a bijection $M \to \mathbb{N}$, we say that M is *countably infinite* and write $\operatorname{card} M = \aleph_0$. If such a bijection does not exist, we say that M is *uncountable*.

If M is finite or countably infinite,w e often ay that M is *countable*.

Counting and Cardinality

2.2.2. Examples.

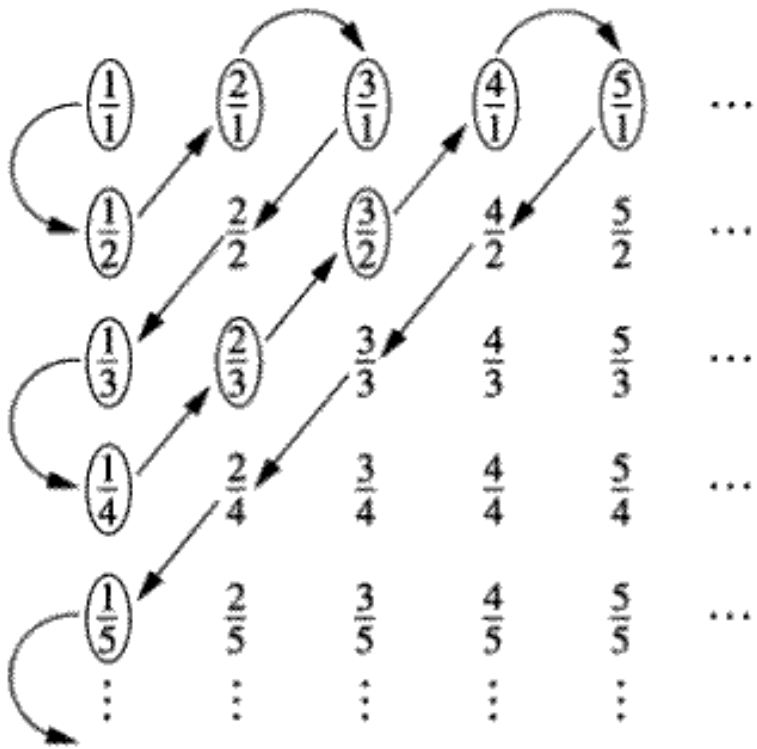
1. The set of bit strings of length 2, $M = \{00, 01, 10, 11\}$ is finite and has cardinality card $M = 4$, because we can define the map $\varphi\colon M \to \{1, 2, 3, 4\}$ by

$$00 \mapsto 1, \qquad 01 \mapsto 2, \qquad 10 \mapsto 3, \qquad 11 \mapsto 4.$$

   It is clear that this map is bijective, so we have counted M. (We say that we have counted M when we have found a bijection φ.)
2. The set of even natural numbers $2\mathbb{N} = \{n \in \mathbb{N}\colon n = 2k,\ k \in \mathbb{N}\}$ is countably infinite; we simply define the bijection $\varphi\colon n \mapsto n/2 + 1$.
3. The set of real numbers between 0 and 1, $M = \{x \in \mathbb{R}\colon 0 \leq x \leq 1\}$ is uncountable. To see this, we use Cantor's diagonalization argument.

Counting and Cardinality

4. The positive rationals are countable:



Cardinality of Power Sets

2.2.3. Theorem. Let M be a finite set with card $M = n$. Then the power set of M, $\mathcal{P}(M)$, has cardinality 2^n.

Proof.
Assume that the elements of M are numbered, writing $M = \{a_1, \ldots, a_n\}$. We associate any element of $\mathcal{P}(M)$ with a bit string of length n as follows: an element $S \in \mathcal{P}(M)$ is a subset $S \subset M$. We associate to S the bit string of length n where the ith bit is 1 if $a_i \in S$ and 0 if $a_i \notin S$. Thus we obtain a bijective map from $\mathcal{P}(M)$ to the bit strings of length n.

Now a bit string of length n is the binary representation of a number between 0 and $2^n - 1$. We associate to each bit string this number plus 1, obtaining a bijection $\mathcal{P}(M) \to \{1, \ldots, 2^n\}$. □

Cardinality of Cartesian Products

2.2.4. Theorem. Let M, N be finite sets with card $M = m$, card $N = n$. Then the cartesian product of M and N, $M \times N$, has cardinality $m \cdot n$.

Proof.
Suppose that $M = \{a_1, \ldots, a_m\}$ and $N = \{b_1, \ldots, b_n\}$. Then we define

$$\varphi\colon M \times N \to \{1, \ldots, m \cdot n\}, \qquad (a_i, b_j) \mapsto (i-1)n + j.$$

The inverse map is

$$\varphi^{-1}\colon \{1, \ldots, m \cdot n\} \to M \times N, \quad x \mapsto \left(a_{1+((x-1)\text{div}n)}, b_{1+((x-1) \mod n)}\right),$$

and it is then easy to see that φ is bijective. □

2.2.5. Remark. Theorem 2.2.4 appears (in a bastardized form) as the "product rule" on page 336 of the textbook.

Cardinality of Cartesian Products

2.2.6. Corollary. Let $M_1, \ldots, M_n$ be a finite number of finite sets with $\operatorname{card} M_k = m_k$. Then the cardinality of the cartesian product is

$$\operatorname{card}(M_1 \times M_2 \times \cdots \times M_n) = \prod_{i=1}^{n} m_i.$$

This follows from Theorem 2.2.4 by induction. We can use Corollary 2.2.6 to obtain an alternate proof for Theorem 2.2.3.

2.2.7. Example. There are 32 microcomputers in a computer center. Each microcomputer has 24 ports. How many ports are there in total?

Each port can be identified by the pair

$$(\text{computer no.}, \text{port no. on computer}).$$

Thus the set of all ports is the cartesian product of the set of all computers with the set of ports on each computer and there are $24 \cdot 32 = 768$ total ports.

Cardinality of Unions of Sets

2.2.8. Theorem. Let M, N be finite disjoint sets with card $M = m$, card $N = n$, $M \cap N = \emptyset$. Then the union of M and N, $M \cup N$, has cardinality $m + n$.

Proof.
Suppose that $M = \{a_1, \ldots, a_m\}$ and $N = \{b_1, \ldots, b_n\}$. Then we define

$$\varphi\colon M \cup N \to \{1, \ldots, m+n\}, \qquad c \mapsto \begin{cases} i & \text{if } c = a_i, \\ m+j & \text{if } c = b_j. \end{cases}$$

The proof of bijectivity is left to the reader. □

2.2.9. Remark. Theorem 2.2.8 is the basis for the "sum rule" on page 338 of the textbook.

Cardinality of Unions of Sets

2.2.10. Corollary. Let $M_1, \ldots, M_n$ be a finite number of disjoint finite sets with card $M_k = m_k$. Then the cardinality of the union is

$$\operatorname{card}\left(\bigcup_{i=1}^{n} M_i\right) = \sum_{i=1}^{n} m_i.$$

2.2.11. Theorem. Let M, N be finite sets with card $M = m$, card $N = n$, $\operatorname{card}(M \cap N) = k$. Then the union of M and N, $M \cup N$, has cardinality $m + n - k$.

Proof.

Suppose that $M = \{a_1, \ldots, a_m\}$ and $N = \{a_1, \ldots, a_k, b_1, \ldots, b_{n-k}\}$, where $k \geq 1$. Then

$$M \cup N = \{a_1, \ldots, a_m, b_1, \ldots, b_{n-k}\}$$

so $\operatorname{card}(M \cup N) = m + n - k$. □

Cardinality of Unions of Sets

Another way of writing Theorem 2.2.11 is

$$\text{card}(M \cup N) = \text{card}\, M + \text{card}\, N - \text{card}(M \cap N). \qquad (2.2.1)$$

The previous theorems are applied in a wide variety of counting problems. We will just give a few examples here; many more can be found in the textbook and the exercises.

Number of Bit Strings with Specified Bits

2.2.12. Example. How many bit strings of length 8 start with a 1 bit or end with the two bits 00?

A bit string of length 8 is an octuple (8-tuple). There is a natural bijection from the set of octuples with first entry equal to 1 to the heptuples (7-tuples):

$$\phi\colon (1, x_1, x_2, \ldots, x_7) \mapsto (x_1, x_2, \ldots, x_7).$$

Since the set of heptuples is $\{0,1\}^7$, by Corollary 2.2.6 the cardinality of the set of bit strings starting with a 1-bit is 2^7. An analogous argument shows that the cardinality of the set of bit strings ending with 00 is 2^6. The problem asks for the union of the these two sets. The cardinality of the set of bit strings both starting with 1 and ending in 00 (the intersection of the previous two sets) is 2^5. Therefore,

$$2^7 + 2^6 - 2^5 = 160$$

bit strings start with a 1 bit or end with the two bits 00.

IPv4

Every computer or other device connected to the internet is assigned an *internet address* or *IP address* (IP stands for "internet protocol"). In the original system still used today, called IPv6, this address consists of 32 bits, or 4 bytes. Each byte (set of 8 bits) can be represented as a decimal number between 0 and 255, and in the *dot-decimal* notation, these numbers are separated by periods. For example, the JI's web server for the SAKAI system has the internet address 202.120.46.185.

The IP address is divided into several parts: the *class* followed (usually) by the *netid* and the *hostid*. The bits in the IP address are numbered 0 through 31.

- For Class A network addresses, the first bit is 0, the next 7 bits give the netid, followed by a 24 bit hostid.
- For Class B network addresses, the first two bits are 10, the next 14 bits give the netid, followed by a 16 bit hostid.
- For Class C network addresses, the first three bits are 110, the next 21 bits give the netid, followed by a 8 bit hostid.

IPv4

- For Class D and E network addresses, the initial bits are 1110 and 11110, respectively, followed by private or multicast addresses. These classes are not used as general internet addresses.

2.2.13. Example. The SAKAI server's address 202.120.46.185 in binary representation is

$$11001010011110000010111010111001.$$

Since the first bits are 110, it is a Class C network address.

There are certain restrictions on the precise addresses permitted:

- 1111111 is not available as a netid for Class A networks.
- No hostids consisting only of 1s or 0s are available on any network.

2.2.14. Example. Under the above restrictions, what is the combined number of Class A, B and C addresses available?

IPv4

First, we calculate the number of Class A addresses. There are $2^7 - 1 = 255$ possible netids, and $2^{24} - 2 = 16\,777\,214$ possible hostids, giving a total of $255 \cdot 16\,777\,214 = 2\,130\,706\,178$ possible Class A addresses. In a similar manner, we calculate that there are $1\,073\,709056$ Class B and $532\,676\,608$ Class C addresses, giving a total of $3\,737\,091\,842$ addresses of Class A, B or C.

This number of addresses has proved too small, and several technical method have been implemented to resolve the problem. The long-term solution is to switch from the IPv4 system to a new 128-bit system called IPv6, which is slowly being implemented. More information can be found at HTTP://EN.WIKIPEDIA.ORG/WIKI/IPV4 and HTTP://EN.WIKIPEDIA.ORG/WIKI/IPV6.

The Pigeonhole Principle

2.2.15. Pigeonhole Principle. Let M be a finite set and $f\colon M \to M$ a map. Then f is surjective if and only if it is injective.

The name "pigeonhole principle" comes from the idea of a set of n pigeons landing in n pigeonholes. Numbering the pigeons $1, \ldots, n$ and the locations $1, \ldots, n$, the association of pigeon to pigeonhole is given by a map

$$f\colon \{1, \ldots, n\} \to \{1, \ldots, n\}.$$

Image source:
http://en.wikipedia.org/wiki/File:TooManyPigeons.jpg.
Used under the license given there

The Pigeonhole Principle states that one pigeon will land in each pigeonhole (f is surjective) if and only if no two pigeons land in the same location (f is injective).

The Pigeonhole Principle

The following theorem is another, equivalent formulation of the pigeonhole principle. We will prove it first, and then show that it implies the pigeonhole principle 2.2.15.

2.2.16. Theorem. Let M, N be finite sets and $f\colon M \to N$ surjective. Then f is injective if and only if $\operatorname{card} M = \operatorname{card} N$.

Proof.

Suppose $M = \{a_1, \ldots, a_m\}$, $\operatorname{card} M = m$ and $\operatorname{card} N = n$. Since f is surjective, it follows that

$$n = \operatorname{card} N = \operatorname{card} \operatorname{ran} f = \operatorname{card}\{f(a_1), \ldots, f(a_m)\} \leq m$$

If f is injective, then the numbers $f(a_1), \ldots, f(a_m)$ are all distinct, and $\operatorname{card}\{f(a_1), \ldots, f(a_m)\} = m$. Thus, $m = n$.

Suppose that $m = n$. Then $\operatorname{card}\{f(a_1), \ldots, f(a_m)\} = m$ so $f(a_1), \ldots, f(a_m)$ are distinct. This implies that f is injective. □

The Pigeonhole Principle

Proof of the Pigeonhole Principle 2.2.15.

Suppose that M is finite and $f\colon M \to M$ is injective. Then $f\colon M \to \operatorname{ran} f$ is surjective and injective. By Theorem 2.2.16, $\operatorname{card}(\operatorname{ran} f) = \operatorname{card} M$. Since $\operatorname{ran} f \subset M$, this implies (why?) that $\operatorname{ran} f = M$. Thus $f\colon M \to M$ is surjective.

Suppose that $f\colon M \to M$ is surjective. Then, by Theorem 2.2.16, f is injective. □

Yet another version of the pigeonhole theorem is the following:

2.2.17. Theorem. Let M, N be finite sets with $\operatorname{card} M > \operatorname{card} N$ and $f\colon M \to N$. Then f is not injective.

The proof is left to the exercises.

The pigeonhole principle seems almost trivial, but it is actually a very deep theorem about the structure of finite sets. It does not hold true for infinite sets, as the injective (but not surjective) map $f\colon \mathbb{N} \to \mathbb{N}$, $n \mapsto n+1$ shows.

Applications of the Pigeonhole Principle

The pigeonhole principle can be used to prove a wide variety of theorems, some of which are quite surprising.

2.2.18. Lemma. Let $n \in \mathbb{Z}_+$. Then there exists a multiple of n such that its decimal expansion contains only the digits 1 and 0.

Proof.
Let $n \in \mathbb{Z}_+$ and consider the set of $n+1$ integers

$$M = \{1,\ 11,\ 111, \ldots, \underbrace{11\ldots1}_{n \text{ times}},\ \underbrace{11\ldots11}_{n+1 \text{ times}}\}.$$

Define the map f: $M \to \{0, \ldots, n-1\}$, $f(x) = x \bmod n$. By Theorem 2.2.17, f is not injective, so there exist two elements in M with the same remainder when divided by n. The larger of these less the smaller is an integer containing only 0s and 1s. Furthermore, it must have remainder 0 when divided by n, i.e., it is a multiple of n. □

Applications of the Pigeonhole Principle

2.2.19. Remark. In many practical applications, the pigeon hole theorem and related statements with respect to counting are formulated as "putting m objects into k boxes." It should be obvious that this is equivalent to finding a function from a set M of card $M = m$ objects to a set N of card $N = n$ boxes.

Given arbitrary sets X, Y and some map $f\colon X \to Y$ it is convenient to define the pre-image of a set $V \subset Y$ as

$$f^{-1}(V) := \{x \in X\colon f(x) \in V\}.$$

If V contains only one element $V = \{v\}$, we often omit the braces, writing $f^{-1}(v)$ instead of $f^{-1}(\{v\})$. For example, defining the function $f\colon \mathbb{R}^2 \to \mathbb{R}$, $f(x, y) = x^2 + y^2$, the set

$$f^{-1}(r^2) = \{(x, y) \in \mathbb{R}^2\colon f(x, y) = r^2\}$$

is a circle of radius r^2 centered at the origin.

The Generalized Pigeonhole Principle

2.2.20. Generalized Pigeonhole Principle. Let M, N be finite sets with cardinalities $\operatorname{card} M = m$ and $\operatorname{card} N = n$, respectively. Then for any function $f\colon M \to N$ there is at least one $n_0 \in N$ such that $\operatorname{card}(f^{-1}(n_0)) \geq \lceil m/n \rceil$.

Proof.
We prove the theorem by contradiction. Suppose that for all $y \in \operatorname{ran} f$

$$\operatorname{card}(f^{-1}(y)) < \lceil m/n \rceil.$$

Then also $\operatorname{card}(f^{-1}(y)) < m/n$, since $\operatorname{card}(f^{-1}(y)) \in \mathbb{N}$. Using $f^{-1}(y) \cap f^{-1}(y') = \emptyset$ for $y \neq y'$,

$$\operatorname{card} M = \operatorname{card}\Big(\bigcup_{y \in \operatorname{ran} f} f^{-1}(y) \Big) = \sum_{y \in \operatorname{ran} f} \operatorname{card}(f^{-1}(y)) < n\frac{m}{n} = m,$$

which is a contradiction. □

Ramsey Theory

We can apply the generalized pigeonhole theorem to a problem that occurs in *Ramsey theory*. In general, Ramsey theory is concerned with the number of elements in a set such that a given property will hold. A particular example is the following problem:

2.2.21. Example. At a party consisting of 6 people, any two people are either friends or enemies. Then there are either three mutual friends or three mutual enemies at the party.

Consider one member of the party, A. The five other members of the party are either friends or enemies of A. By the generalized pigeonhole principle, at least three of them are either friends or enemies of A. Suppose that B, C, D are friends of A. If any two of these three are friends, then they together with A are a group of three mutual friends. If none of B, C, D are friends, then they form a group of three mutual enemies.

If B, C, D are a group of three enemies of A, the proof proceeds in a similar manner.

Ramsey Numbers

For $m, n \in \mathbb{N} \setminus \{0, 1\}$ the *Ramsey number* $R(m, n)$ is defined as the minimum number of people at a party so that there are either m mutual friends or n mutual enemies. By Example 2.2.21 $R(3, 3) \leq 6$. In the exercises you will prove $R(3, 3) > 5$, so $R(3, 3) = 6$.

While a few interesting properties of Ramsey numbers are known, e.g., $R(m, n) = R(n, m)$ and $R(2, n) = n$, only a few Ramsey numbers are actually known. In fact, only nine Ramsey numbers with $3 \leq m \leq n$ have been calculated! For example, $R(4, 4) = 18$. For other numbers, only bounds are known, e.g., $43 \leq R(5, 5) \leq 49$.

Permutations

2.2.22. Definition. Let $\{x_1, \ldots, x_n\}$ be a set of n distinguishable elements ($x_k \neq x_j$ for $j \neq k$). Then an injective map

$$\pi : \{x_1, \ldots, x_n\} \to \{x_1, \ldots, x_n\}$$

is called a *permutation* of these elements.

By the pigeonhole principle, any permutation is automatically bijective.

2.2.23. Remark. A permutation is defined on a set of n distinguishable elements; instead of $\{x_1, \ldots, x_n\}$ we can also simply write $\{1, \ldots, n\}$, replacing the permutation of elements with a permutation of indices.

Recall that a function f is defined as a set of pairs of the form $(x, f(x))$, where x is the independent variable. Thus we could define a permutation π through a set of pairs $\{(1, \pi(1)), \ldots, (n, \pi(n))\}$. In fact, we do represent permutations in this way, but use a different notation, writing

$$\pi = \begin{pmatrix} 1 & 2 & \ldots & n \\ \pi(1) & \pi(2) & \ldots & \pi(n) \end{pmatrix}$$

Permutations in Calculus

2.2.24. Example. Consider the set of 3 objects, $\{1, 2, 3\}$. Then one permutation is given by the identity map,

$$\pi_0 = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix}$$

while other permutations are given by

$$\begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{pmatrix}, \quad \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix}, \quad \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix}, \quad \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{pmatrix}, \quad \begin{pmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \end{pmatrix}.$$

2.2.25. Remark. Instead of writing out the map as above, we often simply give the ordered n-tuple $(\pi(1), \ldots, \pi(n))$. For example, instead of

$$\begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{pmatrix} \qquad \text{we might write} \qquad (3, 1, 2).$$

Permutations in Calculus

2.2.26. Remark. A permutation can be regarded as an *arrangement* of a set; using the notation of Remark 2.2.25, the tuple $(3, 1, 2)$ can be considered on the one hand as the values of a permutation; on the other hand, it represents a specific ordering (or arrangement) of the elements of the set $\{1, 2, 3\}$. In practice, the interpretation as an ordering is used in applications.

2.2.27. Lemma. There are $n!$ permutations π of the set $\{1, \ldots, n\}$.

Proof.
We consider the number of possible values of $\pi(k)$, $k = 1, \ldots, n$. The first value, $\pi(1)$ can be any of the n elements of $\{1, \ldots, n\}$. The second value can be any element of $\{1, \ldots, n\} \setminus \{\pi(1)\}$. Hence there are n possible values of $\pi(1)$, but only $n - 1$ possible values of $\pi(2)$. In general, there are $n - k + 1$ possible values for $\pi(k)$, so in total, there are

$$n \cdot (n-1) \cdots (n-n+1) = n!$$

different choices for the values $(\pi(1), \ldots, \pi(n))$ of a permutation.

Permutations in Combinatorics

In combinatorics, permutations are regarded as *arrangements in a definite order*. Clearly, the ordered tuple $(\pi(1), \ldots, \pi(n))$ is an arrangement of $(1, \ldots, n)$ is a definite order, i.e., the definition based on permutations realizes this goal.

Often one is interested in first selecting $r \leq n$ objects from $\{1, \ldots, n\}$ and then arranging these objects in a definite order. We realize this by the following, more general definition:

2.2.28. Definition. Let $n, r \in \mathbb{N} \setminus \{0\}$ with $r \leq n$. Then an injective map

$$\pi \colon \{1, \ldots, r\} \to \{1, \ldots, n\}$$

is called an *r-permutation* of r elements from $\{1, \ldots, n\}$.

Permutations in Combinatorics

2.2.29. Notation. We again write π as

$$\begin{pmatrix} 1 & 2 & \dots & r \\ \pi(1) & \pi(2) & \dots & \pi(r) \end{pmatrix} \qquad \text{or} \qquad (\pi(1), \dots, \pi(r)),$$

where $\pi(k) \in \{1, \dots, n\}$, $k = 1, \dots, r$.

2.2.30. Example. Let $n = 3$, $r = 2$. Then the following are permutations of two elements from $\{1, 2, 3\}$:

$$(1,2), \qquad (2,1), \qquad (1,3), \qquad (3,1), \qquad (2,3), \qquad (3,2).$$

2.2.31. Theorem. There are

$$n \cdot (n-1) \cdots (n-r+1) = \frac{n!}{(n-r)!}$$

different permutations of r elements from $\{1, \dots, n\}$.

Combinations

A related question arising in combinatorics is the number of **selections** of r elements from a set of n elements, where we do not care about the order. Such selections are called *r-combinations*.

2.2.32. Example. Let $n = 3$, $r = 2$. Then the following are combinations of two elements from $\{1, 2, 3\}$:

$$\{1, 2\}, \qquad \{1, 3\}, \qquad \{2, 3\}.$$

Note that a selection consists of sets (which are unordered) while a permutation consists of ordered tuples.

2.2.33. Definition. A combination of r elements from $\{1, \ldots, n\}$ is subset $A \subset \{1, \ldots, n\}$ with card $A = r$ elements.

Combinations

2.2.34. Theorem. There are

$$\binom{n}{r} = \frac{n!}{r!(n-r)!}$$

combinations of r elements from $\{1, \ldots, n\}$.

Proof.

We first consider permutations of r objects from $\{1, \ldots, n\}$. Every permutation gives us a combination, by identifying

$$(\pi(1), \ldots, \pi(r)) \longrightarrow \{\pi(1), \ldots, \pi(r)\}.$$

Obviously, more than one permutation will give us the same combination. Note that any permutation of $(\pi(1), \ldots, \pi(r))$ will be a permutation of r objects from $\{1, \ldots, n\}$. Furthermore, any permutation of $(\pi(1), \ldots, \pi(r))$ will yield the same combination. For each tuple $(\pi(1), \ldots, \pi(r))$, there are $r!$ permutations of $(\pi(1), \ldots, \pi(r))$, so we need to divide the total number of permutations of r objects from $\{1, \ldots, n\}$ by $r!$. □

Combinations as Permutations of Indistinguishable Objects

A combination of r elements of $\{1, \ldots, n\}$ is similar to a permutation, but without regard to order. Fundamentally, if we do not order the selected objects, we regard them as indistinguishable once they have been selected. (We can not say "Take the fifth selected object," but merely, "Take a selected object." We do not distinguish between the selected objects.)

Therefore, a combination can be regarded as *a permutation of r indistinguishable objects from* $\{1, \ldots, n\}$.

Effectively, this corresponds to a division of $\{1, \ldots, n\}$ into two classes, a subset consisting of r elements and its complement.

Sorting Into Classes

We can hence reformulate Theorem 2.2.34 on combinations with regard to the problem of sorting elements of $\{1, \ldots, n\}$ into two classes (sets) A_1 and A_2 of specified size.

2.2.35. Theorem. Let $\mathcal{N} := \{1, \ldots, n\}$. Then there are

$$\frac{\operatorname{card} \mathcal{N}!}{\operatorname{card} A_1! \operatorname{card} A_2!} = \frac{n!}{n_1! n_2!} = \frac{n!}{r!(n-r)!} = \binom{n}{r}$$

possible ways of dividing $\mathcal{N}$ into two sets A_1 and A_2, where

- $A_1 \subset \mathcal{N}$, $\operatorname{card} A_1 = n_1 = r$,
- $A_2 = \mathcal{N} \setminus A_1$, $\operatorname{card} A_2 = n_2 = n - r$.

Of course, this result can be generalized!

Sorting Into Classes

2.2.36. Theorem. Let $\mathcal{N} := \{1, \ldots, n\}$. Then there are

$$\frac{n!}{n_1! n_2! \ldots n_k!}$$

possible ways of dividing $\mathcal{N}$ into k sets $A_1, \ldots, A_k$, where

- $\mathcal{N} = \bigcup_{i=1}^{k} A_i, \quad \bigcap_{i=1}^{k} A_i = \emptyset$
- $\operatorname{card} A_i = n_i,\ i = 1, \ldots, k.$

2.2.37. Remark. Since we are only interested in dividing a number of elements into classes, and do not distinguish between the elements within each class, this sorting into classes is also called *permutation of indistinguishable objects*.

Sorting Into Classes

Proof of Theorem 2.2.36.

We first count the number of ways of sorting n_1 elements into A_1:

$$\binom{n}{n_1} = \frac{n!}{n_1!(n-n_1)!}.$$

Next we count the ways of sorting n_2 elements of the remaining $n - n_1$ elements into A_2:

$$\binom{n-n_1}{n_2} = \frac{(n-n_1)!}{n_2!(n-n_1-n_2)!}.$$

The total number of ways of sorting n_1 elements into A_1 and n_2 elements into A_2 is then the product

$$\binom{n}{n_1}\binom{n-n_1}{n_2} = \frac{n!}{n_1!(n-n_1)!}\frac{(n-n_1)!}{n_2!(n-n_1-n_2)!} = \frac{n!}{n_1!n_2!(n-n_1-n_2)!}$$

Sorting Into Classes

Proof of Theorem 2.2.36 (continued).

Proceeding in this manner, we find there are

$$\begin{aligned}
\binom{n}{n_1}\prod_{i=1}^{k-1}\binom{n-\sum_{j=1}^{i}n_j}{n_{i+1}} &= \frac{n!}{n_1!(n-n_1)!}\prod_{i=1}^{k-1}\frac{\left(n-\sum_{j=1}^{i}n_j\right)!}{n_{i+1}!\left(n-\sum_{j=1}^{i+1}n_j\right)!} \\
&= \frac{n!}{n_1!(n-n_1)!}\frac{\prod_{i=1}^{k-1}\left(n-\sum_{j=1}^{i}n_j\right)!}{\prod_{i=1}^{k-1}n_{i+1}!\prod_{i=1}^{k-1}\left(n-\sum_{j=1}^{i+1}n_j\right)!} \\
&= \frac{n!}{\prod_{i=1}^{k}n_i!}\frac{\prod_{i=2}^{k-1}\left(n-\sum_{j=1}^{i}n_j\right)!}{\prod_{i=1}^{k-2}\left(n-\sum_{j=1}^{i+1}n_j\right)!} \\
&= \frac{n!}{\prod_{i=1}^{k}n_i!}
\end{aligned}$$

ways of sorting all objects into classes $A_1, \ldots, A_k$, $\operatorname{card} A_i = n_i$. □

Repetition

Recall that r-permutations are arrangements of r objects from a set of $n \geq r$ objects, represented through r-tuples. Similarly, r-combinations are selections of r objects but without regard to order, hence represented as sets of cardinality r. These two problems have one thing in common, which we only mentioned in passing: the r selected/arranged objects are all *distinct*.

What happens if we drop this requirement, i.e., we allow identical objects to appear in our selections/arrangements? This is known as selection/arrangement *with repetition*.

In the case of arrangements with repetition, this is quite easy: in an r-tuple, we now allow any entry to contain an arbitrary element of our set $\{1, \ldots, n\}$, regardless of what is contained in the other entries. Thus, the set of all *r-permutations with repetition* is just $\{1, \ldots, n\}^r$ and has cardinality n^r by Theorem 2.2.4. There are n^r different r-permutations with repetition.

Repetition and Multisets

Dealing with combinations with repetition is a bit more subtle: A 2-combination is, for example, the set $\{a, b\}$, but a 3-combination with repetition might be $\{a, b, b\} = \{a, b\}$. So the "information" that the element b is to be repeated (or selected twice) is lost if we just use set formalism.

A solution is to introduce *multisets*. These are unordered collections of objects (like sets) but also associate to each object a so-called *multiplicity*. For example, the set

$$\{2a, 4b, c\}$$

contains the elements a, b, c with multiplicities $2, 4, 1$, respectively. The multiplicities are generally integers and keep track of "how many times" an element is included in the set. Multisets are related to fuzzy sets (that you encountered in the exercises) and follow a similar calculus. We will define the cardinality of a multiset as the sum of the multiplicities of its elements. Then an *r-combinations with repetition* is simply a multiset of cardinality r, or an r-multiset.

Combinations and Permutations with Repetition

Note that in allowing r-permutations and r-combinations of n elements with repetition, we no longer require $r \leq n$.

2.2.38. Example. Consider the set $S = \{a, b, c\}$. Then

$$(b, a, b, c, a, b, b)$$

is a 7-permutation with repetition of S. The 7-multiset

$$\{2a, 4b, c\}$$

represents the corresponding 7-combination of S.

We have already discussed that there are $n^r = 3^7 = 2187$ possible 7-permutations with repetition. But how many different 7-combinations are there?

Counting Multisets

Let us stay with the special case of Example 2.2.38. It is clear that the number of possible 7-combinations is equal to the number of multisets of cardinality 7. Suppose that we write

$$\{2a, 4b, c\} = \{a, a, b, b, b, b, c\}.$$

Now if we agree to always list the elements of a multiset containing elements of $S = \{a, b, c\}$ in a definite order, we need only write

$$x, x, |, x, x, x, x, |, x$$

to indicate the set $\{2a, 4b, c\}$. By our convention, the first xs will stand for as until a bar $|$ is reached, then the xs will stand for bs etc. We can also leave out the commas. Thus,

$|xxx|xxxx$ stands for the 7-multiset $\{3b, 4c\}$.

Counting Multisets

We see that the multisets are determined entirely by the position of the bars among the xs. In our example, we need 2 bars to indicate the separation between a, b, c and have 7 xs to indicate the elements of the 7-multiset. We can regard this as follows: we have 9 "slots" and each slot is either filled by a bar or an x. Since the xs by themselves can be any element of $S = \{a, b, c\}$, the number of multisets is determined by number of ways to select the two slots that hold the bars. In our example, there are $\binom{9}{2} = 36$ ways to select these slots.

This argument immediately generalizes to yield the following theorem:

2.2.39. Theorem. For $r, n \in \mathbb{Z}_+$ here are

$$\binom{r+n-1}{n-1} = \binom{r+n-1}{r}$$

r-combinations of n elements with repetition. (Or that many r-multisets of n elements.)

Counting Multisets

2.2.40. Example. Consider the equation

$$x_1 + x_2 + x_3 = 11, \qquad x_1, x_2, x_3 \in \mathbb{N}.$$

We are interested in finding the number of solutions to this equation. Note that this equation is equivalent to the following: select a total of 11 items, where x_1 items are of type 1, x_2 are of type 2 and x_3 are of type 3. Thus, each solution corresponds to an 11-multiset of 3 three elements and there are

$$\binom{11+3-1}{11} = 78$$

different solutions.

Lexicographic Ordering

We now know that there are $3! = 6$ permutations of the set $\{1, 2, 3\}$. However, we have not yet discussed a systematic way of obtaining these. In oder to do this, we first need to introduce an ordering of the permutations. We will use the so-called *lexicographic ordering*, which is based on the following definition:

2.2.41. Definition. Let $a = (a_1, a_2, \ldots, a_n)$ and $b = (b_1, b_2, \ldots, b_n)$ be two n-tuples. Then we say that a precedes b, writing $a \prec b$, if either $a_1 < b_1$ or there exists a number $k > 1$ such that $a_i = b_i$ for $1 \leq i \leq k - 1$ and $a_k < b_k$.

2.2.42. Example. The tuple $(2, 3, 4, 1, 5)$ precedes the tuple $(2, 3, 5, 1, 4)$.

Generating Permutations

Given a permutation $a = (a_1, \ldots, a_{n-2}, a_{n-1}, a_n)$ we can ask how to find the "next larger" permutation, i.e., the smallest permutation b with $a \prec b$.

- If $a_{n-1} < a_n$, then $(a_1, \ldots, a_{n-2}, a_n, a_{n-1})$ will succeed a (and there is no smaller successor).
- If $a_{n-1} > a_n$, consider a_{n-2}:
  - If $a_{n-2} < a_{n-1}$, then we obtain a successor as

$$(a_1, \ldots, \min(a_{n-1}, a_n), \min(a_{n-2}, x), \max(a_{n-2}, x))$$

    where $x = \max(a_{n-1}, a_n)$
  - If $a_{n-2} > a_{n-1}$, we consider a_{n-3}.

The procedure we see here is the following:

- Find an index j such that

$$a_j < a_{j+1} \qquad \text{and} \qquad a_{j+1} > a_{j+2} > \cdots > a_n.$$

- Replace a_j by $\min\{a_{j+1}, \ldots, a_n\}$. Then arrange the remaining terms in ascending order in the $j+1$st through nth entry of the permutation.

Generating Permutations

To generate permutations of a set $S = \{1, 2, 3, \ldots, n\}$ we start with the "smallest" permutation,

$$(1, 2, \ldots, n-1, n)$$

and then generate successors by the above procedure until we arrive at the "largest" permutation,

$$(n, n-1, \ldots, 2, 1).$$

2.2.43. Example. The permutations of $(1, 2, 3)$ are (in lexicographic order)

$$(1,2,3), \quad (1,3,2), \quad (2,1,3), \quad (2,3,1), \quad (3,1,2), \quad (3,2,1).$$

Pseudocode for this procedure can be found on page 384 of the textbook.

Generating Combinations

We now investigate how to generate r-combinations from a set of $n \geq r$ elements. We first identify our original n-element set with a tuple, so we consider r-combinations from $S = (a_1, a_2, \ldots, a_n)$. We then need to find a way to encode which elements of S are in the combination C.

One way to do this is to use bit strings of length n with exactly r 1s. Then $a_k \in C$ if and only if the kth element of the bit string is 1.

2.2.44. Example. Let $S = (1, 2, 3, 4)$. Then there are four 3-combinations of S:

$\{1, 2, 3\}$	encoded by	1110,
$\{2, 3, 4\}$	encoded by	0111,
$\{1, 2, 4\}$	encoded by	1101,
$\{1, 3, 4\}$	encoded by	1011.

Since every bit string corresponds to a binary number, we can use the ordering of binary numbers to order combinations. Here, $\{2, 3, 4\}$ is the "first" combination and $\{1, 2, 3\}$ is the "last combination.

Generating Combinations

We can generate successive combinations of $\{1, 2, \ldots, n\}$ by using the following algorithm:

To find the next r-combination after $\{a_1, \ldots, a_r\}$, locate the index i such that $a_k \neq n - r + k$ for $k \leq i$. Then replace a_i with $a_i + 1$ and for all $j = i + 1, \ldots, r$ replace a_j with $a_i + j - i + 1$.

2.2.45. Example. The next 4-combination of the set $\{1, 2, 3, 4, 5, 6\}$ after $\{1, 2, 5, 6\}$ is found as follows:

$$a_1 = 1 \neq n - r + 1 = 3, \quad a_2 = 2 \neq n - r + 2 = 4, \quad a_3 = 5 = n - r + 3 = 5$$

so we see $i = 2$. Then we replace a_2 with 3, a_3 with 4, a_4 with 5 to obtain $\{1, 3, 4, 5\}$. Note that $\{1, 2, 5, 6\}$ corresponds to 110011 and $\{1, 3, 4, 5\}$ corresponds to 101110, which is actually the preceding bit string in lexicographic ordering.

Introduction to Number Theory

Introduction to Counting

Introduction to Discrete Probability

More Counting

Probability - Classical Definition

In the 16^{th} century, the Italian mathematician Cardano, who was a heavy gambler, attempted to use mathematics to describe the outcome of games. He hit upon the following definition, which is really a procedure for calculating probabilities:

2.3.1. Definition. Let A be a random outcome (random event) of an experiment (game) that may proceed in various ways. Assume each of these ways is equally likely. Then the probability of the outcome A, denoted $P[A]$, is

$$P[A] = \frac{\text{number of ways leading to outcome } A}{\text{number of ways the experiment can proceed.}}$$

Classical Probability - Examples

2.3.2. Examples.

1. The experiment consists of flipping a coin. We are interested in the probability of the coin landing heads up. The experiment can proceed in two ways: the coin land heads up or tails up. We assume each event is equally likely, so the classical definition gives

$$P[\text{coin lands heads up}] = \frac{1}{2} = 0.5$$

Classical Probability - Examples

2.3.3. Examples.

2. The experiment consists of rolling two 6-sided dice and summing the results, so the possible outcomes are the numbers $S = 2, 3, \ldots, 12$. We are interested in the outcome $S = 3$. Each die will give results 1, 2, 3, 4, 5 or 6. We assume that each result is equally likely. There are two ways we can get the outcome $S = 3$: either the first die's result is 1 and the second die's result is 2, or the first die gives 2 and the second die gives 1. In total, the experiment can proceed in $6 \times 6 = 36$ different ways. Hence

$$P[3] = \frac{2}{36} = \frac{1}{18} = 0.056$$

Historical Notes

Cardano's work although published, received little attention, and 100 years later, in the middle of the 17^{th} century, the two French mathematicians Fermat and Pascal rediscovered his principles, also by considering games.

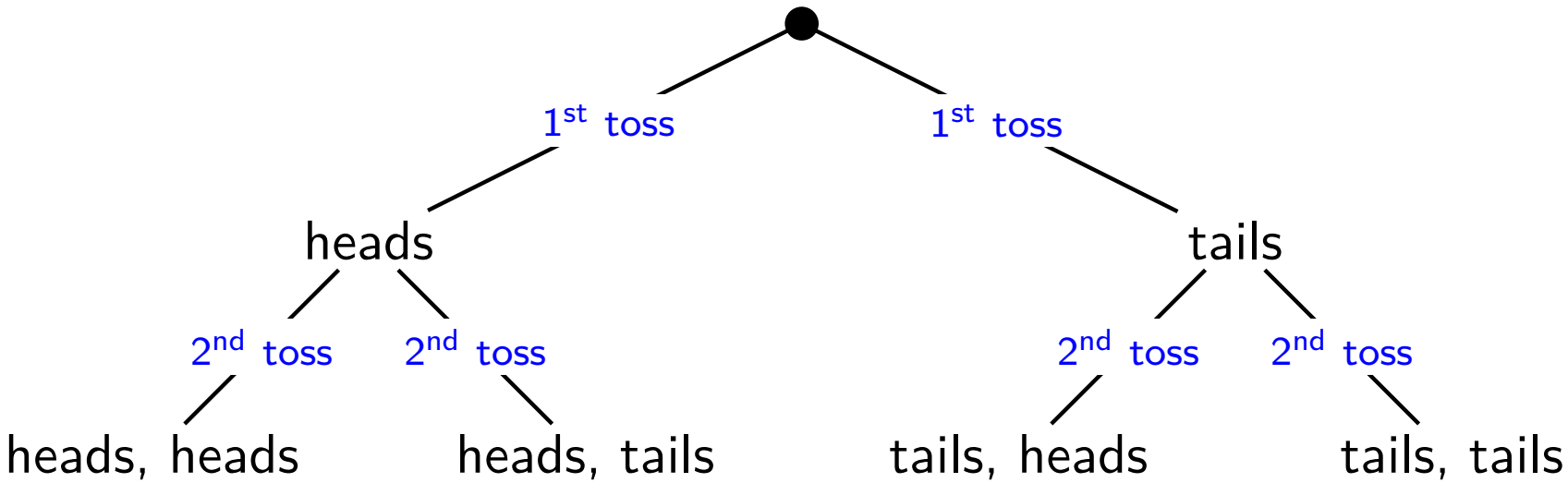
They discussed how to divide the jackpot if a game in progress is interrupted. Imagine that Fermat and Pascal are playing a simple game, whereby a coin is repeatedly tossed. Fermat wins as soon as the coin has turned up heads six times, Pascal wins as soon as the coin has turned up tails six times.

There are 24 gold pieces in the pot. Now the game is interrupted when the coin has already turned up 5 tails and 4 heads. How to divide the pot?

The basic idea is that the pot should be divided proportionally to the probability of each player winning.

Tree Diagrams

Fermat can only win if the coin turns up heads two times in a row. We calculate the probability of this occurring using a tree diagram:



After two tosses, there are four possible outcomes. Only one of the four outcomes involves two heads, so the probability of Fermat winning is 1/4. Therefore, Fermat receives 1/4 of the pot (6 gold pieces), while Pascal receives 3/4 (18 gold pieces).

Sample Spaces and Sample Points

The techniques of counting permutations, combinations and sorting into classes are very useful for evaluating probabilities by the classical definition. First, however, we need to translate physical situations into a mathematical context.

2.3.4. Definition. A *sample space* for an experiment is a set S such that each physical outcome of the experiment corresponds to exactly one element of S.

An element of S is called a sample point.

2.3.5. Remark. Not every element of a sample space needs to correspond to the outcome of an experiment. It is often convenient to select a very large, but simple sample space. For example, for the roll of a six-sided die, we can take $S = \mathbb{N}$, where the result of the die corresponds to the natural numbers 1,2,3,4,5 or 6 and all other natural numbers are not used. All natural numbers in this example are sample points, even though only six actually correspond to physical reality.

Events

2.3.6. Definition. Any subset A of a sample space S is called an *event*.

Two events A_1, A_2 are called *mutually exclusive* if $A_1 \cap A_2 = \emptyset$.

2.3.7. Example. We roll a four-sided die 10 times. Then a possible sample space is $S = \mathbb{N}^{10}$ and a sample point is a 10-tuple, for example $(1, 2, 3, 2, 3, 3, 1, 1, 4, 4) \in \mathbb{N}^{10}$.

This sample point corresponds to first rolling a one, then a 2, next a 3, followed by a 2, a 3, a 3, two ones and two fours.

An event might correspond to "rolling at least two fours" in which case this would be a subset $A \subset S$ such that each 10-tuple in A has at least two entries equal to 4. For example, $(1, 2, 3, 2, 3, 3, 1, 1, 4, 4) \in A$ but $(1, 2, 3, 2, 3, 3, 1, 1, 3, 4) \notin A$.

Image source:
http://commons.wikimedia.org/wiki/File:4-sided_dice_250.jpg.
Used under the license given there

Example

2.3.8. Example. We roll a four-sided die 10 times. What is the probability of obtaining 5 ones, 3 twos, 1 three and 1 four?

There are $4^{10} = 1048576$ possibilities for the 10-tuple of results of the die rolls, corresponding to that many sample points in $S = \mathbb{N}^{10}$ that correspond to physical results. The event A consists of all ordered 10-tuples containing 5 ones, 3 twos, 1 three and 1 four. There are

$$\frac{10!}{5!3!1!1!} = 5040$$

possible ways of obtaining 5 ones, 3 twos, 1 three and 1 four, so there are that many elements in A. The probability is

$$\frac{5040}{1048576} \approx 0.00481 \approx 0.5\%.$$

Classical Probability - Problems

You may have noticed that there are severe problems with the classical definition of probability; it assumes that events are "*equally likely*" - but what does this mean? It means that the probabilities of each event occurring are equal! So the definition of probability already presumes the concept of probability - it is circular.

Such a "definition" is clearly useless!

In the beginning of the 20^{th} century, the Russian mathematician Kolmogorov decided to base probability on a solid mathematical foundation. The classical definition of probability was not only shaky from a logical point of view, it also didn't help much when trying to treat things like throwing a dart at dart board.

He defined probability entirely in the abstract; the physical world is translated into a mathematical set of events and probability is defined as a certain function on this set. This removes all ambiguity from the definition, but does not really tell us what probability *means*.

Axiomatic Definition of Probability

The upshot of Kolmogorov's work for us is that we define probability in the following way:

2.3.9. Definition. Let S be a sample space and $\mathcal{P}(S)$ the power set (set of all subsets) of S. Then a function $P\colon \mathcal{P}(S) \to \mathbb{R}$, $A \mapsto P[A]$ is called a *probability function* (or just *probability*) on S if

1. $P \geq 0$,
2. $P[S] = 1$,
3. For any set of events $\{A_k\} \subset \mathcal{P}(A)$ such that $\bigcap A_k = \emptyset$,

$$P\Big[\bigcup A_k\Big] = \sum P[A_k].$$

Axiomatic Definition of Probability

A subset $A \in \mathcal{P}(S)$ is an event in sample space. We therefore interpret the empty set as the event that "nothing happens" (an experiment has no outcome), the set $A = S$ encompasses the event that "something happens" (the experiment has an outcome) and the event $A^c = S \setminus A$ represents all outcomes that do not include those associated to A. We have

$$P[S] = 1, \qquad P[\emptyset] = 0, \qquad P[A^c] = 1 - P[A],$$

where the first equation is given by definition and the latter two follow easily from the properties of the probability function.

