Discrete mathematics

Petr Kovář
petr.kovar@vsb.cz

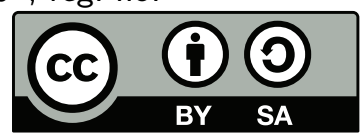
VŠB – Technical University of Ostrava

Winter term 2021/2022
DiM 470-2301/02, 470-2301/04, 470-2301/06

EUROPEAN UNION
European Structural and Investment Funds
Operational Programme Research,
Development and Education

MINISTRY OF EDUCATION,
YOUTH AND SPORTS

The translation was co-financed by the European Union and the Ministry of Education, Youth and Sports from the Operational Programme Research, Development and Education, project "Technology for the Future 2.0", reg. no. CZ.02.2.69/0.0/0.0/18_058/0010212.



About this file

This file is meant to be a guideline for the lecturer. Many important pieces of information are not in this file, they are to be delivered in the lecture: said, shown or drawn on board. The file is made available with the hope students will easier catch up with lectures they missed.

For study the following resources are better suitable:

- Meyer: Lecture notes and readings for an http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-042j-mathematics-for-computer-science-fall-2005/readings/" (weeks 1-5, 8-10, 12-13), MIT, 2005.
- Diestel: Graph theory http://diestel-graph-theory.com/ (chapters 1-6), Springer, 2010.

See also http://homel.vsb.cz/~kov16/predmety_dm.php

Course number: 470-2301/02, 470-2301/04*, 470-2301/06
Credits: 6 credits (2/2/2), *5 credits (2/2/1)
Warrant: Petr Kovář
Lecturer: Petr Kovář/Tereza Kovářová

Web: am.vsb.cz/kovar
Email: Petr.Kovar@vsb.cz
Office: EA536

Classification

Tests

- every week (starting with the third week)
- 2–10 minutes
- evaluation 0/1/2 (no/almost correct/completely correct)
- every other week one additional teoretical question
- we take 4 best 2-point scores and 4 best 3-point scores among 10
- total up to 20 points
- if a student skips a test: 0 points

Typical assignments available at `http://am.vsb.cz/kovar` (in Czech).

Classification (cont.)

Project

- assigned in the second half of the term
- **project**: two or four problems (discrete math & graph theory)
- Bonus **Projects for all who want to learn something** contains two problems (1 discrete mathematics & 1 graph theory)
- total of 10 points
- to receive credit ("zápočet") the project has to be accepted (minimum standards, see web)
- keep the deadline!
- work alone!

Credit ("Zápočet") = at least 10 points and an accepted project

Classification (cont.)

Exam

- examining dates given at the end of the term
- total of 70 points
- sample exam on the web (http://am.vsb.cz/kovar)
- you can use one page A4 with handwritten notes
  definitions, theorems a formulas, but no examples

Literature

In Czech:

- (partially M. Kubesa. Základy diskrétní matematiky, textbook on-line).
- P. Kovář: Algoritmizace diskrétních struktur on-line.
- P. Kovář. Úvod do teorie grafů, textbook on-line.
- P. Kovář: Cvičení z diskrétní matematiky, exercises on-line.
- solved examples as "pencasts" available on-line.

In English:

- Meyer: Lecture notes and readings for an open course (weeks 1-5, 8-10, 12-13), MIT, 2005.
- Diestel: Graph theory on-line preview (chapetrs 1-6), Springer, 2010.

You are free to use any major textbook, but beware: details can differ!
At the exam things will be required as in the lecture.

Office hours

We 9:30–11:00 (?) EA536.

see web: http://am.vsb.cz/kovar

Sample problems

Some problem, we will learn how to solve:

- handshaking problem...
- list all possible tickets in powerball ...
- nine friends exchanging three presents each...
- three lairs and three wells...
- seven bridges of Königsberg...
- missing digits in social security number ("rodné číslo")...
- correcting UPC bar codes...
- Monty Hall...

Additional interesting problems and exercises:
http://am.vsb.cz/kovar.

Chapter 0. Review

- number sets
- set and set operations
- relations
- proof techniques
- mathematical induction

Numbers and interval of integers

Natural numbers and integers

Natural numbers are denoted by $\mathbb{N} = \{1, 2, 3, 4, 5, \ldots\}$
notice! zero is not among them
Natural numbers with zero included denoted by $\mathbb{N}_0 = \{0, 1, 2, 3, 4, 5, \ldots\}$
Integers are denoted by $\mathbb{Z} = \{\ldots, -3, -2, -1, 0, 1, 2, 3, 4, \ldots\}$

Intervals of integers between a and b

is the set $\{a, a+1, \ldots, b-1, b\}$
we denote it by: $[a, b] = \{a, a+1, \ldots, b-1, b\}$

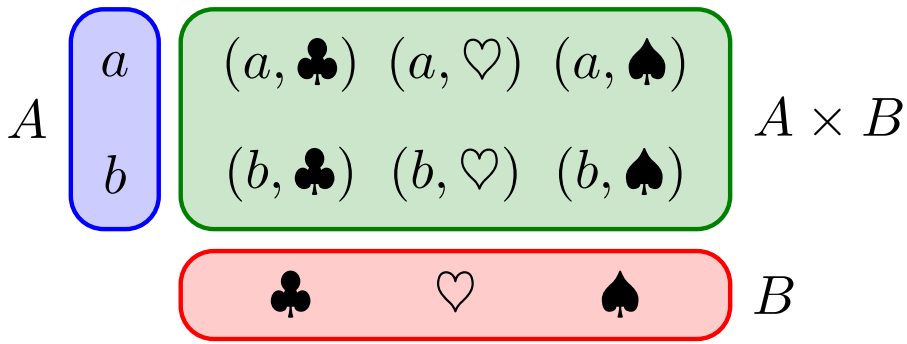
Compare to the notation used for an interval of real numbers (a, b).

Examples

$[3, 7] = \{3, 4, 5, 6, 7\}$ $\quad [-2, -2] = \{-2\}$
$[5, 0] = \emptyset$ (the empty set)

Cartesian product and Cartesian power

Cartesian product of two sets $A \times B = \{(a, b) : a \in A,\ b \in B\}$
is the set of all ordered pairs (a, b) such that $a \in A$ and $b \in B$ in this order.
$A_1 \times A_2 \times \cdots \times A_n = \{(a_1, a_2, \ldots, a_n) : a_i \in A_i,\ i = 1, 2, \ldots, n\}$
For $A_1 = A_2 = \ldots = A_n$ we get the Cartesian power A^n.
We define $A^0 = \{\emptyset\}$, $A^1 = A$.



Cartesian product of sets $A \times B = \{a, b\} \times \{\clubsuit, \heartsuit, \spadesuit\}$.