Note that if $A_1, A_2 \in \mathcal{P}(S)$ are events, the event $A_1 \cap A_2$ includes all sample points in their intersection, i.e., all outcomes of the experiment that are common to the outcomes associated to A_1 and A_2. On the other hand, $A_1 \cup A_2$ represents all outcomes associated to either A_1 or A_2. It is not difficult to show that

$$P[A_1 \cup A_2] = P[A_1] + P[A_2] - P[A_1 \cap A_2]. \tag{2.3.1}$$

Axiomatic Definition of Probability

Note that (2.3.1) implies *Boole's inequality*,

$$P[A_1 \cup A_2] \leq P[A_1] + P[A_2], \tag{2.3.2}$$

for all $A_1, A_2 \in \mathcal{P}(S)$.

Another easy property is that if $A_1 \subset A_2$, then $P[A_1] \leq P[A_2]$.

2.3.10. Example. Suppose that two four-sided dice are rolled. We take the sample space

$$S = \{(1,1),\ (1,2),\ (1,3),\ (1,4),\ (2,1),\ \ldots,(4,3),\ (4,4)\},$$

where each tuple has the form (1^{st} result, 2^{nd} result). We then assign the probability function $P[\{(i,j)\}] = 1/16$ for $i,j = 1,2,3,4$. By defining the probability for each one-element subset of S we are fixing the probability for any event $A \subset S$. Our definition has been chosen to coincide with the probability of obtaining each sample point given by Cardano's classical definition if the dice are fair.

Axiomatic Definition of Probability

Let A_1 be the event that corresponds to the outcome "the sum of the two die rolls is at most 3" and A_2 correspond to the outcome "the two die rolls give the same number". Then

$$A_1 = \{(1,1),(1,2),(2,2)\}, \qquad A_2 = \{(1,1),(2,2),(3,3),(4,4)\}.$$

The probability of these events is calculated as

$$P[A_1] = P[\{(1,1)\}] + P[\{(1,2)\}] + P[\{(2,1)\}] = \frac{3}{16},$$

$$P[A_2] = \frac{4}{16} = \frac{1}{4}.$$

Then we can calculate for example that the probability of "the sum of two die rolls is at most three and both rolls are the same" is

$$P[A_1 \cap A_2] = P[\{1,1\}] = \frac{1}{16},$$

the probability of "the two die rolls are distinct" is

$$P[A_2^c] = 1 - P[A_2] = \frac{3}{4}.$$

Conditional Probability

We have seen from this example how the relations derived from the axioms of Definition 2.3.9 allow us to calculate the probability that

- "event A occurs",
- "event A does not occur",
- "events A and B occur" and
- "event A or event B occurs".

The axioms do not, however, provide us with a way to calculate the probability that "event B occurs if event A has occurred." For this, we need to make an additional definition, which we now try to prepare.

Let us denote by $P[B \mid A]$ the probability that "B occurs given that A has occurred". Let us use Example 2.3.10 for illustration. Suppose we are interested in $P[A_1 \mid A_2]$, i.e., the probability that the sum of the die rolls is less than four given that the rolls are equal. In Cardano's terms, 4 out of 16 outcomes lead to event A_2, and only one out these 4 gives rise to an element in $A_1 \cap A_2 = \{(1,1)\}$.

Conditional Probability

If we count the number of sample points ("outcomes") in A_2 that give $A_1 \cap A_2$ to obtain the probability of obtaining A_1 given that A_2 has occurred, we essentially have

$$P[A_1 \mid A_2] = \frac{\text{card}(A_1 \cap A_2)}{\text{card}\, A_2} = \frac{P[A_1 \cap A_2]}{P[A_2]}.$$

This motivates the following definition:

2.3.11. Definition. Let $A, B \subset S$ be events and $P[A] \neq 0$. Then we define the conditional probability

$$P[B \mid A] := \frac{P[A \cap B]}{P[A]}.$$

Independence of Events

If one event does not influence another, then we say that the two events are *independent*. Mathematically, we express this in the following way.

2.3.12. Definition. Let A, B be two events. We say that A and B are independent if

$$P[A \cap B] = P[A]P[B]. \tag{2.3.3}$$

Equation (2.3.3) is equivalent to

$$P[A \mid B] = P[A] \quad \text{if } P[B] \neq 0,$$
$$P[B \mid A] = P[B] \quad \text{if } P[A] \neq 0.$$

Collisions in Hashing Functions

2.3.13. Example. Recall from Slide 236 that a hashing function associates keys (long numbers) to short numbers, e.g., storage locations. A good hashing function produces few collisions, i.e., mappings of different keys to the same number. We wish to find the probability that there are no hash collisions given that there are m storage locations and that the hashing algorithm has probability $1/n$ of mapping to a particular storage location if there are n locations available.

Let n denote the number of keys that have been mapped by the hashing algorithm and p_n the probability that there is no hash collision for these n keys. Assume that $m \geq n$ storage locations are available. Then $p_1 = 1$ and

$$P[\text{no collision for key } 2] = \frac{m-1}{m},$$

since $m-1$ out of m total locations are available for the second key. (Note that a storage location is used for a key whether or not there is a collision.)

Collisions in Hashing Functions

For the kth key the probability is

$$P[\text{no collision for key } k] = \frac{m-k}{m},$$

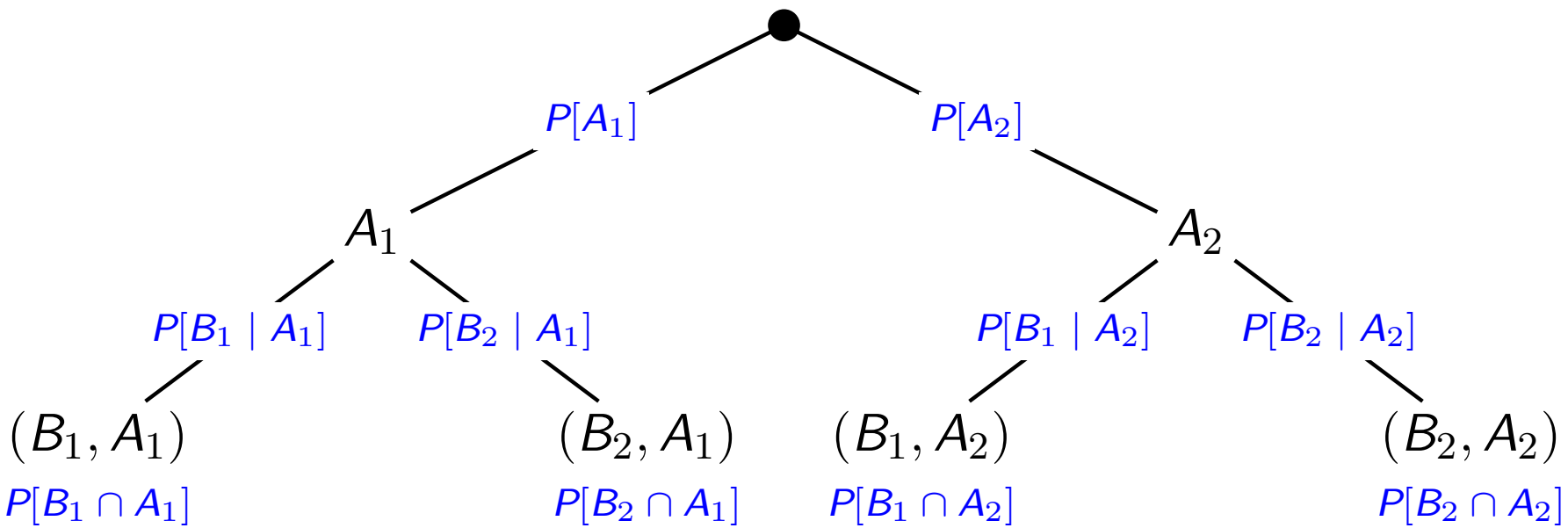
Since the occurrence of a collision is independent of whether a collision has occurred previously, we have

$$\begin{aligned} p_n &= P[\text{no collision for keys } 1, \ldots, n] \\ &= \prod_{k=1}^{n} P[\text{no collision for key } k] \\ &= \frac{m}{m} \cdot \frac{m-1}{m} \cdot \frac{m-2}{m} \cdots \frac{m-n+1}{m} = \frac{m!}{(m-n)!m^n}. \end{aligned}$$

The probability that there is at least one collision is $1 - p_n$. It can be shown that $1 - p_n > 1/2$ if $n \geq 1.177\sqrt{m}$.

Probability Tree Diagrams

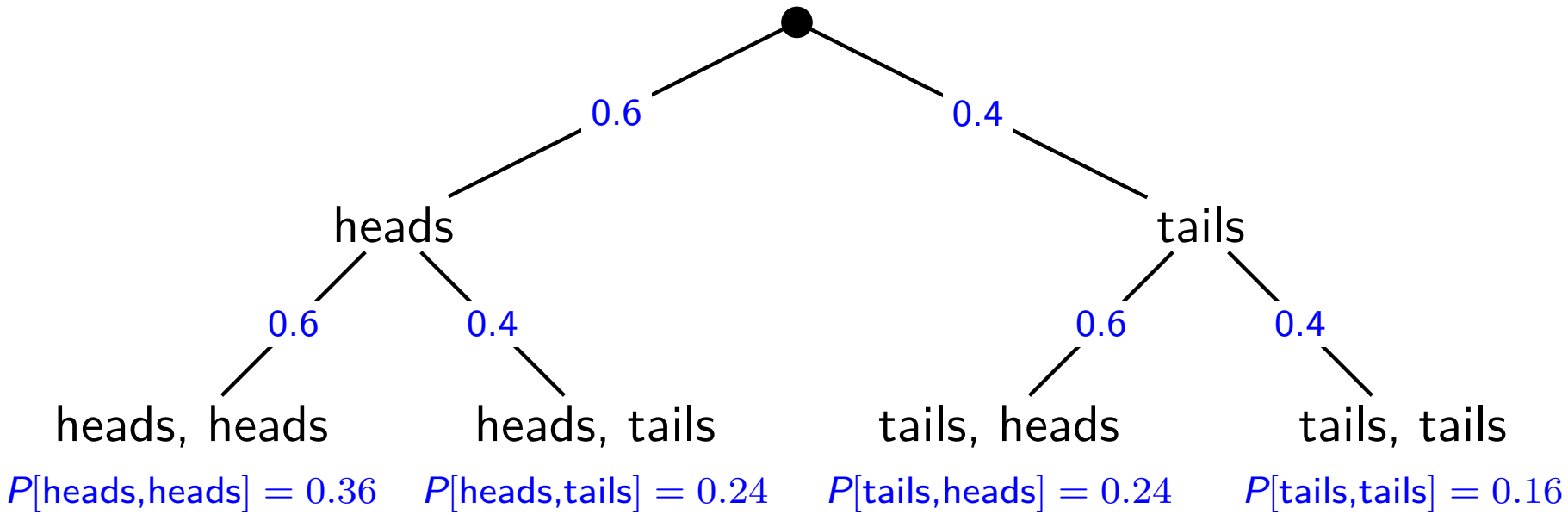
Probability tree diagrams are very well suited to illustrate probabilities. Suppose that we conduct two experiments, where the first experiment has possible outcomes A_1 and A_2, while the second experiment has possible outcomes B_1 and B_2. The the following tree illustrates the experiments and the probabilities associated with the outcomes:



Probability Tree Diagrams

Probabilities are multiplied along each branch to give the final probability, and summed across sub-branches to give unity.

2.3.14. Example. Consider an unfair coin, which turns heads up 60% of the time. We toss the coin twice:



Conditional Probability

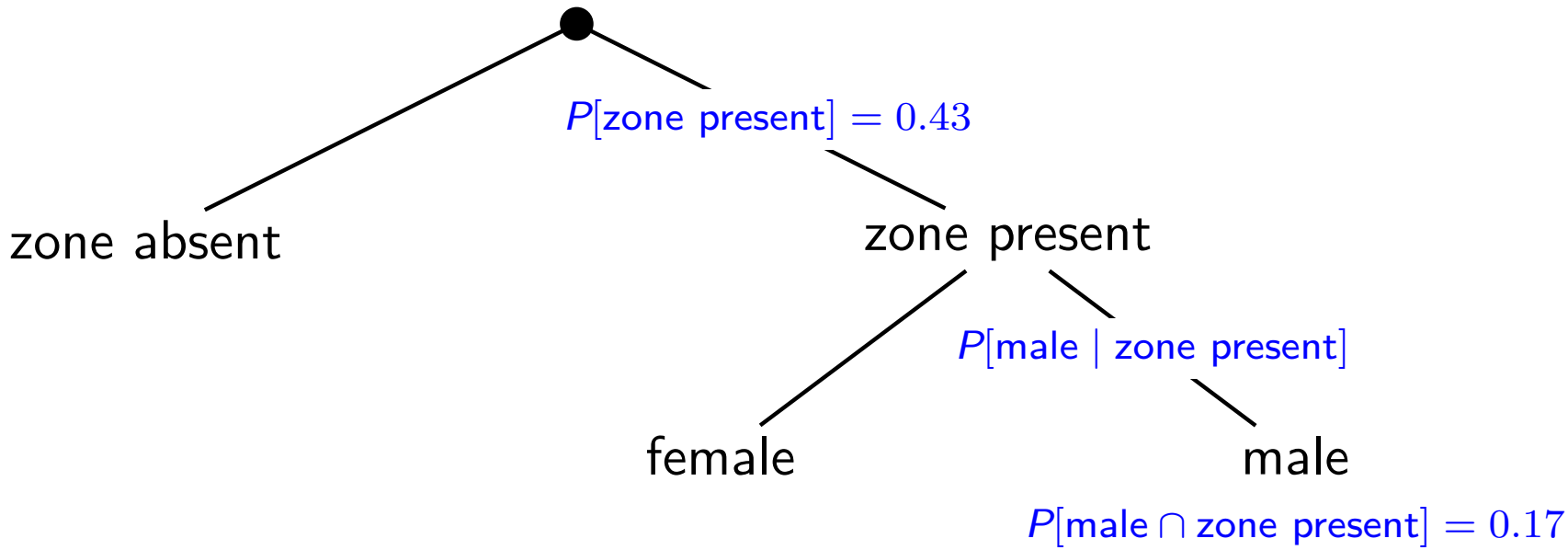
2.3.15. Example. A test for the gender of an unborn child called "starch gel electrophoresis" detects the presence of a protein zone called the pregnancy zone. The following statistical information is known:

- 43% of all pregnant women have the pregnancy zone.
- 51% of all children born are male.
- 17% of all children are male and their mothers have the pregnancy zone.

Given that the zone is present, what is the probability that the child is male?

Conditional Probability

We now use a classical probability tree:



Since we multiply probabilities along branches of probability trees, it is clear that

$$P[\text{male} \mid \text{zone present}] = \frac{P[\text{male} \cap \text{zone present}]}{P[\text{zone present}]} = \frac{0.17}{0.43} \approx 0.40$$

Monte Carlo Algorithms

All the algorithms that we have encountered so far are *deterministic*, i.e., they follow a fixed sequence of steps. Given some input, it is possible to describe precisely what actions the algorithm will take and the output is completely determined.

However, for some applications this procedure is too rigid. A certain randomness in performing one or more steps of the algorithm may be desirable. Such algorithms are called *probabilistic algorithms*.

We will study a specific type of probabilistic algorithm here, called a *Monte Carlo algorithm*. The name stems from the famous Monte Carlo casino in the principality of Monaco, which is located along the mediterranean coast of France.

Image source:
http://commons.wikimedia.org/wiki/File:Monte_Carlo_Casino.jpg.
Used under the license given there

Monte Carlo Algorithms

Monte Carlo algorithms always yield an output, but it is possible that the output is false. The steps involved in these algorithms may be considered to be similar to gambling, whence the name.

A Monte Carlo algorithm performs several iterations, or tests, to determine the truth of some statement. Each test will give either "true" or "unknown". If at least one of the tests yields "true", the algorithm will output "true". If all the tests yield "unknown", then the algorithm ill output "false". The probability that the actual statement is true but a test yields "unknown" instead of "true" is small, and by performing many tests, the probability that the final output is "false" while the statement is actually true can be made very small.

Monte Carlo Algorithms

2.3.16. Example. Consider the problem of determining whether a given number is prime or not. A deterministic algorithm will need $O(2^n)$ steps to solve this problem, which is far too long for practical applications. Instead, a probabilistic algorithm may be used, based on *Miller's test*.

Let $n \in \mathbb{N}$ such that there exists $s \in \mathbb{N}$ and an odd integer t such that $n - 1 = 2^s t$. Then n is said to pass *Miller's test to base b* if either

$$b^t \equiv 1 \mod n \qquad \text{or} \qquad b^{2^j t} \equiv -1 \mod n$$

for some j with $0 \leq j \leq s - 1$.

- If n is prime and $1 < b < n$, then n passes Miller's test to base b.
- If n is composite, then there are fewer than $n/4$ bases b with $1 < b < n$ such that n passes Miller's test to base b.

Based on this, a Monte Carlo algorithm to determine whether n is composite will perform Miller's test for k randomly selected bases b.

Monte Carlo Algorithms

If n fails Miller's test for any of the bases used, the algorithm will return "true" (n is composite). However, if n passes each Miller's test, the answer is still unknown. Nevertheless, the algorithm will return "false" (n is prime) in this case. However, the probability that n is composite and still passes Miller's test each of the k times is

$$p_k = \frac{1}{4^k}.$$

If $k = 30$ tests are performed, $p_k < 10^{-18}$. Then, it is almost certain that a number that the algorithm states is prime actually is so.

The Probabilistic Method

The probabilistic method is a technique for showing the existence of objects with specified properties. It is non-constructive, i.e., it does not give information on how to find/construct these objects but only establishes that they exist.

The method is based on assigning probabilities to all elements of a set S. An object with specified properties exists in S if its probability is greater than zero.

2.3.17. Probabilistic Method. If the probability that an element of a set S does not have a specified property is less than 1, then there exists an element in S with this property.

As a demonstration, we will prove the following statement:

2.3.18. Example. Let $k \in \mathbb{N} \setminus \{0, 1\}$. Then $R(k, k) \geq 2^{k/2}$, where R denotes the Ramsey number introduced on Slide 306.

The Probabilistic Method

It is clear that this statement is true for $k = 2$ and $k = 3$, since $R(2, 2) = 2 = 2^{2/2}$ and $R(3, 3) = 6 > 2^{3/2}$. We will use the probabilistic method to show that if a party has $n < 2^{k/2}$ members, it is possible that no k members are mutual friends or enemies.

Suppose that a party of n people is assembled randomly in such a way that it is equally likely for any two people at the party to be friends or enemies. There are $\binom{n}{k}$ different sets of k people at this party, which we denote by $S_1, S_2, \ldots, S_{\binom{n}{k}}$. Let E_i be the event that all k members of S_i are either mutual friends or enemies. Then the probability that there are either k mutual friends or enemies in S is

$$P[\text{there exist } k \text{ mutual friends or enemies}] = P\left[\bigcup_{i=1}^{\binom{n}{k}} E_i\right].$$

The Probabilistic Method

There are $\binom{k}{2} = k(k-1)/2$ pairs of people in each S_i, and the probability that they are enemies is $1/2$. Thus, the probability that all members if S_i are enemies is $2^{-k(k-1)/2}$. The probability that they are all friends is also $2^{-k(k-1)/2}$, so

$$P[E_i] = 2 \cdot 2^{-k(k-1)/2}.$$

Using Boole's inequality (2.3.2) and $\binom{n}{k} \leq n^k/2^{k-1}$ (see assignments),

$$P\Big[\bigcup_{i=1}^{\binom{n}{k}} E_i\Big] \leq \sum_{i=1}^{\binom{n}{k}} P[E_i] \leq 2^{1-k(k-1)/2} \frac{n^k}{2^{k-1}} = 2^{2-k/2} \frac{n^k}{2^{k^2/2}}.$$

Now if $n < 2^{k/2}$ and $k \geq 4$,

$$P\Big[\bigcup_{i=1}^{\binom{n}{k}} E_i\Big] < 1.$$

The Probabilistic Method

We have therefore shown that for a party whose members are randomly assigned friendship or enmity with each other,

$$P[\text{there exist } k \text{ mutual friends or enemies in a party of } n \text{ members}] < 1.$$

However, by Cardano's definition of probability,

$$= \frac{P[\text{there exist } k \text{ mutual friends or enemies in a party of } n \text{ members}]}{\text{number of parties containing groups of } k \text{ mutual friends/enemies}}{\text{number of all possible parties with } n \text{ members}}$$

Since the quotient is less than 1 if $n < 2^{k/2}$ and $k \geq 4$ it follows that there must exist some party of n members not containing a group of k mutual friends or enemies and we see that $R(k, k) \geq 2^{k/2}$.

Introduction to Number Theory

Introduction to Counting

Introduction to Discrete Probability

More Counting

Recurrence Relations

We will now return to counting problems. One type of problem of significant interest concerns *recurrence relations*.

2.4.1. Definition. A recurrence relation for a sequence (a_n) is an equation of the form

$$a_n = f(a_0, \ldots, a_{n-1}), \qquad \text{for } n \geq k \tag{2.4.1}$$

for some $k \in \mathbb{N}$. A sequence (a_n) is called a solution to the recurrence relation if the terms of the sequence satisfy (2.4.1). Numbers $a_0, \ldots, a_{k-1}$ that are given together with (2.4.1) are called *initial conditions* for the recurrence relation.

We have used recurrence relations before to define sequences, cf. Example 1.4.6. We will study this and other examples more closely.

Fibonacci Numbers

2.4.2. Example. Fibonacci originally posed the following problem in his book *Liber Abbaci* (Book of Calculations) in 1202:

> *A certain man put a pair of rabbits in a place surrounded on all sides by a wall. How many pairs of rabbits can be produced from that pair in a year if it is supposed that every month each pair begets a new pair, which from the second month on becomes productive?*

The solution to this problem leads to the sequence of Fibonacci numbers: In month 1, there is $f_0 = 1$ pair, in month 2 there is $f_1 = 1$ pair, but in month 3 there will be $f_3 = 2$ pairs, of which one is productive. Now in month 4 the productive pair produces another pair, so there are $f_4 = 3$ pairs. In general, every month the number of pairs of rabbits is equal to the number of pairs of the previous month, plus the number of pairs of two months ago, which have since become productive. Hence,

$$f_n = f_{n-1} + f_{n-2}, \qquad n \geq 2, \qquad f_0 = 0,\ f_1 = 1.$$

Tower of Hanoi

2.4.3. Example. The tower of Hanoi is a famous problem: n concentric disks of increasing radius are placed upon one of three pegs. The goal is to move the stack of disks from Peg 1 to Peg 2, but in such a way that

- only one disk is moved at a time and
- a larger disk is never placed on top of a smaller disk.

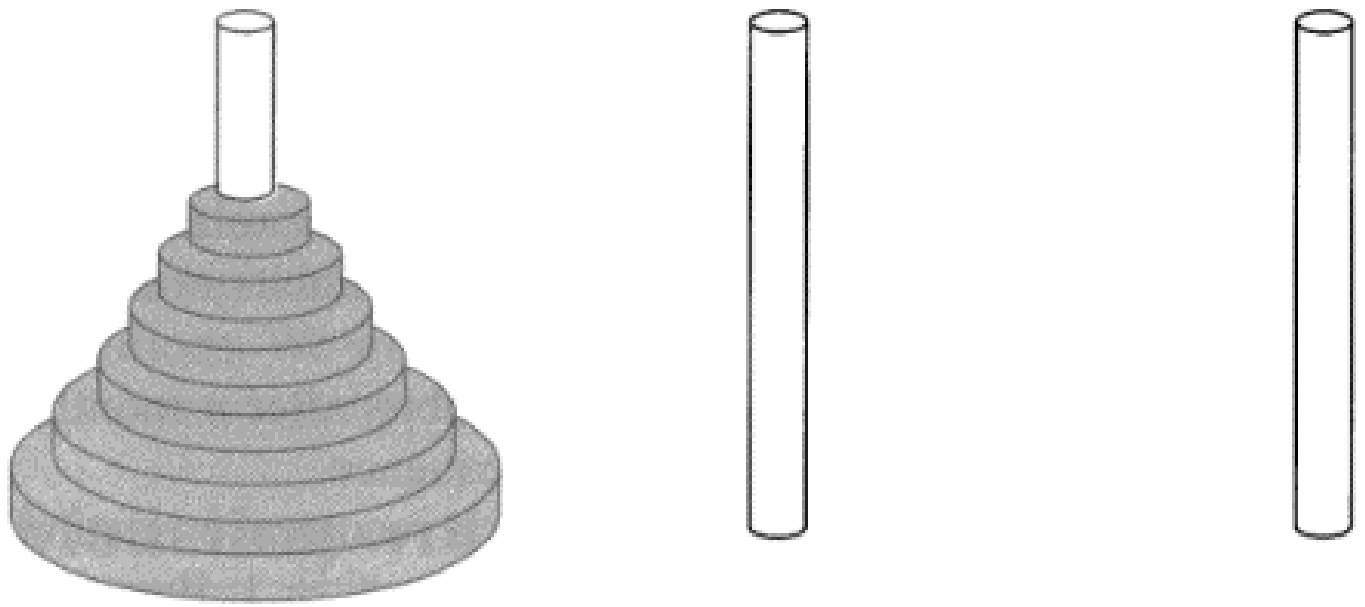


Image source: Rosen's textbook, page 452. Fair use copyright exemption applies.

Tower of Hanoi

Our goal is to find a recurrence relation for the number H_n of moves required to move a stack of n disks form Peg 1 to Peg 2. Suppose we need H_{n-1} moves to transfer the stack of $n-1$ disks to Peg 3:

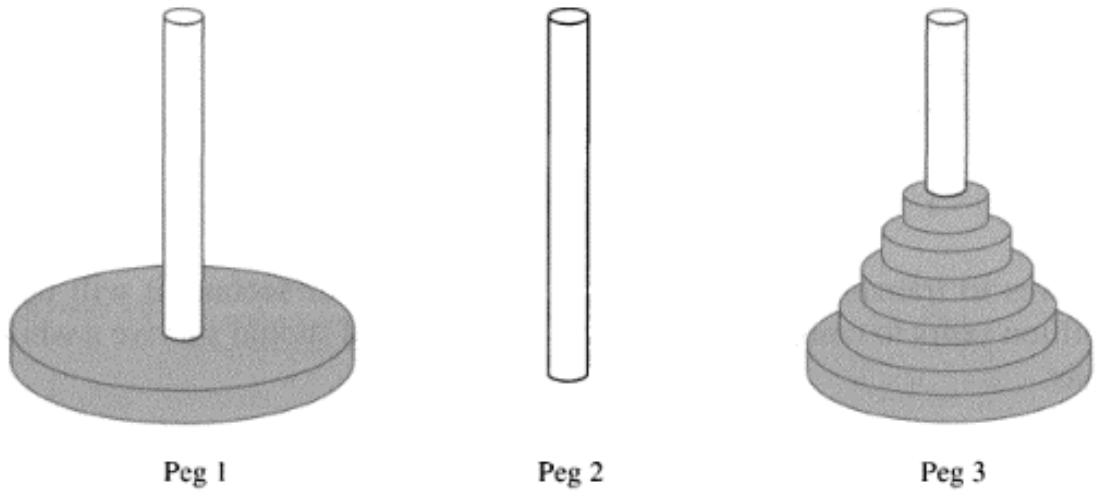


Image source: Rosen's textbook, page 453. Fair use copyright exemption applies.

We can then transfer the largest disk to Peg 2, and then again use H_{n-1} moves to transfer the other disks to Peg 2 on top of the largest disk. Thus,

$$H_n = 2H_{n-1} + 1, \qquad n \geq 2, \qquad H_1 = 1.$$

Tower of Hanoi

The first few terms of (H_n) are

$$1,\ 3,\ 7,\ 15,\ 31, \ldots$$

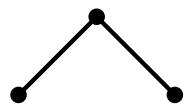
It seems that $H_n = 2^n - 1$. In fact, it is easy to prove this by induction: $H_1 = 2^1 - 1 = 1$, and if $H_{n-1} = 2^{n-1} - 1$, then

$$H_n = 2H_{n-1} + 1 = 2 \cdot (2^{n-1} - 1) + 1 = 2^n - 1.$$

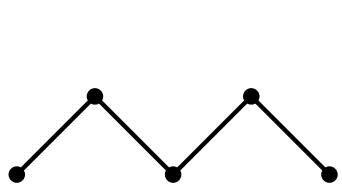
Counting Mountains

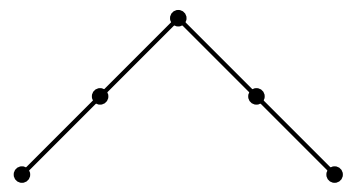
2.4.4. Example. Consider the following problem: We wish to draw "mountains" using upstrokes and downstrokes, like this:

- One upstroke, one downstroke:



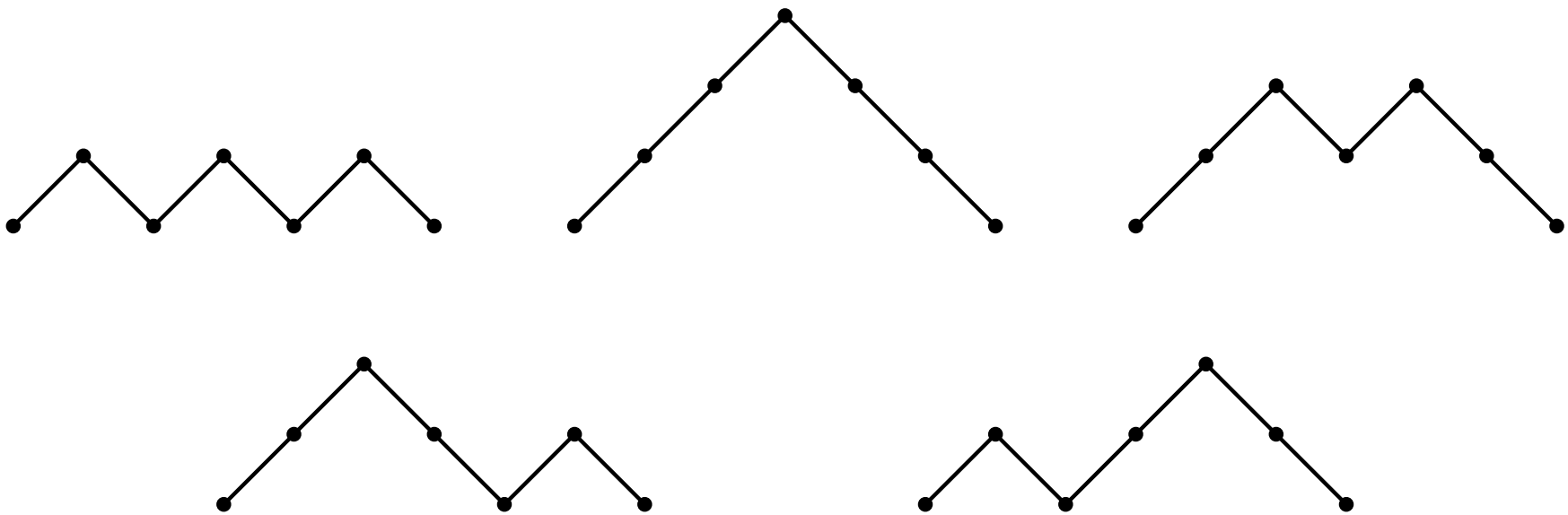
- Two upstrokes, two downstrokes:





Counting Mountains

- Three upstrokes, three downstrokes:



Counting Mountains

The mountains must each consist of n upstrokes and n downstrokes, and there may be no "valleys," i.e., no stroke may lie under the starting point.

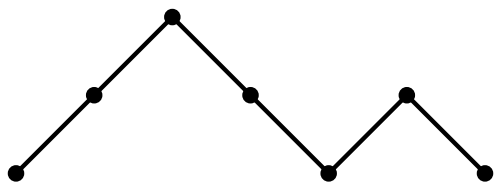
That means that the following constructions are not allowed:

The question is: for given n, how many such mountains using n up- and n downstrokes can we draw?

The answer is related to many diverse combinatorial problems.

Counting Mountains

We would like to consider every mountain as a string or word consisting of the letters $\uparrow$ and $\downarrow$ (if you prefer 0 and 1, this can be a bit string). For instance, we would write



as $\uparrow\uparrow\downarrow\downarrow\uparrow\downarrow$.

However, not all such words do not lead to admissible mountains. For example, $\uparrow\downarrow\downarrow\uparrow\downarrow\uparrow$ does not give a mountain.

Dyck Words and Catalan Numbers

Clearly, we need to impose two conditions on a word w:

2.4.5. Definition. Let $\{\uparrow, \downarrow\}$ be an alphabet. A *Dyck word* is the empty string $\emptyset$ or a word w satisfying

1. $\#(\downarrow) = \#(\uparrow) = n$
2. In any word consisting of the first k letters of w, $\#(\uparrow) \geq \#(\downarrow)$ $(k = 1, \ldots, 2n)$

The number of Dyck words of length $2n$ is called the nth *Catalan number* and denoted C_n.

Hence, any mountain corresponds to a unique Dyck word.

It is clear that $C_1 = 1$. We now find a recursion relation for C_n. For this, note that the set of Dyck words can be described as follows:

- The empty string $\emptyset$ is a Dyck word,
- If w_1 and w_2 are Dyck words, then $\uparrow w_1 \downarrow w_2$ is a Dyck word.

Dyck Words and Catalan Numbers

It is easy to see (using structural induction) that the recursive definition of Dyck words satisfies the properties of Definition 2.4.5. On the other hand, we have the following result:

2.4.6. Theorem. Every Dyck word can we written as $\uparrow w_1 \downarrow w_2$ for some (possibly empty) Dyck words w_1, w_2.

The proof is left to the exercises. From Theorem 2.4.6 we obtain a recurrence relation for the Catalan numbers:

If w is a Dyck word of length $l(w) = 2n$, we can decompose it as $\uparrow w_1 \downarrow w_2$, where $l(w_1) + l(w_2) = 2(n-1)$. Thus the total number of Dyck words of length $2n$ is found by summing over the number of Dyck words w_1 and w_2 with total length $2(n-1)$:

$$C_n = \sum_{k=0}^{n-1} C_k C_{n-1-k}, \qquad n \geq 2. \tag{2.4.2}$$

Catalan Numbers

Catalan numbers appear in many contexts, e.g., as

- The number of ways to parenthesize the product of $n+1$ numbers (cf. Example 8, page 456 of *Rosen*),
- The number of non-crossing handshakes of $2n$ people seated across a circular table,
- The number of triangulations of a polygon with $n+2$ sides,
- The number of full binary trees with n internal nodes,
- The number of rooted trees with n edges.

We will solve the recurrence relation (2.4.2) in the following chapters. Interested students are hereby challenged to try to find a solution by themselves without using external references!

Existence and Uniqueness for Recurrence Relations

2.4.7. Lemma. Let $a_0, \ldots, a_{k-1} \in \mathbb{R}$ be given. Then there exists a unique sequence (a_n) such that a_n satisfies (2.4.1).

Proof.
The existence of a unique sequence (a_n) can be shown by strong induction in n: first, $a_0, \ldots, a_{k-1}$ are given (and exist uniquely). Suppose that for some $n \geq k$ we know that $a_k, \ldots, a_{n-1}$ exist uniquely such that (2.4.4) is satisfied with n replaced by $k, \ldots, n-1$, respectively. Then we define

$$a_n := f(a_0, \ldots, a_{n-1})$$

and we see that a_n exists uniquely. This is precisely the procedure used to verify that inductive definitions make sense. □

Linear Recurrence Relations

2.4.8. Definition. A linear recurrence relation with constant (real) coefficients of degree $k \in \mathbb{N}$ is a recurrence relation of the form

$$a_n = c_1 a_{n-1} + \cdots + c_k a_{n-k} + F(n) \qquad \text{for } n \geq k, \tag{2.4.3}$$

where $c_1, \ldots, c_k \in \mathbb{R}$, $c_k \neq 0$, and $F\colon \mathbb{N} \to \mathbb{R}$ is a sequence. If $F(n) = 0$ for all n we say that (2.4.4) is *homogeneous*, otherwise *inhomogeneous*.

We will first treat the homogeneous case

$$a_n = c_1 a_{n-1} + \cdots + c_k a_{n-k}. \tag{2.4.4}$$

The basic approach for solving (2.4.4) is to make the ansatz

$$a_n = r^n \tag{2.4.5}$$

and attempt to find a number $r \in \mathbb{R}$ such that (2.4.5) satisfies (2.4.4).

Linear Homogeneous Recurrence Relations of Degree 2

This leads to the *characteristic equation* for (2.4.4),

$$r^k - c_1 r^{k-1} - \cdots - c_{k-1} r - c_k = 0, \tag{2.4.6}$$

whose left-hand side is called the *characteristic polynomial*. The solutions of the characteristic equation are called *characteristic roots* of (2.4.4).

We will first treat recurrence relations of degree 2; those of degree 1 are easily solvable, while those of higher degree imply discussing roots of polynomials of degree ≥ 3, which is fairly involved.

2.4.9. Theorem. Let $c_1, c_2 \in \mathbb{R}$ such that $r^2 - c_1 r - c_2 = 0$ has two distinct roots r_1 and r_2. Then the sequence (a_n) is a solution of the recurrence relation $a_n = c_1 a_{n-1} + c_2 a_{n-2}$ if and only if

$$a_n = \alpha_1 \cdot r_1^n + \alpha_2 \cdot r_2^n, \qquad \alpha_1, \alpha_2 \in \mathbb{R}, \ n \in \mathbb{N}. \tag{2.4.7}$$

Linear Homogeneous Recurrence Relations of Degree 2

Proof.

The proof is in two parts: first, we show that (2.4.7) actually solves the recurrence relation $a_n = c_1 a_{n-1} + c_2 a_{n-2}$; then, we show that every solution of $a_n = c_1 a_{n-1} + c_2 a_{n-2}$ is of the form (2.4.7).

Suppose that $a_n = \alpha_1 r_1^n + \alpha_2 r_2^n$. Since r_1, r_2 are roots of $r^2 - c_1 r - c_2 = 0$ we have

$$\begin{aligned} c_1 a_{n-1} + c_2 a_{n-2} &= c_1(\alpha_1 r_1^{n-1} + \alpha_2 r_2^{n-1}) + c_2(\alpha_1 r_1^{n-2} + \alpha_2 r_2^{n-2}) \\ &= \alpha_1 r_1^{n-2}(c_1 r_1 + c_2) + \alpha_2 r_2^{n-2}(c_1 r_2 + c_2) \\ &= \alpha_1 r_1^{n-2} r_1^2 + \alpha_2 r_2^{n-2} r_2^2 \\ &= a_n \end{aligned}$$

so the recurrence relation is satisfied. This shows that (2.4.7) solves the recurrence relation.

Linear Homogeneous Recurrence Relations of Degree 2

Proof (continued).

Now let (a_n) be a solution to the recurrence relation. By Lemma 2.4.7 this sequence is unique and determined by a_0 and a_1. We thus need to show that we can find α_1 and α_2 such that

$$a_0 = \alpha_1 + \alpha_2, \qquad a_1 = \alpha_1 r_1 + \alpha_2 r_2.$$

If this is possible, it follows that $a_n = \alpha_1 r_1^n + \alpha_2 r_2^n$ for all $n \in \mathbb{N}$. A simple calculation yields

$$\alpha_1 = \frac{a_1 - a_0 r_2}{r_1 - r_2}, \qquad \alpha_2 = \frac{a_0 r_1 - a_1}{r_1 - r_2}$$

so

$$a_n = \frac{a_1 - a_0 r_2}{r_1 - r_2} r_1^n + \frac{a_0 r_1 - a_1}{r_1 - r_2} r_2^n.$$

□

Linear Homogeneous Recurrence Relations of Degree 2

2.4.10. Example. The Fibonacci sequence satisfies $f_0 = 0$, $f_1 = 1$ and $f_n = f_{n-1} + f_{n-2}$. The characteristic roots are

$$r_{1,2} = \frac{1 \pm \sqrt{5}}{2},$$

so

$$f_n = \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2} \right)^n - \frac{1}{\sqrt{5}} \left(\frac{1-\sqrt{5}}{2} \right)^n$$

Homogeneous Recurrence Relations of Degree 2

If there is only a single characteristic root of multiplicity 2, then we have the following result:

2.4.11. Theorem. Let $c_1, c_2 \in \mathbb{R}$, $c_2 \neq 0$ such that $r^2 - c_1 r - c_2 = 0$ has a single roots r_0. Then the sequence (a_n) is a solution of the recurrence relation $a_n = c_1 a_{n-1} + c_2 a_{n-2}$ if and only if

$$a_n = \alpha_1 \cdot r_0^n + \alpha_2 \cdot n r_0^n, \qquad \alpha_1, \alpha_2 \in \mathbb{R}, \ n \in \mathbb{N}.$$

The proof is left to the exercises.

2.4.12. Example. We consider the recurrence relation $a_n = 6a_{n-1} - 9a_{n-2}$ with initial conditions $a_0 = 1$ and $a_1 = 6$. There is only one characteristic root, $r = 3$. Then

$$a_n = \alpha_1 3^n + \alpha_2 n 3^n.$$

To find α_1, α_2 we use $a_0 = \alpha_1 = 1$ and $a_1 = 3\alpha_1 + 3\alpha_2 = 6$. This gives

$$a_n = (n+1)3^n.$$

Homogeneous Recurrence Relations of Degree k

We will no prove the following result, which generalizes the solution ansatz to recurrence relations of degree k:

2.4.13. Theorem. Let $c_1, c_2, \ldots, c_k \in \mathbb{R}$ such that

$$r^k - c_1 r^{k-1} - \cdots - c_{k-1} r - c_k = 0$$

has t distinct roots $r_1, \ldots, r_t$ with multiplicities $m_1, \ldots, m_t$, respectively. Then the sequence (a_n) is a solution of the recurrence relation $a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k}$ if and only if

$$\begin{aligned} a_n = {} & (\alpha_{1,0} + \alpha_{1,1} n + \cdots + \alpha_{n,m_1-1} n^{m_1-1}) \cdot r_1^n \\ & + (\alpha_{2,0} + \alpha_{2,1} n + \cdots + \alpha_{2,m_2-1} n^{m_2-1}) \cdot r_2^n \\ & + \cdots + (\alpha_{t,0} + \alpha_{t,1} n + \cdots + \alpha_{t,m_t-1} n^{m_t-1}) \cdot r_t^n, \quad n \in \mathbb{N}. \end{aligned}$$

where $\alpha_{i,j} \in \mathbb{R}$, $1 \leq i \leq t$, $0 \leq j \leq m_i - 1$.

Homogeneous Recurrence Relations of Degree k

2.4.14. Example. We want to solve

$$a_n = -3a_{n-1} - 3a_{n-2} - a_{n-3}, \qquad a_0 = 1, \qquad a_1 = -2, \qquad a_2 = -1.$$

The characteristic is root is found from

$$r^3 + 3r^2 + 3r + 1 = (r+1)^3 = 0$$

so we have $r = -1$ with multiplicity 3. The solution of the recurrence relation is then

$$a_n = \alpha_{1,0}(-1)^n + \alpha_{1,1}n(-1)^n + \alpha_{1,2}n^2(-1)^n.$$

To find the constants we solve

$$\begin{aligned} a_0 &= \alpha_{1,0} = 1, \\ a_1 &= -\alpha_{1,0} - \alpha_{1,1} - \alpha_{1,2} = -2, \\ a_2 &= \alpha_{1,0} + 2\alpha_{1,1} + 4\alpha_{1,2} = -1 \end{aligned}$$

to obtain

$$a_n = (1 + 3n - 2n^2)(-1)^n.$$

Inhomogeneous Recurrence Relations

In the inhomogeneous case we want to solve

$$a_n = c_1 a_{n-1} + \cdots + c_k a_{n-k} + F(n) \qquad \text{for } n \geq k \qquad (2.4.8)$$

with constants $c_1, \ldots, c_k \in \mathbb{R}$. The general solution to (2.4.8) will be the sum of a particular solution and solutions of the associated homogeneous equation. This is analogous to the solution of inhomogeneous systems of linear algebraic equations or of inhomogeneous systems of linear ordinary differential equations.

2.4.15. Theorem. If (a_n^{part}) is a particular solution of (2.4.8), then every solution is of the form $(a_n) = (a_n^{\text{part}} + a_n^{\text{hom}})$, where (a_n^{hom}) solves (2.4.8) with $F(n) = 0$ for all n.

We omit the proof, which is similar to the corresponding statements for systems of ODEs or linear equations.

Inhomogeneous Recurrence Relations

2.4.16. Example. Consider the recurrence equation $a_n = 3a_{n-1} + 2n$ with initial value $a_1 = 3$. The associated homogeneous equation $a_n = 3a_{n-1}$ has general solution $a_n^{\text{hom}} = \alpha \cdot 3^n$. A particular solution can be found by educated guessing: since $F(n) = 2n$ is a polynomial of degree 1, try setting $a_n^{\text{part}} = cn + d$ for $c, d \in \mathbb{R}$ to be determined. Then

$$cn + d = 3(c(n-1) + d) + 2n \qquad \Rightarrow \qquad (2 + 2c)n + (2d - 3c) = 0,$$

so we have $c = -1$ and $d = -3/2$. Thus,

$$a_n = \alpha \cdot 3^n - n - 3/2$$

solves the recurrence relation. The initial condition $a_1 = 3$ then gives $\alpha = 11/6$, so

$$a_n = \frac{11}{2} 3^{n-1} - n - \frac{3}{2}.$$

Inhomogeneous Recurrence Relations

It is no accident that in this example we were able to find a particular solution by guessing it to be a polynomial. In fact, if $F(n)$ is an exponential function, then we can also find a particular solution of exponential form. In fact, the following theorem holds.