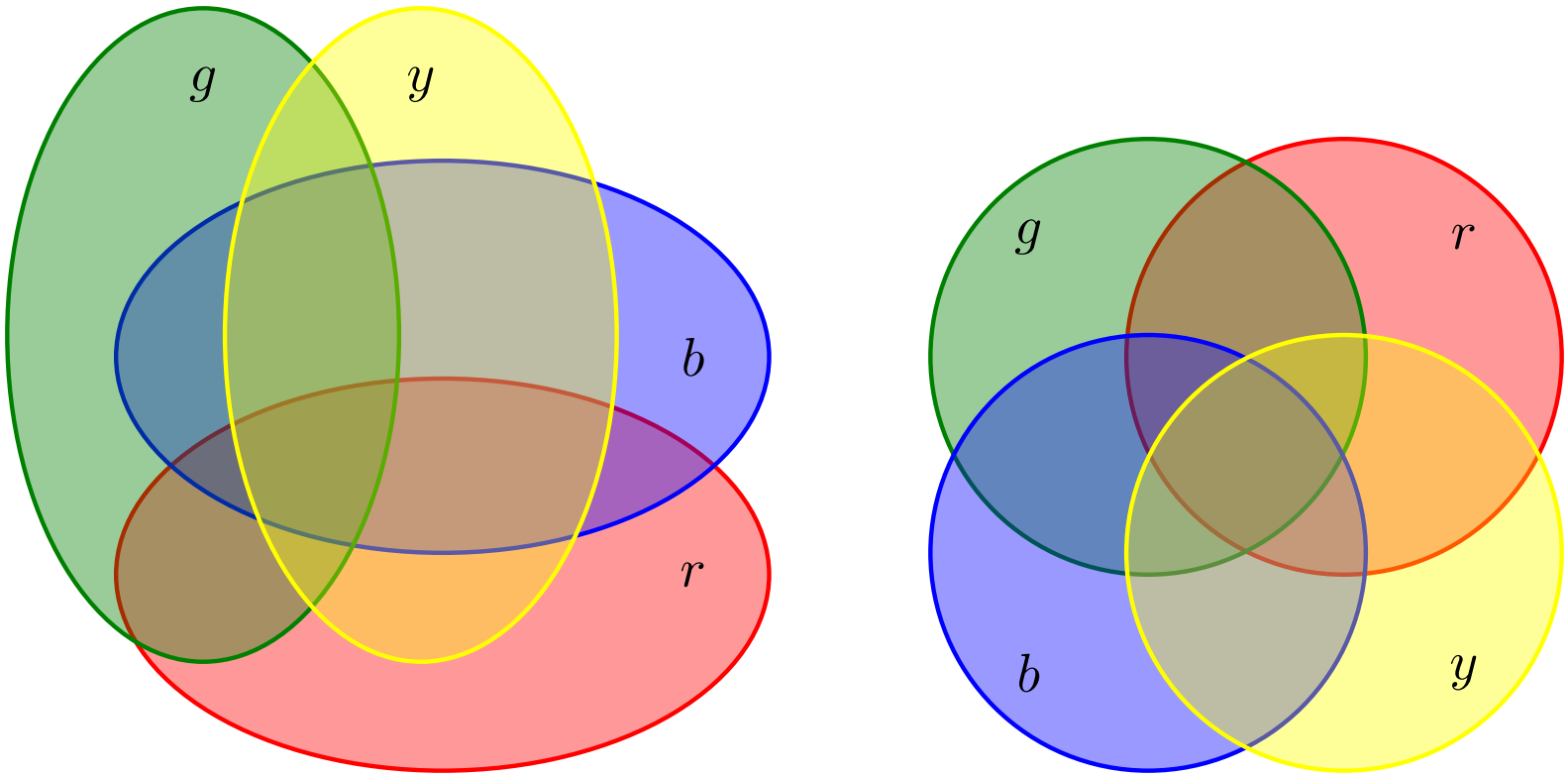
Power set of A

is the set of all subsets of A

$$2^A = \{X : X \subseteq A\}.$$

A family of sets over A

or a family of subsets of A is some $\mathcal{T} \subseteq 2^A$.
We prefer the term "family of sets" to "set of sets".



All subsets of the set of colors $C = \{r, g, b, y\}$.

Generalized unions and intersections

Generalized union $\bigcup_{i=1}^{n} X_i$ and intersection $\bigcap_{i=1}^{n} X_i$ of sets.

Given an index set J, we can write $\bigcup_{j \in J} X_j$ and $\bigcap_{j \in J} X_j$.

Examples

$$A_i = \{1, 2, \ldots, i\}, \quad \text{for each } i \in \mathbb{N}$$

$$\bigcup_{i=1}^{5} A_i = \{1, 2, 3, 4, 5\}, \quad \bigcap_{i=1}^{5} A_i = \{1\}, \quad \bigcap_{i=1}^{\infty} A_i = \{1\}$$

Questions

What is $\bigcap_{j \in J} A_j$ for $J = \{2, 5\}$?

What is $\bigcup_{j \in J} A_j$ for $J = \mathbb{N}$?

Definition

(Homogenous) binary relation R on the set A is a subset of the Cartesian product $A \times A = A^2$, i.e.

$$R \subseteq A^2.$$

Definition

(Homogenous) n-ary relation S on the set A is a subset of the Cartesian power $A \times A \times \cdots \times A = A^n$, i.e.

$$S \subseteq A^n.$$

Example

- Relation between students, with the same grade in DiM.
- Relation between pairs of students, who has a higher score.
- Relation between documents with similar terms (plagiarism)...

Binary relation is a special case of an n-ary relation. (unary, ternary, ...). (Homogenous) relations on a given set are special case of (heterogenous) relation between sets. In greater detail in another course.

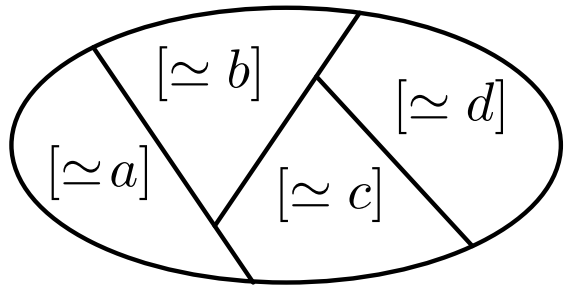
Equivalence relation

Definition

Equivalence on the set A is a *reflexive, symmetric, and transitive* binary relation on the set A. We denote it by $\simeq$.

Definition

Let $\simeq$ be an equivalence relation on the set A. An equivalence class of x (denoted by $[\simeq x]$) is the subset of A defined by $[\simeq x] = \{z \in A : z \simeq x\}$.



Equivalence relation expresses "having the same property".

Examples

- congruence relation $\equiv$ (same remainder after division by n)
- relation among students "having the same grade in DIM"
- relation "synonyms in a language" is (often) an equivalence

Partial ordering

Ordering and equivalence are among the most common relations.

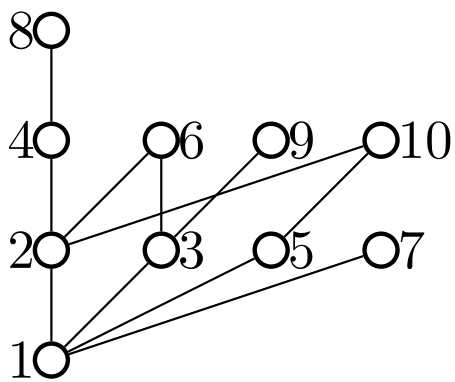
Definition

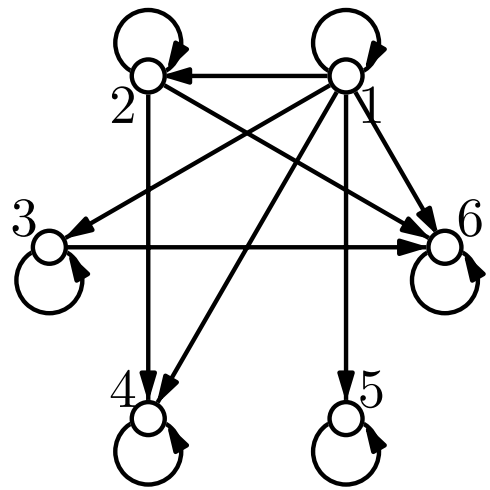
Partial ordering $\preceq$ on the set A is *reflexive, antisymmetric, and transitive* binary relation on the set A. The set with the relation is called a poset.

The word *partial* emphasizes the fact, that the relation does not have to be *linear* relation on A, i.e. not every pair of elements has necessarily to be related. Neither xRy nor yRx.

Partial orderings can be illustrated by a Hasse diagram

- if $x \preceq y$, then the element y will be drawn higher than x,
- elements x and y will be connected by a line if $x \preceq y$. We omit all lines that follow from transitivity.





0.4.2. Concept of a mathematical proof

Theorem (claims) in mathematics are usually of the form of a conditional statement: $P \Rightarrow C$

Precisely formulated premise (or hypothesis) P, under which the conclusion (consequence) C holds.

Detailed description how to obtain the conclusion from the premises is called a proof.

Mathematical proof

of some statement C is a finite sequence of steps including:

- axioms – or postulates that are considered true (the set of postulates differs for various disciplines*),
- hypothesis P is an assumption on which we work,
- statement derived from previous by some correct rule (depends on logic used).

The last step is a conditional statement with *conclusion* C.

* Discrete mathematics relies on Peano axioms, geometry is build upon five Euklid's postulates, ...

What could I need a proof for?

"What is the use of a newborn?"

- correctly understand the limitations of various method
- arguments for/against a presented solution
- comparison of quality of different solutions
- 100% validity of an algorithm may be required (autopilot, intensive care unit)

Mathematical induction

Mathematical induction is a common proof technique used to prove propositional functions with a natural parameter n, denoted by $P(n)$.

Mathematical induction

Let $P(n)$ be a propositional function with an integer parameter n.
Suppose:

- *Basis step:*
  The proposition $P(n_0)$ is true, where $n_0 = 0$ or 1, or some integer.
- *Inductive step:*
  Assume the Inductive hypothesis: $P(n)$ holds for some n.
  Show, that for all $n > n_0$ if $P(n)$ holds, then also $P(n+1)$ holds.

Then $P(n)$ is true for all integers $n \geq n_0$.

Mathematical induction can be used also to prove validity of algorithms.

A few examples follow...

Wait a minute!

But...

- we verify the Basis step,
- we verify the Inductive step (using the Inductive hypothesis),

... how come this implies the validity for infinity many values!?!

Example

How high can you climb a ladder?

Suppose we can

- mount the first step,
- standing on rung n climb the rung $n+1$.

...thus, we can reach any rung of the ladder!

Theorem

The sum of the first n even natural numbers is $n(n+1)$.

$2+4+6=12=3\cdot 4$

$2+4+6+8+10+12+14+16+18+20=110=10\cdot 11$

Proof by mathematical induction based on n:

We prove $\forall n \in \mathbb{N}$ the following holds $\sum_{i=1}^{n} 2i = n(n+1)$.

- *Basis step:* For $n=1$ claim P(1) gives "$2=1\cdot 2$".
- *Inductive step:* Does $P(n)$ imply $P(n+1)$?

  I.e. does $\sum_{i=1}^{n} 2i = n(n+1)$, imply $\sum_{i=1}^{n+1} 2i = (n+1)(n+2)$?

  We state *Inductive hypothesis* P(n):
  Suppose $\exists n \in \mathbb{N} : \sum_{i=1}^{n} 2i = n(n+1)$.
  Now
  $\sum_{i=1}^{n+1} 2i = \sum_{i=1}^{n} 2i + 2(n+1) \overset{IH}{=} n(n+1)+2(n+1) = (n+1)(n+2)$.
  We have shown the correctness of the formula for the sum of the first $n+1$ evens using the formula for the sum of the first n evens.

By mathematical induction the claim holds $\forall n \in \mathbb{N}$. $\square$

Strong mathematical induction compared to mathematical induction

Mathematical induction

Let $P(n)$ be a propositional function with an integer parameter n.
Suppose:

- *Basis step:*
  The proposition $P(n_0)$ is true, where $n_0 = 0$ or 1, or some integer n_0.
- *Inductive step:*
  Assume the Inductive hypothesis: $P(n)$ holds for some n.
  Show, that for all $n > n_0$ if $P(n)$ holds, then also $P(n+1)$ holds.

Then $P(n)$ is true for all integers $n \geq n_0$.

Strong mathematical induction

- *Basis step:* The proposition $P(n_0)$ is true.
- *Inductive step:*
  Inductive hypothesis: Assume $P(k)$ holds for all $n_0 \leq k < n$.
  Show, that also $P(n)$ is true.

Then $P(n)$ is true for all integers $n \geq n_0$.

Example

There are always $pr - 1$ breaks necessary to split a chocolate bar of $p \times r$ squares.

By *strong* induction on $n = pr$:

- *Basis step:*
  For $n_0 = 1$ we have a bar with only one square, there are no breaks necessary ($pr - 1 = 0$).
- *Inductive step:*
  Suppose now the claim holds for *any* chocolate bars with less than n squares. Take any bar with n squares. We break this bar into two parts of s or t squares, respectively, where $1 \leq s, t < n$ and $s + t = n$. By Inductive hypothesis we can break each part by $s - 1$ or $t - 1$ breaks, respectively. There is a total of $(s - 1) + (t - 1) + 1 = s + t - 1 = n - 1$ breaks necessary.

The proof is complete by strong induction for all positive p, r. □

Chapter 1. Sequences

- sequences
- sums and products
- arithmetic progression
- geometric progression
- ceiling and floor functions

Sequence

is an ordered list of objects, called elements.
We denote it by $(a_i)_{i=1}^n = (a_1, a_2, \ldots, a_n)$.

- in real analysis defined as mappings $p : \mathbb{N} \to \mathbb{R}$
- we distinguish first, second, third, ... element in the sequence.
- indices are natural numbers, usually starting at 1
- elements in a sequence can repeat (in contrary to sets)
- sequences can be finite $(a_1, a_2, \ldots, a_n)$
  and infinite $(a_1, a_2, \ldots)$, the sequence can even be empty
  (we focus mainly on finite sequences)

Examples

(x, v, z, v, y)
$(2, 3, 5, 7, 11, 13, 17, 19, 23, 29)$
$(2, 3, 5, 7, 11, 13, 17, 19, 23, 29, \ldots)$
$(1, -1, 1, -1, 1, -1, 1, -1, \ldots)$

A sequence is given by: listing the elements, recurrence relations or a formula for the n-th element

Sums

Sum of a sequence is denoted by

$$\sum_{i=1}^{n} a_i = a_1 + a_2 + \cdots + a_{n-1} + a_n$$

$$\sum_{i \in J} a_i = a_{i_1} + a_{i_2} + \cdots + a_{i_n}, \text{ where } J = \{i_1, i_2, \ldots, i_n\}.$$

Question

$$\sum_{i \in \{1,3,5,7\}} i^2 = ?$$

Example

$$\sum_{i=1}^{n} i = ?$$

Product

Product of elements in a sequence is denoted by

$$\prod_{i=1}^{n} a_i = a_1 \cdot a_2 \cdot \cdots \cdot a_{n-1} \cdot a_n$$

$$\prod_{i \in J} a_i = a_{i_1} \cdot a_{i_2} \cdot \cdots \cdot a_{i_n}, \text{ where } J = \{i_1, i_2, \ldots, i_n\}$$

Examples

$$\sum_{i=2}^{5} \ln(i) = \ln\left(\prod_{i=2}^{5} i\right) = \ln(2 \cdot 3 \cdot 4 \cdot 5) = \ln 120$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n}(i \cdot j) = \sum_{i=1}^{n}\left(i \cdot \sum_{j=1}^{n} j\right) = \left(\sum_{i=1}^{n} i\right) \cdot \left(\sum_{j=1}^{n} j\right) = \left(\frac{1}{2}n(n+1)\right)^2$$

$$\text{empty sum } \sum_{i=3}^{2} i = 0 \qquad \text{empty product } \prod_{i=3}^{2} i = 1$$

Examples

$$\sum_{i=1}^{n}(i+j) = \sum_{i=1}^{n} i + \sum_{i=1}^{n} j = \frac{n}{2}(n+1) + nj$$

$$J = \{2, 8, 12, 21\}, \qquad \sum_{j \in J} j = 2 + 8 + 12 + 21 = 43$$

Questions

$$\sum_{i=1}^{5} \ln(i) = ? \qquad \sum_{i=1}^{100} i = ?$$

$$\prod_{i=1}^{6} i = ? \qquad \prod_{i=1}^{n} i = ?$$

$$\prod_{i=1}^{n}(n-i) = ? \qquad \sum_{i=1}^{n}(n+1-i) = ?$$

Question

Can you find a sequence $(a_i)_{i=1}^n$, such that $\sum_{i=1}^n a_i < \sum_{i=1}^n (-a_i)$?