Inhomogeneous Recurrence Relations

2.4.17. Theorem. Consider the inhomogeneous equation (2.4.8) with

$$F(n) = s^n \sum_{j=0}^{t} b_j n^j, \qquad s, b_0, \ldots, b_t \in \mathbb{R}.$$

A particular solution to (2.4.8) then has the following form:

- If s is not a characteristic root of the associated homogeneous equation, then there exist numbers $p_0, \ldots, p_t \in \mathbb{R}$ such that

$$a_n^{\text{part}} = s^n \sum_{j=0}^{t} p_j n^j.$$

- If s is a characteristic root of multiplicity m, then there exist numbers $p_0, \ldots, p_t \in \mathbb{R}$ such that

$$a_n^{\text{part}} = n^m s^n \sum_{j=0}^{t} p_j n^j.$$

Inhomogeneous Recurrence Relations

2.4.18. Example. Consider the recurrence relation

$$a_n = 6a_{n-1} - 9a_{n-2} + F(n),$$

where we are interested in the cases

1. $F(n) = 3^n$,
2. $F(n) = n3^n$,
3. $F(n) = n^2 2^n$,
4. $F(n) = (n^2 + 1)3^n$.

The associated homogeneous equation is $a_n = 6a_{n-1} - 9a_{n-2}$ which has a single characteristic root $r = 3$ of multiplicity 2. We then have particular solutions of the following forms:

1. $a_n^{\text{part}} = p_0 n^2 3^n$,
2. $a_n^{\text{part}} = (p_0 + p_1 n)n^2 3^n$,
3. $a_n^{\text{part}} = (p_0 + p_1 n + p_2 n^2)2^n$,
4. $a_n^{\text{part}} = (p_0 + p_1 n + p_2 n^2)n^2 3^n$,

Divide-and-Conquer Algorithms

The principle of "divide and conquer", (originating from *Divide et impera*, divide and rule, in Latin) goes back to the ancient Romans who used political, economic and military means to break up alliance of opponents or concentrations of power so that each part was less powerful than themselves.

In computer algorithms, *divide-and-conquer* algorithms are those that break up a problem into several smaller instances and tackle each instance separately. They are similar to recursive algorithms, but "multi-branched", i.e., referring to several instances of themselves, not just one, with smaller input.

We are interested in analyzing the time complexity of divide-and-conquer algorithms. It turns out that recurrence relations are well-suited to describing the number of steps needed for such an algorithm to complete and that we can find quite general formulas for big-Oh estimates for many classes of algorithms.

Divide-and-Conquer Algorithms

2.4.19. Examples.

1. The Binary Search Algorithm 1.7.7 reduces the problem of locating an element of a list to locating it in a list of half the length. Two comparisons are needed to do this, one to determine which half-list to search, the other to determine whether any terms remain in the list. Therefore, the number of comparisons needed to search a list of length n is given by
$$f(n) = f(n/2) + 2.$$
Since only one list of length $n/2$ is searched, this is actually a recursive algorithm, a special case of a divide-and-conquer algorithm.
2. A divide-and-conquer algorithm to find the minimum and maximum of a list of length $n = 2k$ might split the list into two equally sized sublists, find the minimum and maximum in each sublist and compare the values of the two sublists. The number of comparisons needed is then given by
$$f(n) = 2f(n/2) + 2.$$

Divide-and-Conquer Algorithms

3. There exists a divide-and-conquer algorithm for integer multiplication which is faster than Algorithm 2.1.31. Suppose that a and b are integers with binary representations of length $2n$ (adding leading zeroes if necessary),

$$a = (a_{2n-1} \dots a_1 a_0)_2, \qquad b = (b_{2n-1} \dots b_1 b_0)_2.$$

Setting

$$A_0 = (a_{n-1} \dots a_0)_2, \qquad A_1 = (a_{2n-1} \dots a_n)_2,$$
$$B_0 = (b_{n-1} \dots b_0)_2, \qquad B_1 = (b_{2n-1} \dots b_n)_2$$

we can write

$$a = 2^n A_1 + A_0, \qquad b = 2^n B_1 + B_0.$$

Divide-and-Conquer Algorithms

Then

$$a \cdot b = (2^{2n} + 2^n)A_1B_1 + 2^n(A_1 - A_0)(B_0 - B_1) + (2^n + 1)A_0B_0,$$

so the multiplication of two integers of length $2n$ is reduced to summing three multiplications of integers of length n. The number of operations required for the multiplications by powers of 2 (shifts in the binary expansion) and addition are proportional to n, so the total number of operations is given by

$$f(2n) = 3f(n) + Cn$$

for some $C \in \mathbb{N}$.

4. The Merge Sort Algorithm 1.7.19 reduces the problem of sorting a list with n elements to that of sorting two lists with $n/2$ elements. It then uses fewer than n comparisons to merge the two sorted lists. Therefore, it uses fewer than $M(n)$ comparisons, where

$$M(n) = 2M(n/2) + n.$$

Time Complexity of Divide-and-Conquer Algorithms

2.4.20. Theorem. Let f be an increasing function that satisfies the recurrence relation

$$f(n) = af(n/b) + c, \qquad a \geq 1,\ b \in \mathbb{N} \setminus \{0, 1\},\ c > 0$$

whenever n is divisible by b. Then

$$f(n) = \begin{cases} O(n^{\log_b a}) & \text{if } a > 1, \\ O(\log n) & \text{if } a = 1. \end{cases}$$

Furthermore, whenever $a > 1$ and $n = b^k$ for some $k \in \mathbb{Z}_+$,

$$f(n) = C_1 n^{\log_b a} + C_2 \tag{2.4.9}$$

with

$$C_1 = f(1) + \frac{c}{a-1}, \qquad C_2 = -\frac{c}{a-1}.$$

Time Complexity of Divide-and-Conquer Algorithms

Proof.

We first note that if $f(n) = af(n/b) + g(n)$ for some function g,

$$\begin{aligned} f(n) &= af\left(\frac{n}{b}\right) + g(n) = a\left(af\left(\frac{n}{b^2}\right) + g\left(\frac{n}{b}\right)\right) + g(n) \\ &= a^2 f\left(\frac{n}{b^2}\right) + ag\left(\frac{n}{b}\right) + g(n) \\ &\vdots \\ &= a^k f\left(\frac{n}{b^k}\right) + \sum_{j=0}^{k-1} a^j g\left(\frac{n}{b^j}\right) \end{aligned}$$

for any $k \in \mathbb{N}$. If $n = b^k$ for some $k \in \mathbb{N}$, we obtain

$$f(n) = a^k f(1) + \sum_{j=0}^{k-1} a^j g\left(\frac{n}{b^j}\right). \tag{2.4.10}$$

Time Complexity of Divide-and-Conquer Algorithms

Proof (continued).

We will now prove the various statements of the theorem.

(i) Suppose that $a = 1$. If $n = b^k$ for some $k \in \mathbb{N}$, (2.4.10) gives

$$f(n) = f(1) + c \cdot k = f(1) + c \cdot \log_b n = O(\log n).$$

Furthermore, if $b^k < n < b^{k+1}$ for some $k \in \mathbb{N}$ we use that f is increasing to obtain

$$f(n) \leq f(b^{k+1}) = f(1) + c(k+1) = f(1) + c + c \cdot \log_b n = O(\log n).$$

This proves the statement of the theorem for $a = 1$.

Time Complexity of Divide-and-Conquer Algorithms

Proof (continued).

(ii) Now let $a > 1$ and $n = b^k$. Then (2.4.10) gives

$$f(n) = a^k f(1) + c\sum_{j=0}^{k-1} a^j = a^k f(1) + c\frac{a^k - 1}{a-1}$$
$$= a^k \left(f(1) + \frac{c}{a-1} \right) - \frac{c}{a-1}$$
$$= C_1 n^{\log_b a} + C_2,$$

where we have used $a^k = a^{\log_b n} = n^{\log_b a}$. This proves (2.4.9). If $b^k < n < b^{k+1}$ for some $k \in \mathbb{N}$ we use that f is increasing to obtain

$$f(n) \le f(b^{k+1}) = a^k \left(af(1) + \frac{ac}{a-1} \right) - \frac{c}{a-1} = O(n^{\log_b a}). \qquad \square$$

Time Complexity of Divide-and-Conquer Algorithms

We now apply Theorem 2.4.20 to the first two of the Examples 2.4.19:

2.4.21. Examples.

1. The comparisons needed by the Binary Search algorithm satisfy $f(n) = f(n/2) + 2$, so
$$f(n) = O(\log n).$$

2. The comparisons needed for finding the minimum and maximum of a list of integers satisfy $f(n) = 2f(n/2) + 2$, so
$$f(n) = O(n^{\log_2 2}) = O(n).$$

In order to treat the other examples, we need a more general theorem.

The Master Theorem

The following result is called the "Master Theorem" because it can be used to obtain the time complexity of many divide-and-conquer algorithms. There exist more powerful variants that give big-Θ estimates.

2.4.22. Master Theorem. Let f be an increasing function that satisfies the recurrence relation

$$f(n) = af(n/b) + cn^d, \qquad a \geq 1,\ b \in \mathbb{N} \setminus \{0,1\},\ c > 0,\ d \geq 0$$

whenever $n = b^k$. Then

$$f(n) = \begin{cases} O(n^d) & \text{if } a < b^d, \\ O(n^d \log n) & \text{if } a = b^d, \\ O(n^{\log_b a}) & \text{if } a > b^d, \end{cases}$$

The proof of the Master Theorem is relegated to the exercises.

The Master Theorem

We now apply the Master Theorem to the remaining two Examples 2.4.19:

2.4.23. Examples.

1. The number of comparisons needed by the Merge Sort algorithm is bounded by $M(n)$, where $M(n) = 2M(n/2) + n$, so

$$M(n) = O(n \log n).$$

2. The number of operations needed for the fast integer multiplication satisfies $f(n) = 3f(n/2) + Cn$, so

$$f(n) = O(n^{\log_2 3}) = O(n^{1.585}).$$

This is a significant improvement on the "classical" algorithm, which uses $O(n^2)$ operations.

The Master Theorem

A more complicated algorithm that is analyzed using the Master Theorem is presented on Page 479 of the textbook. There, the *closest-pair problem* is shown to be solvable with $O(n \log n)$ operations, instead of the naively expected $O(n^2)$ operations.

Generating Functions

In order to analyze more complicated recurrence relations (e.g., non-linear relations such as (2.4.2)) and other counting problems, we introduce the concept of a *generating function*.

2.4.24. Definition. Let (a_n) be a sequence of real numbers. If the formal power series

$$G(x) = \sum_{n=0}^{\infty} a_n x^n = a_0 + a_1 x + a_2 x^2 + \dots$$

has a radius of convergence $\varrho > 0$ we say that G is a *generating function* for (a_n). A generating function for a finite tuple $(a_1, \dots, a_k)$ of real numbers is obtained by first extending the tuple to a sequence by setting $a_n = 0$ for $n > k$ and then applying the above definition.

Generating Functions

2.4.25. Examples.

1. The geometric series formula yields

$$\frac{1}{1-x} = \sum_{n=0}^{\infty} x^n,$$

   so $G(x) = 1/(1-x)$ is the generating function for the sequence (a_n) with $a_n = 1$ for all $n \in \mathbb{N}$.

2. Similarly, for $a \neq 0$ the function $G(x) = 1/(1-ax)$ is the generating function for the sequence $(1, a, a^2, a^3, \ldots)$.

Generating Functions

3. By the binomial theorem,

$$(1+x)^n = \sum_{k=0}^{n} \binom{n}{k} x^k$$

so the function $G(x) = (1+x)^n$ is the generating function for the $n+1$-tuple

$$\binom{n}{0}, \binom{n}{1}, \ldots, \binom{n}{n}.$$

The Cauchy Product

Generating functions are power series with non-vanishing radii of convergence. In order to work with them effectively, we need to recall some background from calculus. In particular, we have the following result regarding the product of two power series:

2.4.26. Theorem. Let $\sum a_k$ and $\sum b_k$ be series that both converge absolutely. Then the *Cauchy product* $\sum c_k$,

$$c_k := \sum_{i+j=k} a_i b_j$$

converges absolutely and $\sum c_k = \left(\sum a_k\right)\left(\sum b_k\right)$. The sequence

$$(a_k) * (b_k) := (c_k), \qquad c_k := \sum_{i+j=k} a_i b_j,$$

is called the *convolution* of the sequences (a_k) and (b_k).

The Cauchy Product

2.4.27. Example. The Cauchy product is often used in calculus to establish the functional equation for the exponential function. The exponential function can be defined to be

$$\exp(x) = \sum_{n=0}^{\infty} \frac{x^n}{n!}.$$

Then

$$\begin{aligned}
(\exp x)(\exp y) &= \left(\sum_{n=0}^{\infty} \frac{x^n}{n!}\right)\left(\sum_{m=0}^{\infty} \frac{y^m}{m!}\right) = \sum_{n=0}^{\infty}\left(\left(\frac{x^k}{k!}\right) * \left(\frac{y^k}{k!}\right)\right)_n \\
&= \sum_{n=0}^{\infty} \sum_{l+m=n} \frac{1}{l!m!} x^l y^m = \sum_{n=0}^{\infty} \frac{1}{n!} \sum_{l+m=n} \frac{n!}{l!m!} x^l y^m \\
&= \sum_{n=0}^{\infty} \frac{1}{n!}(x+y)^n \\
&= \exp(x+y)
\end{aligned}$$

The Cauchy Product

This yields the functional equation

$$(\exp x)(\exp y) = \exp(x+y).$$

We also remark that $\exp x = e^x$ is the generating function for the sequence (a_n) with $a_n = 1/n!$.

2.4.28. Corollary. Let $\sum a_k x^k$ and $\sum b_k x^k$ be power series that both converge absolutely for $|x| < \varrho$. Then

$$\Big(\sum_{l=0}^{\infty} a_l x^l\Big)\Big(\sum_{m=0}^{\infty} b_m x^m\Big) = \sum_{n=0}^{\infty} c_n x^n$$

with

$$c_n = \sum_{i+j=n} a_i b_j$$

The Cauchy Product

2.4.29. Example. For any $n \in \mathbb{Z}_+$ and $k = 0, \ldots, n$,

$$\binom{n}{0}\binom{n}{k} + \binom{n}{1}\binom{n}{k-1} + \cdots + \binom{n}{k}\binom{n}{0} = \binom{2n}{k}.$$

We can write this identity as

$$\sum_{i=0}^{k} \binom{n}{i}\binom{n}{k-i} = \sum_{i+j=k} \binom{n}{i}\binom{n}{j} = \binom{2n}{k} \tag{2.4.11}$$

To prove (2.4.11), we consider the generating function of the finite sequence $\binom{2n}{k}$, $k = 0, \ldots, 2n$, given by

$$G(x) = (1+x)^{2n} = \sum_{k=0}^{2n} \binom{2n}{k} x^k. \tag{2.4.12}$$

The Cauchy Product

For $k = 0, \ldots, n$ the right-hand-side of (2.4.11) is given by the coefficient of x^k in $G(x)$. Now

$$\begin{aligned}(1+x)^{2n} &= \big((1+x)^n\big)^2 \\ &= \Big(\sum_{l=0}^{n} \binom{n}{l} x^l\Big)\Big(\sum_{m=0}^{n} \binom{n}{m} x^m\Big) \\ &= \sum_{k=0}^{2n}\Big(\sum_{i+j=k} \binom{n}{i}\binom{n}{j}\Big)x^k.\end{aligned}$$

Comparing with (2.4.12) gives (2.4.11). The formula (2.4.11) was known to the Chinese mathematician Zhu Shijie (朱世杰) who lived around AD 1300. Setting $k = n$ we obtain the special case

$$\sum_{k=0}^{n} \binom{n}{k}^2 = \binom{2n}{n},$$

which is known as *Vandermonde's convolution formula*.

The Binomial Series

We will need some more background from basic calculus to employ generating functions to their full potential. Recall that the generalized binomial coefficients are defined by

$$\binom{\alpha}{0} := 1, \qquad \binom{\alpha}{j} := \frac{\alpha(\alpha-1)\dots(\alpha-j+1)}{j!}, \qquad j \in \mathbb{N},\ \alpha \in \mathbb{R}.$$

The following theorem should be known from calculus:

2.4.30. Theorem. Let $-1 < x < 1$ and $\alpha \in \mathbb{R}$. Then

$$(1+x)^\alpha = \sum_{n=0}^{\infty} \binom{\alpha}{n} x^n$$

where the series on the right converges absolutely.

The Binomial Series

2.4.31. Example. For $\alpha = 1/2$, we have the binomial series

$$\sqrt{1+x} = \sum_{n=0}^{\infty} \binom{1/2}{n} x^n.$$

The first term, $n = 0$, evaluates to

$$\binom{1/2}{0} x^0 = 1.$$

For $n \geq 1$ we have

$$\begin{aligned}
\binom{1/2}{n} &= \frac{1/2(1/2-1)(1/2-2)\dots(1/2-(n-1))}{n!} \\
&= (-1)^{n-1}\frac{1/2(1-1/2)(2-1/2)\dots(n-1-1/2)}{n!} \\
&= \frac{(-1)^{n-1}}{2^n}\frac{1(2-1)(4-1)\dots(2(n-1)-1)}{n!}
\end{aligned}$$

The Binomial Series

Continuing,

$$\begin{aligned}\binom{1/2}{n} &= \frac{(-1)^{n-1}}{2^n}\frac{1(2-1)(4-1)\dots(2(n-1)-1)}{n!}\\ &= \frac{(-1)^{n-1}}{2^n}\frac{1\cdot 1\cdot 3\cdot 5\cdots(2n-3)}{n!}\\ &= \frac{(-1)^{n-1}}{2^n}\frac{1}{2^{n-1}(n-1)!}\frac{1\cdot 2\cdot 3\cdot 4\cdot 5\cdots(2n-2)}{n!}\\ &= -2\frac{(-1)^n}{4^n}\frac{1}{n}\frac{(2n-2)!}{(n-1)!(n-1)!} = -2\frac{(-1)^n}{4^n}\frac{1}{n}\binom{2n-2}{n-1}.\end{aligned}$$

It follows that

$$\sqrt{1+x} = 1 - 2\sum_{n=1}^{\infty}\frac{(-1)^n}{4^n}\binom{2n-2}{n-1}\frac{x^n}{n} \tag{2.4.13}$$

for $|x| < 1$.

The Catalan Numbers

We will now find a closed formula for the Catalan numbers C_n. Recall that they satisfy the recurrence relation (2.4.2),

$$C_{n+1} = \sum_{k=0}^{n} C_k C_{n-k} = \sum_{j+k=n} C_k C_j, \qquad n \in \mathbb{N}.$$

with $C_0 = 1$. Define the generating function of the catalan numbers to be the formal power series

$$c(x) = \sum_{n=0}^{\infty} C_n x^n.$$

Then

$$c(x)^2 = \Big(\sum_{j=0}^{\infty} C_j x^j\Big)\Big(\sum_{k=0}^{\infty} C_k x^k\Big) = \sum_{n=0}^{\infty}\Big(\sum_{j+k=n} C_k C_j\Big)x^n$$

$$= \sum_{n=0}^{\infty} C_{n+1} x^n = \frac{1}{x}(c(x) - 1)$$

The Catalan Numbers

It follows that

$$c(x) = 1 + xc(x)^2.$$

This implies that

$$c(x) = \frac{1 - \sqrt{1-4x}}{2x} = \frac{2}{1 + \sqrt{1-4x}}$$

From the last quotient we see that $c(x)$ actually has a power series expansion at $x = 0$. Expanding the first quotient using (2.4.13),

$$\begin{aligned} c(x) &= \frac{1}{2x}(1 - \sqrt{1-4x}) = \frac{1}{2x} \cdot 2 \sum_{n=1}^{\infty} \frac{(-1)^n}{4^n} \binom{2n-2}{n-1} \frac{(-4x)^n}{n} \\ &= \sum_{n=1}^{\infty} \binom{2n-2}{n-1} \frac{x^{n-1}}{n} \\ &= \sum_{n=0}^{\infty} \binom{2n}{n} \frac{x^n}{n+1} = \sum_{n=0}^{\infty} C_n x^n. \end{aligned}$$

The Catalan Numbers

This shows that

$$C_n = \frac{1}{n+1}\binom{2n}{n}.$$

Generating Functions and Counting

Generating functions can also be used directly in certain counting problems. We illustrate through an example:

2.4.32. Example. We want to find the number of solutions of

$$e_1 + e_2 + e_3 = 17$$

where

$$2 \le e_1 \le 5, \qquad 3 \le e_2 \le 6, \qquad 4 \le e_3 \le 7. \tag{2.4.14}$$

This is a more complicated version of Example 2.2.40. The solution will be the coefficient of x^{17} in the expansion of

$$(x^2 + x^3 + x^5)(x^3 + x^4 + x^5 + x^6)(x^4 + x^5 + x^6 + x^7),$$

since that coefficients will count all products of monomials $x^{e_1} x^{e_2} x^{e_3} = x^{e_1+e_2+e_3}$ with (2.4.14). For a more complicated application, we refer to Example 12 of Section 7.4 of the textbook.

The Inclusion-Exclusion Principle

In Theorem 2.2.11 we have proven that for two finite sets A_1, A_2,

$$\operatorname{card}(A \cup B) = \operatorname{card} A + \operatorname{card} B - \operatorname{card}(A \cap B). \tag{2.4.15}$$

For the following arguments we will write $|A|$ for $\operatorname{card} A$ for brevity. It turns out that (2.4.15) can be generalized; for instance,

$$\begin{aligned} |A \cup B \cup C| = {} & |A| + |B| + |C| - |A \cup B| - |B \cup C| - |A \cup C| \\ & + |A \cap B \cap C|. \end{aligned}$$

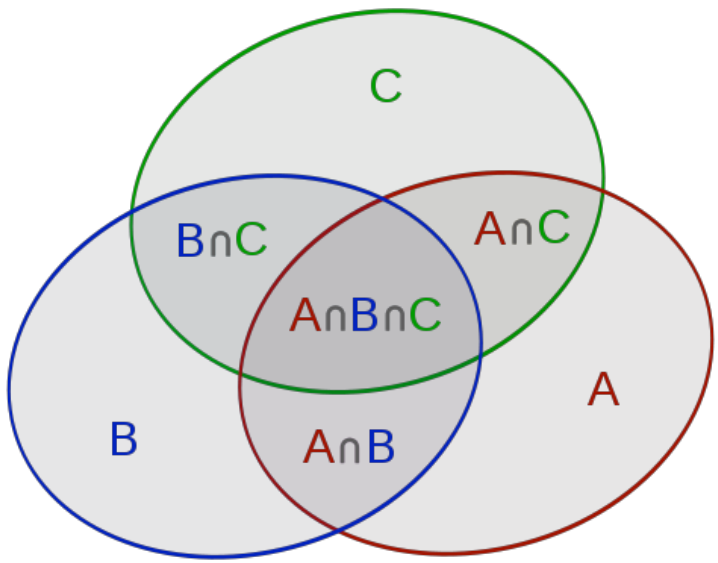


Image source: http://commons.wikimedia.org/wiki/File:Inclusion-exclusion.svg Used under the license given there

The Inclusion-Exclusion Principle

The general theorem which governs the cardinality of the union of n sets is called the *Inclusion-Exclusion Principle*

2.4.33. Inclusion-Exclusion Principle. For any sets $A_1, \ldots, A_n$, $n \in \mathbb{N}$,

$$\begin{aligned} |A_1 \cup A_2 \cup \ldots \cup A_n| = & \sum_{1 \le i \le n} |A_i| - \sum_{1 \le i < j \le n} |A_i \cap A_j| \\ & + \sum_{1 \le i < j < k \le n} |A_i \cap A_j \cap A_k| \\ & - + \ldots + (-1)^{n+1} |A_1 \cap A_2 \cap \ldots \cap A_n| \end{aligned} \tag{2.4.16}$$

The Inclusion-Exclusion Principle

Proof.

We will show that the right-hand side of (2.4.16) counts every element of the union $\bigcup A_i$ exactly once. Suppose that an element a is contained in exactly r of the sets $A_1, \ldots, A_n$, $1 \leq r \leq n$. Then the first sum counts a exactly $r = \binom{r}{1}$ times. The second sum counts a $\binom{r}{2}$ times, since there are $\binom{r}{2}$ pairs of sets which both contain a. In general, a sum involving the intersection of m sets will count a $\binom{r}{m}$ times.

Therefore, the number of times that the right-hand side (2.4.16) counts a is

$$\begin{aligned}\binom{r}{1} - \binom{r}{2} + - \cdots + (-1)^{r+1}\binom{r}{r} &= 1 - \sum_{m=0}^{r}(-1)^m\binom{r}{m} \\ &= 1 - (1-1)^r = 1 \qquad \square\end{aligned}$$

The Probabilistic Inclusion-Exclusion Principle

Another formulation of the theorem is useful in probability theory:

2.4.34. Inclusion-Exclusion Principle. Let S be a sample space and $A_1, \ldots, A_n \subset S$ events such that $P(A_i) \in [0,1]$, $i = 1, \ldots, n$. Then

$$P(A_1 \cup A_2 \cup \ldots \cup A_n) = \sum_{1 \le i \le n} P(A_i) - \sum_{1 \le i < j \le n} P(A_i \cap A_j) + \sum_{1 \le i < j < k \le n} P(A_i \cap A_j \cap A_k) - + \ldots + (-1)^{n+1} P(A_1 \cap A_2 \cap \ldots \cap A_n) \quad (2.4.17)$$

Since we are not assuming that S is a countable set, proving this version of the theorem is best done using induction.

The Matching Problem

An application of the inclusion-exclusion principle is the *matching problem*.

2.4.35. Example. Suppose that someone writes n letters and addresses n corresponding envelopes, but then puts the letters into the envelopes randomly. Of interest is the probability that there is at least one match, i.e., that at least one letter is put into the correct envelope. In particular, how does this probability behave as n becomes large?

Let A_i be the event that the ith letter is put into the correct envelope. We want to find $P(A_1 \cup A_2 \cup \cdots \cup A_n)$, so we apply the inclusion-exclusion formula. First we note that $P(A_i \cap A_j)$ does not depend on i and j, i.e., the probability that the correct letters are put into any given two envelopes doesn't depend on their numbers. We set $P(A_i \cap P_j) =: p_2$, so

$$\sum_{1 \le i < j \le n} P(A_i \cap A_j) = \binom{n}{2} p_2.$$

A similar argument applies to the other probabilities in (2.4.17).

The Matching Problem

We then have

$$\begin{aligned}
&P(A_1 \cup A_2 \cup \ldots \cup A_n) \\
&= \sum_{1 \le i \le n} P(A_i) - \sum_{1 \le i < j \le n} P(A_i \cap A_j) + \sum_{1 \le i < j < k \le n} P(A_i \cap A_j \cap A_k) \\
&\quad - + \ldots + (-1)^{n+1} P(A_1 \cap A_2 \cap \ldots \cap A_n) \\
&= \binom{n}{1} p_1 - \binom{n}{2} p_2 + \binom{n}{3} p_3 - + \cdots + (-1)^{n+1} \binom{n}{n} p_n.
\end{aligned}$$

It is clear that $p_1 = P(A_i)$, the probability of putting the correct letter into the ith envelope, is $1/n$. However, $p_2 = P(A_i \cap A_j) \neq P(A_i)P(A_j)$ because the events A_i and A_j are not independent.

Let us consider the probability p_r of $P(A_{i_1} \cap \cdots \cap A_{i_r})$. There are $n!$ ways of matching letters to envelopes, but we require r envelopes to have the correct letters. Thus, there remain $(n-r)!$ ways of matching the other letters to envelopes.

The Matching Problem

It follows that

$$p_r = \frac{(n-r)!}{n!}.$$

Then

$$\begin{aligned} P(A_1 \cup A_2 \cup \ldots \cup A_n) &= \binom{n}{1} p_1 - \binom{n}{2} p_2 + - \cdots + (-1)^{n+1} \binom{n}{n} p_n \\ &= -\sum_{r=1}^{n} (-1)^r \binom{n}{r} \frac{(n-r)!}{n!} = -\sum_{r=1}^{n} \frac{(-1)^r}{r!}. \end{aligned}$$

Since $e^{-1} = \sum_{r=0}^{\infty} (-1)^r / r!$, we have

$$P(A_1 \cup A_2 \cup \ldots \cup A_n) \approx 1 - \frac{1}{e} \approx 0.63212.$$

The approximation is actually quite good for small n, e.g., for $n = 7$ we obtain a probability of 0.63214.

Inclusion-Exclusion (Alternative Form)

Suppose that S is a finite set and $A_1, \ldots, A_n \subset S$. Then

$$A_1 \cap A_2 \cap \cdots \cap A_n = S \setminus (A_1^c \cup A_2^c \cup \cdots \cup A_n^c)$$

and, therefore,

$$\begin{aligned} |A_1 \cap A_2 \cap \ldots \cap A_n| &= |S| - |A_1^c \cup A_2^c \cup \ldots \cup A_n^c| \\ &= |S| - \sum_{1 \le i \le n} |A_i^c| + \sum_{1 \le i < j \le n} |A_i^c \cap A_j^c| \\ &\quad - + \ldots + (-1)^n |A_1^c \cap A_2^c \cap \ldots \cap A_n^c|. \end{aligned} \tag{2.4.18}$$

We can apply (2.4.18) to certain counting problems.

2.4.36. Example. We are interested in the number of primes less than or equal to n for a given $n \in \mathbb{N} \setminus \{0, 1\}$. By Theorem 2.1.16, to check whether a number p is prime, it suffices to check whether it is a multiple of any number less than $\sqrt{p}$.

Prime Numbers and the Sieve of Eratosthenes

Suppose we are interested in the number of primes less than $n = 120$. Then we need to eliminate from $S = \{2, 3, \ldots, 120\}$ all composite numbers, i.e., all numbers with prime factors less than $\lfloor \sqrt{120} \rfloor = 10$. These are the numbers $2, 3, 5, 7$, so let

$$A_1 := \{q \in S \colon 2 \mid q\}, \qquad A_2 := \{q \in S \colon 3 \mid q\},$$
$$A_3 := \{q \in S \colon 5 \mid q\}, \qquad A_4 := \{q \in S \colon 7 \mid q\}.$$

Then a prime number in S will be equal to 2, 3, 5 or 7 or an element of $A_1^c \cap A_2^c \cap A_3^c \cap A_4^c$. We apply (2.4.18):

$$|A_1^c \cap A_2^c \cap A_3^c \cap A_4^c| = |S| - \sum_{i=1}^{4} |A_i| + \sum_{1 \le i < j \le 4} |A_i \cap A_j| - \sum_{1 \le i < j < k \le 4} |A_i \cap A_j \cap A_k| + |A_1 \cap A_2 \cap A_3 \cap A_4| \tag{2.4.19}$$

Prime Numbers and the Sieve of Eratosthenes

Now

$$|S| = 119, \quad |A_1| = 60, \quad |A_2| = 40, \quad |A_3| = 24, \quad |A_4| = 17.$$

Furthermore,

$$\begin{aligned}
A_1 \cap A_2 &= \{q \in S\colon 6 \mid q\}, & |A_1 \cap A_2| &= 20,\\
A_1 \cap A_3 &= \{q \in S\colon 10 \mid q\}, & |A_1 \cap A_3| &= 12,\\
A_1 \cap A_4 &= \{q \in S\colon 14 \mid q\}, & |A_1 \cap A_4| &= 8,\\
A_2 \cap A_3 &= \{q \in S\colon 15 \mid q\}, & |A_2 \cap A_3| &= 8,\\
A_2 \cap A_4 &= \{q \in S\colon 21 \mid q\}, & |A_2 \cap A_4| &= 5,\\
A_3 \cap A_4 &= \{q \in S\colon 35 \mid q\}, & |A_3 \cap A_4| &= 3,\\
A_1 \cap A_2 \cap A_3 &= \{q \in S\colon 30 \mid q\}, & |A_1 \cap A_2 \cap A_3| &= 4,\\
A_1 \cap A_2 \cap A_4 &= \{q \in S\colon 42 \mid q\}, & |A_1 \cap A_2 \cap A_4| &= 2,\\
A_1 \cap A_3 \cap A_4 &= \{q \in S\colon 70 \mid q\}, & |A_1 \cap A_3 \cap A_4| &= 1,\\
A_2 \cap A_3 \cap A_4 &= \{q \in S\colon 105 \mid q\}, & |A_2 \cap A_3 \cap A_4| &= 1
\end{aligned}$$

Prime Numbers and the Sieve of Eratosthenes

and, finally,

$$A_1 \cap A_2 \cap A_3 \cap A_4 = \{q \in S\colon 210 \mid q\}, \qquad |A_1 \cap A_2 \cap A_3 \cap A_4| = 0.$$

Inserting these values into (2.4.19), we obtain

$$\begin{aligned} |A_1^c \cap A_2^c \cap A_3^c \cap A_4^c| &= 119 - 60 - 40 - 24 - 17 \\ &\quad + 20 + 12 + 8 + 8 + 5 + 3 - 4 - 2 - 1 - 1 + 0 \\ &= 26 \end{aligned}$$

It follows that there are $4 + 26 = 30$ primes less than or equal to 120.

The *Sieve of Eratosthenes* is a simple algorithm that allows us to find these primes. The sieve finds all prime numbers less than n by eliminating (sifting) all multiples of integers less than $\sqrt{n}$.

We illustrate by finding the prime numbers less than or equal to 120. It suffices to eliminate from the list $\{2, 3, \ldots, 120\}$ all numbers that are multiples of $2, \ldots, \lfloor\sqrt{120}\rfloor = 2, \ldots, 10$.

The Sieve of Eratosthenes

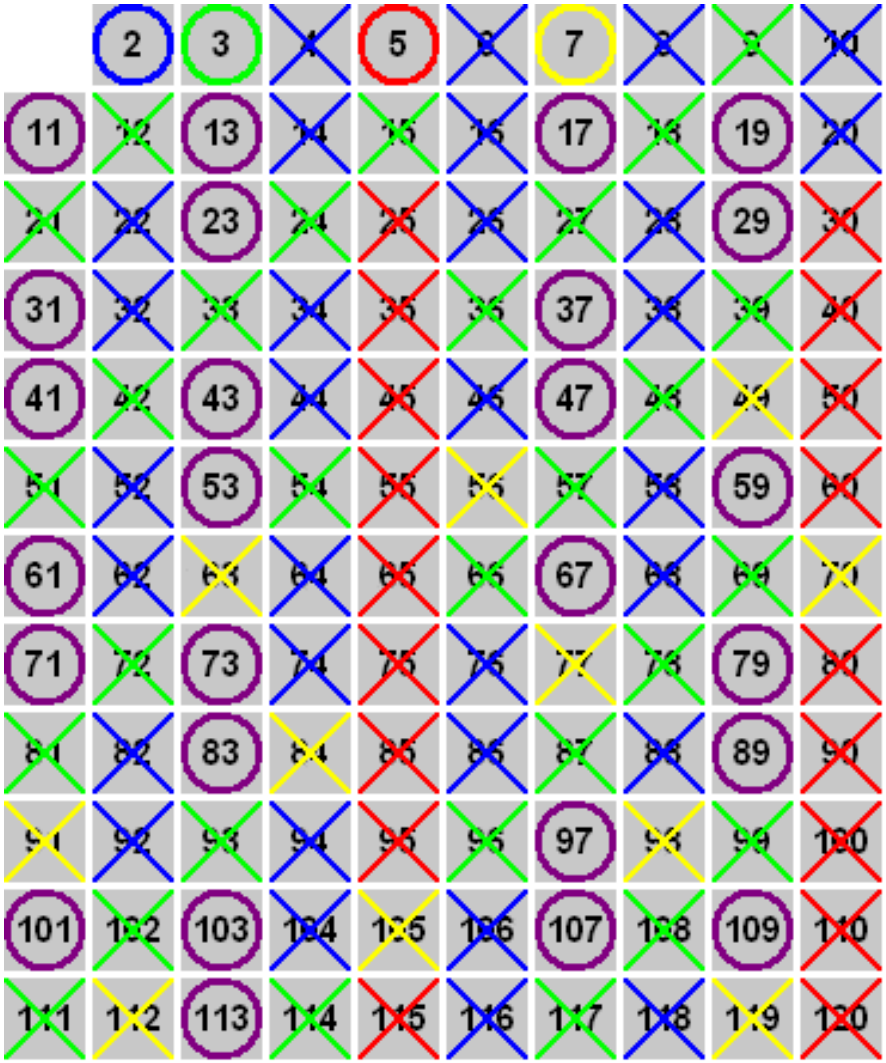


Image source: http://commons.wikimedia.org/wiki/File:New_Animation_Sieve_of_Eratosthenes.gif
Used under the license given there

Part III

Relations, Graphs and Trees

Relations

Graphs

Trees

Relations

Graphs

Trees

Matrices

We assume familiarity from linear algebra with matrices and their properties. We briefly recall some definitions that will be relevant to us.

A matrix with m rows and n columns is called an $m \times n$ matrix and denoted by

$$A = (a_{ij})_{\substack{1 \le i \le m \\ 1 \le j \le n}} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}, \qquad a_{ij} \in \mathbb{R}.$$

The set of all $m \times n$ matrices with real coefficients will be denoted by $\mathrm{Mat}(m \times n, \mathbb{R})$, or by $\mathrm{Mat}(m \times n)$ for short. We will often write $A = (a_{ij})$ if the range of the indices i and j is understood.

Matrices

If $A, B \in \mathrm{Mat}(m \times n)$, $\lambda \in \mathbb{R}$, we define the *component-wise sum* and *scalar multiple*

$$A + B = (a_{ij}) + (b_{ij}) = (a_{ij} + b_{ij}) \in \mathrm{Mat}(m \times n),$$
$$\lambda A = \lambda \cdot (a_{ij}) = (\lambda \cdot a_{ij}) \in \mathrm{Mat}(m \times n).$$

If $A \in \mathrm{Mat}(m \times n)$ and $B \in \mathrm{Mat}(n \times p)$ we define the *matrix product*

$$A \cdot B = (a_{ij}) \cdot (b_{jk}) = (c_{ik}) \in \mathrm{Mat}(m \times p)$$

where

$$c_{ik} := \sum_{j=1}^{n} a_{ij} b_{jk}.$$

The matrix product is associative, but not commutative. If $A \in \mathrm{Mat}(n \times n)$ (i.e., A is a *square matrix*) then $A \cdot A \in \mathrm{Mat}(n \times n)$.

Matrices

For $A \in \mathrm{Mat}(n \times n)$ we define

$$A^k := \underbrace{A \cdot A \cdots A}_{k \text{ times}} \qquad \text{for } k \in \mathbb{Z}_+$$

and

$$A^0 := \mathbb{1}, \qquad \mathbb{1} := (\delta_{ij}).$$

Here

$$\delta_{ij} := \begin{cases} 1 & \text{for } i = j, \\ 0 & \text{for } i \neq j, \end{cases}$$

is the *Kronecker symbol* and $\mathbb{1}$ is called the *unit matrix*.

Matrices

If $A \in \mathrm{Mat}(m \times n)$ we define the transpose of A by

$$A^T = (a_{ij})^T := (a_{ji}) \in \mathrm{Mat}(n \times m).$$

If $A \in \mathrm{Mat}(n \times n)$ and $A = A^T$ we say that A is *symmetric*. If $A = -A^T$ we say that A is *antisymmetric*. The *trace* of $A \in \mathrm{Mat}(n \times n)$ is defined by

$$\mathrm{tr}\, A = \mathrm{tr}(a_{ij}) := \sum_{i=1}^{n} a_{ii}.$$

Zero-One Matrices

In discrete mathematics, we will often use *zero-one* matrices, i.e., matrices whose entries consist only of zeroes and ones. We denote the set of these matrices by $\mathrm{Mat}(m \times n, \mathbb{Z}_2)$ or $\mathrm{Mat}(m \times n, \{0,1\})$.

Let $b_1, b_2 \in \{0,1\}$. Then we define

$$b_1 \wedge b_2 := \min(b_1, b_2) = \begin{cases} 1 & \text{if } b_1 = b_2 = 1, \\ 0 & \text{otherwise,} \end{cases}$$

$$b_1 \vee b_2 := \max(b_1, b_2) = \begin{cases} 0 & \text{if } b_1 = b_2 = 0, \\ 1 & \text{otherwise.} \end{cases}$$

We extend these operations to zero-one matrices by applying them in each component: if $A, B \in \mathrm{Mat}(m \times n, \mathbb{Z}_2)$, then

$$A \wedge B = (a_{ij}) \wedge (b_{ij}) := (a_{ij} \wedge b_{ij}), \tag{3.1.1}$$

$$A \vee B = (a_{ij}) \vee (b_{ij}) := (a_{ij} \vee b_{ij}). \tag{3.1.2}$$

The Boolean Product

We call $A \wedge B$ the *meet* of A and B and $\vee$ the *join* of A and B.

If $A \in \mathrm{Mat}(m \times n, \mathbb{Z}_2)$ and $B \in \mathrm{Mat}(n \times p, \mathbb{Z}_2)$ we define the *boolean (matrix) product*

$$A \odot B = (a_{ij}) \odot (b_{jk}) = (c_{ik}) \in \mathrm{Mat}(m \times p)$$

where

$$\begin{aligned} c_{ik} &:= (a_{i1} \wedge b_{1k}) \vee (a_{i2} \wedge b_{2k}) \vee \cdots \vee (a_{in} \wedge b_{nk}) \\ &= \max_{1 \le j \le n} \min(a_{ij}, b_{jk}). \end{aligned}$$

The boolean product is associative, but not commutative. For $A \in \mathrm{Mat}(n \times n, \mathbb{Z}_2)$ we define

$$A^{[k]} := \underbrace{A \odot A \odot \cdots \odot A}_{k \text{ times}} \qquad \text{for } k \in \mathbb{Z}_+$$

and $A^{[0]} := \mathbb{1}$.

The Boolean Product

3.1.1. Example. Let

$$B = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix}.$$

Then

$$B^{[2]} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \end{pmatrix}, \qquad B^{[3]} = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix},$$

$$B^{[4]} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}, \qquad B^{[5]} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

Now for all $k > 5$, $B^{[k]} = B^{[5]}$.

Relations

We have already defined the concept of a relation on Slide 88 and used equivalence relations to construct the sets of integers and rational numbers. Let us briefly recall some key definitions.

3.1.2. Definition. Let M, N be sets. Then a relation R *from* M *to* N is a subset of $M \times N$.

- If $M = N$, we say that R is a *relation on* M;
- If $(a, a) \in R$ for all $a \in M$ then R is *reflexive*;
- If $(a, b) \in R$ implies $(b, a) \in R$ for all $a, b \in M$ then R is *symmetric*;
- If $(a, b) \in R$ and $(b, c) \in R$ implies $(a, c) \in R$ for all $a, b, c \in M$ then R is *transitive*;
- If $(a, b) \in R$ and $(b, a) \in R$ implies $a = b$ for all $a, b \in M$ then R is *antisymmeric*.

Relations

3.1.3. Example. Define the relation R on $\mathbb{Z}_+$ by $(a, b) \in R \quad \Leftrightarrow a \mid b$. Then

- R is reflexive, because $a \mid a$ for all $a \in \mathbb{Z}$,
- R is not symmetric, because $1 \mid 2$ but $2 \nmid 1$,
- R is transitive, because $a \mid b$ and $b \mid c$ implies $a \mid c$,
- R is antisymmetric, because $a \mid b$ and $b \mid a$ implies $a = b$.

If we define the same relation on $\mathbb{Z}$, then it is not antisymmetric, because $2 \mid (-2)$ and $(-2) \mid 2$.