Question

Can you find a sequence $(a_i)_{i=1}^n$, such that $\sum_{i=1}^n a_i > 0$ and $\prod_{i=1}^n a_i < 0$?

Question

Does there exist a sequence of *positive* numbers $(a_i)_{i=1}^n$, such that $\sum_{i=1}^n a_i > \prod_{i=1}^n a_i$?

Arithmetic progression

Certain sequences for special progressions and we know several their properties.

Arithmetic progression

The sequence (a_i) is an arithmetic progression if its terms are

$$a, \quad a+d, \quad a+2d, \quad a+3d, \ldots$$

Real numbers a, d are the first term and the difference of the progression, respectively.

Notice that the sequence (a_i) is an arithmetic progression, if there exists a real number d, such that for all $i > 1$ is $a_i - a_{i-1} = d$.

Every subsequent term arises by adding (the same!) difference d to the previous term.

Finite arithmetic progressions are also considered.
We have n terms

$$a, a+d, a+2d, \ldots, a+(n-1)d.$$

Examples

$-2, 3, 8, 13, 18, \ldots$ first term -2, difference 5

$-3, 2, 7, 12, 17, \ldots$ first term -3, difference 5

$20, 9, -2, -13, -24, \ldots$ first term 20, difference -11

$\sqrt{2}, \sqrt{2}, \sqrt{2}, \sqrt{2}, \sqrt{2}, \ldots$ first term $\sqrt{2}$, difference 0

Examples

Find the n-th term of the progressions a_n from previous example

$-2, 3, 8, 13, 18, \ldots$ $a_n = -2 + (n-1)5$

$-3, 2, 7, 12, 17, \ldots$ $a_n = -3 + (n-1)5$

$20, 9, -2, -13, -24, \ldots$ $a_n = 20 - (n-1)11$

$\sqrt{2}, \sqrt{2}, \sqrt{2}, \sqrt{2}, \sqrt{2}, \ldots$ $a_n = \sqrt{2}$

Example

Which sequence is given by the n-th term $a_n = -8 + 5n$?

Second progression $-3, 2, 7, 12, 17, \ldots$

Summing n terms of an arithmetic progression

$$a_1 + a_2 + \cdots + a_n = \sum_{i=1}^{n} a_i$$

In this case

$$a + (a + d) + \cdots + a + (n-1)d = \sum_{i=1}^{n} (a_1 + (i-1)d)$$

holds

$$\sum_{i=1}^{n} (a_1 + (i-1)d) = \frac{n}{2}(a_1 + a_n) = \frac{n}{2}(2a_1 + (n-1)d) = na_1 + \frac{n(n-1)d}{2}.$$

Sum of certain consecutive n terms of an arithmetic progression

$$\sum_{i=k}^{k+n-1} a_i = \frac{n}{2}(a_k + a_{k+n-1}) = \frac{n}{2}(2a_k + (n-1)d) = na_k + \frac{n(n-1)d}{2}.$$

Notes

The sum of an infinite arithmetic progression generally does not exist.

Sequence of partial sums

- diverges to $+\infty$ for $d > 0$,
- diverges to $-\infty$ for $d < 0$,
- for $d = 0$ diverges to $+\infty$ or to $-\infty$ or converges based on a_1.

Arithmetic progression with first term a and difference d can be given by a recurrence relation

$$a_n = a_{n-1} + d, \quad a_1 = a.$$

Example savings

Example

Uncle Scrooge has 4 514 cents in his safe. Every week he adds 24 cents to the safe. What is the formula for a_n?

$$4\ 514, 4\ 538, 4\ 562, 4\ 586, \cdots = 4\ 514 + 24(n-1) = 4\ 490 + 24n.$$

Example

Uncle Scrooge has 4 514 cents in his safe. The pocket money of each of his three nephews is 1 cent, but every week he increases the pocket money by one cent.
a) Evaluate the total pocket money in the n-th week.
b) Evaluate the number of cents in the safe in the n-th week.

a) pocket money $k = 3 + 3(n-1) = 3n$
b) in safe $s = 4\ 514 - 3n$

Geometric progression

Geometric progression

The sequence (a_i) is a geometric progression if its terms are

$$a, \quad a \cdot q, \quad a \cdot q^2, \quad a \cdot q^3, \ldots$$

Real numbers a, q are the first term and the common ration of the progression, respectively.

Notice that the sequence (a_i) is a geometric progression if there exists a real number q, such that for all $i > 1$ is $\frac{a_i}{a_{i-1}} = q$.

Every subsequent term arises by multiplying the previous term by (the same!) common ratio q.
Finite geometric progressions are also considered. We have n terms

$$a, \quad a \cdot q, \quad a \cdot q^2, \ldots, \quad a \cdot q^{n-1}.$$

Question

Can a progression be both geometric and arithmetic at the same time? If yes, can you find different solutions? Infinitely many?

Examples

$2, 10, 50, 250, 1250, \ldots$ first term 2, common ratio 5

$9, 6, 4, \frac{8}{3}, \frac{16}{9}, \ldots$ first term 9, common ratio $\frac{2}{3}$

$4, -2, 1, -\frac{1}{2}, \frac{1}{4}, \ldots$ first term 4, common ratio $-\frac{1}{2}$

$\sqrt{2}, \sqrt{2}, \sqrt{2}, \sqrt{2}, \sqrt{2}, \ldots$ first term $\sqrt{2}$, common ratio 1

Examples

Find the n-th term of the progressions a_n from previous example

$2, 10, 50, 250, 1250, \ldots$ $a_n = 2 \cdot 5^{n-1}$

$9, 6, 4, \frac{8}{3}, \frac{16}{9}, \ldots$ $a_n = 9 \cdot \left(\frac{2}{3}\right)^{n-1} = \frac{27}{2} \cdot \left(\frac{2}{3}\right)^{n}$

$4, -2, 1, -\frac{1}{2}, \frac{1}{4}, \ldots$ $a_n = 4 \cdot \left(-\frac{1}{2}\right)^{n-1} = -8 \cdot \left(-\frac{1}{2}\right)^{n}$

$\sqrt{2}, \sqrt{2}, \sqrt{2}, \sqrt{2}, \sqrt{2}, \ldots$ $a_n = \sqrt{2}$

Example

Which sequence is given by the n-th term $a_n = \left(\frac{1}{2}\right)^{n}$? $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \ldots$

Sum of n terms of a geometric progression

$$a_1 + a_2 + \cdots + a_n = \sum_{i=1}^{n} a_i$$

In our case

$$a + (a \cdot q) + \cdots + a \cdot q^{n-1} = \sum_{i=1}^{n} (a_1 \cdot q^{i-1})$$

for $q \neq 1$ holds

$$\sum_{i=1}^{n} (a_1 \cdot q^{i-1}) = a_1 \frac{q^n - 1}{q - 1}.$$

For $q = 1$ is the progression both arithmetic and geometric; we use a different formula.

Question

How does the sum of first n terms of a geometric progression with common ratio 1 look like?

Notes

The sum of an infinite geometric progression

- generally does not exist for $|q| \geq 1$,
- for $q = 1$ the sequence is constant; the sum depends on a_1,
- for $q = -1$ the sequence oscillates, there is no sum
- for $|q| < 1$ the sum is finite $\frac{a_1}{1-q}$

Sequence of partial sums of an infinite geometric progression

- diverges for $q \geq 1$,
- oscillates (and does not converge for $q \leq -10$,
- converges to $\frac{a_1}{1-q}$ for $|q| < 1$.

A geometric progression with first term a and common ratio q can be described recursively

$$a_n = a_{n-1} \cdot q, \quad a_1 = a.$$

Example savings

Example

Uncle Scrooge has 4 514 cents in a bank. Every year he get an interest of 2 percent (no rounding). What is the formula for the amount a_n (after n years)?

$$4\ 514, 4\ 604.3, 4\ 696.4, 4\ 790.3, 4\ 886.1, 4\ 983.8, \cdots = 4\ 514 \cdot 1.02^{n-1}.$$

Example

Uncle Scrooge has 4 514 cents in his safe. The pocket money of each of his three nephews is 1 cent, but every week he doubles the pocket money of each.

a) Evaluate the total pocket money in the n-th week.

b) Evaluate the number of cents in the safe in the n-th week.

a) pocket money $k = 3 \cdot 2^{n-1} = \frac{3}{2} \cdot 2^n$

b) in safe $s = 4\ 514 - 3 \cdot 2^{n-1}$

Example

We tilt the pendulum to 5 cm height. Due friction each sway of the pendulum looses one fifth of it energy. Describe the sequence of heights to which the pendulum rises after each sway.

$$5 \text{ cm}, \quad 4 \text{ cm}, \quad \frac{16}{5} \text{ cm}, \quad \frac{64}{25} \text{ cm}, \quad \frac{256}{125} \text{ cm}, \ldots, \quad 5 \cdot \left(\frac{4}{5}\right)^{n-1} \text{ cm}$$

first term 5 cm,
common ratio $\frac{4}{5}$.

Question

After how many tilts will the pendulum stop?

Integer part of a real number

$\lfloor x \rfloor$ floor function for a real number x
$\lceil x \rceil$ ceiling function for a real number x

Example

$\lfloor 3.14 \rfloor = 3 \qquad \lfloor -3.14 \rfloor = -4$
$\lfloor x \rfloor = \lceil x \rceil \quad \Rightarrow \quad x \in \mathbb{Z}$

Question

Gives the expression $\lceil \log n \rceil$ the number of digits of n (in decimal system)?

If not, can you find a "correct" formula?

Question

$\left\lceil \frac{n}{n+1} \right\rceil = ?$, where $n \in \mathbb{N}$ (what if $n \in \mathbb{N}_0$)

Next lecture

Arrangements and selections

- multiplication principle (of independent selections)
- method of double counting

Discrete mathematics

Petr Kovář
petr.kovar@vsb.cz

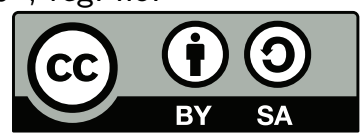
VŠB – Technical University of Ostrava

Winter term 2021/2022
DiM 470-2301/02, 470-2301/04, 470-2301/06

EUROPEAN UNION
European Structural and Investment Funds
Operational Programme Research,
Development and Education

MINISTRY OF EDUCATION,
YOUTH AND SPORTS

The translation was co-financed by the European Union and the Ministry of Education, Youth and Sports from the Operational Programme Research, Development and Education, project "Technology for the Future 2.0", reg. no. CZ.02.2.69/0.0/0.0/18_058/0010212.



About this file

This file is meant to be a guideline for the lecturer. Many important pieces of information are not in this file, they are to be delivered in the lecture: said, shown or drawn on board. The file is made available with the hope students will easier catch up with lectures they missed.

For study the following resources are better suitable:

- Meyer: Lecture notes and readings for an http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-042j-mathematics-for-computer-science-fall-2005/readings/" (weeks 1-5, 8-10, 12-13), MIT, 2005.
- Diestel: Graph theory http://diestel-graph-theory.com/ (chapters 1-6), Springer, 2010.

See also http://homel.vsb.cz/~kov16/predmety_dm.php

Chapter 2. Arrangements and selections

- selections: permutations, k-permutations and k-combinations
- two basic counting principles
  multiplication principle (of independent selections)
  method of double counting
- permutations, k-permutations and k-combinations with repetition

Arrangements and selections

We count the number of selections from a given set

- ordered arrangements / unordered selections,
- with repetition / without repeating elements.

Today:

- permutations (without repetition)
- combinations (without repetition)
- k-permutations (without repetition)
- $+$ problems leading to counting selections

Beware! While solving real life problems we usually need to split a complex problem into several sub-cases,

- complex selections/arrangements, during discussions

we have to distinguish common/different properties.

Definition

Permutation of an n-element set X is an *(ordered) arrangement of all n elements* from X (without repetition).

The total number of possible permutations of an n-element set is

$$P(n) = n \cdot (n-1) \cdot (n-2) \cdot \cdots \cdot 2 \cdot 1 = n!$$

- the first element is chosen among n possibilities
- the second element is chosen among $n-1$ possibilities
- the third element is chosen among $n-2$ possibilities...

Problems, described by permutations (without repetition)

- number of orderings of elements from a set
- number of bijections from an n-element set onto another n-element set
- number of ways how to order the cards in a deck
- distribution of numbers at a start of a marathon
- distributing keys in a fully occupied hotel

Definition

Combination (or k-combination) from a set X is an *(unordered) selection of k distinct elements* from a given set X (a k-element subset of X.)

The number of k-combinations from an n-element set

$$C(n,k) = \frac{n!}{k! \cdot (n-k)!} = \binom{n}{k}$$

- $n!$ different orderings (permutations) of X
- we choose first k elements (not distinguishing their $k!$ orderings)
- we discard the last $n-k$ elements (not distinguishing their $(n-k)!$ orderings)

Problems, described by combinations (without repetition)

- number of k-element subsets of an n-element set
- binomial coefficients: coefficient at x^k in $(x+1)^n$

$$(x+1)^n = \sum_{k=0}^{n} \binom{n}{k} x^k$$

Definition

k-permutation from a set X is an *ordered arrangement of k elements from an n-element set X* (without repetition)
(sequence of k-elements from X).

The number of k-permutations from an n-element set

$$V(n,k) = n \cdot (n-1) \cdot \cdots \cdot (n-k+1) = \frac{n!}{(n-k)!}$$

- the first element is chosen among n possibilities
- the second element is chosen among $n-1$ possibilities
- ...
- the k-th element is chosen among $n-k+1$ possibilities.

or

- $n!$ possibilities how to order elements of X
- we take only first k elements
- we discard the last $n-k$ elements (not distinguishing their $(n-k)!$ orderings)

Problems, described by k-permutations (without repetition)

- setting up an k-element sequence from n elements
- number of injections (one-to-one mappings) from an k-element set to an n-element set
- number of different race outcomes (trio on a winner's podium)
- distributing keys in a partially occupied hotel

Examples

- team of four among ten employees
  calculation using *k-combinations* (we do not distinguish ordering)

$$C(10,4) = \binom{10}{4} = \frac{10!}{4! \cdot 6!} = \frac{10 \cdot \cancel{9} \cdot \cancel{8} \cdot 7}{\cancel{4} \cdot \cancel{3} \cdot \cancel{2}} = \frac{10 \cdot 3 \cdot 7}{1} = 210$$

- number of matches in a tennis tournament of seven players
  calculation using *k-combinations* (2-element subsets in a 7-element set)

$$C(7,2) = \binom{7}{2} = 21$$

Examples

- number of possible orders after a tournament of seven players
  calculation based on *permutations*

$$P(7) = 7! = 5040$$

- number of triples on the winners podium in a tournament of seven
  calculation by *3-permutations*, because "the order does matter"

$$V(7,3) = \frac{7!}{4!} = 7 \cdot 6 \cdot 5 = 210$$

Complex selections and arrangements

In some cases we add and in some cases we multiply the number of selections or arrangements to obtain the result. How to recognize which is correct?