Since relations from M to N are subsets of $M \times N$, we can combine two relations R_1 and R_2 from M to N in the same way as sets: we can define

$$R_1 \cup R_2, \qquad R_1 \cap R_2 \qquad \text{and} \qquad R_1 \setminus R_2. \tag{3.1.3}$$

Relations

3.1.4. Example. Let R_1, R_2 be two relations on $\{2,3,4,5,6\}$ defined by

$$(a,b) \in R_1 \quad \Leftrightarrow \quad a \mid b, \qquad (a,b) \in R_2 \quad \Leftrightarrow \quad (2 \mid a) \wedge (2 \mid b).$$

Then

$$R_1 = \{(2,2),(2,4),(2,6),(3,3),(3,6),(4,4),(5,5),(6,6)\},$$
$$R_2 = \{(2,2),(2,4),(2,6),(4,2),(4,4),(4,6),(6,2),(6,4),(6,6)\}.$$

and

$$\begin{aligned} R_1 \cup R_2 &= \{(2,2),(2,4),(2,6),(3,3),(3,6),(4,2),(4,4),(4,6),\\ &\qquad (5,5),(6,2),(6,4),(6,6)\},\\ R_1 \cap R_2 &= \{(2,2),(2,4),(2,6),(4,4),(6,6)\},\\ R_1 \setminus R_2 &= \{(3,3),(3,6),(5,5)\},\\ R_2 \setminus R_1 &= \{(4,2),(4,6),(6,2),(6,4)\}. \end{aligned}$$

Composing Relations

We can also find the composition of relations, as we do with functions:

3.1.5. Definition. Let R_1 be a relation from M to N, R_2 a relation from N to P. Then we define the *composition of R_2 with R_1*,

$$R_2 \circ R_1 = \left\{(m,p) \in M \times P\colon \exists_{n \in N} (m,n) \in R_1 \wedge (n,p) \in R_2\right\}.$$

Composing Relations

3.1.6. Example. Let $R_1 = \{(x, y) \in \mathbb{R}^2 : x^2 = y\}$ and $R_2 = \{(x, y) : y^2 = x\}$. Then

$$\begin{aligned} R_2 \circ R_1 &= \left\{(x, y) \in \mathbb{R}^2 : \exists_{z \in \mathbb{R}} (x, z) \in R_1 \wedge (z, y) \in R_2\right\}, \\ &= \left\{(x, y) \in \mathbb{R}^2 : \exists_{z \in \mathbb{R}} x^2 = z \wedge z = y^2\right\} \\ &= \left\{(x, y) \in \mathbb{R}^2 : x^2 = y^2\right\} \end{aligned}$$

Composing Relations

If R is a relation on a set M, we can compose it with itself. In general, we define:

3.1.7. Definition. Let R be a relation on a set M. Then

$$R^1 := R, \qquad R^{n+1} := R^n \circ R, \qquad n \in \mathbb{Z}_+.$$

3.1.8. Examples.

1. Let $R = \{(x, y) \in \mathbb{R}^2 : x^2 = y\}$. Then

$$\begin{aligned} R^2 = R \circ R &= \left\{(x, y) \in \mathbb{R}^2 : \exists_{z \in \mathbb{R}} (x, z) \in R \wedge (z, y) \in R\right\}, \\ &= \left\{(x, y) \in \mathbb{R}^2 : \exists_{z \in \mathbb{R}} x^2 = z \wedge z^2 = y\right\} \\ &= \left\{(x, y) \in \mathbb{R}^2 : x^4 = y\right\} \end{aligned}$$

Composing Relations

2. Let $R = \{(a, b) \in \mathbb{N}^2 : a > b\}$. Then

$$\begin{aligned} R^2 &= \left\{(a, b) \in \mathbb{N}^2 : \exists_{c \in \mathbb{N}} (a, c) \in R \wedge (c, b) \in R\right\}, \\ &= \left\{(a, b) \in \mathbb{N}^2 : \exists_{c \in \mathbb{N}} a > c \wedge c > b\right\} \\ &= \left\{(a, b) \in \mathbb{N}^2 : a > b + 2\right\} \\ &\subsetneq R \end{aligned}$$

3. Let $R = \{(a, b) \in \mathbb{N}^2 : a \geq b\}$. Then

$$\begin{aligned} R^2 &= \left\{(a, b) \in \mathbb{N}^2 : \exists_{c \in \mathbb{N}} (a, c) \in R \wedge (c, b) \in R\right\}, \\ &= \left\{(a, b) \in \mathbb{N}^2 : \exists_{c \in \mathbb{N}} a \geq c \wedge c \geq b\right\} \\ &= \left\{(a, b) \in \mathbb{N}^2 : a \geq b\right\} \\ &= R \end{aligned}$$

Composing Relations

3.1.9. Theorem. Let R be a relation on a set M. Then R is transitive if and only if $R^n \subset R$ for all $n \in \mathbb{Z}_+$.

Proof.

($\Rightarrow$) Suppose that R is transitive. We show by induction that $R^n \subset R$ for $n \in \mathbb{Z}_+$. Note that (trivially) this is true for $n = 1$. Now suppose that $R^n \subset R$ and let $(a, b) \in R^{n+1}$. Then there exists a $c \in M$ such that $(a, c) \in R^n$ and $(c, b) \in R$. Since we assumed that $R^n \subset R$ we have $(a, c) \in R$. Since R is transitive, it follows that $(a, b) \in R$. Hence, $R^n \subset R$.

($\Leftarrow$) Suppose that $R^2 \subset R$. Let $(a, b), (b, c) \in R$. Then $(a, c) \in R^2 \subset R$, so R is transitive. □

Representing Relations as Matrices

Suppose M and N are finite sets with card $M = p$, card $N = q$ and that R is a relation from M to N. Label the elements of M and N so that

$$M = \{m_1, \ldots, m_p\}, \qquad N = \{n_1, \ldots, n_q\}.$$

Then we can represent $R \subset M \times N$ through the numbers

$$r_{ij} = \begin{cases} 1 & \text{if } (m_i, n_j) \in R, \\ 0 & \text{otherwise.} \end{cases}$$

and write $R = (r_{ij})$.

3.1.10. Example. Let $M = \{1, 2, 3, 4, 5\}$ and R the relation on M defined by $(a, b) \in R \Leftrightarrow a \mid b$. Then

$$R = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Representing Relations as Matrices

It is easily seen that a relation R on a set M is symmetric if and only if $R = R^T$, i.e., the matrix representing R is symmetric. Furthermore, R is antisymmetric if and only if $\min(r_{ij}, r_{ji}) = 0$ whenever $i \neq j$.

R on M is reflexive if $r_{ii} = 1$ for all i which is equivalent to $\operatorname{tr} R = \operatorname{card} M$.

If $R_1, R_2 \subset M \times N$ are two relations then we have

$$R_1 \cup R_2 = R_1 \vee R_2, \qquad R_1 \cap R_2 = R_1 \wedge R_2,$$

where the left-hand sides of the equations represents the union of the relations (see (3.1.3)) and the right-hand sides represent the join and meet of the matrices (see (3.1.1)). In particular,

$$R_1 \subset R_2 \quad \Leftrightarrow \quad R_1 \cup R_2 = R_2 \quad \Leftrightarrow \quad R_1 \vee R_2 = R_2. \tag{3.1.4}$$

Representing Relations as Matrices

3.1.11. Lemma. Let R_1 be a relation from M to N, R_2 a relation from N to P. Then

$$R_2 \circ R_1 = R_1 \odot R_2, \tag{3.1.5}$$

where the left-hand side is the composition of relations and the right-hand side is the boolean product of the matrices.

The proof is left to the exercises.

3.1.12. Example. Let $M = \{1, 2, 3\}$ and $R_1 = \{(1, 2), (1, 3), (2, 3)\}$, $R_2 = \{(2, 1), (2, 2), (3, 1)\}$. Then

$$R_2 \circ R_1 = \{(1, 1), (1, 2), (2, 1)\}.$$

Writing the relations as matrices,

$$R_1 = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}, \qquad R_2 = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}.$$

Representing Relations as Matrices

Then

$$R_1 \odot R_2 = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

which is the matrix for $R_2 \circ R_1$.

From (3.1.5), (3.1.4) and Theorem 3.1.9 we obtain the following result:

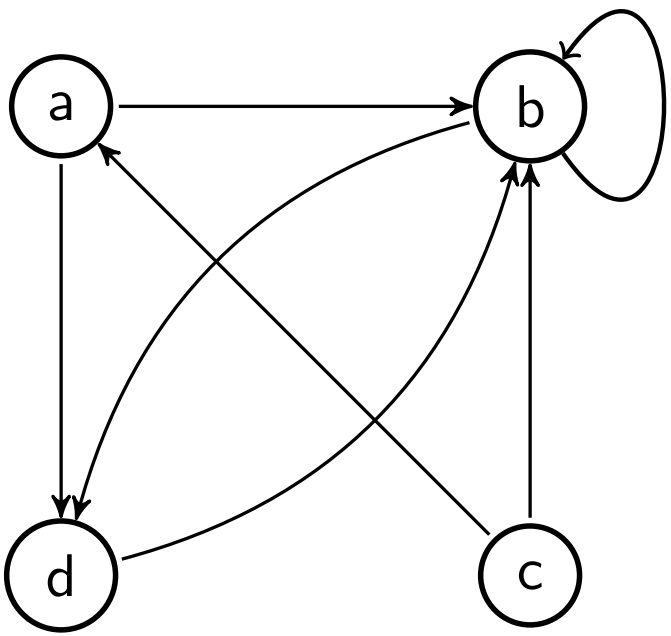
3.1.13. Lemma. Let R be a relation on a finite set M. Then R is transitive if and only if

$$R^{[n]} \vee R = R$$

for all $n \in \mathbb{N}$.

Digraphs

3.1.14. Definition. A *directed graph* or *digraph* $G = (V, E)$ consists of a set V of *vertices* (or *nodes*) together with a set E of ordered pairs of elements of V (i.e., $V \subset E^2$) called *edges* (or *arcs*). An edge (a, b) is said to have *initial vertex* a and *terminal vertex* b.



3.1.15. Example. The figure at left shows the directed graph with vertices $V = \{a, b, c, d\}$ and edges

$$E = \{(a, b), (a, d), (b, b), (b, d), (c, a), (c, b), (d, b)\}.$$

Representing Relations as Digraphs

Directed Graphs can be used to represent a relation R on a set M by taking $V = M$ as the vertices and $E = R$ as the edges.

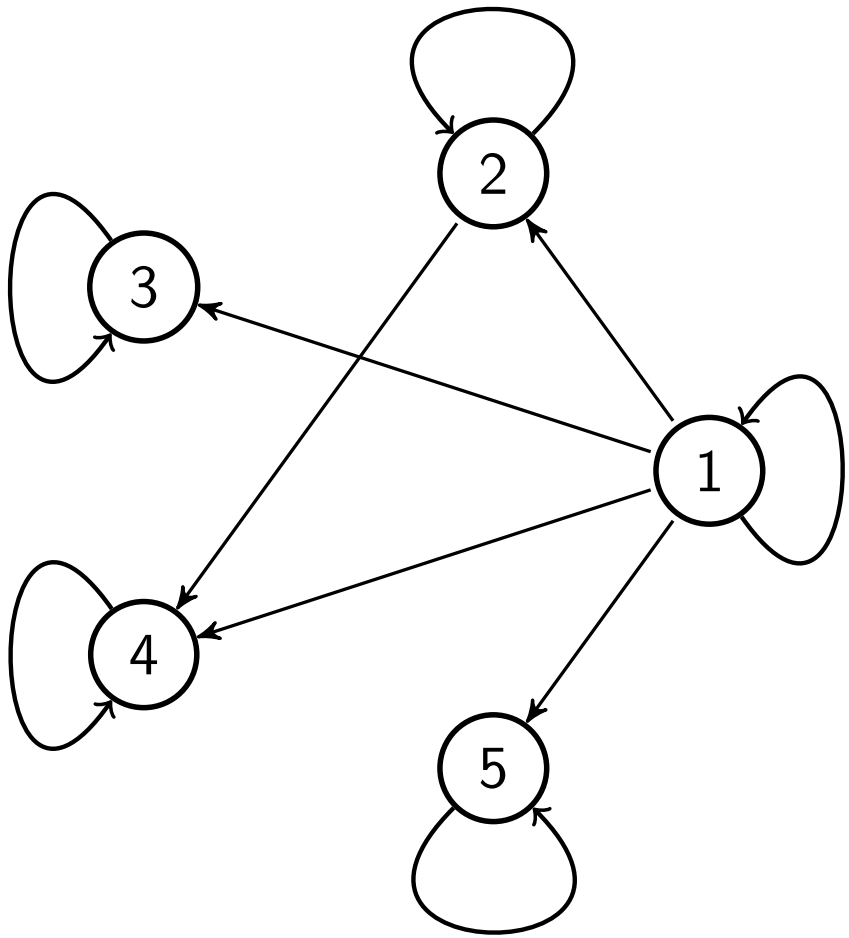
3.1.16. Example. Consider the relation of Example 3.1.10 on

$$M = \{1, 2, 3, 4, 5\}$$

given by

$$(a, b) \in R \quad \Leftrightarrow \quad a \mid b.$$

The figure at right shows R as a directed graph with vertices $V = M$ and edges $E = R$.



Closures of Relations

When a relation R does not have a certain property (e.g., symmetry) then we may be able to add points to the relation to generate a new relation $\widetilde{R} \supset R$ that does have this property. The smallest relation $\widetilde{R}$ that then has this property is called the *closure of* R with respect to that property.

3.1.17. Example. Let $M = \{1, 2, 3\}$ and consider the relation R defined by $(a, b) \in R \Leftrightarrow a < b$. Here,

$$R = \{(1, 2), (1, 3), (2, 3)\}.$$

This relation is not reflexive, but we can add the missing points:

$$\widetilde{R} = \{(1, 1), (1, 2), (1, 3), (2, 2), (2, 3), (3, 3)\}$$

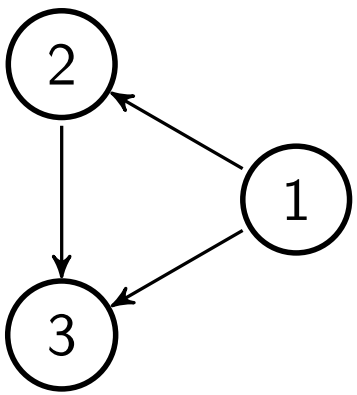
is the *reflexive closure* of R. On the other hand, R is not symmetric. By adding other points, we obtain

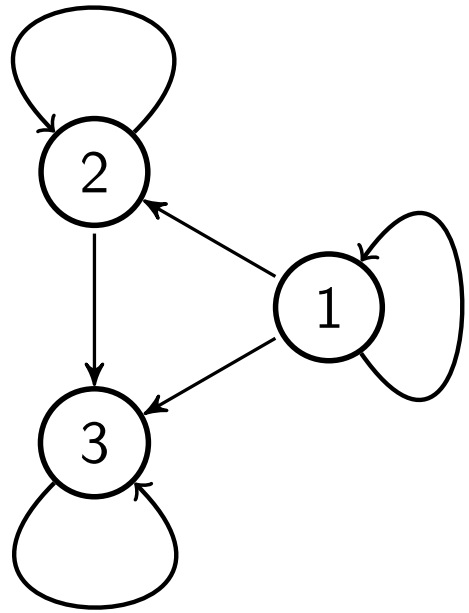
$$\widetilde{\widetilde{R}} = \{(1, 2), (1, 3), (2, 1), (2, 3), (3, 1), (3, 2)\},$$

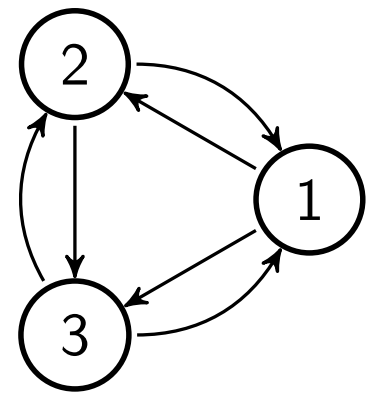
the *symmetric closure* of R.

Closures of Relations

Below, we show the digraphs for R, the reflexive closure $\widetilde{R}$ and the symmetric closure $\widetilde{\widetilde{R}}$ of R







The graphs show that it is not difficult to construct the symmetric and reflexive closure of a relation.

Closures of Relations

3.1.18. Definition. Let M be a set and R a relation on M. Let $\widetilde{R}$ be a relation on M such that

1. $\widetilde{R}$ has a certain property,
2. $R \subset \widetilde{R}$,
3. any relation S that has the certain property and contains R (i.e., $R \subset S$) also contains $\widetilde{R}$.

Then $\widetilde{R}$ is called the closure of R with respect to this property.

If M is a set we define the *diagonal relation*

$$\Delta := \{(a, a) \in M^2 : a \in M\}.$$

For $R \in M^2$ we define

$$R^{-1} := \{(a, b) \in M^2 : (b, a) \in R\}.$$

Closures of Relations

3.1.19. Lemma. Let M be a set and R a relation on M. Then

- $R \cup \Delta$ is the reflexive closure of R,
- $R \cup R^{-1}$ is the symmetric closure of R.

We now turn to the question of *transitive closures*. Given a relation R, what is the smallest relation $\widetilde{R}$ containing R such that $\widetilde{R}$ is transitive?

3.1.20. Example. Let $M = \{1, 2, 3, 4\}$ and

$$R = \{(1, 3), (1, 4), (2, 1), (3, 2)\}.$$

This relation is not transitive, because $(2, 1) \in R$ and $(1, 3) \in R$ but $(2, 3) \notin R$. So let's add the "missing" pairs:

$$\widetilde{R} = \{(1, 3), (1, 4), (2, 1), (3, 2), (1, 2), (2, 3), (2, 4), (3, 1)\}.$$

But $\widetilde{R}$ is still not transitive, because it contains $(3, 1)$ and $(1, 4)$, but not $(3, 4)$! This shows that finding the transitive closure of a relation is more complicated than finding the symmetric or reflexive closure.

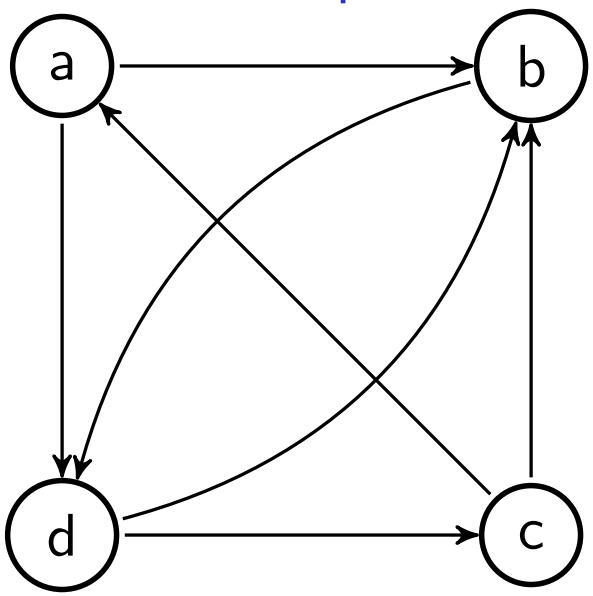
Paths in Directed Graphs

3.1.21. Definition. Let $G = (V, E)$ be a directed graph. Then a tuple of edges $(e_1, \ldots, e_n)$, $e_j \in E$, $j = 1, \ldots, n$, $n \in Z_+$ is called a *path of length n from a to b* if

- $e_j = (x_{j-1}, x_j)$, $j = 1, \ldots, n$ and
- $x_0 = a$ and $x_n = b$.

We also denote this path by writing $(x_0, \ldots, x_n)$. For any $a \in V$ we define the empty set as the *path of length* 0 *from a to a*.

3.1.22. Example.



Consider the graph shown at left. Then a path from a to b is given, for example, by (a, d, c, b, d, b) corresponding to the edges $\big((a, d), (d, c), (c, b), (b, d), (d, b)\big)$.

The tuple (a, b, c) is not a path, because (b, c) is not an edge of the digraph.

Paths in Relations

We define a path in a relation on a set analogously to a path in a directed graph:

3.1.23. Definition. Let M be a set, R a relation on M and $G = (M, R)$ the graph associated to the relation. Then a tuple $(x_0, \ldots, x_n)$ is said to be a path in R if it is a path in G.

3.1.24. Theorem. Let M be a set, R a relation on M and $a, b \in M$. Then there is a path of length n from a to b in R if and only if $(a, b) \in R^n$.

Proof.
We prove the statement by induction. For $n = 1$, a path of length $n = 1$ from a to b is the tuple (a, b). It exists if and only if $(a, b) \in R^1 = R$, so the statement is true for $n = 1$.

The Connectivity Relation

Proof (continued).

Now assume that the statement is true for n. There exists a path of length $n+1$ from a to b if and only there exists a $c \in \mathbb{R}$ such that there is a path of length 1 from a to c and a path of length n from c to b. By our inductive hypothesis, this is the case if and only if $(a, c) \in R$ and $(c, b) \in R^n$. But there exists such a c if and only if $(a, b) \in R \circ R^n = R^{n+1}$. □

3.1.25. Definition. Let M be a set and R a relation on M. Then the *connectivity relation for* R is the relation R^* defined by

$$(a, b) \in R^* \quad \Leftrightarrow \quad \text{there exists a path in } R \text{ joining } a \text{ to } b. \tag{3.1.6}$$

It follows directly from Theorem 3.1.24 that

$$R^* = \bigcup_{k=1}^{\infty} R^k. \tag{3.1.7}$$

The Transitive Closure

3.1.26. Theorem. Let M be a set and R a relation on M. Then R^* is the transitive closure of R.

Proof.

We show that R^* is transitive and that every transitive relation that contains R also contains R^*. Suppose that $(a, b), (b, c) \in R^*$. Then there exists a path from a to b and a path from b to c in R. Joining these two paths, we obtain a path from a to c, so $(a, c) \in R^*$. Hence, R^* is transitive.

Suppose S is a transitive relation and $R \subset S$. Then, by Theorem 3.1.9, $S^n \subset S$. It follows that

$$S^* = \bigcup_{k=1}^{\infty} S^k \subset S.$$

If $R \subset S$ then $R^* \subset S^*$, because any path in R is also a path in S. Hence, $R^* \subset S^* \subset S$. □

The Transitive Closure

Since we have now established that R^* is the transitive closure of R, we will now turn to computing the connectivity relation in practice.

3.1.27. Lemma. Let M be a set with cardinality $\operatorname{card} M = n$, $a, b \in M$ and R a relation on M. If there is a path in R of length at least one connecting a to b, then there is also a path of length not exceeding n connecting a to b.

Furthermore, if $a \neq b$ and there is a path of length at least one connecting a and b, then there is such a path of length not exceeding $n - 1$.

3.1.28. Corollary. Let M be a set with cardinality $\operatorname{card} M = n$ and R a relation on M. From Theorem 3.1.24, Lemma 3.1.27 and (3.1.6) it follows that

$$R^* = \bigcup_{k=1}^{n} R^k. \tag{3.1.8}$$

The Transitive Closure

Proof of Lemma 3.1.27.

Suppose there is at least one path connecting a and b. Assume that the length of the shortest such path is m, so that the path is given by $(x_0, x_1, \ldots, x_m)$ with $x_0 = a$ and $x_m = b$.

Assume that $a = b$ and $m \geq n + 1$. Since card $M = n$, by the Pigeonhole Principle at least two vertices in the path are equal. Suppose that $x_i = x_j$ with $0 \leq i < j \leq m - 1$. Then the path contains a circuit (sub-path) from x_i to itself. This circuit can be deleted to give a path $(x_0, \ldots, x_i, x_{j+1}, \ldots, x_m)$ of shorter length. This process can be repeated until we obtain a path from a to b of length $m \leq n$.

The case $a \neq b$ is left to the reader. □

The Transitive Closure

Let M be a set with cardinality $\operatorname{card} M = n$ and R a relation on M. Representing R as a matrix, we obtain the transitive closure through

$$R^* = R \vee R^{[2]} \vee \cdots \vee R^{[n]}.$$

3.1.29. Example. Let $M = \{1, 2, 3\}$ and $R = \{(1,1), (1,3), (2,2), (3,1), (3,2)\}$. To find the transitive closure of R, we compute

$$R^* = R \vee R^{[2]} \vee R^{[3]} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{pmatrix} \vee \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} \vee \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

so $R^* = R \cup \{(1,2), (3,3)\}$ is the transitive closure of R.

The Transitive Closure

3.1.30. Example. Let $M = \{1, 2, 3\}$ and $R = \{(1, 3), (2, 1), (3, 1), (3, 2)\}$. To find the transitive closure of R, we compute

$$R^* = R \vee R^{[2]} \vee R^{[3]} = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix} \vee \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \end{pmatrix} \vee \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

so $R^* = R \cup \{(1, 2), (1, 1), (2, 2), (2, 3), (3, 3)\}$.

Computing the transitive closure in this way uses $O(n^4)$ bit operations. There is a more efficient algorithm, called Warshall's algorithm, that uses only $O(n^3)$ operations. The algorithm is discussed in the textbook, and you are encouraged to have a look at it there.

Partial Orderings

3.1.31. Definition. let M be a set and R a relation on M that is reflexive, transitive and antisymmetric. Then R is called a *partial ordering* on M and the pair (M, R) is called a *partially ordered set* or *poset*.

3.1.32. Examples.

1. The relation $\leq$ is a partial ordering on $\mathbb{Z}$. Here "the relation $\leq$" is supposed to mean "the set of all pairs $(a, b) \in \mathbb{Z}^2$ such that $a \leq b$." We will use this type of shorthand throughout this section.
2. Let M be a set. The relation $\subset$ is a partial ordering on the power set $\mathcal{P}(M)$.
3. The relation $|$ is a partial ordering on $\mathbb{Z}_+$.

Total Orderings

Given a poset M we will use the general symbol $\preccurlyeq$ to indicate the ordering relation if no specific set/relation (as in the above examples) is provided. Thus, a poset is denoted $(M, \preccurlyeq)$.

3.1.33. Definition. Let $(M, \preccurlyeq)$ be a poset.

1. Let $a, b \in M$. If $a \preccurlyeq b$ or $b \preccurlyeq a$, then a and b are called *comparable*, otherwise they are called *incomparable*.
2. If any two elements of M are comparable we say that $(M, \preccurlyeq)$ is a *totally ordered* or *linearly ordered* set and the relation $\preccurlyeq$ is called a *total* or *linear* ordering. A totally ordered set is also called a *chain*.

3.1.34. Examples.

1. The relation $\leq$ is a total ordering on $\mathbb{Z}$.
2. Let M be a set. The relation $\subset$ is not a total ordering on the power set $\mathcal{P}(M)$.
3. The relation $|$ is not a total ordering on $\mathbb{Z}_+$.

Well-Ordered Sets

3.1.35. Definition. Let $(M, \preccurlyeq)$ be a poset. Then $(M, \preccurlyeq)$ is called a *well-ordered set* if $\preccurlyeq$ is a total ordering and every non-empty subset of M has a least element.

3.1.36. Examples.

1. The poset $(\mathbb{Z}, \leq)$ is not well-ordered, because it does not have a least element.
2. The poset $(\mathbb{N}, \leq)$ is well-ordered.
3. The poset $(\mathbb{N}^2, \preccurlyeq)$ with

$$(a_1, a_2) \preccurlyeq (b_1, b_2) \quad \Leftrightarrow \quad (a_1 < b_1) \vee \big((a_1 = b_1) \wedge (a_2 \leq b_2)\big)$$

is well-ordered. (This is called the *lexicographic ordering*.)

Well-Ordered Induction

Recall that the principle of induction on the natural numbers is equivalent to the well-ordering principle. It seems plausible that a version of induction should hold on general well-ordered sets, and this is indeed the case.

If $(M, \preccurlyeq)$ is a poset and $a, b \in M$, we define

$$a \prec b \qquad :\Leftrightarrow \qquad (a \preccurlyeq b) \wedge (a \neq b).$$

3.1.37. Theorem. Suppose that $(M, \preccurlyeq)$ is a well-ordered set. Then the predicate $P(x)$ is true for all $x \in M$ if

$$\forall_{y \in M} \quad \left(\forall_{x \prec y} P(x) \right) \Rightarrow P(y) \tag{3.1.9}$$

Well-Ordered Induction

Proof.

Suppose that (3.1.9) is true and that there exists some $z \in M$ so that $P(z)$ is false. Then the set $A = \{x \in M : \neg P(x)\}$ is non-empty. Since $(M, \preccurlyeq)$ is well-ordered, there exists a least element a of A. Since a is the least element of A, it follows that $P(x)$ is true for all $x \prec a$. But then by (3.1.9) $P(a)$ must be true. This gives a contradiction. □

3.1.38. Remark. For this type of induction, we do not need a "basis step", i.e., we do not need to show that $P(x)$ is true for some initial element of M. For example, suppose that a is the least element of M. Then the statement $\underset{x \prec a}{\forall} P(x)$ in (3.1.9) is vacuously true (there is no element $x \prec a$) so once (3.1.9) has been established, $P(a)$ is true automatically.

Well-Ordered Induction

3.1.39. Example. Suppose that for $n, m \in \mathbb{N}$ we define

$$a_{m,n} := \begin{cases} 0 & \text{if } n = m = 0, \\ a_{m-1,n} + 1 & \text{if } n = 0 \text{ and } m > 0, \\ a_{m,n-1} + n & \text{if } n > 0. \end{cases}$$

We want to prove that

$$a_{m,n} = m + \frac{n(n+1)}{2} \qquad m, n \in \mathbb{N}. \tag{3.1.10}$$

We can use the lexicographic ordering on $\mathbb{N}^2$ introduced in Example 3.1.36 to apply well-ordered induction. In particular,

$$(m, n) \prec (m_0, n_0) \quad \Leftrightarrow \quad (m < m_0) \vee \big((m = m_0) \wedge (n < n_0)\big).$$

Take $(m_0, n_0) \in \mathbb{N}^2$ and suppose that (3.1.10) holds for all $(m, n) \prec (m_0, n_0)$.

Well-Ordered Induction

1. If $(m_0, n_0) = (0, 0)$ the statement follows immediately.
2. Consider the case $n_0 = 0$ and $m_0 > 0$. Then $a_{m_0,n_0} = a_{m_0-1,n_0} + 1$. Since $(m_0 - 1, n_0) \prec (m_0, n_0)$, we have

$$a_{m_0,n_0} = a_{m_0-1,n_0} + 1 = m_0 - 1 + \frac{n_0(n_0+1)}{2} + 1 = m_0 + \frac{n_0(n_0+1)}{2}$$

   so (3.1.10) is true for $(m, n) = (m_0, n_0)$ in this case.
3. Now consider the case $n_0 > 0$. Then $a_{m_0,n_0} = a_{m_0,n_0-1} + n_0$. Since $(m_0, n_0 - 1) \prec (m_0, n_0)$, we have

$$a_{m_0,n_0} = a_{m_0,n_0-1} + n_0 = m_0 + \frac{n_0(n_0-1)}{2} + n_0 = m_0 + \frac{n_0(n_0+1)}{2}$$

   so (3.1.10) is true for $(m, n) = (m_0, n_0)$ in this case also.

This completes the proof by induction.

General Lexicographic Ordering

We can define a lexicographic ordering on the cartesian product of n posets as follows:

3.1.40. Definition. Let $(A_1, \preccurlyeq_1), \ldots, (A_n, \preccurlyeq_n)$ be a family of n posets. Then the set $A_1 \times \cdots \times A_n$ together with the ordering relation

$$(a_1, \ldots, a_n) \preccurlyeq (b_1, \ldots, b_n) \quad :\Leftrightarrow \quad \Big(\underset{1 \le i \le n}{\forall} a_i = b_i \Big) \vee \big(a_1 \prec_1 b_1 \big) \vee \Big(\underset{j \in \{2,\ldots,n\}}{\exists} a_j \prec_j b_j \wedge \underset{i<j}{\forall} a_i = b_i \Big). \tag{3.1.11}$$

becomes a poset. The ordering relation (3.1.11) is called the *lexicographic ordering* for the cartesian product $A_1 \times \cdots \times A_n$.

If $A_1 = A_2 = \cdots = A_n$ and we have have the same partial order $\preccurlyeq$ on each A_k, we use the same symbol $\preccurlyeq$ for the partial order on the cartesian product $A_1 \times \cdots \times A_n$ given by (3.1.11).

Lexicographic Ordering of Strings

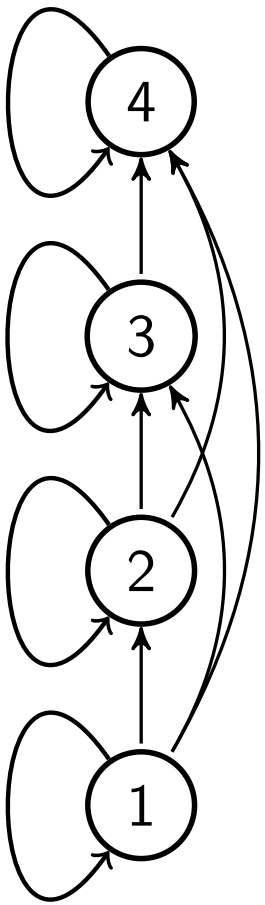
3.1.41. Definition. Let $\mathcal{A}$ be an alphabet and $\preccurlyeq$ an ordering relation on the alphabet. Let $a = a_1 a_2 \dots a_m$ and $b = b_1 b_2 \dots b_n$ be strings of length n and m from $\mathcal{A}$. Let $t = \min(m, n)$. Then we define

$$a \prec b \quad :\Leftrightarrow \quad \big((a_1, \dots, a_t) = (b_1, \dots, b_t) \wedge m < n\big) \\ \vee \big((a_1, \dots, a_t) \prec (b_1, \dots, b_t)\big).$$

3.1.42. Example. Let $\mathcal{A} = \{0, 1\}$ with the partial ordering $\preccurlyeq$ induced by $0 \prec 1$. Consider the bit strings $a = 110011$, $b = 11001$, $c = 1100001$, $d = 1110$. Then $c \prec b \prec a \prec d$.

Hasse Diagrams

Consider the digraph for the poset $(\{1, 2, 3, 4\}, \leq)$:

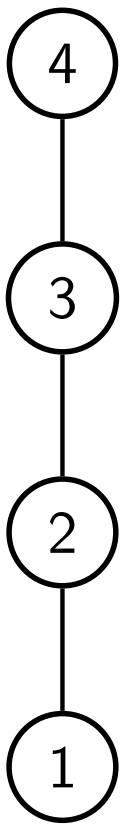


If we know that this is the graph of a partial ordering, we do not need to show all the edges:

- Since $\leq$ is reflexive, we know that loops must be present and we can omit them,
- Since $\leq$ is transitive, we can omit the edges that must be present due to transitivity.
- If we agree that the arrows always point upwards, we can omit the arrowheads.

Hasse Diagrams

Applying these principles we end up with the following graph, called a *Hasse diagram*:

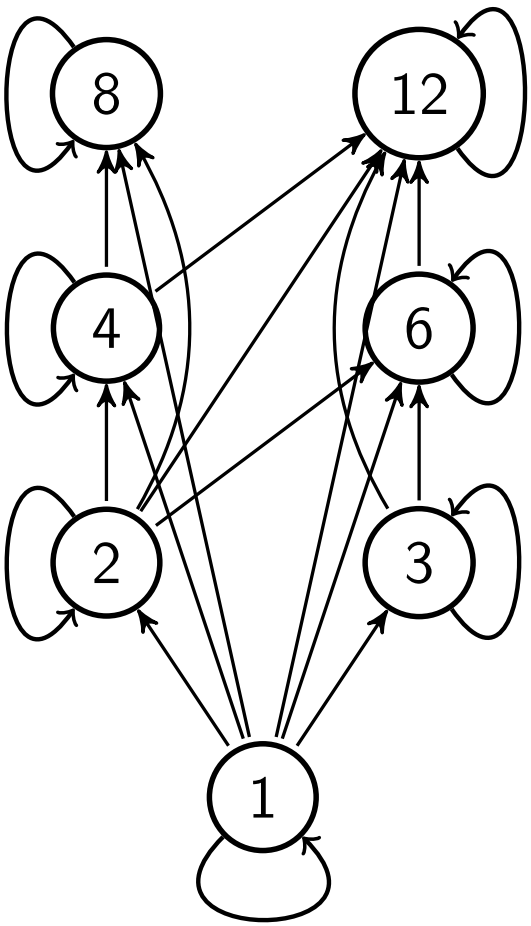


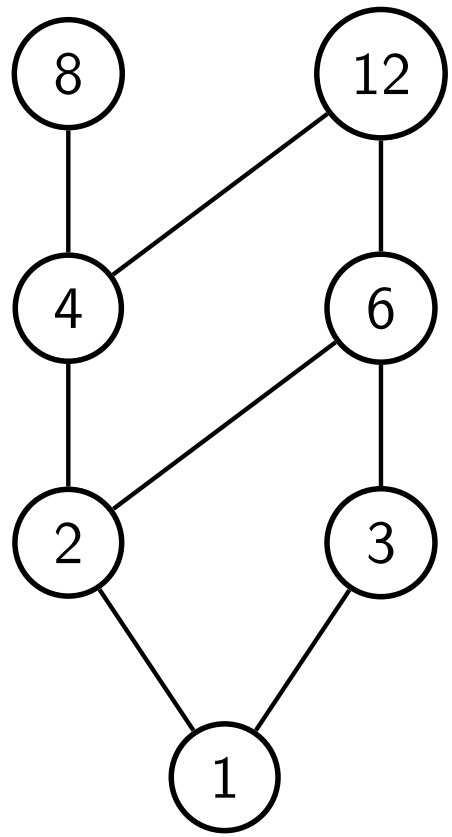
We can perform this basic procedure with any graph of an ordering relation. It is easy to see that the original graph of the partial ordering relation can be regained from the Hasse diagram, so no information is lost by simplifying the graph in this way.

The Hasse diagran can be very useful to identify basic properties of the relation

Hasse Diagrams

3.1.43. Example. Consider the poset $(\{1, 2, 3, 4, 6, 8, 12\}, |)$. Then the graph and the Hasse diagram of the relation are





Maximal and Minimal Elements

3.1.44. Definition. Let $(M, \preccurlyeq)$ be a poset and suppose $a \in M$. We say that

1. The element a is *maximal* if there exists no $b \in M$ such that $a \prec b$.
2. The element a is *minimal* if there exists no $b \in M$ such that $b \prec a$.
3. The element a is the *greatest element of* M if $b \preccurlyeq a$ for all $b \in M$.
4. The element a is the *least element of* M if $a \preccurlyeq b$ for all $b \in M$.

3.1.45. Remark. The greatest and least elements, if they exist, are unique (see exercises).

3.1.46. Example. Consider the poset $(\{1, 2, 3, 4, 6, 8, 12\}, |)$ of Example 3.1.43. There,

- The element 1 is minimal;
- The elements 8 and 12 are maximal;
- The element 1 is the least element;
- There is no greatest element.

Subsets and Bounds

3.1.47. Definition. Let $(M, \preccurlyeq)$ be a poset and suppose that $A \subset M$.

1. An element $u \in M$ such that $a \preccurlyeq u$ for all $a \in A$ is is called an *upper bound for* A.
2. An element $l \in M$ such that $l \preccurlyeq a$ for all $a \in A$ is is called a *lower bound for* A.
3. An upper bound x for A such that $x \preccurlyeq u$ for all upper bounds u of A is called a *least upper bound for* A. We then write $x = \sup A$.
4. A lower bound x for A such that $l \preccurlyeq x$ for all lower bounds l of A is called a *greatest lower bound for* A. We then write $x = \inf A$.

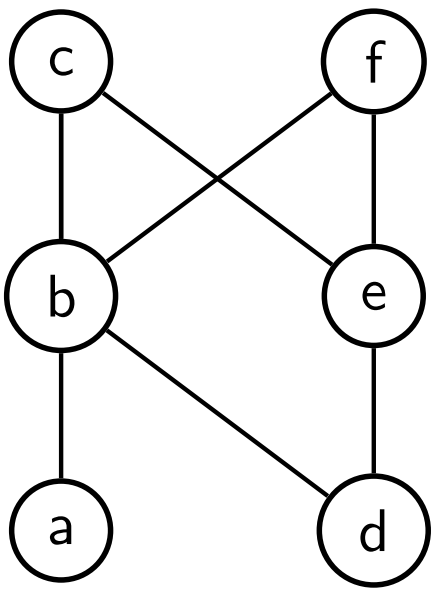
3.1.48. Remark. The greatest and least upper bounds, if they exist, are unique (see exercises).

Subsets and Bounds

3.1.49. Example. Consider the poset $(\{1, 2, 3, 4, 6, 8, 12\}, |)$ of Example 3.1.43 and let $A = \{1, 2, 3\}$. Then

- The elements 6 and 12 are upper bounds for A;
- The element 1 is the (only) lower bound for A;
- $\inf A = 1$, $\sup A = 6$.

3.1.50. Example. Consider the poset with Hasse diagram as shown below:



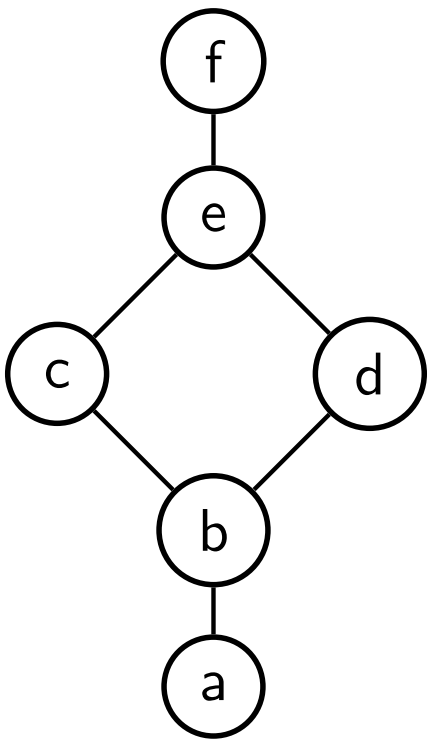
The subset $A = \{a, b, e\}$ has two upper bounds, c and f, but no least upper bound.

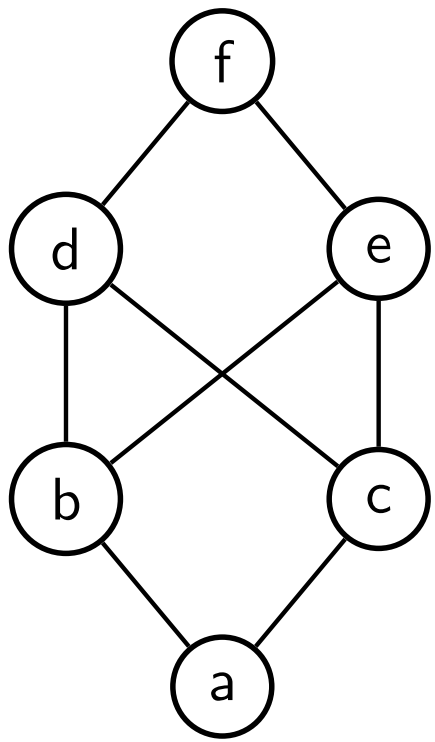
There is no lower bound for A.

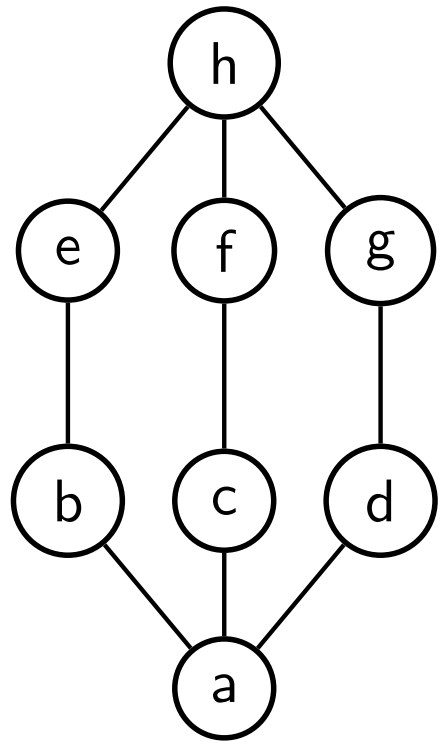
Lattices

3.1.51. Definition. Let $(M, \preccurlyeq)$ be a poset and suppose that every pair of elements of M has a greatest lower and a least upper bound. Then M is called a *lattice*.

3.1.52. Example. The left and right diagrams are lattices, the middle one is not ($\sup\{b, c\}$ does not exist) .







Topological Sorting

Lattices are used in problems involving information flow - we will study some examples in the exercises.

We now turn to another problem: Let $(M, \preccurlyeq)$ be a poset which is not totally ordered. A total ordering $\preccurlyeq_t$ such that $a \preccurlyeq_t b$ whenever $a \preccurlyeq b$ is said to be *compatible* with $\preccurlyeq$. The problem is to find a compatible total ordering for a poset $(M, \preccurlyeq)$. This is called *topological sorting* of $(M, \preccurlyeq)$. In our discussion, the following result will be sueful.

3.1.53. Lemma. Let $(M, \preccurlyeq)$ be a finite, non-empty poset. Then there exists a minimal element of M.

Proof.

Choose some $a \in M$. If a is minimal, we are finished. If it is not minimal, there exists some $a_1 \prec a$. We repeat this process with a_1. Since M is finite, we must obtain a minimal element after a finite number of iterations. □

Topological Sorting

We now give a topological sorting algorithm as follows: Let $(M, \preccurlyeq)$ be a finite, non-empty poset.

1. Find a minimal element a_1 of $(M, \preccurlyeq)$.
2. Define $(M', \preccurlyeq')$, where $M' = M \setminus \{a_1\}$ and $\preccurlyeq'$ actually is the restriction of $\preccurlyeq$ from M to M'. This is also a poset.
3. If $M' \neq \emptyset$, repeat Step 1 with $(M, \preccurlyeq)$ replaced by $(M', \preccurlyeq')$, obtaining a least element a_2 of M'.

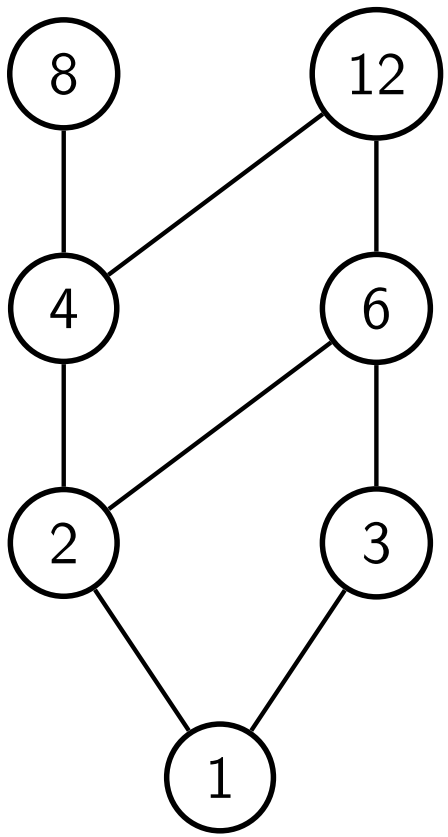
The above loop continues until $M = \emptyset$. We obtain a sequence of elements $a_1, \ldots, a_n$, $n = \operatorname{card} M$, and define a total ordering $\preccurlyeq_t$ by

$$a_1 \prec_t a_2 \prec_t a_3 \prec_t \cdots \prec_t a_n.$$

It is easy to see that $\preccurlyeq_t$ is compatible with $\preccurlyeq$: if $a \preccurlyeq b$, then a will have been selected before b, so $a \preccurlyeq_t b$.