Sum rule

Suppose there are n_1 selections (arrangements) obtained in one way and n_2 selections (arrangements) obtained in another way, where no selection (arrangement) can be obtained in both ways, then the total number of selections (arrangements) is $n_1 + n_2$.
"EITHER n_1 ways OR n_2 further ways."

Product rule

Suppose a selection (arrangement) can be broken into a sequence of two selections (arrangements). If the first stage can be obtained in n_1 ways and the second stage can be obtained in n_2 ways for each way (independently) of the first stage, then the total number of selections (arrangements) is $n_1 \cdot n_2$.
"First n_1 ways AND then n_2 ways."

If a selection is broken into two disjoint sets of selections, then we add the number of selections.

Example

In the game „člověče nezlob se“ we roll an ordinary dice and move a peg by the indicated number of fields. If we roll a 6 in the first roll, we roll an additional time. By how many fields can we move the peg in one round?

We distinguish two cases:

- if there is not a 6 in the first roll, we move by 1 up to 5 fields,
- if there is a 6 in the first roll, we move by $6+1$ up to $6+6$ fields.

There are 11 possibilities: 1, 2, 3, 4, 5, (no 6!) 7, 8, 9, 10, 11, 12.

If a selection can be broken into two stages (subselections), then we multiply the number of selections.

Example

The coach of a hockey team sets up a formation (three forwards, two full-backs and a goalkeeper). He has a team of 12 forwards, 8 full-backs, and two goalkeepers.
How many different formations can he set up?

Because there is no relation between the choice of full-backs, forwards, and goalkeepers we can count as follows

$$\binom{12}{3} \cdot \binom{8}{2} \cdot \binom{2}{1} = \frac{12 \cdot 11 \cdot 10}{6} \cdot \frac{8 \cdot 7}{2} \cdot 2 = 220 \cdot 28 \cdot 2 = 12320.$$

There are altogether 12 320 different formations.

When two selections are not independent...

we cannot just multiply the counts of each (sub)selection.

Example

The coach of a hockey team sets up a formation (three forwards, two full-backs and a goalkeeper). He has a team of 11 forwards, 8 full-backs, 1 universal player (either a full-bak or a forward), and two goalkeepers. How many different formations can he set up?

- choose 3 forwards: $\binom{12}{3}$
- choose 2 full-backs: $\binom{8}{2}$ or $\binom{9}{2}$?

It depends, whether the universal player was picked as forward or not.

... solution in the discussion

Question

We roll a dice three times. How many rolls are possible, such that every subsequent roll gives a higher number than the previous one?

Double counting

Suppose each arrangement can be further split into several *finer* ℓ *arrangements*. Moreover, suppose we know how to count the total number of the refined arrangements m. Then the total number of the original arrangements is *given by the ratio* m/ℓ.

Example

We have the characters T, Y, P, I, C. How many different (even meaningless) five-letter words can you construct? We do not distinguish Y and I letters.

If we distinguish all characters, we have $P(5) = 5! = 120$ words.
Not distinguishing Y, I: $TYPIC = TIPYC$.

In the total of $m = 120$ we have every arrangement counted twice $\ell = 2$.
The number of different words is

$$\frac{m}{\ell} = \frac{120}{2} = 60.$$

Arrangements with repetition

So far no repetition of selected elements was allowed
(people, subsets, . . .)
In several problems repetition is expected (rolling dice, characters, . . .)

Example

How many anagrams of the word MISSISSIPPI exist?
(*anagram* is a word obtained by rearranging all characters of a given word)

If no character in "MISSISSIPPI" would repeat, the calculation would rely on permutation. But they repeat: S 4times, I 4times, P 2times.

By double counting:

1. first we distinguish all characters (using colors, indices, etc.)
2. we count all arrangements: $(4+4+2+1)!$
3. divide by the number of indistinguishable arrangements: $4! \cdot 4! \cdot 2! \cdot 1!$

$$\frac{(4+4+2+1)!}{4! \cdot 4! \cdot 2! \cdot 1!} = \frac{11 \cdot 10 \cdot 9 \cdot \cancel{8} \cdot 7 \cdot \cancel{6} \cdot 5}{\cancel{4} \cdot \cancel{3} \cdot \cancel{2} \cdot \cancel{2}} = 11 \cdot 10 \cdot 9 \cdot 7 \cdot 5 = 34650$$

Definition

Permutation with repetition from the set X is an *arrangement* of elements from X in a sequence such that every element from X occurs **a given number of times**. Denote the number of them by $P^*(m_1, m_2, \ldots, m_k)$.

(an arrangement with a given number of copies of elements from X)

The number of all permutations with repetition from a k-element set, where the i-th element is repeated in m_i identical copies ($i = 1, 2, \ldots, k$):

$$P^*(m_1, m_2, \ldots, m_k) = \frac{(m_1 + m_2 + \cdots + m_k)!}{m_1! \cdot m_2! \cdots m_k!}.$$

Examples

- permutation with repetition of 2 elements, one element occurs in k copies the other in $(n - k)$ copies

$$\frac{(k + n - k)!}{k! \cdot (n - k)!} = \binom{n}{k} = C(n, k).$$

  first element = "is", the second element = "is not" an arrangement
- permutation of multisets (in multisets identical copies are allowed)

Example describing the idea of combination with repetition

Example

How man ways are there to select 6 balls of three colors, provided we have an unlimited supply of balls of each color?

We present a beautiful trick, how to count the total number of selections.
Suppose we pick ●, ●, ●, ●, ●, ●
This selection we *can order* (group) based on colors

now we observe, that only the "bars", not the colors are important

The total number of selections is

$$C^*(3,6) = \binom{6+2}{2} = \binom{8}{2} = 28.$$

Definition

A k-combination with repetition from an n-element set X is a **selection** of k elements from X, while each element can occur in **an arbitrary number of identical copies**. The number of them we denote by $C^*(n,k)$.

The total number of all k-element selections with repetition from n possibilities is

$$C^*(n,k) = \binom{k+n-1}{n-1}.$$

- having n "colors", we need $n-1$ bars
- we can "select bars", or "select elements"

$$C^*(n,k) = \binom{k+n-1}{n-1} = \binom{k+n-1}{k}.$$

Problems solved using k-combinations with repetition

- number of ways how to write k using n nonnegative integer summands
- drawing k elements of n kinds provided after each draw we return the elements back to the polling urn

Example

How many ways are there to write k as the sum of n nonnegative integer summands? We distinguish the order of summands!

We have

$$k = x_1 + x_2 + \cdots + x_n.$$

We will select (draw) k ones and distribute them into n boxes (with the possibility of tossing more ones into each box).

- some boxes can remain empty ($0 \in \mathbb{N}_0$)
- we can toss all ones into one box
- we repeat boxes, not ones! (a different problem)

Questions

How many ways are there to write k as the sum of n positive summands?

How many ways are there to write k as the sum of at least n natural summands?

How many ways are there to write k as the sum of at most n natural summands?

Definition

A k-permutation with repetition from an n-element set X is an **arrangement** of k elements from X, while elements **can repeat** in an arbitrary number of identical copies. The number of them we denote by $V^*(n, k)$.

The arrangement is a sequence.

The number of all k-permutations with repetition from n possibilities is

$$V^*(n, k) = \underbrace{n \cdot n \cdots n}_{k} = n^k.$$

Problems solved by k-permutation with repetition

- number of mappings of an k-element set to an n-element set
- cardinality of the Cartesian power $|A^k|$

Question

How many odd-sized subsets has a given set of n elements?

Next lecture

Chapter 3. Discrete probability

- motivation
- sample space
- independent events

Discrete mathematics

Petr Kovář
petr.kovar@vsb.cz

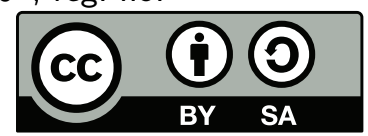
VŠB – Technical University of Ostrava

Winter term 2021/2022
DiM 470-2301/02, 470-2301/04, 470-2301/06

EUROPEAN UNION
European Structural and Investment Funds
Operational Programme Research,
Development and Education

MINISTRY OF EDUCATION,
YOUTH AND SPORTS

The translation was co-financed by the European Union and the Ministry of Education, Youth and Sports from the Operational Programme Research, Development and Education, project "Technology for the Future 2.0", reg. no. CZ.02.2.69/0.0/0.0/18_058/0010212.



About this file

This file is meant to be a guideline for the lecturer. Many important pieces of information are not in this file, they are to be delivered in the lecture: said, shown or drawn on board. The file is made available with the hope students will easier catch up with lectures they missed.

For study the following resources are better suitable:

- Meyer: Lecture notes and readings for an http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-042j-mathematics-for-computer-science-fall-2005/readings/" (weeks 1-5, 8-10, 12-13), MIT, 2005.
- Diestel: Graph theory http://diestel-graph-theory.com/ (chapters 1-6), Springer, 2010.

See also http://homel.vsb.cz/~kov16/predmety_dm.php

Chapter 3. Discrete probability

- describing chance: event, sample space
- independent events
- expected values
- random selections and arrangements

Discrete probability

One of the oldest motivation for counting probabilities is gambling.
Questions

- "What are the chances of rolling dice with a given outcome?"
- "What are the chances of receiving a given hand of cards?"

led to formalizing the terms of chance and probability.

We will deal only with *discrete* cases, i.e. situations, that can be described with one of *finitely many* possibilities.

Overview:

- motivation problems
- sample space (intuition often fails)
- independent events
- expected values
- random selections and arrangements

Motivation problems

Flipping a coin 2 possible outcomes: head / tail (1 / 0)
We expect that both sides are equally likely to be obtained by flipping (we say "with probability $\frac{1}{2}$")

Rolling dice 6 possible outcomes: 1, 2, 3, 4, 5, and 6
each number of points occurs with the same frequency ("probability") $\frac{1}{6}$

Shuffling a deck of cards we expect, that the shuffling is fair, no shuffle is more likely to occur
there are $32! \doteq 2.6 \cdot 10^{35}$ possible outcomes

Powerball Winning Numbers (Tah sportky) drawing balls (6 out of 49)
$\binom{49}{6} = 13\,983\,816$ possible outcomes (Powerball: 5/55 and 1/42)

Our "expectancy of probability" is based on the expectancy of fairness.
Often different outcomes are considered to be equivalent in the terms of frequency of occurrences of a random experiment.

Finite sample space

How to describe "chance" by a mathematical model?
We expect (subjectively) whether an event will occur. We compare with the previous relative frequency of that event, a number between 0 and 1.

$$\text{probability} = \frac{\text{frequence of an event}}{\text{number of experiments}}$$

Definition

Finite sample space is a pair (Ω, P), where Ω is a finite set of elements and P is probability function, which assigns to every subset of Ω a real value (probability) from $\langle 0, 1 \rangle$, s.t. the following hold

- $P(\emptyset) = 0,\ P(\Omega) = 1$
- $P(A \cup B) = P(A) + P(B)$ for disjoint $A, B \subseteq \Omega$

An Event is any subset $A \subseteq \Omega$ and its probability is $P(A)$.

Note: it is enough to assign probabilities to the one element sets, called elementary events, or atomic events, or points of Ω. Then, for $A = \{a_1, \ldots, a_k\} \subseteq \Omega$, by definition $P(A) = P(\{a_1\}) + \cdots + P(\{a_k\})$.

Notice

- One element subsets are called elementary events.
  However, the elements are *not* events nor elementary events.

Example

elementary event: rolling a dice you get 1
event: rolling a dice you get an even number

- *disjoint events* cannot occur at the same time: $A \cap B = \emptyset$

Example

Rolling two dice we have events

- A: you get at least one 6
- B: you get the sum 7

are not disjoint events $P(A \cap B) = \frac{2}{36} = \frac{1}{18}$, but

- C: you get at least one 6
- D: you get the sum 3

are disjoint events $P(C \cap D) = 0$, C and D cannot occur together.

Motivation problems based on the definition

Flipping a coin 2 possible outcomes: head / tail (1 / 0)
$\Omega = \{0, 1\}$ a $P(\{0\}) = P(\{1\}) = \frac{1}{2}$.

Rolling dice $\Omega = \{1, 2, 3, 4, 5, 6\}$ and $P(\{1\}) = \ldots = P(\{6\}) = \frac{1}{6}$
Event "rolling an even number" is given by the subset $\{2, 4, 6\}$.

Shuffling a deck of cards Ω contains all 32! permutations of 32 cards, each permutation has the same probability $\frac{1}{32!}$.
Event "Full House" is formed by the subset of permutations with a triple of cards with the same value and a pair with another value.

Powerball Winning Numbers (Tah sportky) drawing balls (6 out of 49)
Ω contains all 6-combinations of 49 numbers, each with the probability $1/\binom{49}{6}$.

Notice: in all examples the elementary events have the same probability.

Definition

If the probability function $P : P(A) \to \langle 0, 1 \rangle$ is given by

$$P(A) = |A| \,/\, |\Omega|$$

for all $A \subseteq \Omega$, then P is called uniform probability and such sample space Ω is uniform.

- all elementary events have the same probability
- the probability of the event A = relative size of A with respect to Ω

One can specify the probability of elementary events only:
For each elementary event $\{e\} \subset \Omega$ let

$$P(\{e\}) = \frac{1}{|\Omega|}.$$

Example

Random experiment: sum of points while rolling two dice.

The set of all possible sums on two dice $\Omega = \{2, 3, \ldots, 12\}$.
Probabilities of the elementary events differ! There is only one way how to obtain the sum $2 = 1 + 1$, while the sum 7 can be obtained in six ways.

There are $6 \cdot 6 = 36$ possible outcomes, we mentioned that the sum 7 can be obtained in 6 ways, the sum 6 in five ways, etc. We get the following probabilities

$$\begin{aligned}
P(2) = P(12) &= \tfrac{1}{36}, \\
P(3) = P(11) &= \tfrac{2}{36} = \tfrac{1}{18}, \\
P(4) = P(10) &= \tfrac{3}{36} = \tfrac{1}{12}, \\
P(5) = P(9) &= \tfrac{4}{36} = \tfrac{1}{9}, \\
P(6) = P(8) &= \tfrac{5}{36}, \\
P(7) &= \tfrac{6}{36} = \tfrac{1}{6}.
\end{aligned}$$

Example

Random experiment: sum of points while rolling two dice.
(a model with a uniform sample space).

The sample space is $\Omega' = [1,6]^2$ (Cartesian power of the set $\{1,2,3,4,5,6\}$).

The probability of every elementary event is $P(A) = \frac{1}{36}$.