Topological Sorting

3.1.54. Example. Consider the poset $(\{1, 2, 3, 4, 6, 8, 12\}, |)$ of Example 3.1.43. Then a total ordering can be found by successively choosing the least (bottom) elements from the Hasse diagram:



One possible total ordering is given by

$$1 \prec_t 2 \prec_t 3 \prec_t 4 \prec_t 6 \prec_t 8 \prec_t 12$$

Another possibility is

$$1 \prec_t 3 \prec_t 2 \prec_t 6 \prec_t 4 \prec_t 12 \prec_t 8$$

Relations

Graphs

Trees

Graphs

The definition of a graph is similar to that of a directed graph (Definition 3.1.14) with the modification that no "direction" of the edges is defined.

3.2.1. Definition. An *(undirected) graph* $G = (V, E)$ consists of a nonempty set V of *vertices* (or *nodes*) together with a collection E of one or two-element subsets of V called *edges*. An edge $\{a, b\}$ is said to *connect* the *endpoints* a and b.

If card $V < \infty$ we say that G is a *finite graph*, otherwise an infinite graph.

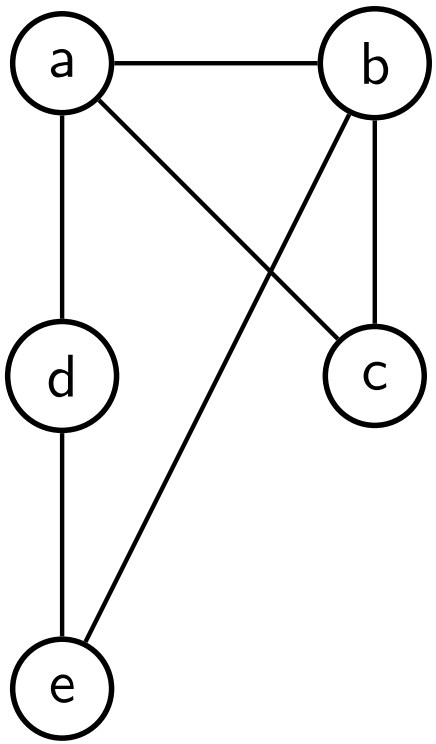
If E contains every edge $\{a, b\}$, $a, b \in V$, at most once and does not contain edges $\{a, a\}$ (i.e., E is a set of two-element subsets of $\mathcal{P}(V)$) we say that G is a *simple graph*.

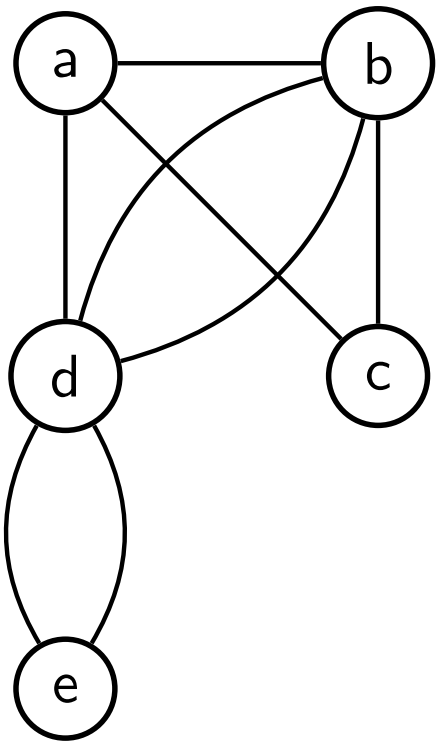
A graph G may also have several edges connecting the same two points. In this case, E is regarded as a multiset and we say that G is a *multigraph*.

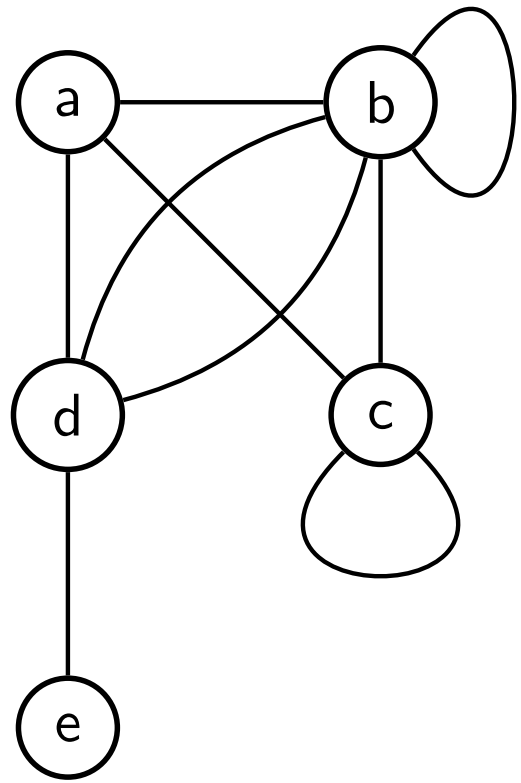
A graph G that contains *loops*, i.e., one-element subsets $\{a\} \subset V$, and for which E is possibly a multiset, is called a *pseudograph*.

Graphs

3.2.2. Example. From left to right: a simple graph, a multigraph and a pseudograph.







Directed Graphs

The following definition of a directed graph is more general than Definition 3.1.14:

3.2.3. Definition. A *directed graph* (or *digraph*) $G = (V, E)$ consists of a nonempty set V of *vertices* (or *nodes*) together with a collection E of ordered pairs of elements of V called *edges* (or *arcs*). An edge (u, v) is said to *start at* u and *end at* v.

If E contains every edge (a, b), $a, b \in V$, at most once and does not contain edges (a, a) (called *loops*) (i.e., $E \subset V^2$) we say that G is a *simple directed graph*.

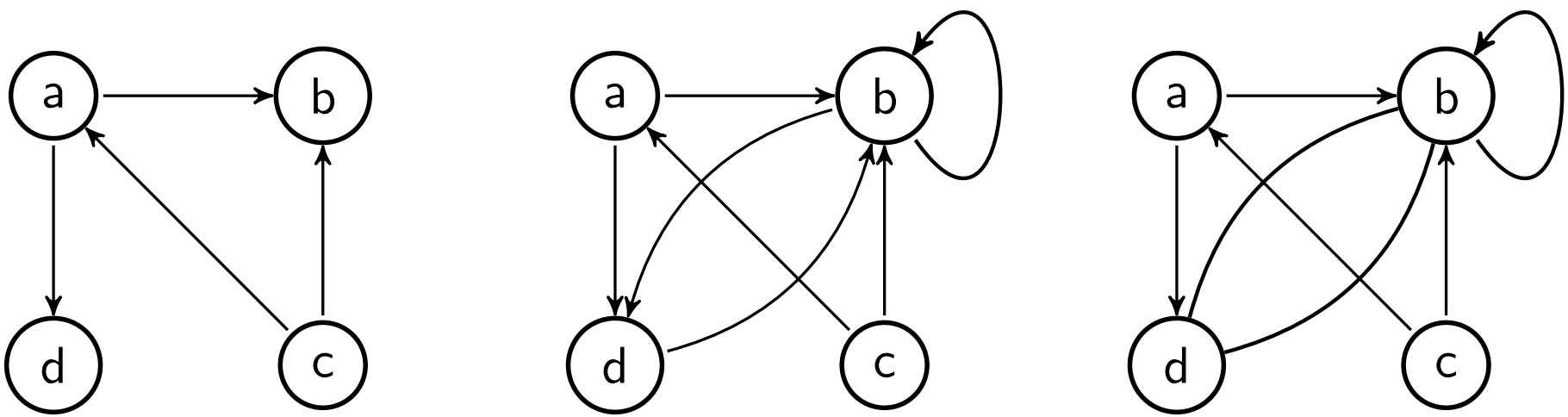
A directed graph G containing loops or more than one edge starting and ending at the same two points, respectively, is called a *directed multigraph*. In this case, E is regarded as a multiset of pairs.

A graph that contains directed a well as undirected edges is called a *mixed graph*.

Directed Graphs

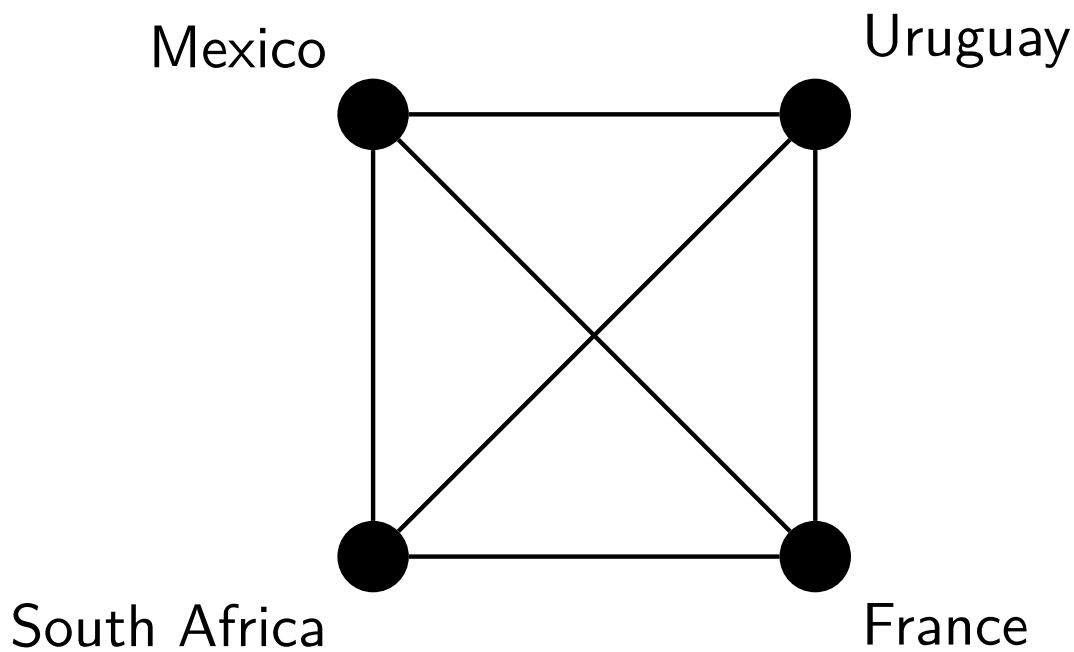
3.2.4. Remark. Note that while a directed multigraph may contain loops, an undirected multigraph may not (such an undirected graph would be called a pseudograph). Since graph theory is still a fairly young discipline, the terminology used varies widely and is not always logical or consistent.

3.2.5. Example. From left to right: a simple digraph, a multiple digraph and a mixed graph.



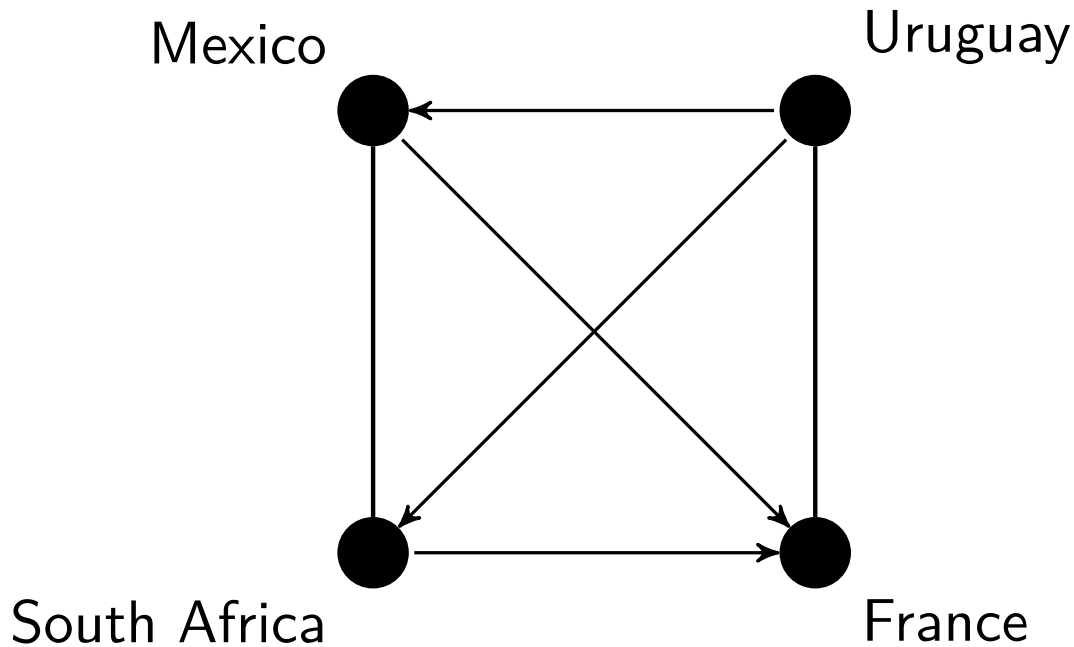
Example: Graphs in Football

3.2.6. Example. In the group stage of the 2010 World Cup, Group A consisted of Uruguay, Mexico, South Africa and France. Each of the teams played against each of the other three teams. (This is called a *round-robin tournament*.) The following undirected simple graph represents the matches:



Example: Graphs in Football

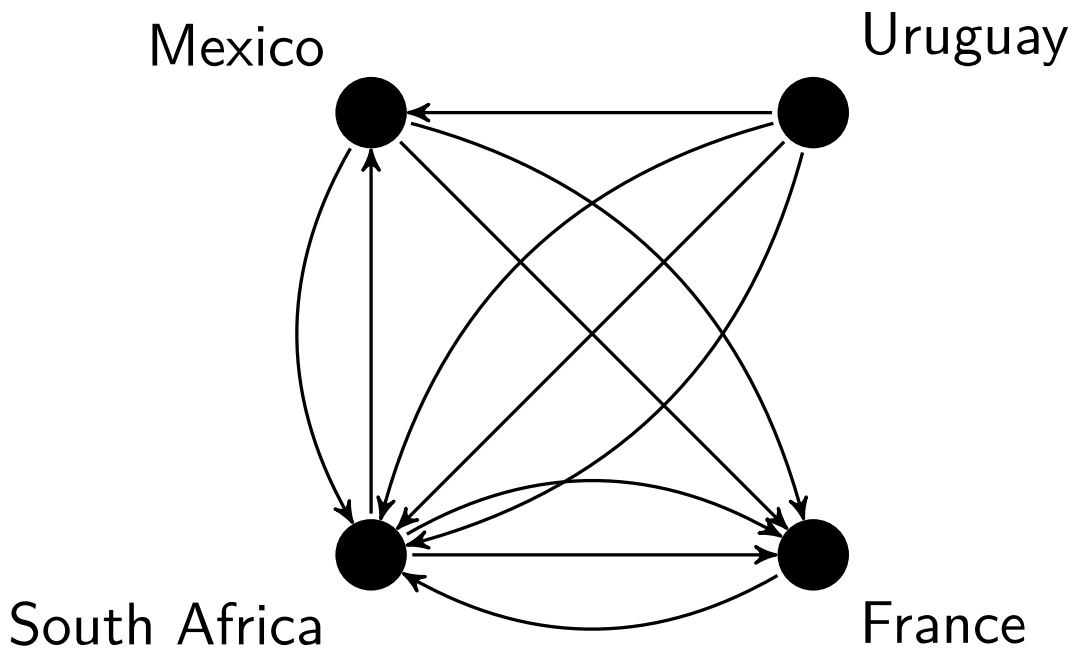
By adding arrows to the edges we can indicate which team eventually won each match:



The graph shows that Uruguay won its match against Mexico, but drew against France. This is a mixed graph.

Example: Graphs in Football

Alternatively, we can indicate the goals scored by each team against the other teams:



The graph shows that Uruguay did not concede a single goal and scored three goals against South Africa. The match between France and Uruguay ended without a goal scored. This is a directed multigraph.

Graph Terminology

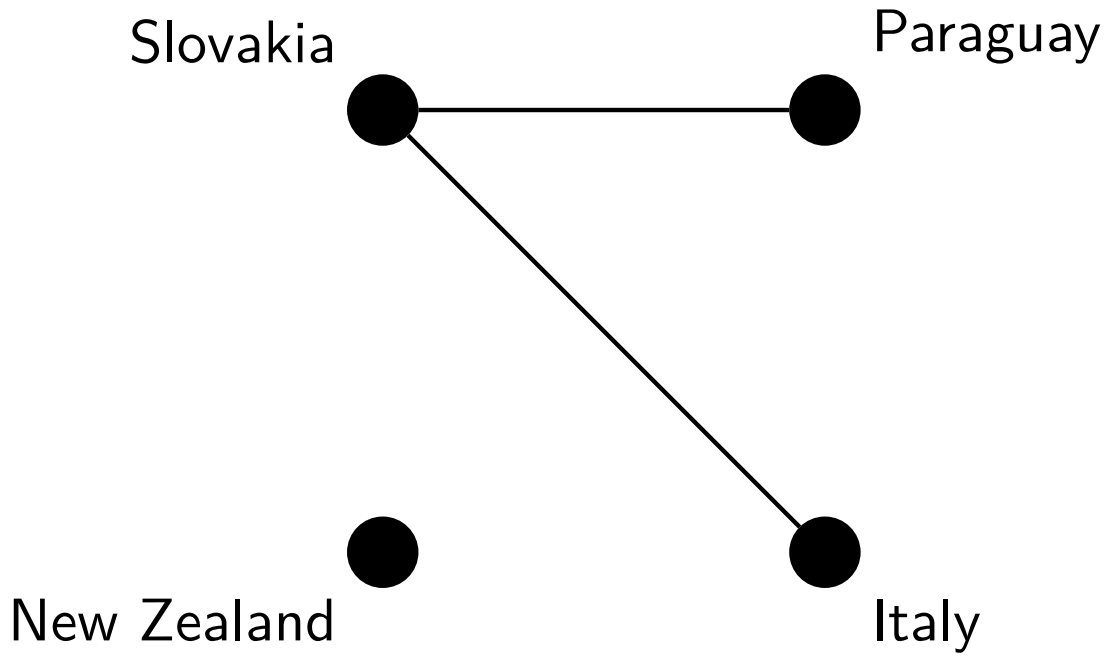
3.2.7. Definition. Two vertices u and v in an undirected graph G are called *adjacent* (or *neighbors*) in G if u and v are endpoints of an edge of G. If e is associated with $\{u, v\}$, the edge e is called incident with u and v. The edge e is also said to *connect* u and v.

The *degree* of a vertex is the number of edges incident with it, except that a loop at a vertex contributes twice to the degree of that vertex.The degree of a vertex v is denoted $\deg(v)$.

A vertex is *isolated* if it has degree zero. It is *pendant* if it has degree one.

Example: Graphs in Football

3.2.8. Example. In the group stage of the 2010 World Cup, Group F consisted of Paraguay, Slovakia, New Zealand and Italy. The following undirected simple graph represents the matches that did not end in a draw:



Here, New Zealand is isolated, while Italy and Paraguay are pendant. The vertex Slovakia has degree 2. Italy and Slovakia are adjacent, but Italy and Paraguay are not.

The Handshaking Theorem

The sum of the degrees of all the vertices of a graph must equal twice the number of edges, since each edge "contributes" two degrees. Suppose that the graph $G = (V, E)$ has $|E|$ edges. Then

$$2|E| = \sum_{v \in V} \deg v.$$

This is known as the *handshaking theorem*. A slightly less obvious statement is the following:

3.2.9. Theorem. An undirected graph has an even number of vertices of odd degree.

Proof.

Let V_1 and V_2 be the set of vertices of even and odd degree in $G = (V, E)$, respectively. Then

$$\underbrace{2|E|}_{\text{even}} = \sum_{v \in V_1} \underbrace{\deg v}_{\text{even}} + \sum_{v \in V_2} \underbrace{\deg v}_{\text{odd}}.$$

□

Digraph Terminology

3.2.10. Definition. An edge (u, v) of a directed graph is said to have *initial vertex* u and *terminal vertex* v. The vertex u is said to be adjacent to v and v is said to be adjacent to u.

The *in-degree* of a vertex v, denoted $\deg^-(v)$, is the number of edges with v as a terminal vertex. The *out-degree* of a vertex v, denoted $\deg^+(v)$, is the number of edges with v as an initial vertex. A loop contributes one to both the in- and the out-degree.

In a digraph (V, E) the sum of the edges is equal to the sum of the in-degrees and equal to the sum of the out-degrees:

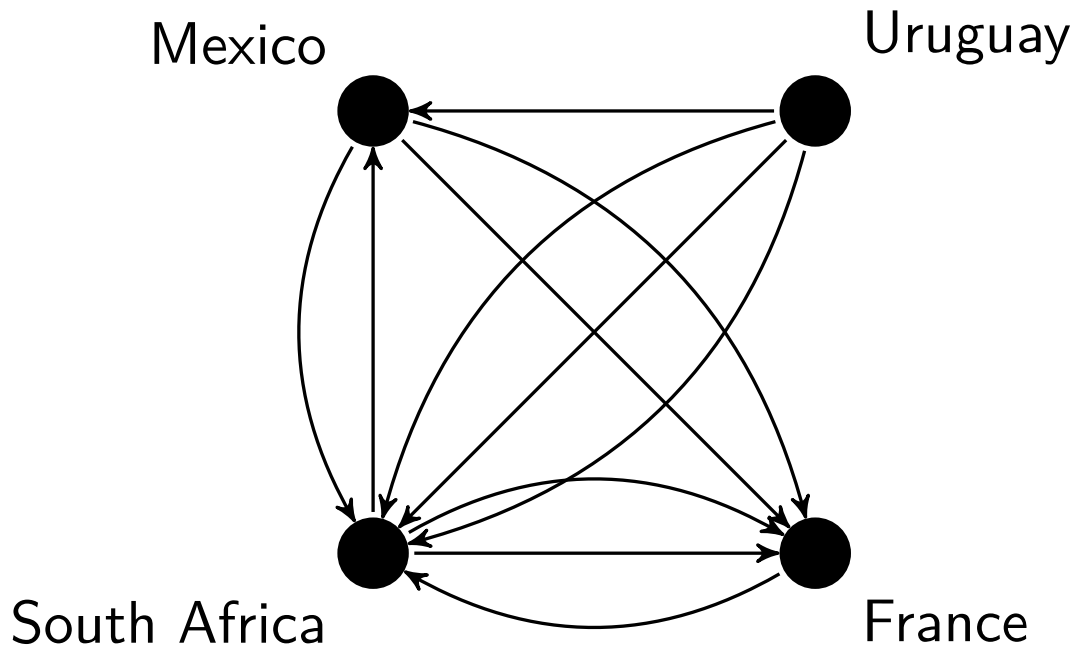
$$|E| = \sum_{v \in V} \deg^-(v) = \sum_{v \in V} \deg^+(v).$$

Digraph Terminology

3.2.11. Example. In the graph below,

$$\deg^-(\text{Uruguay}) = 0, \qquad \deg^+(\text{Uruguay}) = 4.$$

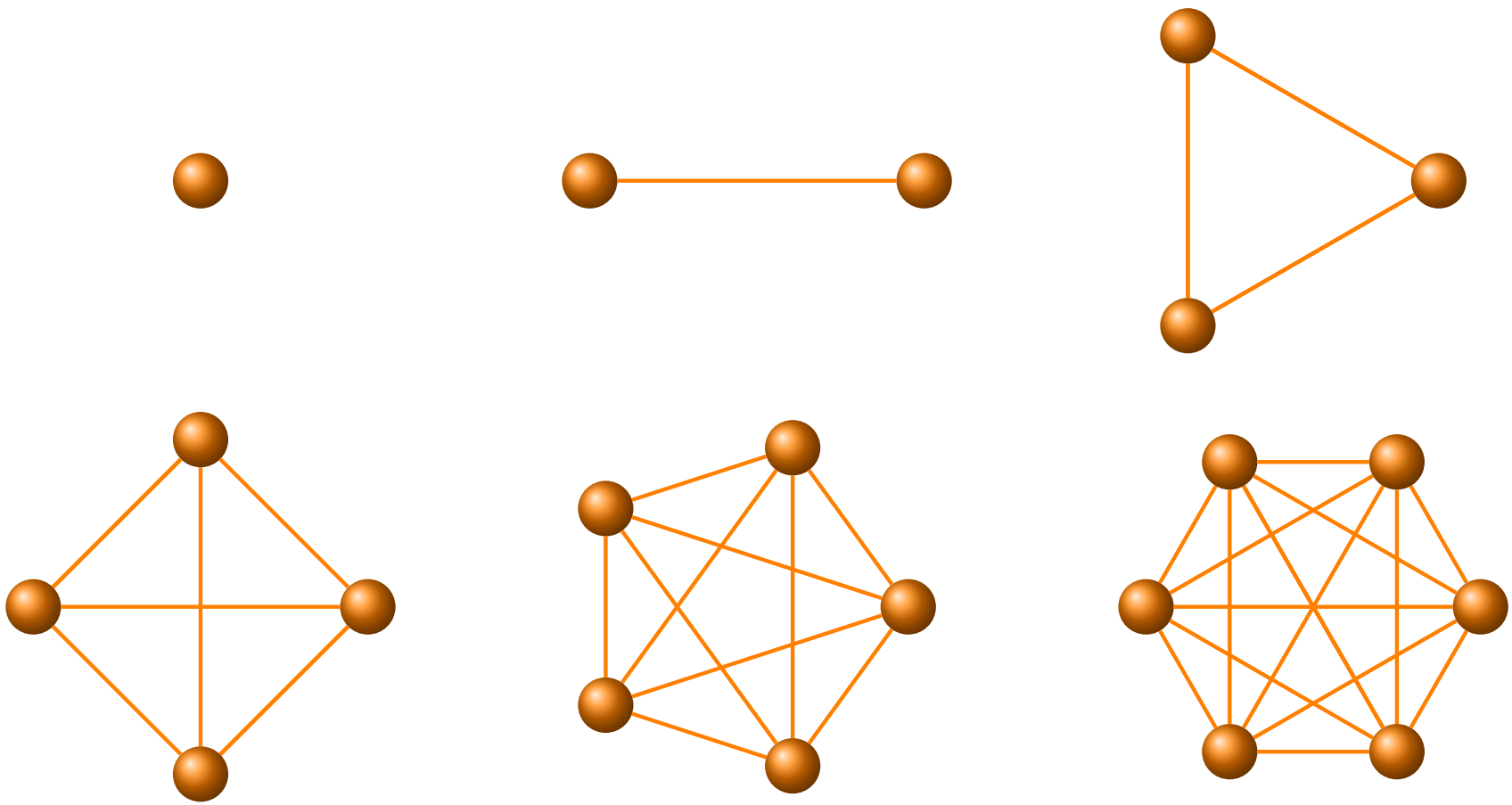
Uruguay and France are the only vertices that are not adjacent.



Complete Graphs

3.2.12. Definition. A *complete graph* K_j, $j \in \mathbb{Z}_+$ is the simple graph with j vertices that contains exactly one edge between each pair of vertices.

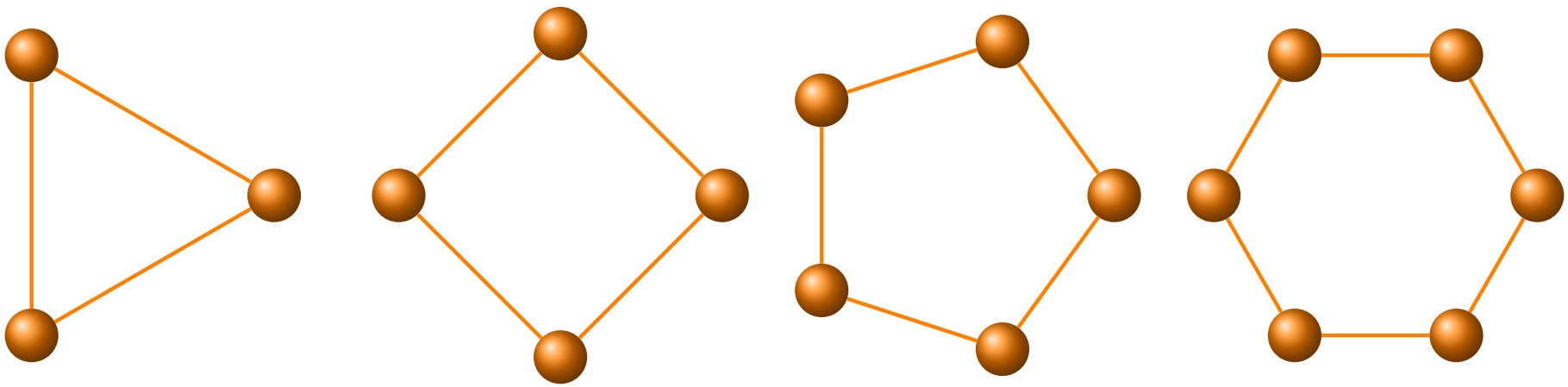
3.2.13. Examples. The complete graphs K_j, $j = 1, 2, 3, 4, 5, 6$:



Cycle Graphs

3.2.14. Definition. The *cycle* C_n, $n \geq 3$, is the graph $G = (V, E)$ with $V = \{v_1, \ldots, v_n\}$ and edges $\{v_j, v_{(j+1) \bmod n}\}$, $j = 1, \ldots, n$.

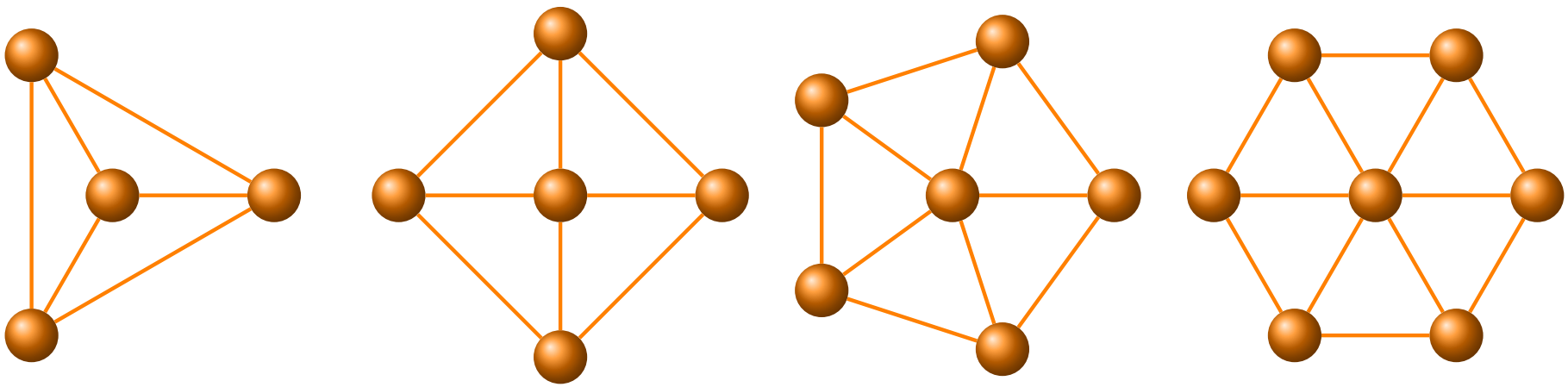
3.2.15. Examples. The cycles C_j, $j = 3, 4, 5, 6$:



Wheel Graphs

3.2.16. Definition. The *wheel* W_n, $n \geq 3$, is the graph $G = (V, E)$ consisting of a cycle C_n with one additional vertex connected to all other vertices.

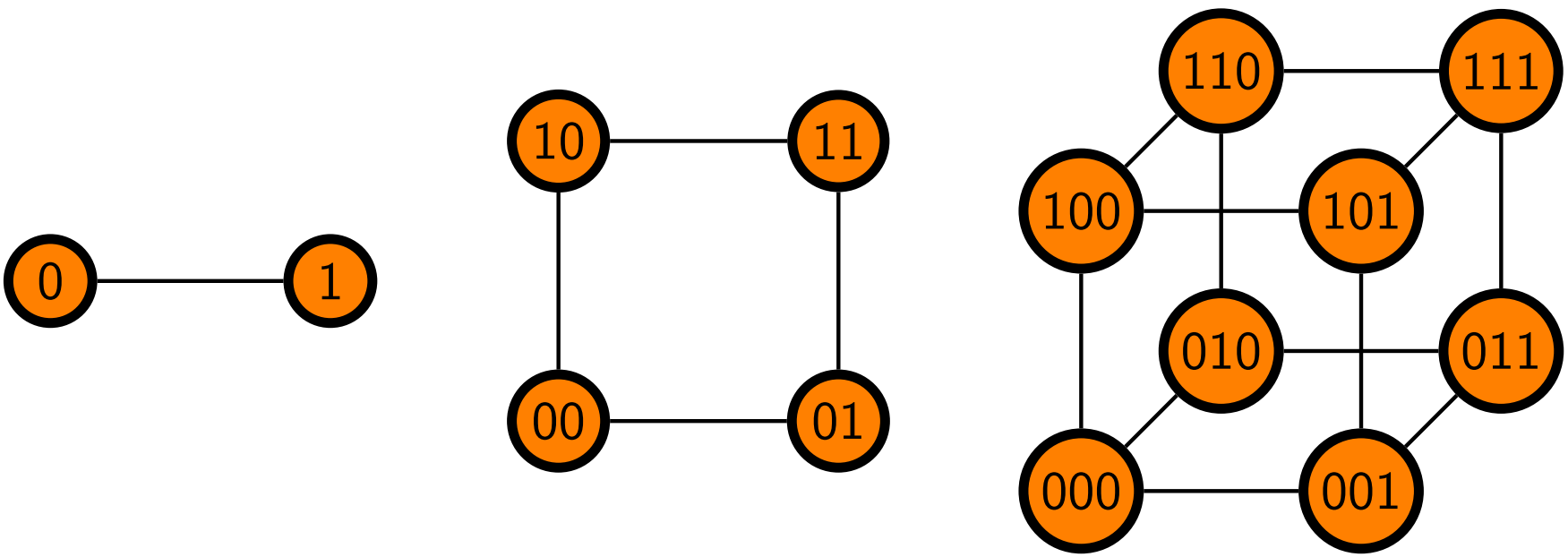
3.2.17. Examples. The wheels W_j, $j = 3, 4, 5, 6$:



Hypercube Graphs

3.2.18. Definition. The *hypercube* Q_n, $n \geq 1$, is the graph $G = (V, E)$ where V is the set of bit strings of length n and any two vertices are joined by an edge if they differ in a single digit.

3.2.19. Examples. The cubes Q_1, Q_2, Q_3:

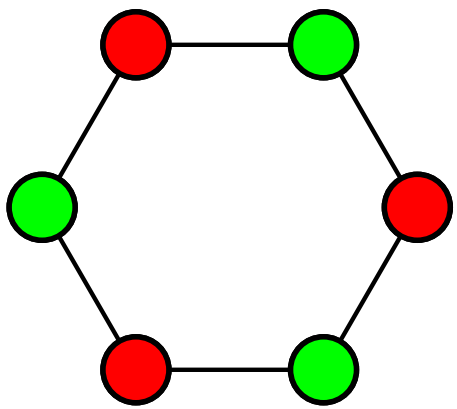


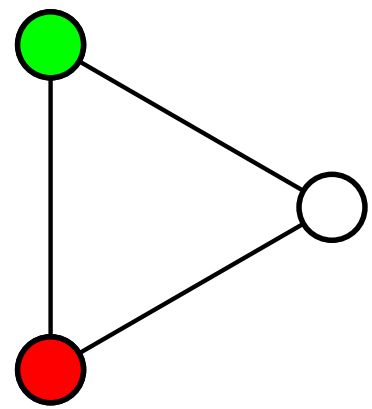
A hypercube Q_n can be constructed from two hypercubes Q_{n-1} by prefixing their vertices with zero and 1, respectively, and then joining the vertices that are identical except for the (new) first digit.

Bipartite Graphs

3.2.20. Definition. A simple graph $G = (V, E)$ is called *bipartite* if its vertex set can be partitioned into two disjoint subsets V_1 and V_2 such that every edge in E connects a vertex in V_1 to a vertex in V_2. We then call the pair (V_1, V_2) a *bipartition* of G.

3.2.21. Example. The cycle C_6 is bipartite, while the cycle C_3 is not bipartite.





Bipartite Graphs

Example 3.2.21 suggests the following theorem:

3.2.22. Theorem. A simple graph is bipartite if and only if it is possible to assign one of two different colors to each vertex of the graph so that no two adjacent vertices are assigned the same color.

Proof.

($\Rightarrow$) Assume that the graph $G = (V, E)$ is bipartite. Then $V = V_1 \cup V_2$, where every vertex in V_1 is adjacent only to an edge in V_2 and vice-versa. We assign one color to all vertices in V_1 and the other color to all vertices in V_2.

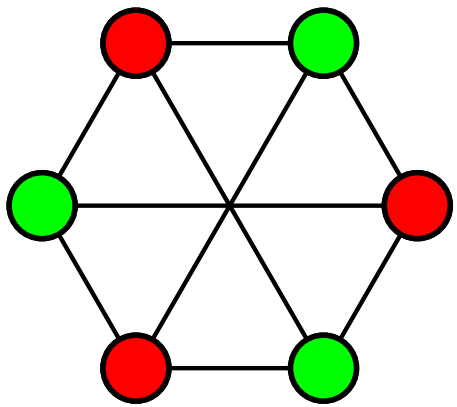
($\Leftarrow$) Suppose that $G = (V, E)$ and $V = V_1 \cup V_2$, where V_1 consist of vertices of one color and V_2 consists of vertices of the other color and no two vertices of the same color are adjacent. Then every vertex in V_1 is adjacent only to a vertex in V_2 and vice-versa, so (V_1, V_2) is a bipartition of G.

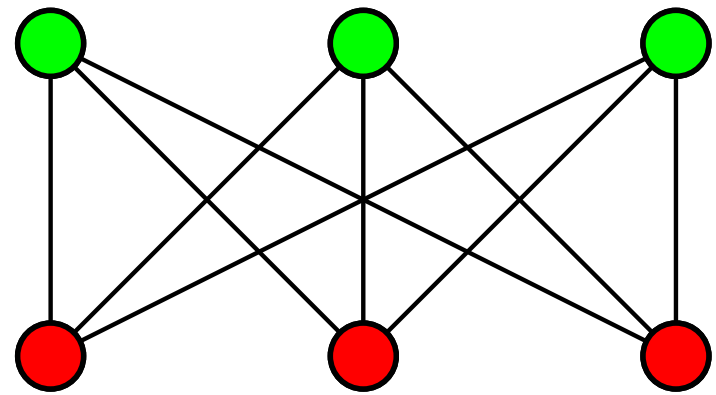
Complete Bipartite Graphs

Recall from Definition 3.2.12 that a complete graph is a graph where any two vertices are connected by exactly one edge.

3.2.23. Definition. The *complete bipartite graph* $K_{m,n}$, $m, n \in \mathbb{Z}_+$, is a simple graph having a bipartition (V_1, V_2) such that $|V_1| = m$, $|V_2| = n$, and every vertex in V_1 is adjacent to every vertex in V_2.

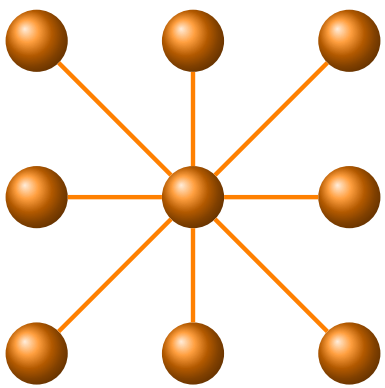
3.2.24. Example.
Below are two ways of drawing the graph $K_{3,3}$:



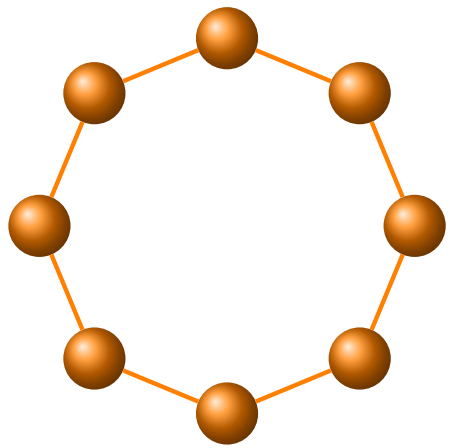


Some Applications of Graphs

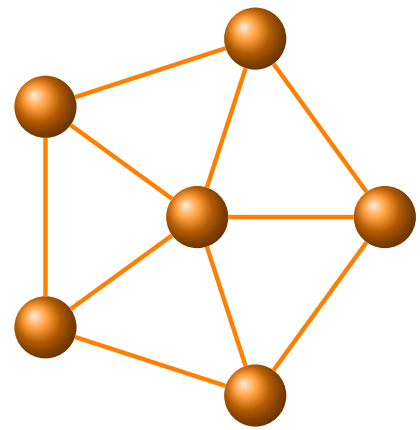
3.2.25. Example. Local-Area Networks (LANs) are interconnections of computers or computing devices (such as printers, routers etc.). There are three basic ways (topologies) of arranging the network connections, represented through graphs:



Star Topology (bipartite complete graph $K_{1,n}$)



Ring Topology (ring graph C_n)



Hybrid Topology (wheel graph W_n)

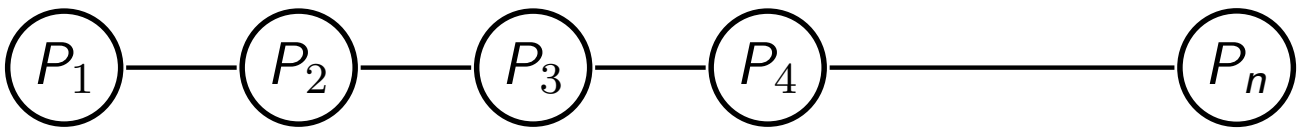
Some Applications of Graphs

3.2.26. Example. In modern computers, calculationally intensive tasks are not handled by a single processor (*serial processing*) but split into multiple sub-tasks and handled by several processors concurrently (*parallel processing*). Since the sub-tasks depend on each other, the processors need to be connected so that they can communicate with each other. The processors thus represent a network. The type of connection within the network will influence the speed of the calculation as well as the type of parallel software that can be used.

The simplest possible network would connect any two processors with each other, forming a complete graph K_n with $\binom{n}{2}$ edges. For 64 processors, this means there would be 2016 connections and any processor would be connected to 63 others. This configuration is often not possible in practice and will always be expensive.

Linear Arrays

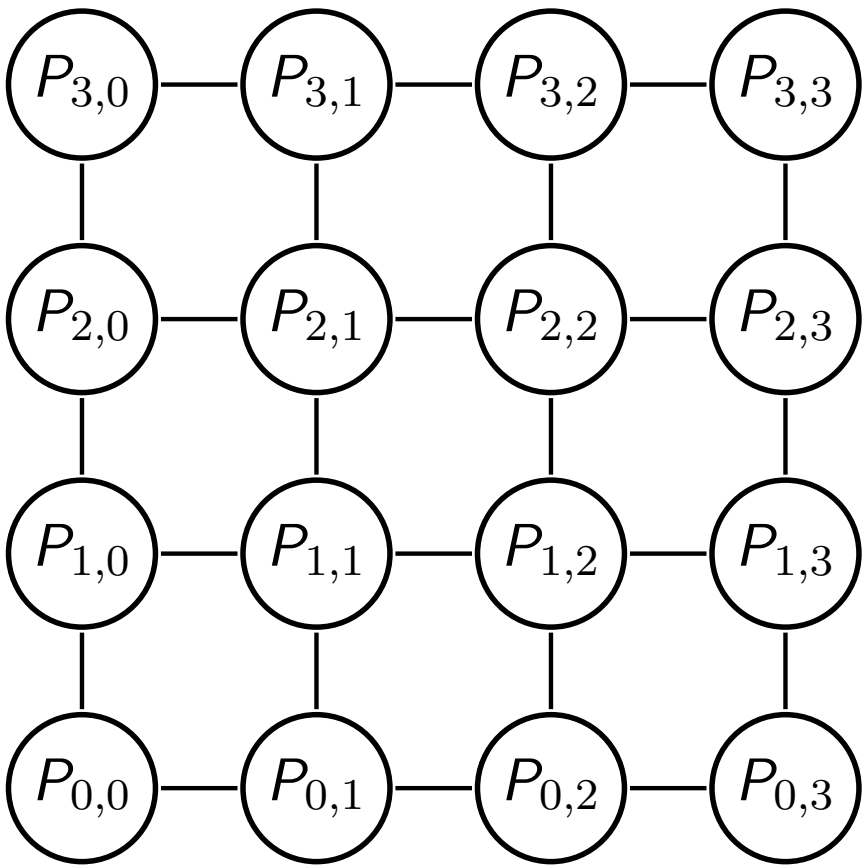
Another simple arrangement is called a *linear array*, where the ith processor is connected to the $(i-1)$st and the $(i+1)$st processor, except that the first is only connected to the second, and the last is only connected to the penultimate processor. If we label our processors $P_1, \ldots, P_n$, the linear array is shown below:



The disadvantage of the linear array is that communication between two processors requires sending signals across many intermediate processors, i.e., requiring many *hops*.

Mesh Networks

If there are $n = m^2$ processors, they can be arranged in a *mesh*. The processors are labeled $P_{i,j}$, $0 \leq i, j \leq m-1$, and each $P_{i,j}$ is connected to $P_{i\pm1,j\pm1}$, i.e., four other processors, as long as they are in the mesh.



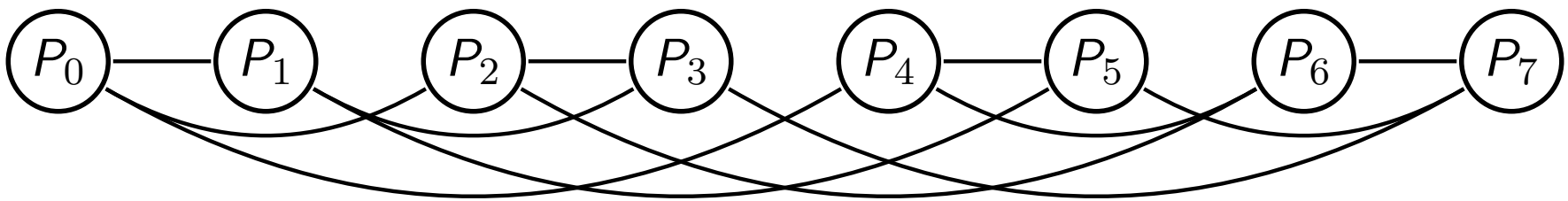
Nodes at the edges of the network are only connected to two or three other nodes.

There is a generalization of the mesh network that ensures that every node is connected to exactly four other nodes (see exercises).

Communication between two arbitrary nodes requires $O(\sqrt{n}) = O(m)$ intermediate links.

Hypercube Networks

Many networks with $n = 2^m$ nodes are arranged in *hypercubes*. The hypercube configuration balances the number of direct connections of each processor with the number of intermediate connections required for communication. A hypercube network for $m = 3$ is shown below:



This is another representation of the hypercube in Example 3.2.19.

Subgraphs

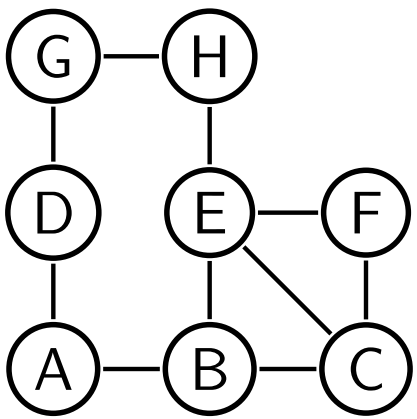
3.2.27. Definition. let $G = (V, E)$ be a graph. Then $H = (W, F)$ is called a *subgraph* of G if $W \subset V$ and $F \subset E$. If $H \neq G$ then H is called a *proper subgraph* of G.

3.2.28. Example. The cycle C_6 is a subgraph of the wheel W_6 which is a subgraph of the complete graph K_7.

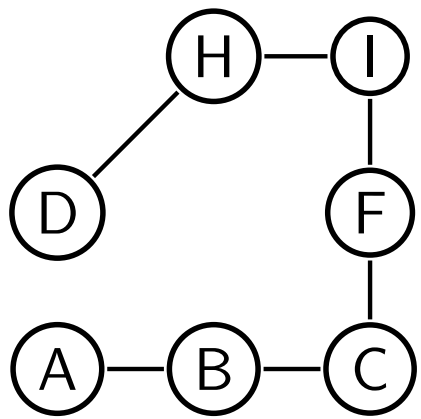
Union of Graphs

3.2.29. Definition. Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be graphs. Then the *union* of G_1 and G_2, denoted by $G_1 \cup G_2$, is the simple graph with vertices $V_1 \cup V_2$ and edges $E_1 \cup E_2$.

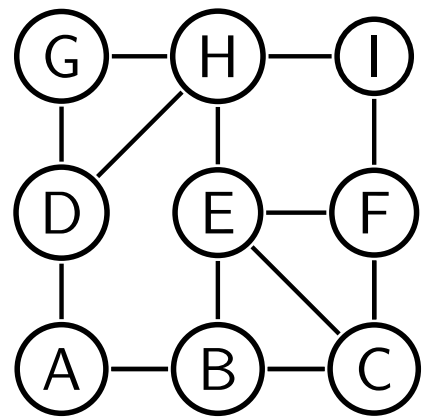
3.2.30. Example.



G_1



G_2

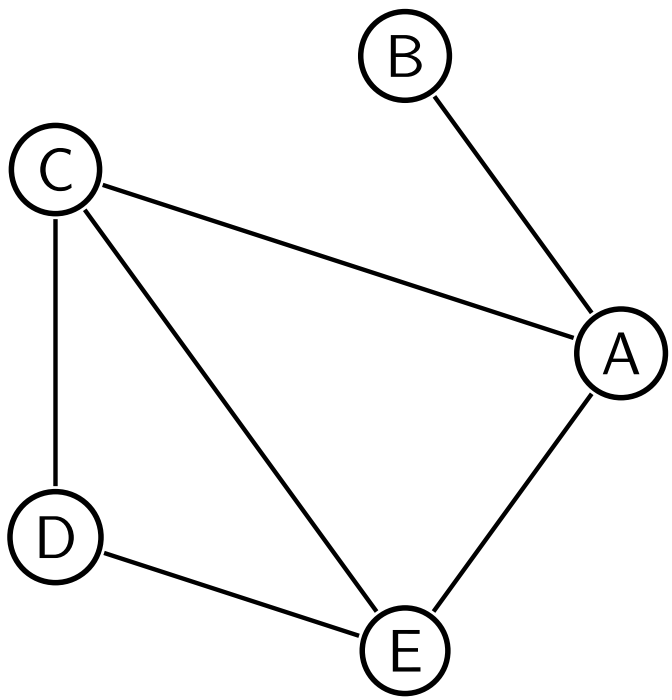


$G_1 \cup G_2$

Adjacency Tables

We will now study several ways of representing graphs. A simple way to do so is using an *adjacency table*, which lists all the vertices of the graph and the vertices adjacent to them.

3.2.31. Example. The following is a simple graph and its adjacency table:

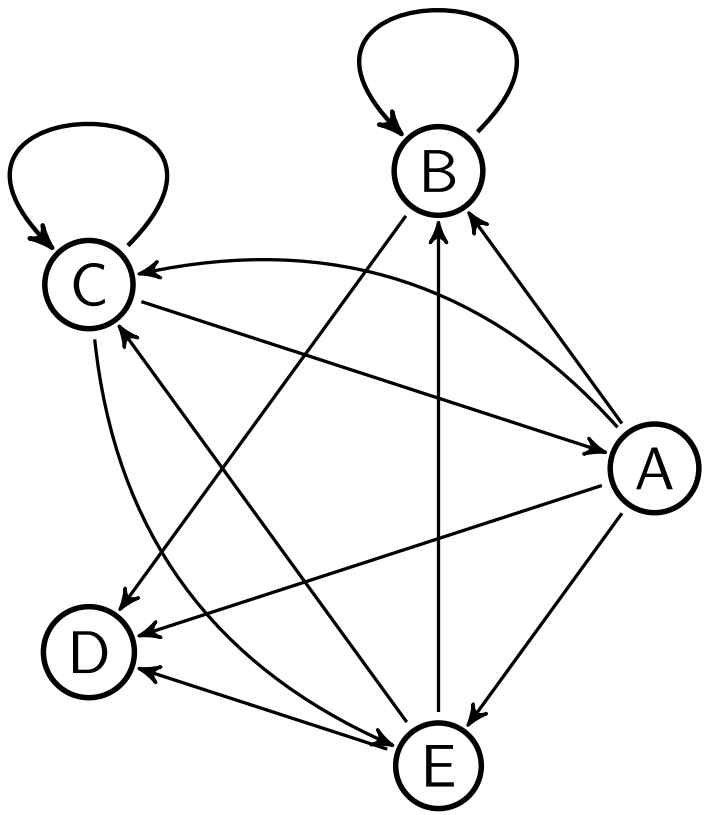


Vertex	Adjacent Vertices
A	B, C, E
B	A
C	A, D, E
D	C, E
E	A, C, D

Adjacency Tables

A similar table can be used for directed graphs.

3.2.32. Example. The following is a directed graph and its adjacency table:



Initial Vertex	Terminal Vertices
A	B, C, D, E
B	B, D
C	A, C, E
D	
E	B, C, D

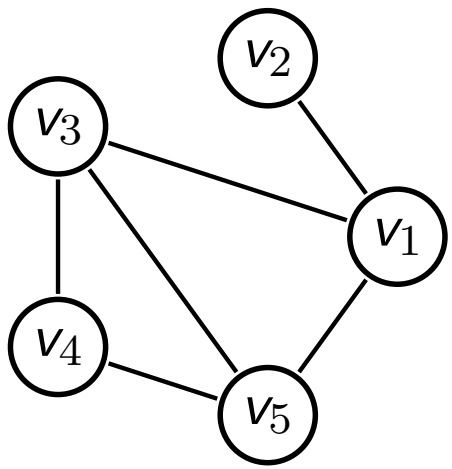
Adjacency Matrices

Instead of tables, matrices can also be used to describe graphs. Assume that $G = (V, E)$, where $V = (v_1, \ldots, v_n)$ has been ordered in some way. We then define the *adjacency matrix* $A_G \in \mathrm{Mat}(n \times n, \mathbb{Z}_2)$ through

$$a_{ij} = \begin{cases} 1 & \text{if } \{v_i, v_j\} \in E, \\ 0 & \text{otherwise.} \end{cases}$$

Note that the adjacency matrix is always symmetric.

3.2.33. Example. The following is a simple graph and its adjacency matrix:



$$A_G = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{pmatrix} \end{array}$$

Adjacency Matrices

In the case of simple graphs, adjacency matrices are zero-one matrices with vanishing trace. However, they can also be used to describe multigraphs and pseudographs, in which case the entries a_{ij} are equal to the multiplicity of the edge $\{v_i, v_j\}$. The adjacency matrix remains symmetric for any type of undirected graph.

An adjacency matrix $A_G \in \mathrm{Mat}(n \times n, \mathbb{Z}_2)$ for a directed graph $G = (V, E)$ is defined by

$$a_{ij} = \begin{cases} 1 & \text{if } (v_i, v_j) \in E, \\ 0 & \text{otherwise.} \end{cases}$$

Clearly, it does not need to be symmetric.

Adjacency Matrices

If $|V| = n$, an adjacency matrix will contain n^2 entries, while an adjacency table will have $c \cdot n$ entries, where $c = \max_{1 \leq i \leq n} \deg(v_i)$. Therefore, if a (pseudo-)graph has only few edges, $c \ll n$, (it is *sparse*) it may be more efficient to use adjacency tables to describe the graph. On the other hand, in this case most of the entries of the adjacency matrix will be zero (such a matrix is also called sparse) and there are special techniques to work with sparse matrices efficiently.

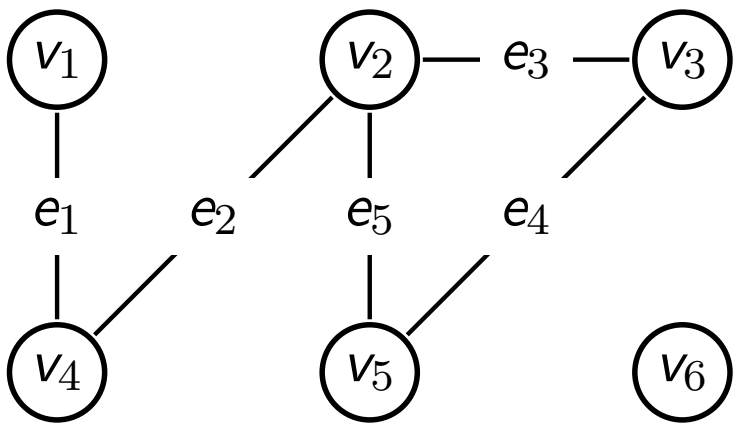
If the graph is *dense* (contains many edges) it is preferable to use an adjacency matrix instead of an adjacency table as information is contained in a more accessible form in the matrix.

Incidence Matrices

Graphs can also be represented using *incidence matrices*. If $G = (V, E)$ is an undirected (pseudo-)graph, $V = \{v_1, \ldots, v_n\}$, and $E = \{e_1, \ldots, v_m\}$, we define $M_G \in \mathrm{Mat}(m \times n, \mathbb{Z}_2)$ by

$$m_{ij} = \begin{cases} 1 & \text{when the edge } e_j \text{ is incident with the vertex } v_i \\ 0 & \text{otherwise.} \end{cases}$$

3.2.34. Example. The following is a simple graph and its incidence matrix:



$$M_G = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{array} \begin{array}{c} \begin{array}{ccccc} e_1 & e_2 & e_3 & e_4 & e_5 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

Isomorphisms of Graphs

Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be simple graphs. Suppose we have a bijective map

$$\varphi\colon V_1 \to V_2,$$

i.e., to every vertex in V_1 we associate precisely one vertex in V_2 and vice-versa. This map induces an injective map

$$\varphi^*\colon E_1 \to V_2 \times V_2, \qquad (a, b) \mapsto (\varphi(a), \varphi(b)).$$

If $\operatorname{ran} \varphi^* = E_2$, we can regard the graph G_2 as being the image of G_1 under (φ, φ^*), where φ maps the vertices of G_1 into those of G_2 and φ^* does the same for the edges.

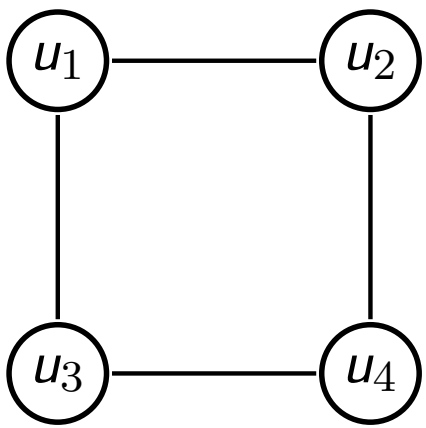
Isomorphisms of Graphs

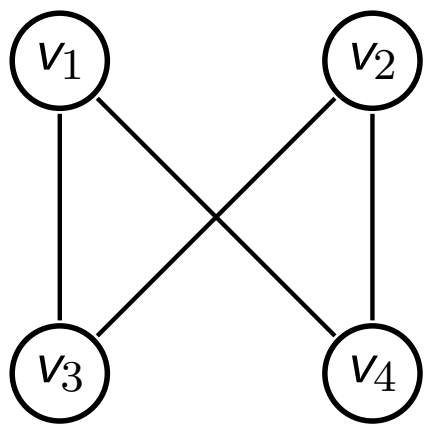
3.2.35. Definition. Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two simple graphs. Then we say that G_1 and G_2 are *isomorphic* if there exists a bijective function $\varphi\colon V_1 \to V_2$ such that the induced map

$$\varphi^*\colon E_1 \to E_2, \qquad (a, b) \mapsto (\varphi(a), \varphi(b))$$

is bijective. Such a function φ is called a *(graph) isomorphism*.

3.2.36. Example. Consider the two graphs $G = (U, E)$, $H = (V, F)$ below:





Isomorphisms of Graphs

We now show that the graphs G and H are isomorphic: define $\varphi\colon U \to V$ by setting

$$\varphi\colon u_1 \mapsto v_1, \qquad u_2 \mapsto v_4, \qquad u_3 \mapsto v_3, \qquad u_4 \mapsto v_2.$$

Now

$$E = \big\{\{u_1, u_2\}, \{u_1, u_3\}, \{u_2, u_4\}, \{u_3, u_4\}\big\}$$

and

$$\begin{aligned} F &= \big\{\{v_1, v_3\}, \{v_1, v_4\}, \{v_2, v_3\}, \{v_2, v_4\}\big\} \\ &= \big\{\{\varphi(u_1), \varphi(u_3)\}, \{\varphi(u_1), \varphi(u_2)\}, \{\varphi(u_3), \varphi(u_4)\}, \{\varphi(u_2), \varphi(u_4)\}\big\}. \end{aligned}$$

It is easy to see that $(a, b) \in E$ if and only if $(\varphi(a), \varphi(b)) \in F$, so φ is a graph isomorphism. This shows that G and H are isomorphic.

Isomorphisms of Graphs

An equivalent way to see that φ is an isomorphism is to write down the adjacency matrix for the ordered set of vertices $U = (u_1, \ldots, u_4)$ and compare with the matrix for the graph $(\varphi(U), \varphi^*(E))$, where

$$\begin{aligned}
\varphi(U) &= (\varphi(u_1), \varphi(u_2), \varphi(u_3), \varphi(u_4)) \\
&= (v_1, v_4, v_3, v_2), \\
\varphi^*(E) &= \big\{\{\varphi(u_1), \varphi(u_3)\}, \{\varphi(u_1), \varphi(u_2)\}, \{\varphi(u_3), \varphi(u_4)\}, \{\varphi(u_2), \varphi(u_4)\}\big\} \\
&= \big\{\{v_1, v_3\}, \{v_1, v_4\}, \{v_2, v_3\}, \{v_2, v_4\}\big\}
\end{aligned}$$

If the matrices are identical, then φ is an isomorphism. In our case, we have

$$A_{(U,E)} = \begin{array}{c} \\ u_1 \\ u_2 \\ u_3 \\ u_4 \end{array} \begin{array}{c} \begin{array}{cccc} u_1 & u_2 & u_3 & u_4 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \end{array} = \begin{array}{c} \\ v_1 \\ v_4 \\ v_3 \\ v_2 \end{array} \begin{array}{c} \begin{array}{cccc} v_1 & v_4 & v_3 & v_2 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \end{array} = A_{(\varphi(U),\varphi^*(E))}$$

Graph Invariants

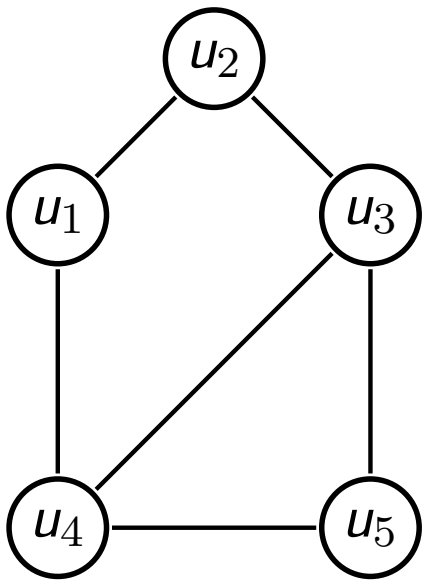
Properties of graphs that are preserved by graph isomorphisms are called *graph invariants*. Two obvious examples of graph invariants are

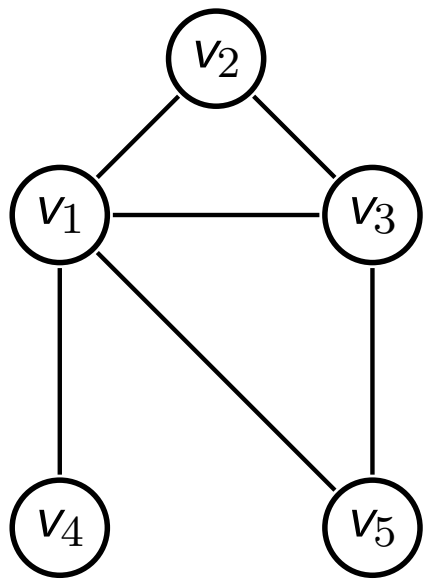
- The number of edges of a graph,
- The total number of vertices in a graph

The second invariant can be made more precise:

- For any $k \in \mathbb{N}$, the number of vertices of degree k is a graph invariant.

3.2.37. Example. Consider the two graphs $G = (U, E)$, $H = (V, F)$ below:

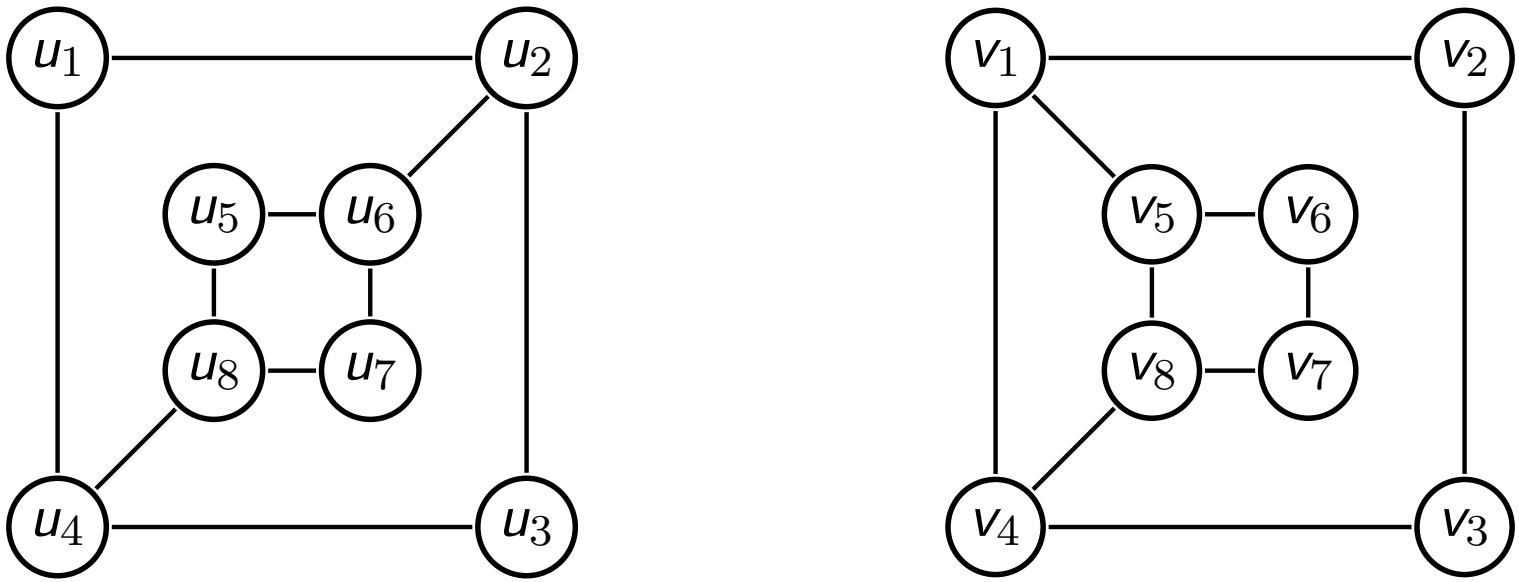




Graph Invariants

The two graphs are not isomorphic because H has a vertex of degree 4 (v_1), while G has no such vertex.

3.2.38. Example. Consider the two graphs $G = (U, E)$, $H = (V, F)$ below:



Both graphs have the same number of vertices of degrees 1, 2 and 3, so they might conceivably be isomorphic. In order to show that they are in fact not isomorphic, we need a stronger argument.

Isomorphism of Subgraphs

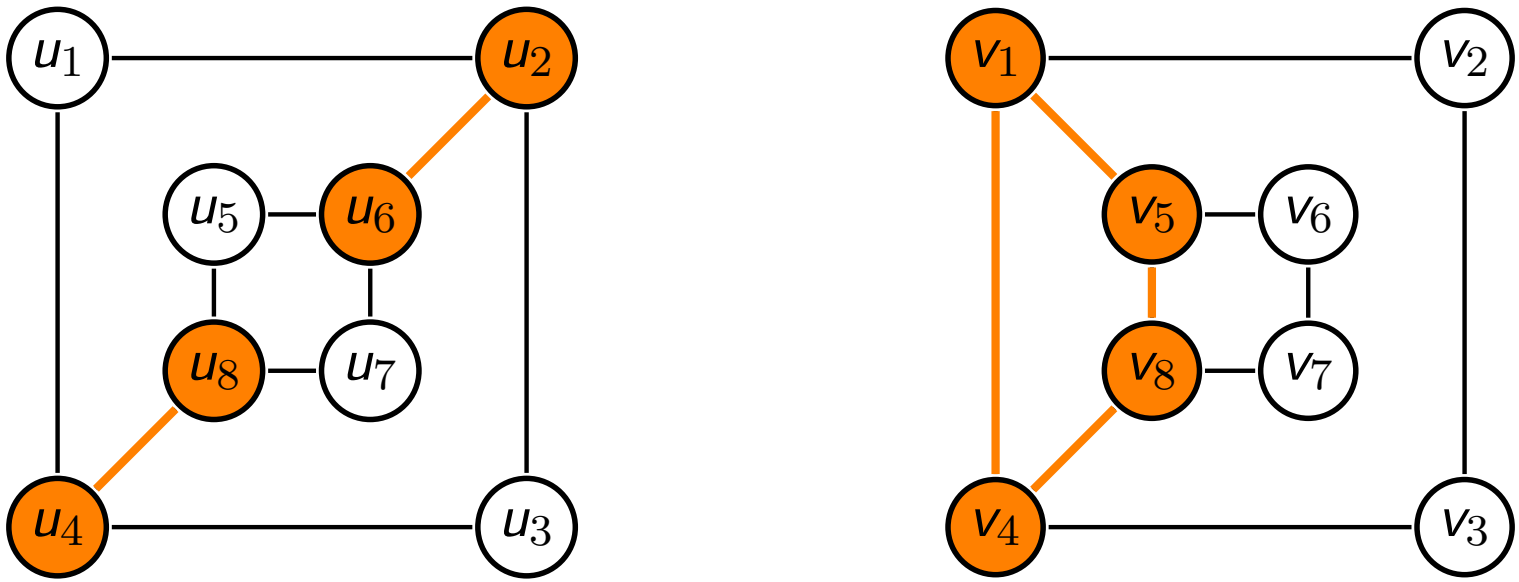
Suppose that there exists a graph isomorphism $\varphi\colon U \to V$, $\varphi_*\colon E \to F$. Let $U' \subset U$ and $G' = (U', E')$ be a subgraph of (U, E). Then φ restricted to U', denoted $\varphi|_{U'}$, has range contained in V and induces an isomorphism onto a subgraph of (V, F).

In our example, consider the subgraph (U', E') consisting of all vertices of degree $k \in \mathbb{N}$. Then a graph isomorphism φ, if one exists, can be restricted to U' and its range must be the subgraph of V consisting of vertices of degree k. We hence make the following note:

3.2.39. Lemma. If two graphs are isomorphic, then all subgraphs consisting of vertices of degree $k \in \mathbb{N}$ must be isomorphic.

Isomorphism of Subgraphs

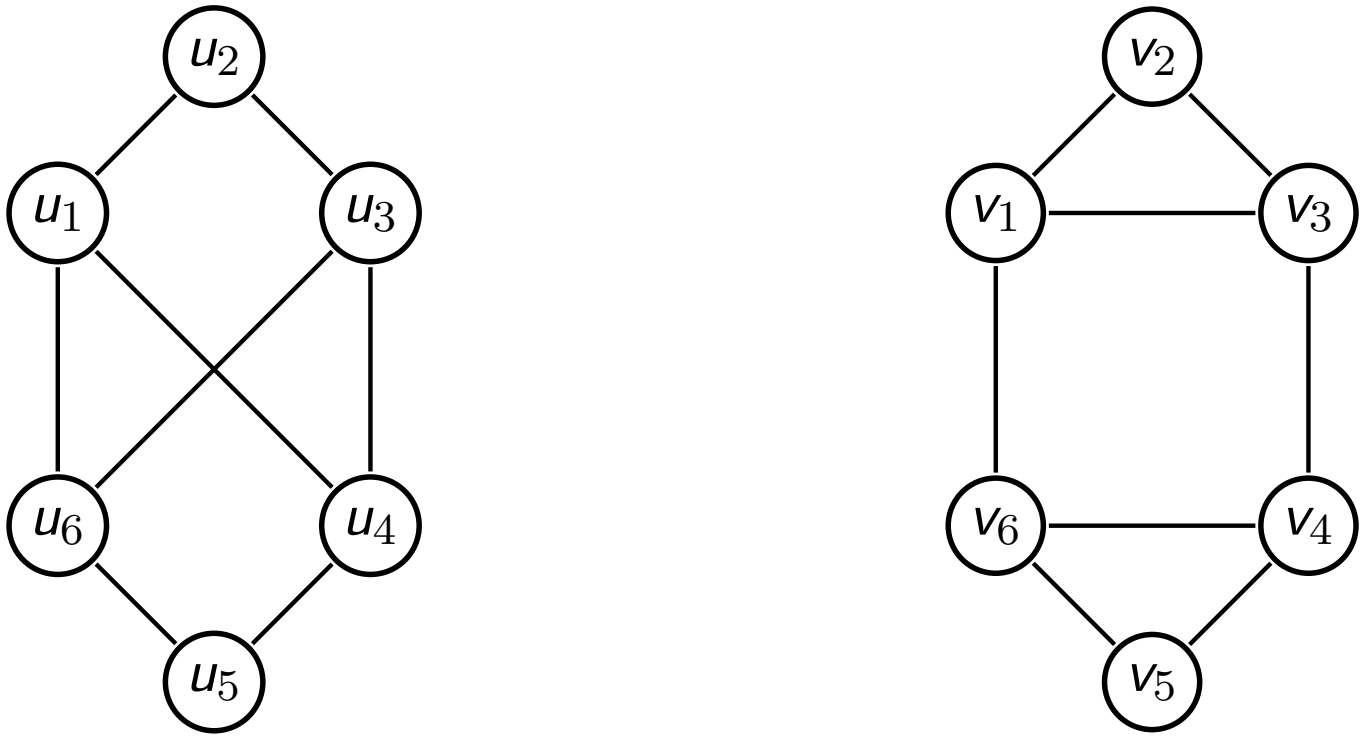
Returning to Example 3.2.38, the subgraphs consisting of vertices of degree 3 and their common edges are marked in orange:



It is obvious that the two subgraphs are not isomorphic, so the graphs G and H can not be isomorphic either.

A More Difficult Problem of Isomorphism

3.2.40. Example. Consider the two graphs $G = (U, E)$, $H = (V, F)$ below:



Here the subgraphs of of vertices of degree 2 and 3 are isomorphic (for the latter, see Example 3.2.36). However, we may suspect that G and H are not themselves isomorphic; proving this requires us to introduce the concept of a path, which we have already encountered for directed graphs.

Paths

3.2.41. Definition. Let $G = (U, E)$ be an undirected graph. Let $u, v \in U$.

- A *path of length* $n = 1$ *from* u *to* v is a set containing the single edge $e \in E$ associated with $\{u, v\}$.
- A *path of length* $n \geq 2$ *from* u *to* v is a tuple of edges $(e_1, e_2, \ldots, e_n)$ such that
  - e_1 is associated with $\{u, x_1\}$,
  - e_k is associated with $\{x_{k-1}, x_k\}$ for $k = 2, \ldots, n-1$,
  - e_n is associated with $\{x_{n-1}, v\}$

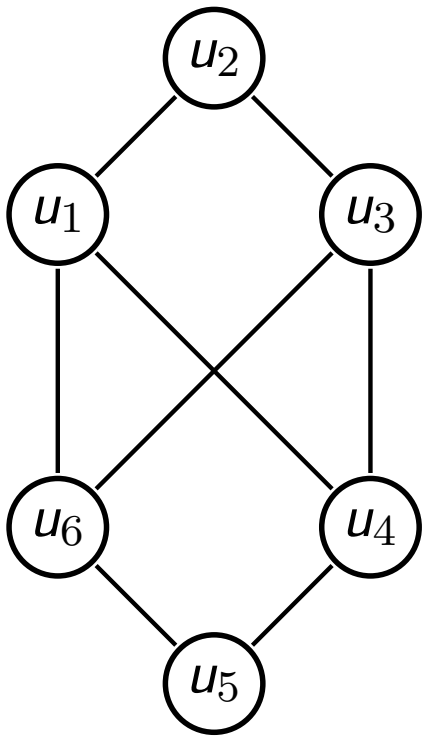
  where $x_1, \ldots, x_{n-1} \in U$ and $e_1, \ldots, e_n \in E$.
- A path is said to *pass through* the vertices $u, x_1, \ldots, x_{n-1}, v$ and *traverse* the edges $e_1, \ldots, e_n$.
- If G is a simple graph, we denote the path simply by the tuple of vertices it passes through, i.e., by $(u, x_1, \ldots, x_{n-1}, v)$.
- A path with $u = v$ is called a *circuit*.
- If the tuple of edges does not contain any edge more than once, the path is said to be *simple*.

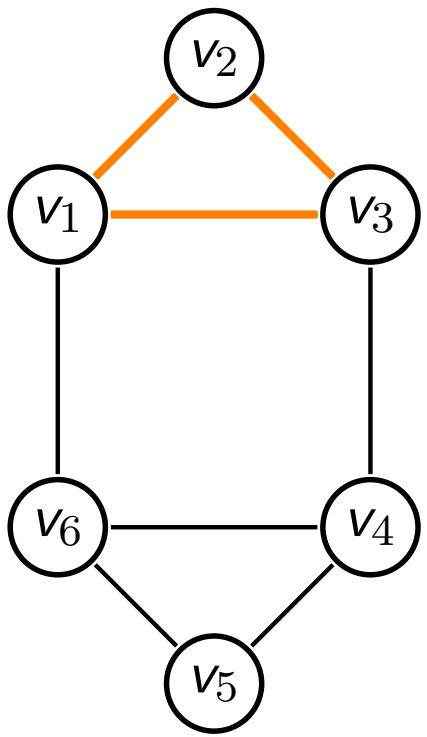
Paths

In the assignments, you will prove the following result:

3.2.42. Theorem. The existence of a simple circuit of length $k \in \mathbb{Z}_+$ is a graph invariant.

3.2.43. Example. The graphs shown below are not isomorphic, because one has a simple path of length 3 (marked in orange) while the other does not:



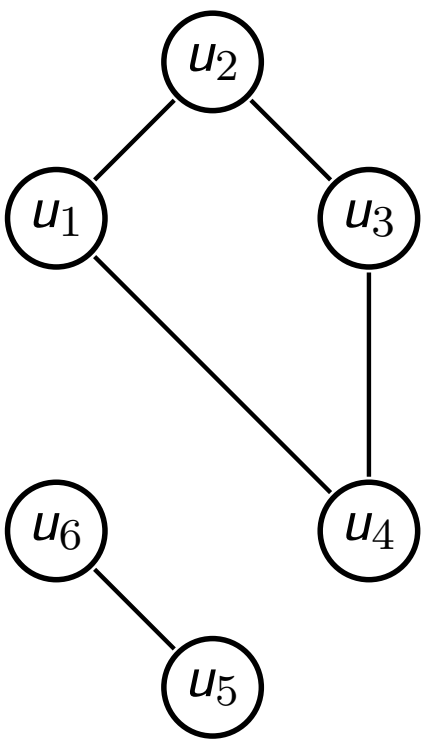


Connectedness

3.2.44. Definition. An undirected graph is called *connected* if there is a path between any two distinct vertices.

A *connected component* of a graph is a connected subgraph that is not a proper subgraph of another connected subgraph.

3.2.45. Example. Consider the graph $G = (U, E)$ shown below:



The graph is not connected, because there is no path from u_1 to u_5 that traverses edges in E.

There are two connected components: the subgraph given by $U' = \{u_5, u_6\}$ with edge set $E' = \{\{u_5, u_6\}\}$ and the subgraph $U'' = \{u_1, u_2, u_3, u_4\}$ with edge set $E'' = \{\{u_1, u_2\}, \{u_2, u_3\}, \{u_3, u_4\}, \{u_1, u_4\}\}$.

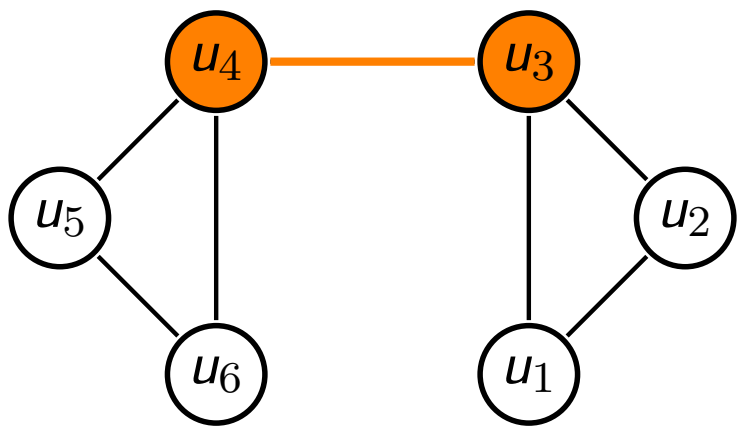
$U''' = \{u_1, u_2, u_3\}$ with $E''' = \{\{u_1, u_2\}, \{u_2, u_3\}\}$ is not a connected component, because it a subgraph of the connected subgraph (U'', E'').

Cut Vertices and Cut Edges

The removal of a vertex v of a graph G together with all incident edges gives a subgraph of G. If this subgraph has more connected components than G, we say that v is a *cut vertex* or an *articulation point*.

Similarly, an edge whose removal gives a subgraph with more connected components is called a *cut edge*.

3.2.46. Example. In the graph below, the cut edges and cut vertices have been marked in orange:



Euler Paths and Circuits

3.2.47. Definition. Let G be a finite multigraph.

1. An *Euler circuit* in G is a simple circuit containing every edge of G.
2. An *Euler path* in G is a simple path containing every edge of G.

Famously, Leonhard Euler studied the seven bridges of Königsberg in 1735. An Euler path corresponds to walking across each bridge exactly once:

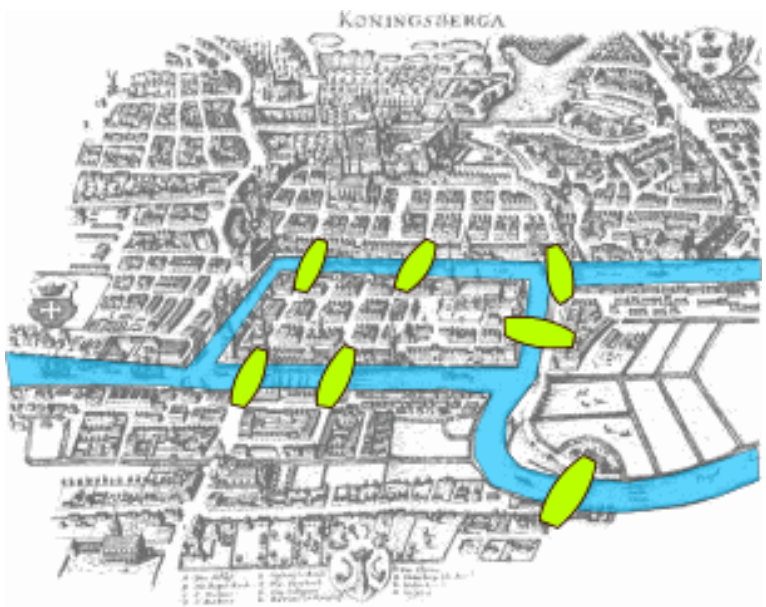
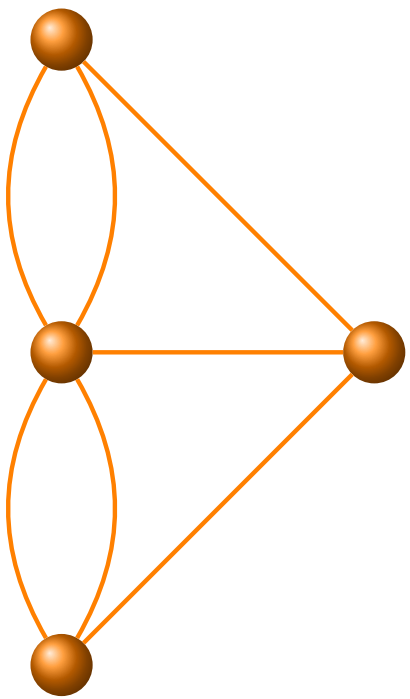


Image source:
http://en.wikipedia.org/wiki/File:Konigsberg_bridges.png.
Used under the license given there



Euler Paths and Circuits

3.2.48. Theorem. A finite connected multigraph with at least two vertices has an Euler circuit if and only if each of its vertices has even degree.

Proof.

($\Rightarrow$) Suppose that the multigraph has an Euler circuit starting at some vertex v_1. We then "follow" the Euler circuit and count 1 degree whenever we "enter" a vertex and 1 more degree when we "leave" each vertex. We count one degree for v_1 as we start by leaving v_1. Entering some other vertex, we count 1 as we enter and 1 as we leave. Hence, by traversing the edges of the graph along the circuit, we count two degrees whenever we pass through a vertex. Finally, we "enter" the vertex v_1 at the final edge of the circuit, counting another degree. Therefore, each vertex of the graph must have an even degree.

Euler Paths and Circuits

($\Leftarrow$) Suppose the multigraph G is connected and that every vertex has an even degree. We choose some vertex v_0 and arbitrarily choose an edge $\{v_0, v_1\}$. This is possible, because the graph is connected and has at least two vertices. We then choose an edge incident to v_1, which is also incident to some vertex v_2. We continue to choose edges in this way until we reach a vertex v_n which has no more edges incident to it that we have not yet used in our path.

This will occur because G is assumed to be finite and hence contains only a finite number of edges.

We now claim that $v_n = v_0$: The path "uses" one degree of each vertex to enter and one degree to "leave" each vertex. Since every vertex has an even degree, for every edge that is used to "enter" the vertex, there exists exactly one edge to "leave" the vertex. Therefore, if the path arrives at a vertex that can not be "left" by an unused edge, it must be v_0.

Euler Paths and Circuits

We have now constructed a circuit in the graph G. However, it may not be an Euler circuit, because it might not include every edge in G. Let H be the subgraph of G that contains the edges not already used. Then we have an Euler circuit of $H \setminus G$. Let w_0 be a vertex in H that is connected to a vertex in $G \setminus H$. Such a vertex exists because G is connected. Repeat the above procedure to find a circuit in H starting and ending at w_0 and then splice this circuit into the circuit of $H \setminus G$. We then obtain a new (large) circuit in G.

If there remain unused edges, we repeat the above step until no more edges remain and our circuit becomes an Euler circuit in G. Our procedure terminates because the graph G is finite. □

The construction of an Euler circuit can be made into a workable algorithm. We also note that every circuit is of course a path, so the existence of an Euler circuit implies the existence of an Euler path.

Euler Paths and Circuits

3.2.49. Theorem. A finite connected multigraph has an Euler path but not an Euler circuit if and only if it has exactly two vertices of odd degree.

Proof.

($\Rightarrow$) Suppose that the multigraph has an Euler path starting at some vertex v_1 and ending at some vertex v_n. Then the vertex v_1 must have odd degree, because it is left once more than it is entered. A similar statement applies to v_n. All other vertices are entered and then left an equal number of times. Thus, v_1 and v_n have odd degree while all other vertices have even degree.

($\Leftarrow$) Suppose that the multigraph G has two vertices, v_1 and v_n, of odd degree and that all other vertices have even degree. Add an edge $\{v_1, v_n\}$ to G. Then every vertex in G has even degree and there exists an Euler circuit starting at v_1 and ending at v_n. Removing the edge $\{v_1, v_n\}$ from the Euler circuit produces the desired Euler path. (Why exactly?) □

Hamilton Paths and Circuits

3.2.50. Definition. Let G be a finite graph.

1. A *Hamilton path* in G is a simple path that passes through every vertex of G exactly once.
2. A *Hamilton circuit* in G is a simple circuit that passes through every vertex of G exactly once.

There are no known useful necessary and sufficient conditions for the existence of Hamilton circuits, but there are a few giving sufficient conditions:

3.2.51. Dirac's Theorem. Let G be a simple graph with $n \geq 3$ vertices such that the degree of each vertex is at least $n/2$. Then G has a Hamilton circuit.

3.2.52. Ore's Theorem. Let G be a simple graph with $n \geq 3$ vertices such that the sum of the degree of any two non-adjacent vertices is at least n. Then G has a Hamilton circuit.

Hamilton Paths and Circuits

Clearly, Dirac's Theorem is a Corollary of Ore's Theorem. Ore's Theorem will be proven in the exercises.

Hamiltonian circuits appear in many applications. A famous problem that we will look at later is the *traveling salesman problem*, which asks for the perfect route a traveling salesman should take to visit a set of cities.

The Icosian Game

William Hamilton studied such paths and circuits and their symmetry properties. Out of his research he developed a game called the *Icosian game* or "A Voyage Round the World" which challenges a person to connect pegs on the edges of a dodecahedron using a string in such a way that all pegs are touched exactly once by the string.

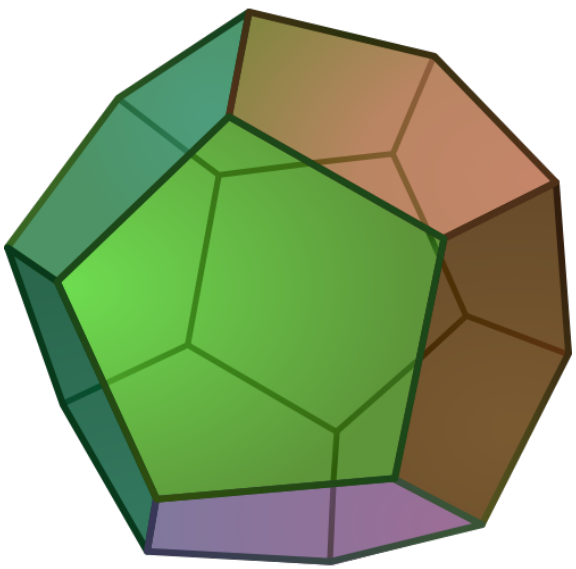


Image source:
http://commons.wikimedia.org/wiki/File:
POV-Ray-Dodecahedron.svg
Used under the license given there

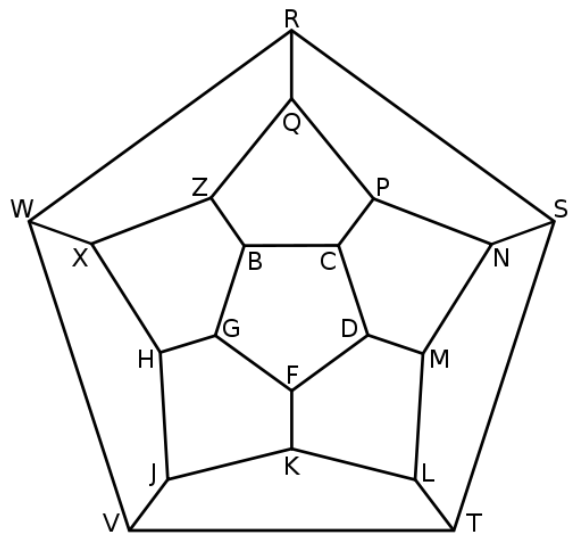


Image source:
http://commons.wikimedia.org/wiki/File:
Icosian_grid_small_with_labels.svg
Used under the license given there

The Icosian Game

A solution is shown below:

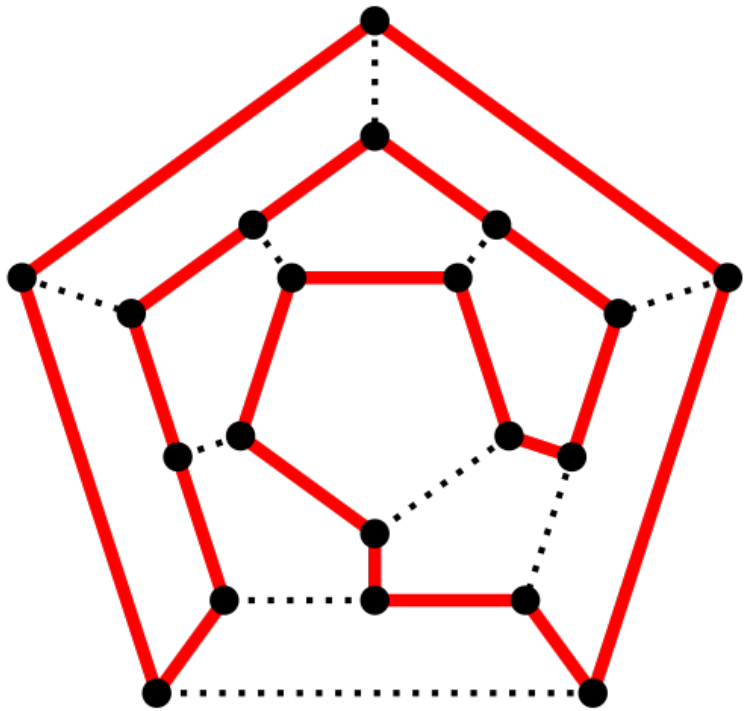


Image source:
http://commons.wikimedia.org/wiki/File:Hamiltonian_path.svg
Used under the license given there

A three-dimensional image of the solution can be found at http://mathworld.wolfram.com/IcosianGame.html

Weighted Graphs

A weighted graph is a graph where each edge is assigned a weight. For example, the image below gives the distances (in miles) between various cities in the United States:

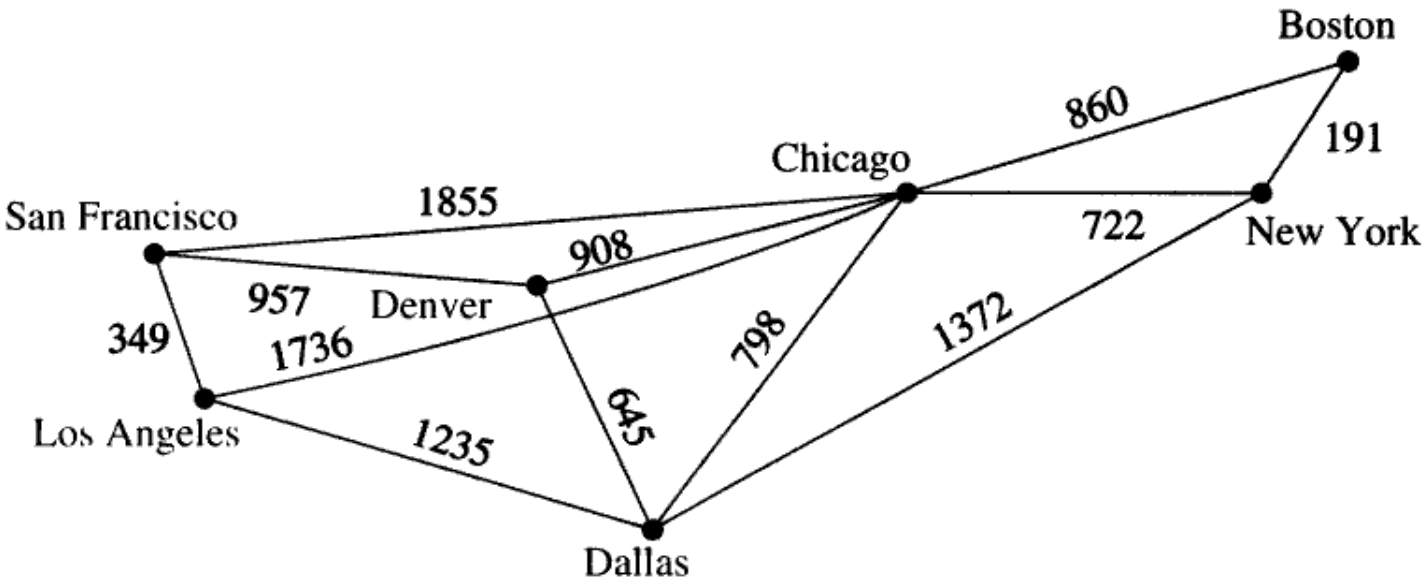


Image source: Rosen, K.H., *Discrete Mathematics*, 6th Ed., McGraw-Hill International Edition 2007
Used under fair use exemption.

We are interested in paths in weighted graphs. The *length* of a path in a weighted graph is given by the sum of the weights of the edges comprising the path. An empty path consisting of no edges is said to have length infinity.

Shortest Paths in Weighted Graphs

Suppose we are given an undirected, connected, simple, weighted path. Given two vertices a and z, our goal is to find the shortest path (i.e., the path of least length) that joins a to z. We will here present an algorithm due to Dijkstra that finds this shortest path.

The algorithm proceeds in several iterations. In the kth iteration, we will form a distinguished set of vertices, called S_k. Furthermore, in each iteration we will assign the labels $L_k(v)$ to the vertex v.

We start with the $k = 0$th iteration, setting

$$S_0 := \emptyset, \qquad L_0(v) = \begin{cases} 0 & v = a, \\ \infty & v \neq a. \end{cases}$$

We find S_{k+1} from S_k, $k \in \mathbb{N}$, by adding a vertex u to S_k with the smallest label. Then, we update the labels of all vertices not in S_{k+1} so that

$$L_{k+1}(v) = \min\{L_k(v), L_k(u) + w(u, v)\} \tag{3.2.1}$$

where $w(u, v)$ denotes the weight of the edge $\{u, v\}$.

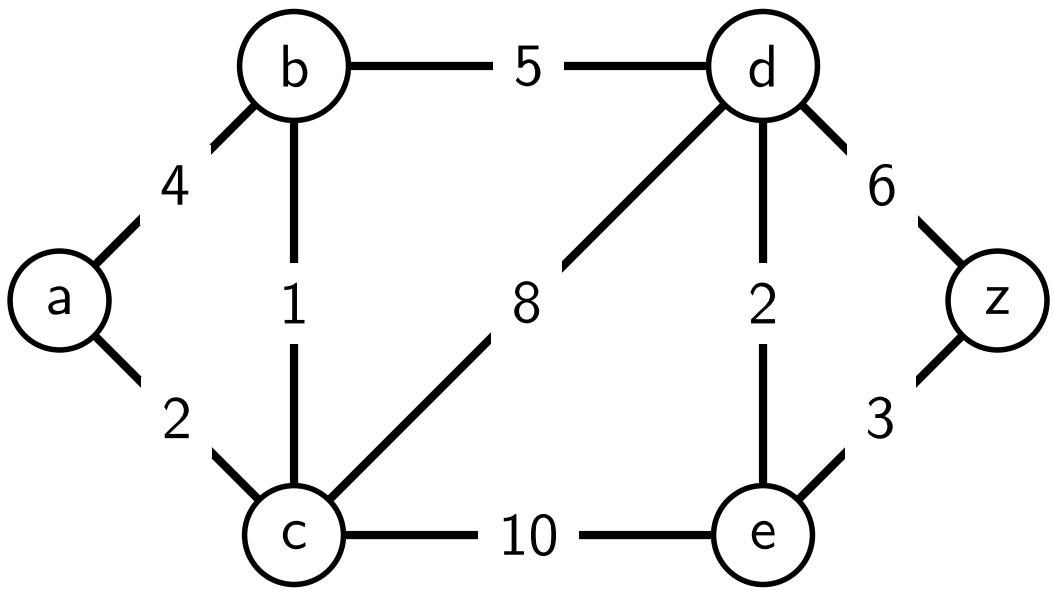
Dijkstra's Algorithm

We claim that

$$L_k(v) = \text{length of the shortest path from } a \text{ to } v \text{ that contains only vertices in } S_k.$$

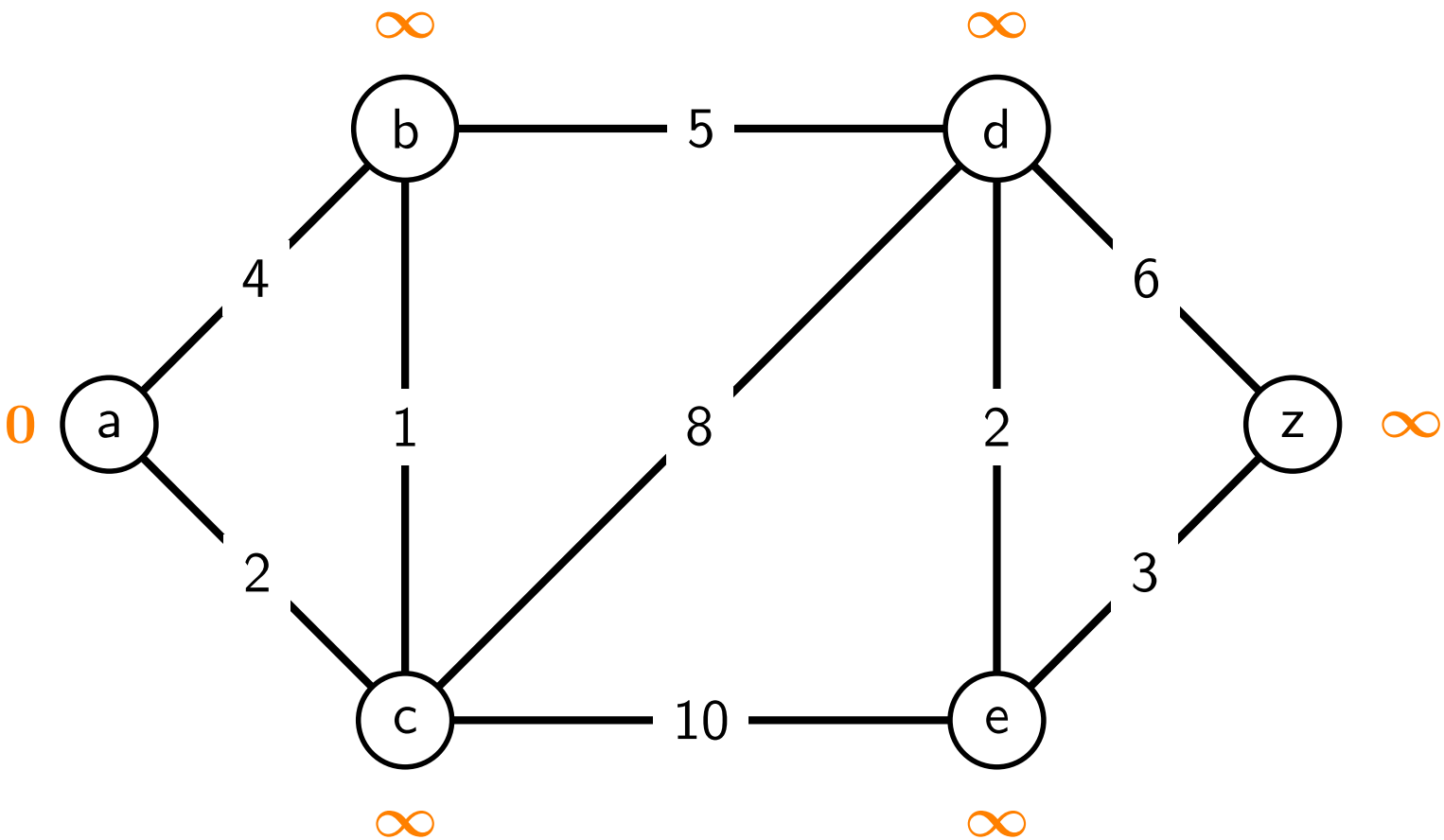
Once z is added to S_k, the length of the shortest path is given by the label of z.

3.2.53. Example. We will find the shortest path from a to z in the graph given below:



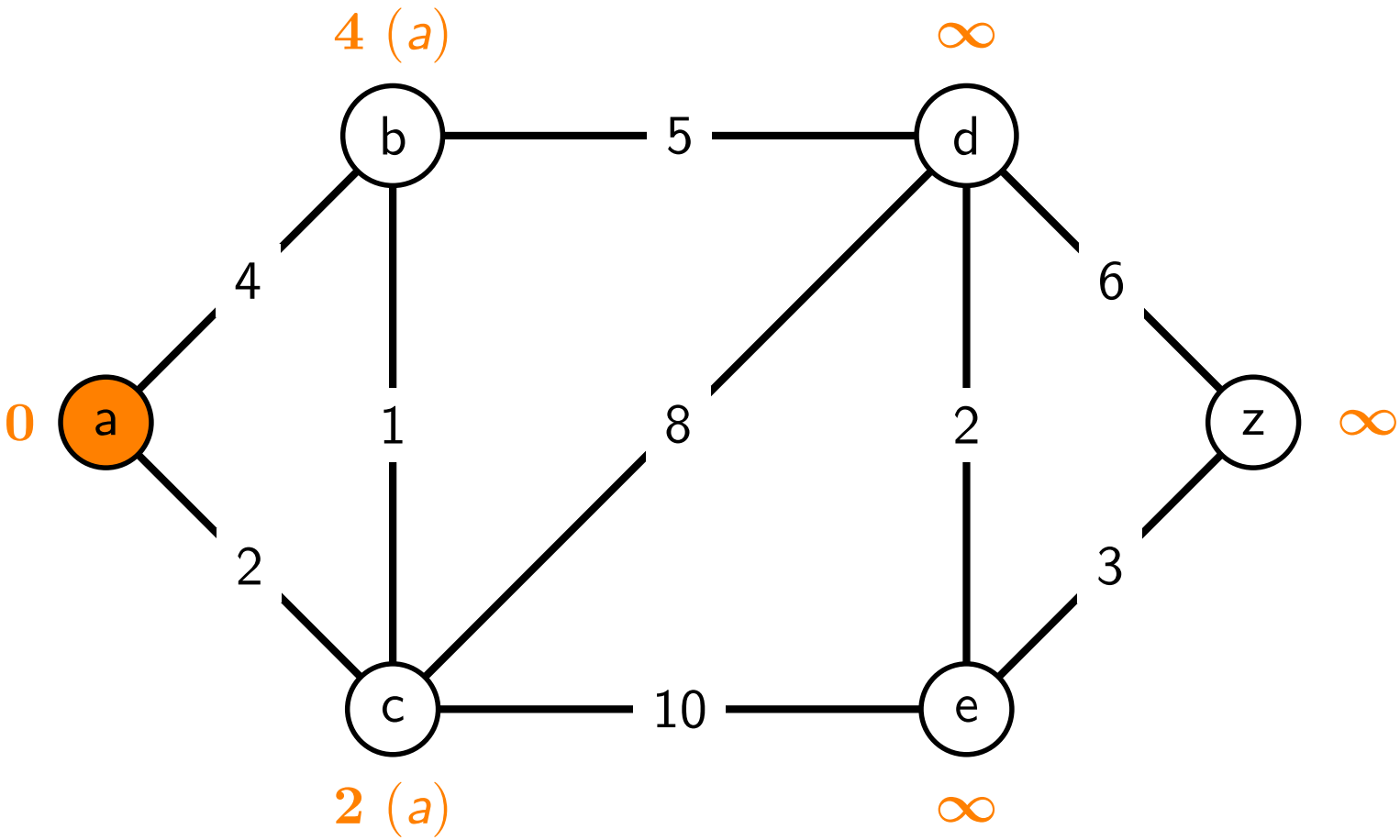
Dijkstra's Algorithm - 0^{th} Iteration

$S_0 = \emptyset$.



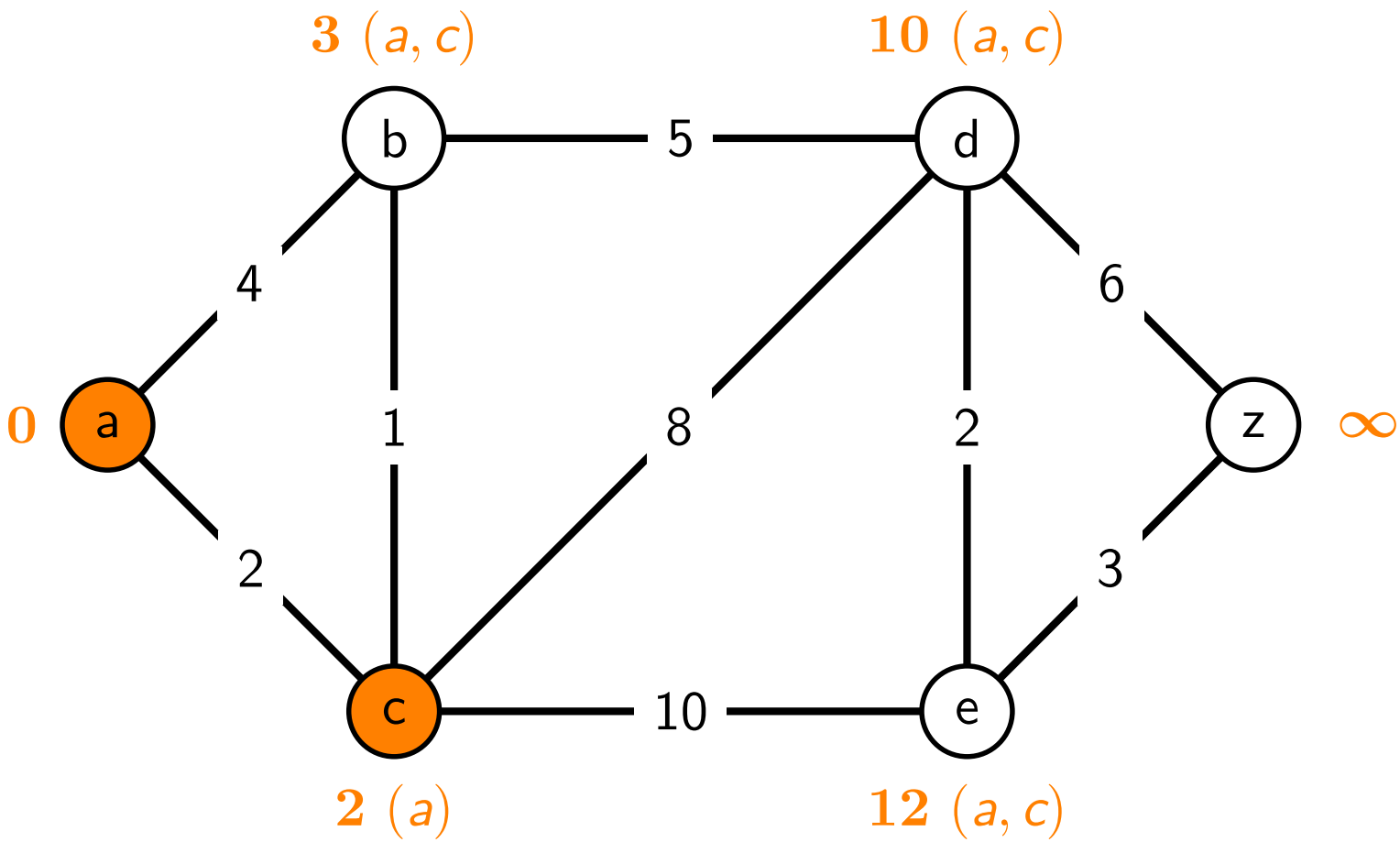
Dijkstra's Algorithm - 1st Iteration

$S_1 = \{a\}$.



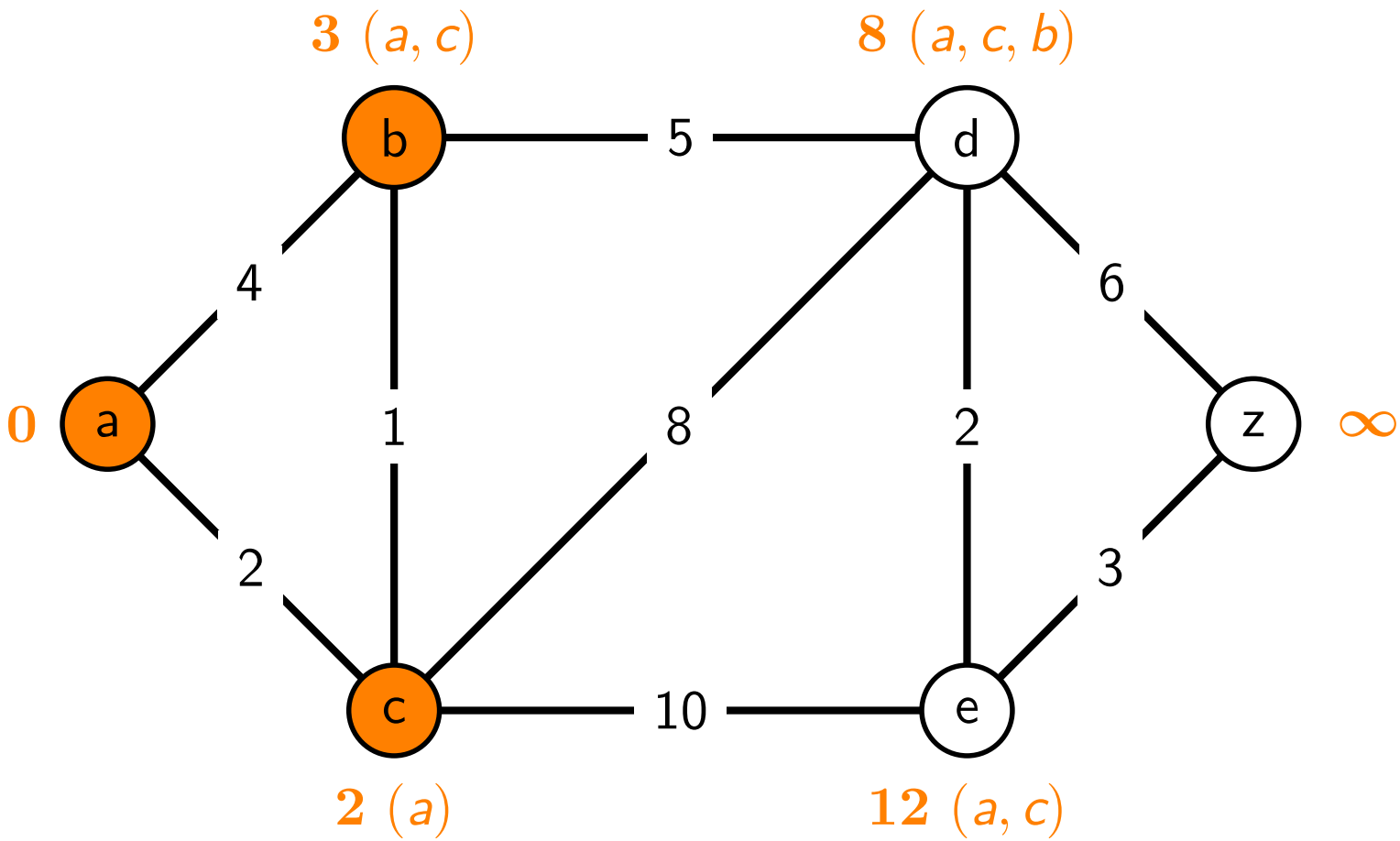
Dijkstra's Algorithm - 2nd Iteration

$S_2 = \{a, c\}$.



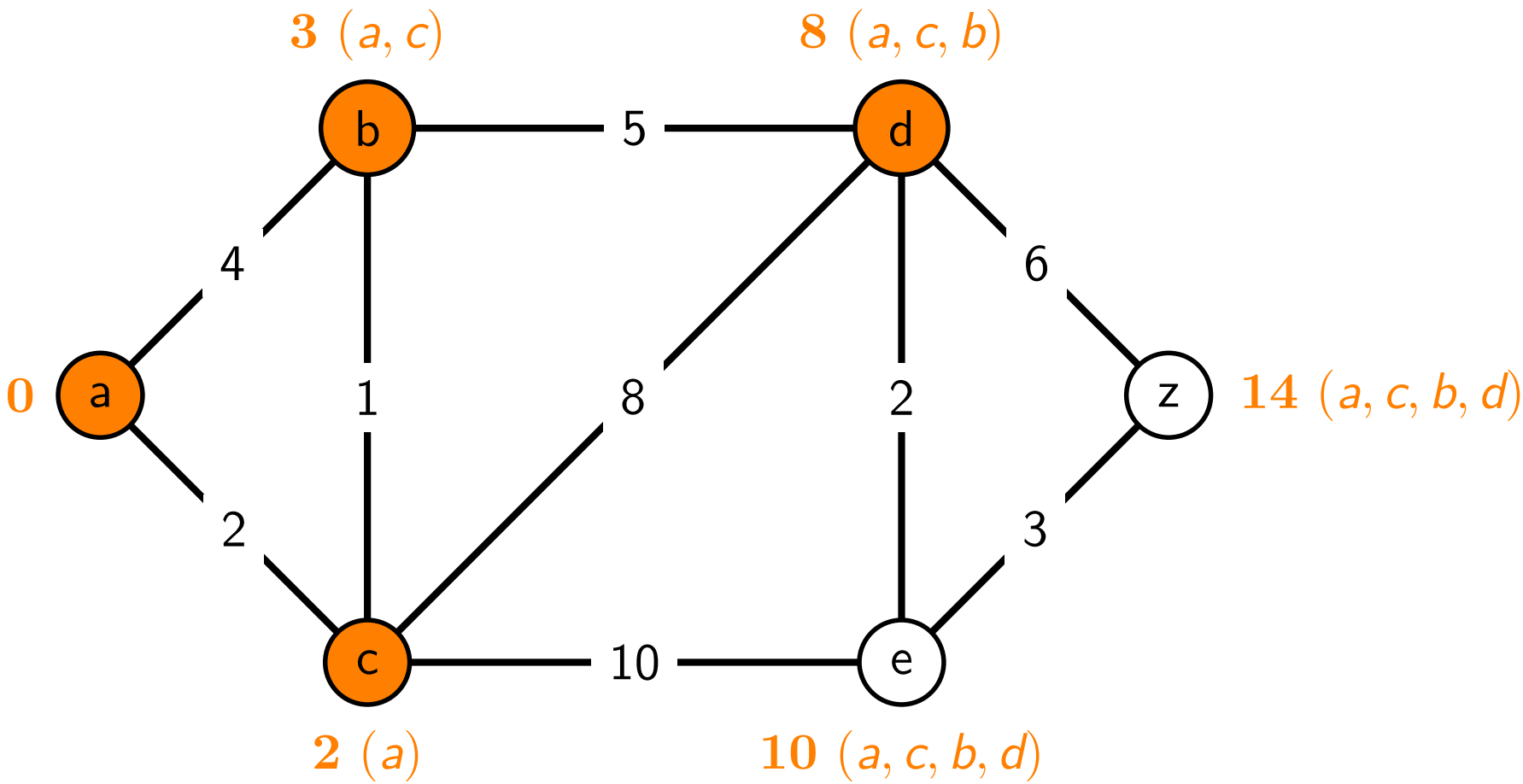
Dijkstra's Algorithm - 3rd Iteration

$S_3 = \{a, b, c\}$.



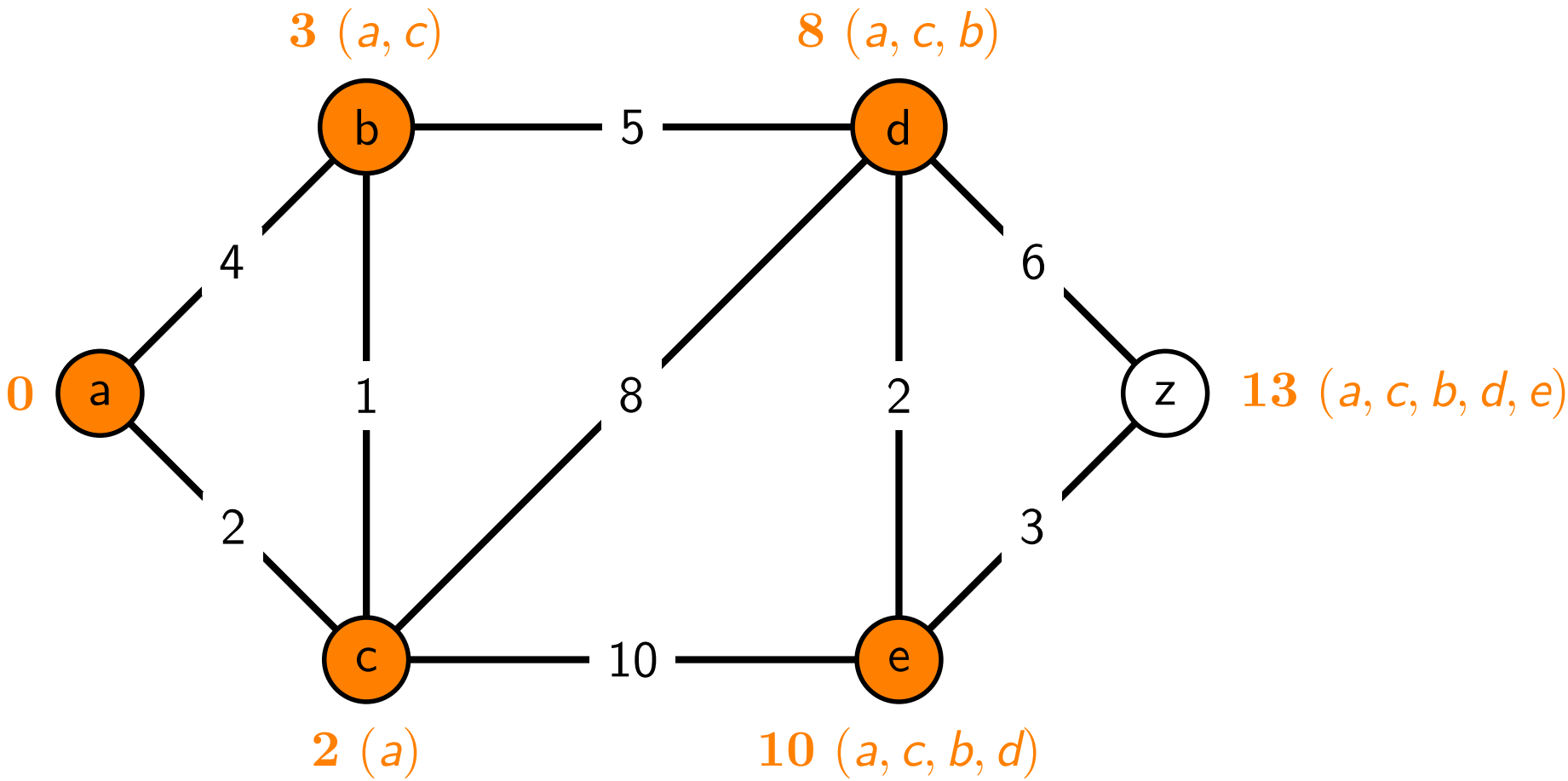
Dijkstra's Algorithm - 4th Iteration

$S_4 = \{a, b, c, d\}$.



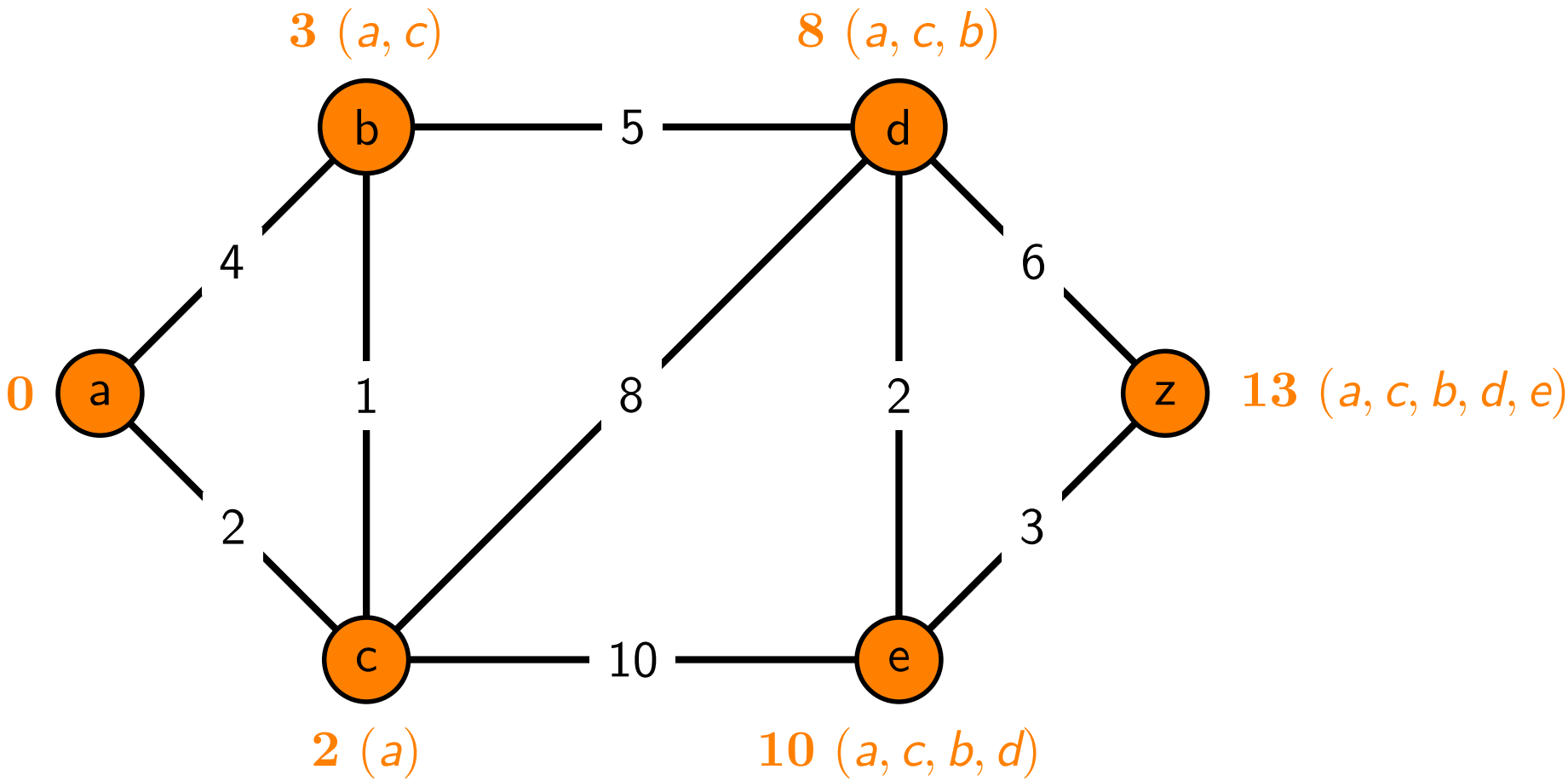
Dijkstra's Algorithm - 5th Iteration

$S_5 = \{a, b, c, d, e\}$.



Dijkstra's Algorithm - 6th Iteration

$S_5 = \{a, b, c, d, e, z\}$.



Dijkstra's Algorithm

To establish that Dijkstra's algorithm actually gives shortest path from a to z, we will prove the following result:

3.2.54. Theorem. Let G be an undirected, weighted, connected graph and a a vertex in G. At the kth iteration of Dijkstra's algorithm,

(i) The label $L_k(v)$ of every vertex $v \in S_k$ is the length of a shortest path from a to v;

(ii) The label $L_k(w)$ of every vertex $w \notin S_k$ is the length of a shortest path from a to w that contains only (besides w itself) vertices in S_k.

Proof.

We prove these statements by induction in $k \in \mathbb{N}$. For $k = 0$, $S_0 = \emptyset$, so (i) is vacuously true. Furthermore, there is no path from a to a vertex other than a using vertices in S_0, so, by definition, its length is ∞. It follows that (i) and (ii) are true for $k = 0$.

Dijkstra's Algorithm

Proof (continued).

Assume that (i) and (ii) are true for $k \in \mathbb{N}$. The vertices in S_k are hence labeled with the length of the shortest path from a. Let $u \in S_{k+1}$ but $u \notin S_k$, so u is the vertex added in the kth iteration. Thus, u has a smallest label of all vertices not in S_k.

It is clear (from (ii)) that u is labeled with the length of a shortest path containing only vertices in S_k. We claim that the label of u is actually the length of a shortest path without regard to where the path's vertices lie. We show this by contradiction: Suppose that there exists a path of length less than $L_k(u)$ from a to u containing a vertex not in S_k. Let v be the first vertex along this path which is not in S_k. Then $L_k(v) < L_k(u)$ so v should have been added to S_k instead of u, giving a contradiction. Hence the label of u is the shortest path from a to u. This implies (i) with k replaced by $k+1$.

Dijkstra's Algorithm

Proof (continued).

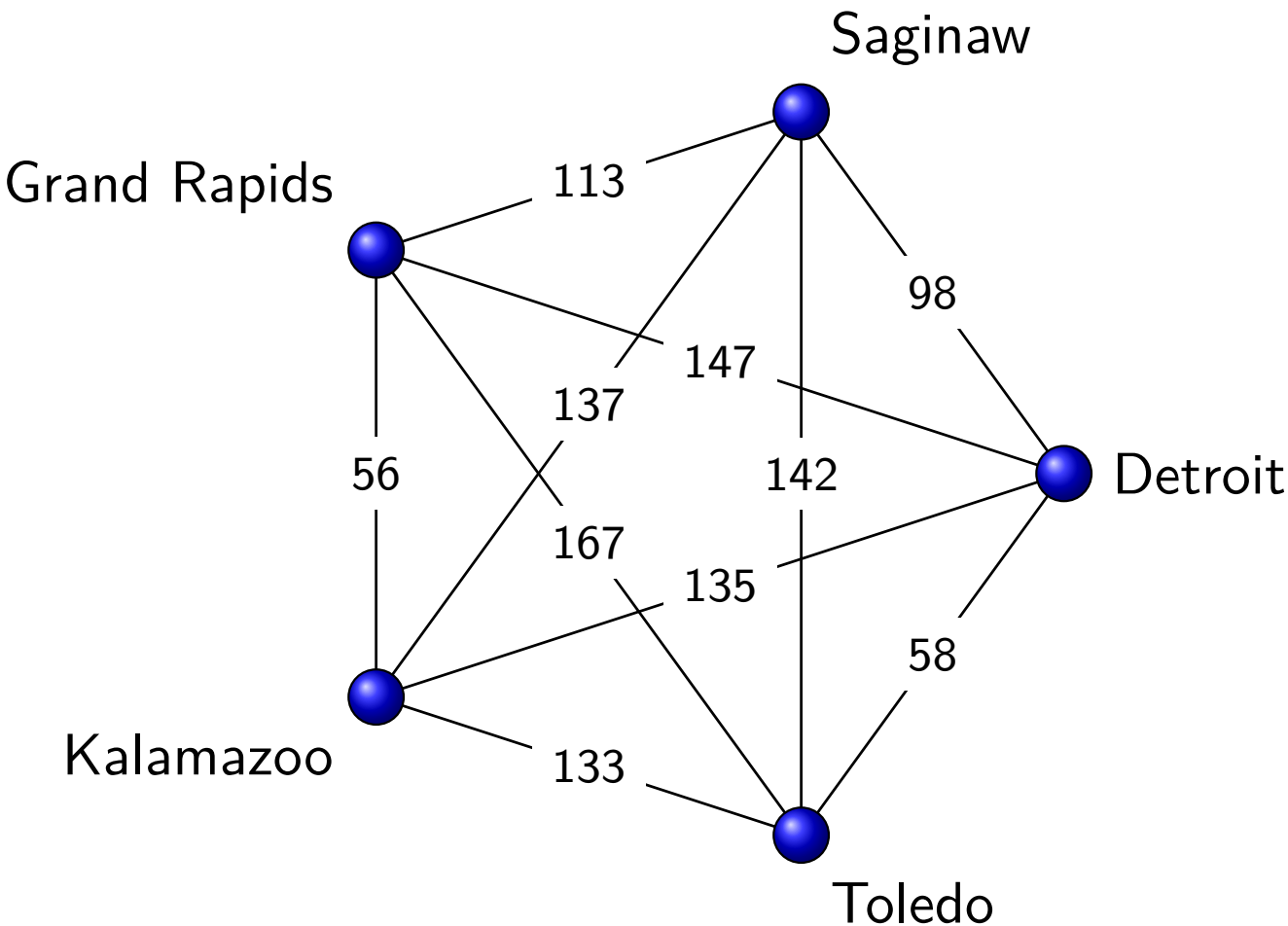
Let $v \notin S_{k+1}$. A shortest path from a to v containing only elements of S_{k+1} either contains u or it does not. If it does not contain u, then $L_{k+1}(v) = L_k(v)$ is the length of the shortest path from a to v since we assumed that (ii) is true for k. If it does contain u, the path is made up of a shortest path from a to u followed by the edge from u to v. (Why?) But then the length of this path is just given by the right-hand side of (3.2.1), which is equal to the label $L_{k+1}(v)$. Thus, (ii) is true with k replaced by $k+1$. □

3.2.55. Corollary. Dijktsra's algorithm finds the length of a shortest path between two vertices in a connected simple undirected weighted graph.

3.2.56. Theorem. Dijkstra's algorithm uses $O(n^2)$ operations (additions and comparisons) to find the length of a shortest path between two vertices in a connected simple undirected weighted graph consisting of n vertices.

The Traveling Salesman Problem

Consider the following problem: a traveling salesman needs to visit the cities Detroit, Toledo, Grand Rapids, Saginaw and Kalamazoo, shown in the graph below with the distances from each other:



The Traveling Salesman Problem

The traveling salesman problem asks: in which order should the cities be visited to minimize the total distance travelled, assuming that he starts and ends his tour in Detroit? In other words, we are looking for the shortest Hamilton circuit of the graph starting and ending in Detroit. In our example, the graph is complete, so we can visit the cities in any order. By checking each possible route individually, it turns out that the circuit

Detroit - Grand Rapids - Toledo - Saginaw - Kalamazoo - Detroit

is longest, taking 728 miles, while

Detroit - Toledo - Kalamazoo - Grand Rapids - Saginaw - Detroit

is shortest with a length of 458 miles.

The Traveling Salesman Problem

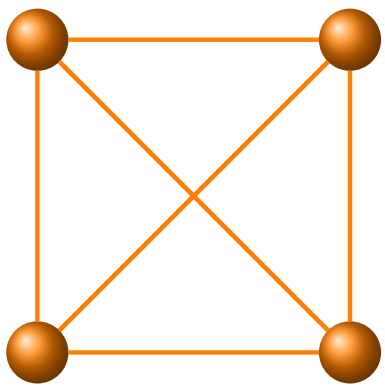
To solve the traveling salesman problem naively in a graph with n vertices, one must examine $(n-1)!/2 = O(n!)$ different Hamilton circuits – a prohibitive task for any realistic n. Due to the great practical importance of the problem, more efficient algorithms have been developed. These, however, are all exponential in time, and in fact, it has been shown that the problem belongs to the class of NP-complete problems.

There exist polynomial-time algorithms that do not find the shortest possible route (of weight $w_{\min}$) but rather a route of weight less than $c \cdot w_{\min}$, for example with $c = 3/2$. In practice, there are algorithms that solve the problem in graphs of 1000 vertices within 2% of the shortest path length in a few minutes of computer time.

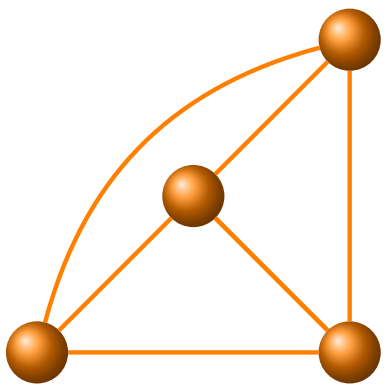
Planar Graphs

3.2.57. Definition. A graph is called *planar* if it can be drawn in the plane without any edges crossing. Such a drawing is called a *planar representation* of the graph.

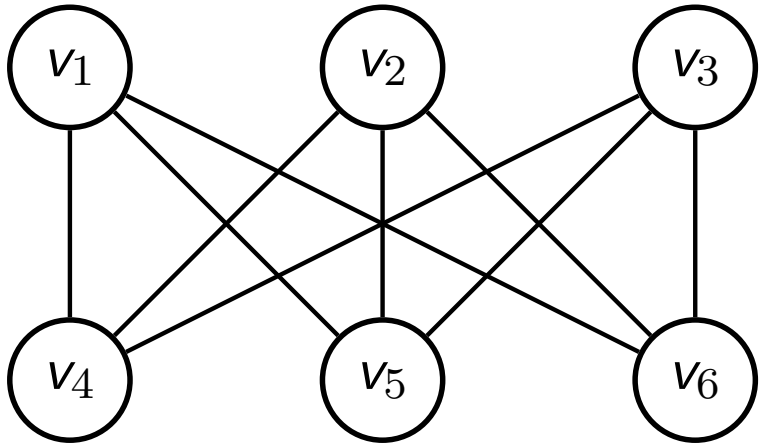
3.2.58. Example. The complete graph K_4 is planar:



can be drawn as

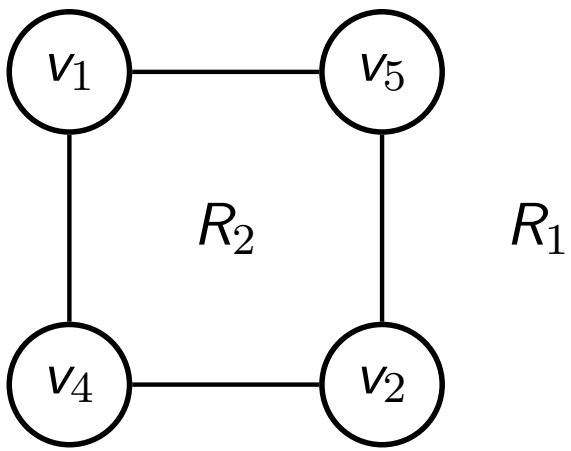


Planar Graphs



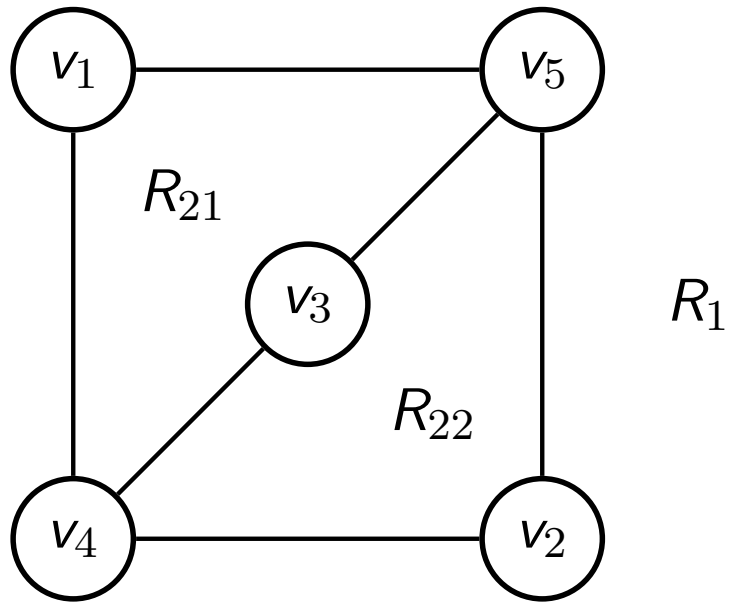
3.2.59. Example. The complete bipartite graph $K_{3,3}$ (see Example 3.2.24) is shown at right:

We will show that it is non-planar. In any planar representation, the vertices v_1 and v_2 are connected to both v_4 and v_5, splitting the plane into two regions, R_1 and R_2:



Planar Graphs

Now suppose the vertex v_3 is in the region R_2. The region R_2 is then divided into two subregions, R_{21} and R_{22}, as shown below:



Now there is no way to place the final vertex v_6 without two edges crossing: If $v_6 \in R_1$, the edge $\{v_3, v_6\}$ must cross one of the edges of the path $(v_1, v_5, v_2, v_4, v_1)$; if $v_6 \in R_{22}$, the edge $\{v_1, v_6\}$ must cross another edge; if $v_6 \in R_{21}$, the edge $\{v_2, v_6\}$ must cross another edge. A similar construction applies if $v_3 \in R_1$.

Euler's Formula

A planar representation of a finite graph in the plane splits the plane into *regions*: polygons whose edges are the edges of the graph, as well as one unbounded region.

3.2.60. Euler's Formula. Let G be a finite connected planar simple graph with e edges and v vertices. Let r be the number of regions in a planar representation of G. Then

$$r = e - v + 2.$$

Proof.

Let $G = (V, E)$ be given in a planar representation. We will construct a finite sequence $G_1, G_2, \ldots, G_e = G$ of subgraphs of G as follows: Let G_1 be the subgraph consisting of one edge of G and the two vertices incident to this edge. $G_{k+1} = (V_{k+1}, E_{k+1})$ is obtained from $G_k = (V_k, E_k)$ as follows: select any edge in $E \setminus E_k$ incident to a vertex in V_k and add it to E_k. If this edge is incident to a vertex not in V_k, add the vertex to V_k. This construction is possible because G is connected.

Euler's Formula

Let r_k, e_k and v_k denote the number of regions, edges and vertices, respectively, of G_k. We will prove by induction that

$$r_k = e_k - v_k + 2 \tag{3.2.2}$$

for all $k \in \mathbb{Z}_+$. For $k = 1$ we have one edge and two vertices and a single region, so the formula holds. Now assume that $r_k = e_k - v_k + 2$ and that the edge $\{a_{k+1}, b_{k+1}\}$ is added to G_k. We consider two cases:

(i) $a_{k+1}, b_{k+1} \in V_k$. Since the graph is planar, the vertices a_{k+1} and b_{k+1} must lie on the boundary of a common region R (otherwise the edge $\{a_{k+1}, b_{k+1}\}$ would intersect some other edge). The addition of the edge $\{a_{k+1}, b_{k+1}\}$ then splits R into two subregions. Hence,

$$r_{k+1} = r_k + 1, \qquad e_{k+1} = e_k + 1, \qquad v_{k+1} = v_k$$

and (3.2.2) remains true for k replaced with $k + 1$.

Euler's Formula

Proof (continued).

(ii) $a_{k+1} \in V_k, , b_{k+1} \notin V_k$. In this case, b_{k+1} must lie in a region that has a_{k+1} on its boundary. Hence, the edge $\{a_{k+1}, b_{k+1}\}$ does not produce a new boundary to a region and

$$r_{k+1} = r_k, \qquad e_{k+1} = e_k + 1, \qquad v_{k+1} = v_k + 1.$$

Again, (3.2.2) remains true for k replaced with $k+1$ and the proof is complete. □

3.2.61. Corollary. If G is a connected planar simple graph with e edges and $v \geq 3$ vertices, then $e \leq 3v - 6$.

The proof of Corollary 3.2.61 uses the concept of the *degree* of a region: this is the number of edges that lie on the boundary of the region, where an edge that occurs twice on the boundary contributes two to the degree.

Euler's Formula

Proof.

Let G be a connected simple planar graph with at least three vertices. Then G divides the plane into $r \in \mathbb{Z}_+$ regions, where the degree of each region is at least three. The sum of the degrees of all regions is twice the number of edges. It follows that

$$2e = \sum_{R_i} \deg(R_i) \geq 3r.$$

Using Euler's formula,

$$e - v + 2 = r \leq \frac{2e}{3}$$

from which $e \leq 3v - 6$ follows. □

Euler's Formula

3.2.62. Corollary. If G is a connected planar simple graph then G has a vertex of degree not exceeding five.

Proof.
If G has one or two vertices the result is obviously true. If G has three or more vertices, we can apply Corollary 3.2.61 to give

$$2e \leq 6v - 12.$$

If the degree of every vertex were at least six, by the Handshaking Theorem we would have

$$2e = \sum_{v_i} \deg(v_i) \geq 6v,$$

giving a contradiction. □

Euler's Formula

3.2.63. Corollary. Let G be a connected planar simple graph with e edges and $v \geq 3$ vertices. If G has no circuits of length three, then $e \leq 2v - 4$.

The proof is left to the reader.

3.2.64. Examples.

(i) The complete graph K_5 is non-planar, because it has $v = 5$ vertices and $e = 10$ edges, so $3v - 6 = 9 \not\geq 10 = e$.

(ii) From Example 3.2.59 we know that $K_{3,3}$ is non-planar. However, it has $v = 6$ vertices and $e = 9$ edges and hence satisfies $e \leq 3v - 6$, so Corollary 3.2.61 can not be used to show that it is non-planar.

(iii) The graph $K_{3,3}$ is bipartite, so it has no circuits of length three (why?). By Corollary 3.2.63 it should satisfy the inequality $e \leq 2v - 4$ if it were planar. However, $e = 9 \not\leq 8 = 2v - 4$, so it is non-planar.

Elementary Subdivisions and Homeomorphic Graphs

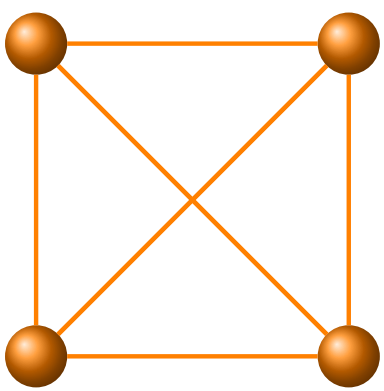
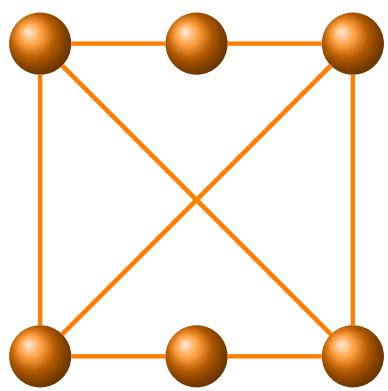
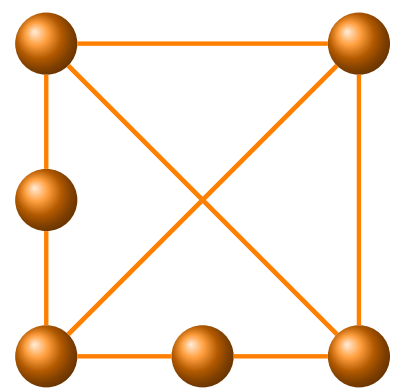
3.2.65. Definition. Let $G = (V, E)$ be a graph, $v_1, v_2 \in V$ and $\{v_1, v_2\} \in E$. Then the graph $\widetilde{G} = (\widetilde{V}, \widetilde{E})$ given by

$$\widetilde{V} = V \cup \{w\}, \qquad \widetilde{E} = (E \setminus \{v_1, v_2\}) \cup \big\{\{v_1, w\}, \{w, v_2\}\big\}$$

is said to be obtained from G through an *elementary subdivision*. We say that G is obtained from itself through an *empty elementary subdivision*.

Two graphs G_1, G_2 are said to be *homeomorphic* if there exists a graph G such that G_1 and G_2 can each be obtained from G through successive (possibly empty) elementary subdivisions.

3.2.66. Example. The following graphs are all homeomorphic:

Elementary Subdivisions and Homeomorphic Graphs

Two graphs that are homeomorphic are either both planar or non-planar:

3.2.67. Lemma. Let G_1 and G_2 be homeomorphic graphs. Then G_1 is planar if and only if G_2 is planar.

Proof.

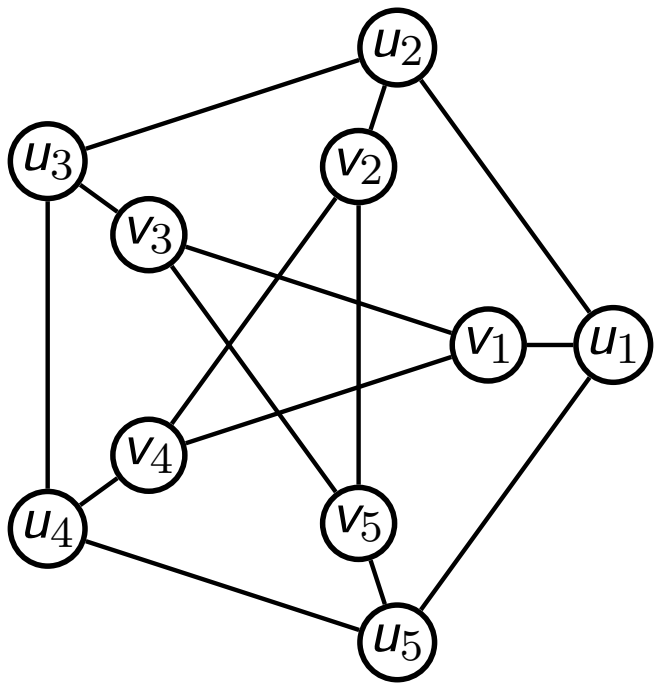
It is sufficient to show that if G_1 is planar, then G_2 is also planar. Let G be the graph from which G_1 and G_2 were obtained through elementary subdivisions. Suppose that G_1 is planar. Then G must also be planar (it simply contains fewer vertices through which the edges pass; the geometry of the edges is the same). For the same reasons the elementary subdivisions of G that yield G_2 do not cause a crossing of edges, so G_2 is also planar. □

It is clear that a graph G is non-planar if a subgraph of G is non-planar. In particular, if a subgraph of G is homeomorphic to a non-planar graph, then G is non-planar. There is a (perhaps surprising) result from Kuratowski that actually gives a specific criterion along these lines:

Kuratowski's Theorem

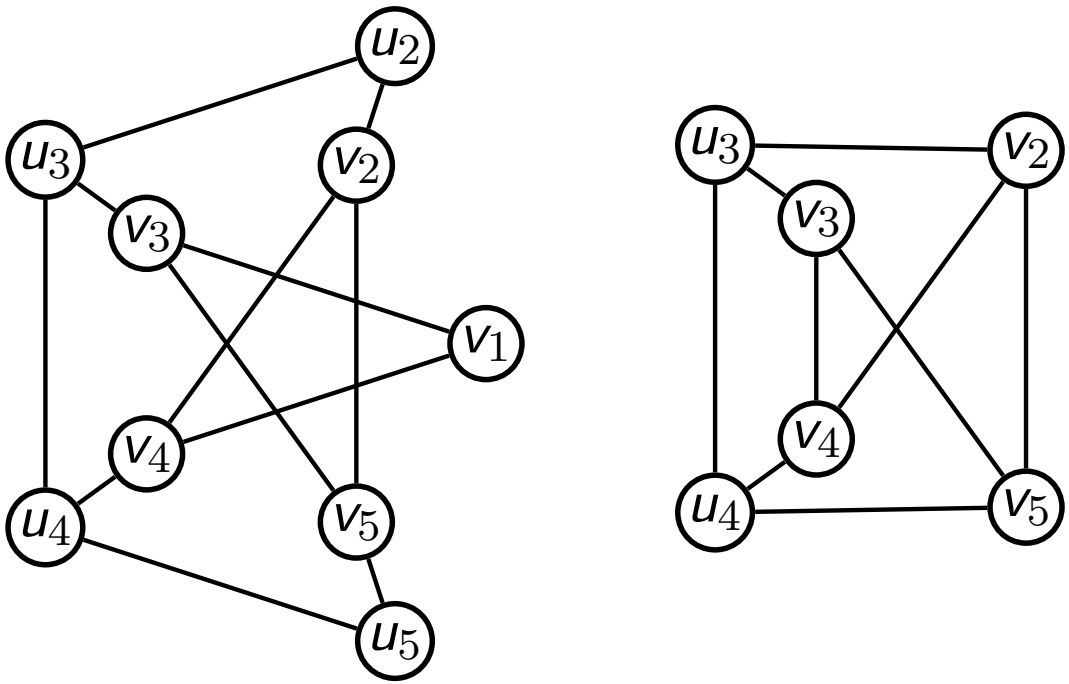
3.2.68. Kuratowski's Theorem. A graph is non-planar if and only if it contains a subgraph that is homeomorphic to $K_{3,3}$ or K_5.

3.2.69. Example. Consider the *Petersen graph*, shown below. It has no circuits of length 3, $v = 10$ vertices and $e = 15$ edges. It satisfies $e \leq 2v - 4$, so Corollary 3.2.63 does not preclude it from being planar. We will establish that it is actually non-planar.



Kuratowski's Theorem

Consider the subgraph obtained by removing u_1 and the three vertices adjacent to it (below left):



Removing the vertices u_2, u_5 and v_1 yields a homeomorphic graph. It is easy to see that this graph is actually (isomorphic to) the complete bipartite graph $K_{3,3}$ (with bipartition $(\{u_3, v_4, v_5\}, \{v_2, v_3, u_4\})$), so the Petersen graph is non-planar.

Maps and Dual Graphs

Consider a map of regions, as shown below:

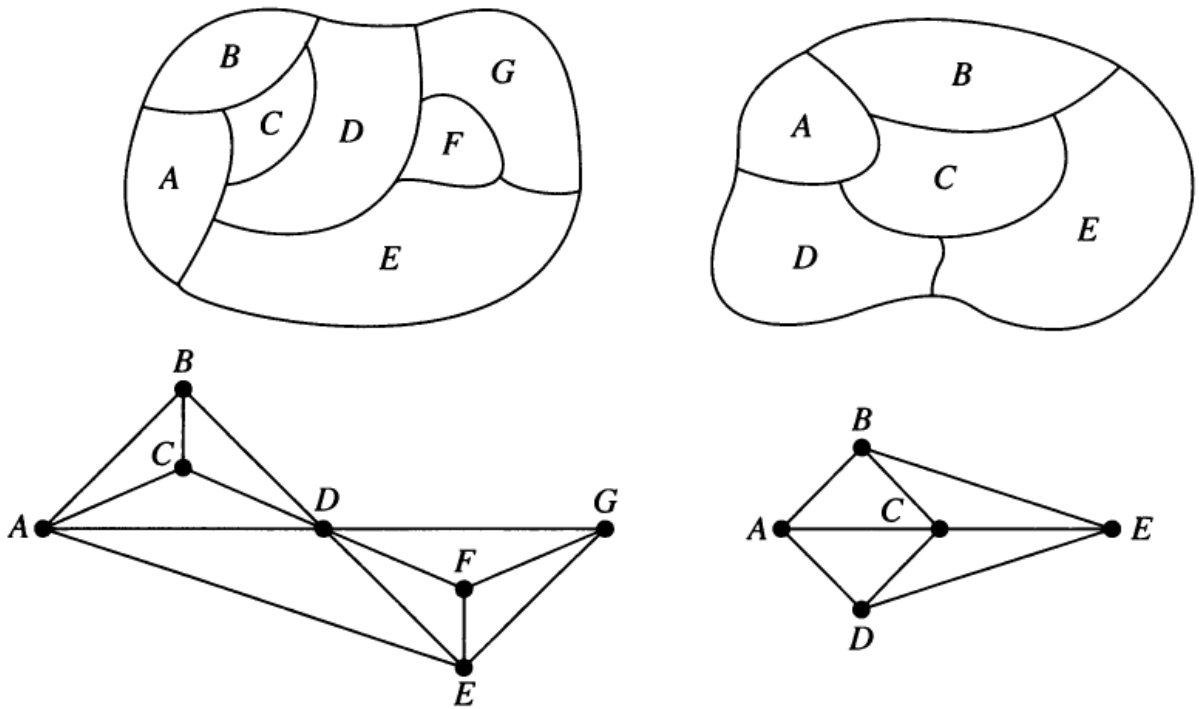


Image source: Rosen, K.H., *Discrete Mathematics*, 6th Ed., McGraw-Hill International Edition 2007. Used under fair use exemption.

The *dual graph* is constructed by identifying regions with vertices and adding edges between those vertices whose regions have a common boundary. It is clear that the dual graph is always planar.

Maps and Dual Graphs

We are interested in coloring a map (or, equivalently, its dual graph) so that no two adjoining regions (or adjacent vertices) have the same color. We formalize this as follows:

3.2.70. Definition. A *coloring* of a graph is an assignment of colors to vertices so that no two adjacent vertices have the same color.

Let G be a graph. The *chromatic number* $\chi(G)$ is the least number of colors needed for a coloring of the graph.

3.2.71. Examples.

1. $\chi(K_n) = n$,
2. $\chi(C_n) = 2$ if $n \geq 2$ and n is even,
3. $\chi(C_n) = 3$ if $n \geq 3$ and n is odd,
4. $\chi(W_n) = \chi(C_n) + 1$,
5. $\chi(K_{m,n}) = 2$.

Maps and Dual Graphs

The following theorem was first formulated as a conjecture in the 1850s and was only proven in 1976 with the aid of computers. The (at the time controversial) proof is the first example of a significant theorem that was proven in a way that would make it all but impossible for a human being to verify the result by hand.

3.2.72. Four Color Theorem. The chromatic number of a planar graph is no greater than four.

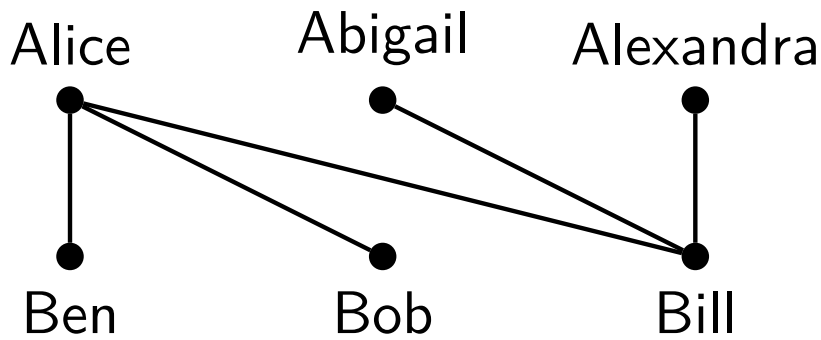
The Marriage Theorem

Reference A. Bogomolny, *Marriage Theorem* from Interactive Mathematics Miscellany and Puzzles, http://www.cut-the-knot.org/arithmetic/elegant.shtml

Suppose we have a group of n boys and n girls. Every girl writes a list of the boys that she likes and is willing to marry. We assume that all the boys are gentlemen and will not refuse the offer of a girl. Is there a criterion that allows us to determine, based on the lists submitted by the girls, if every girl will be able to marry a boy?

The Marriage Theorem

3.2.73. Example. Suppose we have three girls, Alice, Abigail and Alexandra, and three boys, Ben, Bill and Bob. Alice likes all of the boys, but Abigail and Alexandra only like Bill. It is clear that there is no match up so that every girl gets the boy she likes:



We see that this situation can be modeled by a bipartite graph. Furthermore, because every boy will accept a girl's offer to marry him, the problem is actually symmetric; we could also say that a pair (girl,boy) is *compatible* if both are willing to marry. In our formulation we merely say that a pair is compatible based on the choice of the girl; we could also, of course, reverse this without change to the problem.

The Marriage Theorem

It turns out that a "perfect matching", so that every girl gets a boy (and vice-versa) is possible if the *marriage condition* holds:

3.2.74. Marriage Condition. Suppose we have a set of n girls and n boys. We say that the marriage condition is satisfied if every set of r girls, $1 \leq r \leq n$, collectively likes at least r boys.

3.2.75. Marriage Theorem. Suppose we have a set of n girls and n boys. Then every girl will be able to marry a boy if and only if the marriage condition is satisfied.

The Marriage Theorem

Proof.

It is clear that the marriage condition is necessary. If a group of r girls collectively likes fewer than r boys, it is clearly impossible to find one boy for each of the r girls.

We show that the condition is also sufficient by strong induction in n. If $n = 1$, the condition states that the one girl likes the one boy, so a perfect match is possible. Suppose that for all $1 \leq r < n$ the marriage theorem has been established, i.e., if the marriage condition holds for r boys and r girls, then a perfect match is possible. Now assume that the marriage condition holds for n girls and n boys. We consider two cases:

(i) Every set of $r < n$ girls likes more than r boys. Select an arbitrary (boy,girl) pair and match them. Then there remain $n - 1$ girls and $n - 1$ boys that still satisfy the marriage condition. By the induction hypothesis, we can find a perfect matching in this group, so we have a matching of all $2n$ boys and girls.

The Marriage Theorem

Proof (continued).

(ii) There is a set of $1 < r_0 < n$ girls that likes exactly r_0 boys. By the induction hypothesis these $2r_0$ boys and girls can be matched up perfectly. We now need to match up the remaining $n - r_0$ girls with boys.

Consider any s of the remaining $n - r_0$ girls. From the marriage condition it follows that r_0 matched girls plus these s girls must collectively like at least $r_0 + s$ boys. Since the r_0 married girls only like the r_0 boys that they were matched with, the s girls must like at least s additional boys. Thus, any group of s girls among the $n - r_0$ remaining girls likes at least s boys among the $n - r_0$ remaining boys. It follows that the marriage condition is satisfied for the remaining $n - r_0$ girls. By the induction hypothesis, a complete matching is possible for these remaining girls with the remaining boys, providing a complete matching for all the girls. □

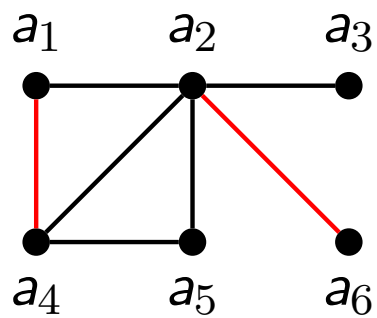
Perfect Matchings

The marriage theorem can also be formulated purely graph theoretically as follows:

3.2.76. Definition. Let $G = (V, E)$ be a graph. Then a *matching* of G is a set of pairwise non-adjacent edges of G, i.e., a subset of E such that no two edges are incident to the same vertex.

A *perfect matching* of G is a matching of G such that every vertex of G is incident to one of the edges in the matching.

3.2.77. Example. Consider the graph shown below:



The red edges represent a matching of the graph. Other matchings exist, but a perfect matching is not possible.

Perfect Matchings

We now define the marriage condition as follows:

3.2.78. Marriage Condition. Let G be a bipartite simple graph consisting of $2n$ vertices such that there exists a bipartition (V_1, V_2) with card $V_1 = \text{card } V_2 = n$. Then the marriage condition is satisfied if and only if the zero-one adjacency matrix of G has no $r \times s$ zero-submatrix with $r + s > 2n$.

It is left to the reader to verify that this is equivalent to the Marriage Condition 3.2.74 when the set of boys and girls with their respective compatibilities is interpreted as a bipartite graph. In this context, the marriage theorem can be stated as follows:

3.2.79. Marriage Theorem. Let G be a bipartite simple graph consisting of $2n$ vertices and assume that the marriage condition is satisfied. Then there exists a perfect matching for G.

The Harem Theorem

There is a generalization of the marriage theorem called the harem (後宮) theorem:

3.2.80. Harem Theorem. Suppose that we have n girls and at least $m \cdot n$ boys. Suppose that every girl likes at least m boys and she can marry as many boys as she wants (boys may not refuse marriage offers). Then every girl will be able to marry m boys she likes if and only if any set of r girls, $1 \leq r \leq n$, collectively likes $m \cdot r$ boys.

The proof is left to the reader. (Hint: make m copies of every girl and apply the marriage theorem.)

The marriage theorem is a fundamental theorem of combinatorics and connected to many other important results in this field.

Relations

Graphs

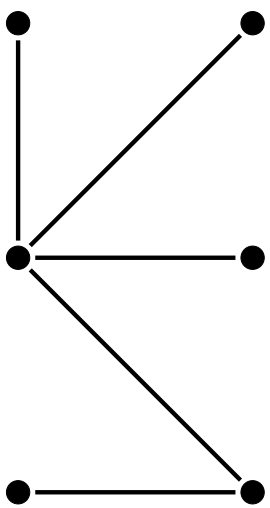
Trees

Trees

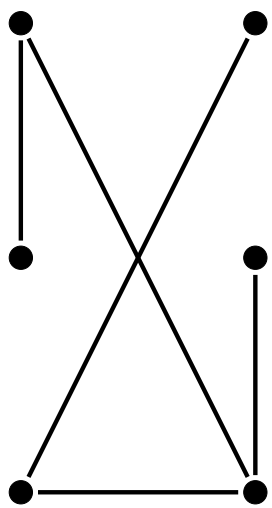
We have encountered rooted trees before in Definition 1.4.12. We will now investigate trees more extensively.

3.3.1. Definition. A *tree* is a connected undirected graph with no simple circuits. A not necessarily connected graph with no simple circuits is called a *forest*.

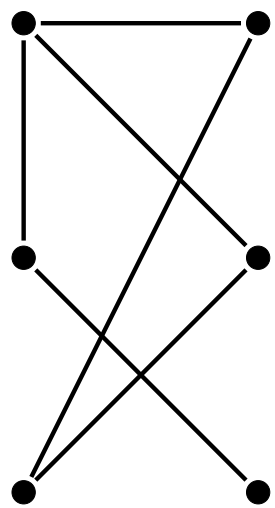
3.3.2. Example. Of the graphs below, G_1 and G_2 are trees, while G_4 is a forest.



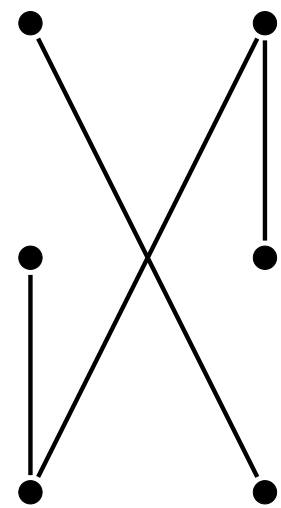
G_1



G_2



G_3



G_4

Trees and Simple Paths

Trees can also be characterized by paths. We first give a theorem on the existence of simple paths in general undirected graphs.

3.3.3. Theorem. There is a simple path between every pair of distinct vertices of a connected undirected graph.

Proof.
Let $G = (V, E)$ be a connected undirected graph and $u \neq v$ two vertices. Since G is connected, there exists at least one path joining u and v. Let $(u, x_1, \ldots, x_{n-1}, v)$ be the tuple of vertices associated to such a path having least length. We then claim that this path is simple. If it is not simple, the $x_i = x_j$ for some $i < j$. However, then the tuple of vertices $(x_{i+1}, \ldots, x_j)$ can be eliminated from the path to give a shorter path, yielding a contradiction. □

Trees and Simple Paths

The following theorem gives an equivalent characterization of trees which is sometimes used as a definition.

3.3.4. Theorem. An undirected graph is a tree if and only if there is a unique simple path between any two of its vertices.

Proof.

($\Rightarrow$) Suppose that T is a tree. Then T is a connected graph with no simple circuits. Let x, y be distinct vertices. Then, since T is connected, there exists a simple path joining x and y by Theorem 3.3.3. This path must be unique, for if there were a second such path the two paths could be joined to yield a circuit. This circuit could further be reduced to a simple circuit (see assignments), yielding a contradiction.

Trees and Simple Paths

Proof (continued).

($\Leftarrow$) Suppose that T is a graph and that there exists a unique simple path between any two vertices of T. Then T is clearly connected. Furthermore, T can not have a simple circuit: if there were such a circuit starting and ending at a vertex x and passing through a vertex y, then there would be two distinct simple paths joining x and y. But this contradicts the assumption that there exists a unique simple path between x and y. $\square$

Rooted Trees

3.3.5. Definition. A *rooted tree* is a tree in which one vertex has been designated as the root and every edge is imbued with a direction in such a way that the root is the initial vertex of any path joining the root to any other vertex.

Rooted trees can also be defined recursively (see Definition 1.4.12). Definition 3.3.5 suffices to imbue each edge of a tree with a unique direction (why?).

Rooted Trees

We next introduce some terminology:

3.3.6. Definition. Let T be a rooted tree with root r.

- If $v \neq r$ is a vertex in T, the *parent* of v is the unique vertex u such that there is a directed edge (u, v) in T.
- If u is the parent of v, the v is a *child* of u.
- If v_1, v_2 have the same parent u, they are called *siblings*.
- If $v \neq r$, the *ancestors* of v are vertices in the unique directed path joining r to v, excluding v and r.
- The *descendants* of a vertex u are all those vertices which have u as an ancestor.
- A vertex with no children is called a *leaf*, otherwise it is called an *internal vertex*.
- If a is a vertex in T, the *rooted subtree* with root a is the tree whose vertices comprise a and all its descendants and whose edges comprise all the edges joining a to these descendants.

m-ary Trees

3.3.7. Definition. A rooted tree is called an *m-ary tree* if every internal vertex has no more than m children. The tree is called a *full m-ary tree* if every internal vertex has exactly m children. A (full) m-ary tree with $m = 2$ is called a (full) *binary tree*.

Full binary trees where previously encountered in Definition 1.4.16.

3.3.8. Definition. An *ordered rooted tree* is a tree where the children of each vertex are ordered.

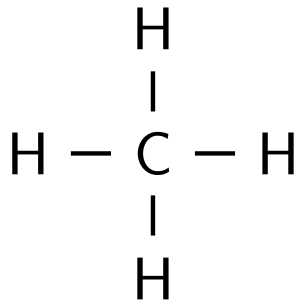
When drawing a rooted tree in the conventional way, we consider the children to be ordered from left to right. Unless stated otherwise, all depicted trees are considered to be ordered in this way.

For (ordered) binary trees we define the *left child* and the *right child* of a root in the obvious way. If u is an internal vertex, the (rooted) subtree with the left/right child of u as its root is called the *left/right subtree of u*.

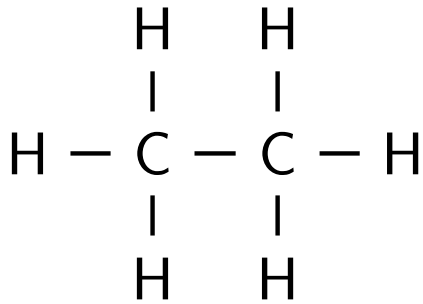
Application to Saturated Hydrocarbons

A *saturated hydrocarbon*, or *alkane*, has the chemical formula C_nH_{2n+2}. Examples include

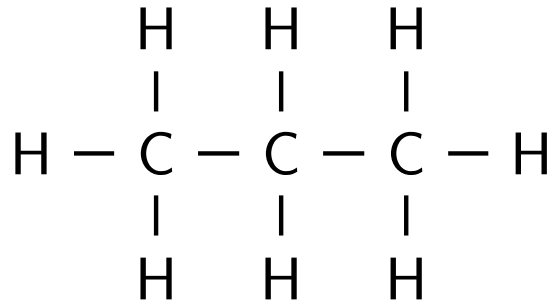
Methane (CH_4)



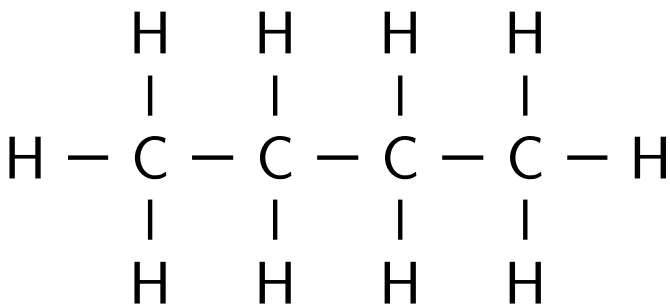
Ethane (C_2H_6)



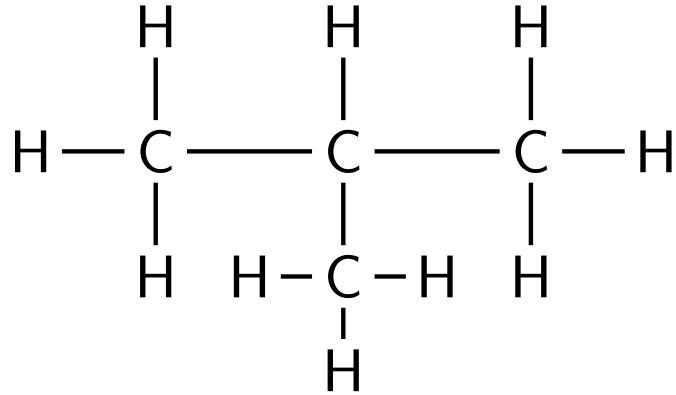
Propane (C_3H_8)



n-Butane (C_4H_{10})



iso-Butane (Methylpropane; C_4H_{10})



Application to Saturated Hydrocarbons

We can define saturated hydrocarbons inductively as follows:

3.3.9. Definition.

(i) The rooted tree consisting of one root and four children is an alkane.

(ii) If T is a rooted tree, then the tree obtained by adding three children to a leaf is an alkane.

In this definition, the internal vertices will represent carbon atoms, while the leaves represent hydrogen atoms.

Definition 3.3.9 will yield alkanes that are different regarded as rooted trees, but whose underlying undirected graphs are isomorphic. These are considered to represent the same molecule.

Alkanes with the same number of internal vertices and the same number of leaves that are not isomorphic are called *stereoisomers*. Finding the number of stereoisomers of a given compound is a non-trivial problem involving graphs and counting. More information on this problem can be found at http://www.cs.uwaterloo.ca/journals/JIS/cayley.html

Counting the Vertices and Edges of Trees

3.3.10. Theorem. A tree with n vertices has $n-1$ edges.

Proof.
The proof follows easily by induction: for $n=1$ the tree consists only of a single vertex and no edges, so the statement is true.

Suppose the statement is true for any tree with n vertices and let T be a tree with $n+1$ vertices. Then T has a leaf, which can be removed along with the vertex joining it to its parent. The resulting tree has $n-1$ vertices, and, by the induction hypothesis, $n-1$ vertices. Restoring the eliminated leaf and its vertex, we see that T has n edges. □

Counting the Vertices and Edges of Trees

3.3.11. Theorem. A full m-ary tree with i internal vertices contains $n = mi + 1$ vertices.

Proof.

There are two types of vertices in a full m-ary tree: the root and vertices that are children of other vertices. Since there are i internal vertices, there must be $m \cdot i$ children. Adding the root, we obtain$n = mi + 1$. □

3.3.12. Theorem. A full m-ary tree with

(i) n vertices has $i = (n-1)/m$ internal vertices and $l = [(m-1)n + 1]/m$ leaves,

(ii) i internal vertices has $n = mi + 1$ vertices and $l = (m-1)i + 1$ leaves,

(iii) l leaves has $n = (ml-1)/(m-1)$ vertices and $i = (l-1)/(m-1)$ internal vertices.

Proof.

The theorem follows from the two equations $n = mi + 1$ and $n = i + l$. □

Counting the Vertices and Edges of Trees

3.3.13. Example. Suppose that someone starts a chain letter. Each person who receives the letter is asked to send it on to four other people. Some people do this, but others do not send any letters. How many people have seen the letter, including the first person, if no one receives more than one letter and if the chain letter ends after there have been 100 people who read it but did not send it out? How many people sent out the letter?

Balanced m-ary Trees

3.3.14. Definition. Let T be a rooted tree with root r.

- The *level* of a vertex $v \neq r$ is the length of the path joining r to v. The level of r is defined to be zero.
- The *height* of T is the maximum of all the levels of the vertices of T.
- A rooted m-ary tree of height h is said to be *balanced* if all leaves have levels h or $h-1$.

3.3.15. Theorem. An m-ary tree of height h has at most m^h leaves.

Proof.
We prove the statement by strong induction in h. The statement is true for $h = 0$, because the tree then consists of a single vertex, the root. This is a leaf, so the number of leaves is $m^0 = 1$.

Balanced m-ary Trees

Proof (continued).

Assume T is a tree of height h. Removing the root and all edges originating at the root, one obtains a forest of at most m tress of height at most $h-1$. Each tree has at most m^{h-1} leaves by the induction hypothesis. Hence, the total number of leaves is at most $m \cdot m^{h-1} = m^h$. □

3.3.16. Corollary.

(i) If an m-ary tree of height h has l leaves, then $h \geq \lceil \log_m l \rceil$.

(ii) If the m-ary tree is full and balanced, then $h = \lceil \log_m l \rceil$.

Proof.

Statement (i) follows directly from $l \leq m^h \Leftrightarrow h \geq \log_m l$. Statement (ii) will be proven in the assignments. □

Binary Search Trees

Suppose we have a totally ordered set of objects that we want to enter into a database for easy lookup. One technique involves the use of binary trees as follows:

1. The first object becomes the root of the search tree.
2. The second object becomes a left child if it precedes the root, or a right child if it succeeds it.
3. The third and further objects are sorted along the tree, moving left if they precede a vertex or right if they succeed it.

3.3.17. Example. We enter the words mathematics, physics, geography, zoology, meteorology, geology, psychology, and chemistry (in this order) into a search tree, using the standard lexicographic ordering of words.

Locating and Adding Items to a Binary Search Tree

3.3.18. Algorithm. Locating and adding items to a binary search tree

procedure: *insertion*(T: binary search tree, x: item)
  $v :=$ root of T {a vertex not present in T has the value *null*}
  while $v \neq$ *null* and *label*$(v) \neq x$
    begin
      if $x <$ *label*(v) **then**
        if left child of $v \neq$ *null* **then** $v :=$ left child of v
        else add new vertex as a left child of v and set $v :=$ *null*
      else
        if right child of $v \neq$ *null* **then** $v :=$ right child of v
        else add new vertex as a right child of v and set $v :=$ *null*
  if root of $T =$ *null* then add a vertex v to the tree and label it with x
  else if $v =$ *null* or *label*$(v) \neq x$ **then** label new vertex with x and let v be this new vertex
  {v is the location of x}

Locating and Adding Items to a Binary Search Tree

We now analyze the computational complexity of Algorithm 3.3.18. First, given a binary tree T with n labeled vertices, we add unlabeled vertices to it so that every labeled vertex has two children. This gives a full binary tree, which we denote by U. The labeled vertices are the interior vertices of U. The advantage of this procedure is that when locating or adding an item, we never increase the number of vertices in U.

The most comparisons needed to add an item is the length of the longest path in U from its root to a leaf, i.e., the heigh h of U. The vertices of T are the interior vertices of U, so U has n interior vertices. By Theorem 3.3.12, U then has $n+1$ leaves. Corollary 3.3.16 gives

$$h \geq \lceil \log_2(n+1) \rceil.$$

If T is balanced we actually have equality, so the algorithm requires at most $\lceil \log(n+1) \rceil = O(\log n)$ comparisons. As items are added to a search tree, it may become unbalanced. There are algorithms (discussed in courses on data structures) that can "rebalance" a tree in this case.

Decision Trees

A rooted tree in which each internal vertex corresponds to a decision, with a subtree at these vertices for each possible outcome of the decision, is called a *decision tree*. The possible solutions of the problem correspond to the paths to the leaves of this rooted tree. This is illustrated through the following example:

3.3.19. Example. Suppose that among eight nominally identical coins one is counterfeit and lighter than the others. This coin may be found by weighing any number of the eight coins on a balance scale. We wish do determine the minimum number of weighings needed to find the counterfeit coin.

Finding a Counterfeit Coin

Every weighing corresponds to an internal vertex in a decision tree and every result is a child of this weighing and also corresponds to the next weighing. Each weighing has three possible outcomes:

- "left side is lighter"
- "right side is lighter"
- "no side is lighter"

so the decision tree is a 3-ary tree. The final result after the weighings ("coin no. n is counterfeit") corresponds to a leaf in the tree. There are 8 possible leaves, so, by Corollary 3.3.16 the height of the tree is at least $\lceil \log_3 8 \rceil = 2$. Hence, at least 2 weighings are necessary.

Finding a Counterfeit Coin

A naive approach easily shows that the coin can be found in three weighings. However, it is also possible to do so in just two weighings, as illustrated below:

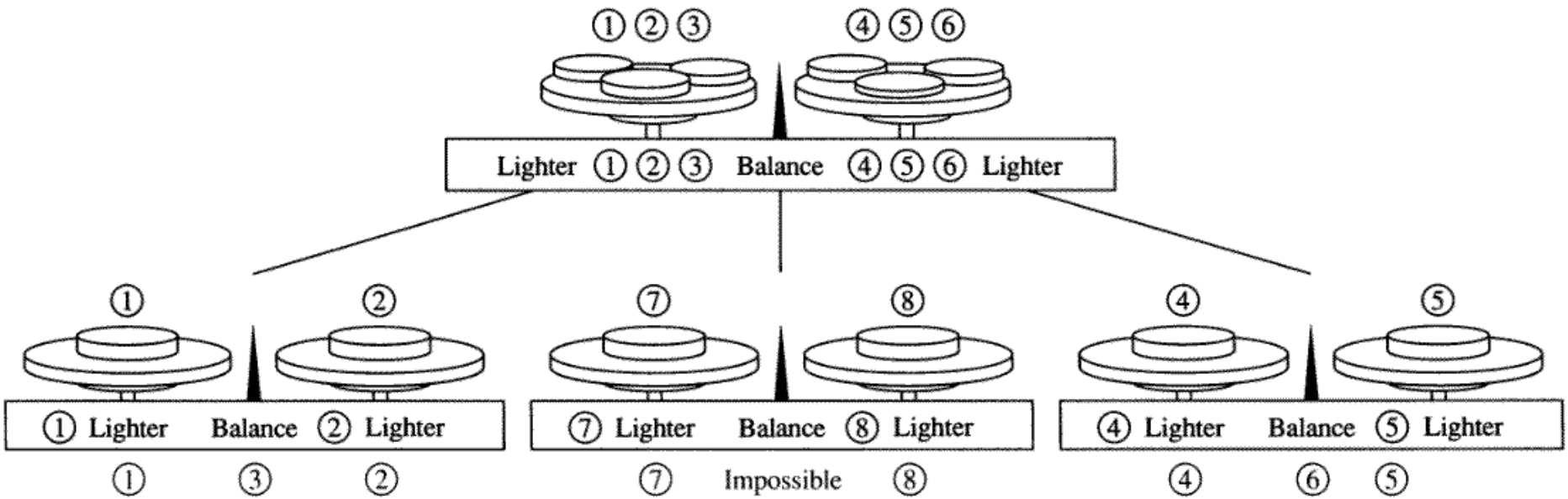


Image source: Rosen, K.H., *Discrete Mathematics*, 6th Ed., McGraw-Hill International Edition 2007. Used under fair use exemption.

Complexity of Binary Sorting Algorithms

Consider a sorting algorithm based on comparing two configurations of a given list (usually by comparing two possible positions of a single list item). Such an algorithm is called a *binary sorting algorithm*, because each comparison will designate one of two possible list configurations as being "more sorted". All of the sorting algorithms we have studied are of this type.

We can model any binary sort algorithm using a binary decision tree. For a list of n distinct elements there are $n!$ different arrangements, one of which will correspond to the sorted list. Hence, the decision tree has $n!$ leaves and, by Corollary 3.3.16 the height of the tree is at least $\lceil \log_2 n! \rceil$ and at least that many comparisons are needed.

Since $\log n! = \Theta(n \log n)$, it follows that the worst-case time complexity of any binary sort is $\Omega(n \log n)$. Thus, it is impossible to find a binary search algorithm that has a lower time complexity than $O(n \log n)$. The merge sort (Algorithm 1.7.19) may be considered optimal in this sense.