Events for each sum are subset in Ω'

$$
\begin{aligned}
S_1 &= \emptyset, \\
S_2 &= \{(1,1)\}, \\
S_3 &= \{(1,2),(2,1)\}, \\
&\vdots \\
S_7 &= \{(1,6),(2,5),(3,4),(4,3),(5,2),(6,1)\}, \\
S_8 &= \{(2,6),(3,5),(4,4),(5,3),(6,2)\}, \\
&\vdots \\
S_{12} &= \{(6,6)\}.
\end{aligned}
$$

Probabilities of the events $S_1, \ldots, S_{12}$ are same as in the previous model.

We prefer sample spaces with uniform probability.

Definition

Complementary event to the even A is denoted by $\overline{A}$ and the following holds $\overline{A} = \Omega \setminus A$.

Theorem

The probability of the complementary event $\overline{A}$ to the event A is $P(\overline{A}) = 1 - P(A)$.

Example

We roll an ordinary dice.
Let A be the event "rolling a dice we got 1 or 2",
the complementary event $\overline{A}$ is the event "we rolled neither 1 nor 2", which essentially means "we rolled 3, 4, 5, or 6."

Handy for evaluating probabilities in uniform samplespaces.
Usually, we count selections or arrangements.

Conditional probability of event A given B occurs

If event B has nonzero probability, then the conditional probability of event A given B occurs we denote by $P(A|B)$ and it is given by

$$P(A|B) = \frac{P(A \cap B)}{P(B)}.$$

The conditional probability of event A given B occurs is the probability of the event A, provided the event B occurred.

Example

What is the probability that rolling two dice we get the sum 7, provided that on some dice we got 5.
It is easy to evaluate $P(A \cap B) = \frac{2}{36}$ $(5+2$ or $2+5)$.
When evaluating $P(B)$ don't count $5+5$ twice, thus $P(B) = \frac{11}{36}$.
We get $P(A|B) = \dfrac{P(A \cap B)}{P(B)} = \dfrac{\frac{2}{36}}{\frac{11}{36}} = \dfrac{2}{11}$.

Beware! "Intuition" is often misleading

In the previous example we showed, that when rolling two dice the conditional probability of A "the sum is 7" provided the event B "there is a 5 on at least one dice", je $P(A|B) = \frac{2}{11}$.

But this is *not* the correct solution in the following problem:

Example

A croupier rolls two dice, checks the outcames and announces "There is at least one 5. What is the probability of the sum being 7?"

Probability $\frac{2}{11}$ is the solution to a similar but differently worded problem:

Example

A croupier rolls two dice, he will check the outcome for 5. If no 5 is obtained, he rolls again, until a 5 appears. Then he announces "There is at least one 5. What is the probability of the sum being 7?"

Reason:

It is not a conditional probability $P(A|B)$ if the event B may/did not occur and probability $P(B)$ is zero.

Independent events

It is intuitively clear what independence of events is. Informally:
The probability of one event is not influenced by the result of another event.
(similar to *independent selections/arrangements* from Chapter 2.)

Independent events:

- two consecutive rolls of a dice,
- one roll with two or more dice,
- rolling a dice and shuffling cards,
- drawing from a urn and tossing the ball back
  (two draws in Powerball)

Dependent events:

- top and bottom face numbers on a dice,
- first and second card in a deck,
- consecutive drawings from an urn while keeping drawn balls outside.

Definition

Independent events A, B are such two events, that

$$P(A \cap B) = P(A) \cdot P(B).$$

Alternative definition:

Event A is independent of event B, if the probability of occurrence of event A while event B occurs is the same as the probability of event A

$$\frac{P(A \cap B)}{P(B)} = \frac{P(A)}{P(\Omega)} \qquad \text{or} \qquad P(A|B) = P(A).$$

Both definitions are equivalent.

Questions

From the "Sportka" urn (we draw one ball and then another ball)

- what is the probability that the first ball is 1?
- what is the probability that the second ball is 2?
- what is the probability that second ball is 2, provided first ball was 1?
- how do the probabilities change if we return the first ball before drawing the second ball?

Expected value

Let us explore random experiments whose outcome is a (natural) number. Let us concentrate on problems with finitely many possible outcomes.

Definition of a random variable X

The outcome of a random experiment, which gives a number as a result will we call a random variable X.

Definition of expected value

Let X be a random variable that can have k possible outcomes from the set $\{h_1, h_2, \ldots, h_k\}$, where h_i occurs with the probability p_i, a $p_1 + p_2 + \cdots + p_k = 1$. *The expected value of* X is the number

$$E(X) = EX = \sum_{i=1}^{k} p_i h_i = p_1 \cdot h_1 + p_2 \cdot h_2 + \cdots + p_k \cdot h_k.$$

Thus, it represents the average amount one "expects" as the outcome of the random trial when identical experiments are repeated many times.

Example

What is the expected value when rolling a dice?

$$E(K) = \frac{1}{6} \cdot 1 + \frac{1}{6} \cdot 2 + \frac{1}{6} \cdot 3 + \frac{1}{6} \cdot 4 + \frac{1}{6} \cdot 5 + \frac{1}{6} \cdot 6 = \frac{21}{6} = 3.5.$$

Example

What is the expected value when rolling a dice on which 6 has the probability of occurrence twice as high as any other number?

$$E(K) = \frac{1}{7} \cdot 1 + \frac{1}{7} \cdot 2 + \frac{1}{7} \cdot 3 + \frac{1}{7} \cdot 4 + \frac{1}{7} \cdot 5 + \frac{2}{7} \cdot 6 = \frac{27}{7} \doteq 3.8571.$$

Example

There is a 20\$ a 5\$ and a 1\$ banknote. We pick a banknote by random. Every bigger value has a two times bigger probability to be chosen than the smaller. What is the expected value of the banknote M?

$$p_1 = \frac{1}{7},\ p_5 = \frac{2}{7},\ p_{20} = \frac{4}{7}, \quad E(M) = \frac{1}{7} \cdot 1 + \frac{2}{7} \cdot 5 + \frac{4}{7} \cdot 20 = \frac{91}{7} = 13.$$

Sum of expected values

For any two random variables X, Y the following holds

$$E(X + Y) = E(X) + E(Y).$$

Product of expected values

For any two independent random variables X, Y the following holds

$$E(X \cdot Y) = E(X) \cdot E(Y).$$

Example

What is the expected value of the sum while rolling two dice?

Using the results from the previous example and by Sum of expected values theorem

$$E(K_1 + K_2) = E(K_1) + E(K_2) = 3.5 + 3.5 = 7.$$

Examples

What is the expected value of the product of numbers of points while rolling two dice?

Using the first example and by Product of expected values theorem

$$E(K_1 \cdot K_2) = E(K_1) \cdot E(K_2) = 3.5 \cdot 3.5 = 12.25.$$

Similarly $E(K_1 \cdot K_2) = \sum_{i=1}^{6} \sum_{j=1}^{6} \frac{1}{36} \cdot i \cdot j = \frac{441}{36} = \frac{49}{4} = 12.25$

What is the expected value of the product of the numbers of points on the top and bottom face while rolling one dice?

The expected value on the top face is 3.5 and on the bottom face also 3.5. But the expected value of their product is not $3.5 \cdot 3.5 = 12.25$, since the two variables are not independent.

We can evaluate the expected value by definition

$$\frac{1}{6}(1 \cdot 6 + 2 \cdot 5 + 3 \cdot 4 + 4 \cdot 3 + 5 \cdot 2 + 6 \cdot 1) = \frac{56}{6} \doteq 9.3333.$$

Carefully about the independence of events while multiplying expected values!

Random selections and arrangements

Frequently used *finite uniform random selections* in discrete mathematics:

Random subset Given an n-element set we pick any of the 2^n subsets, each with the probability 2^{-n}.

Random permutation Among all $n!$ permutations of a given n-element set we pick one with the probability $1/n!$.

Random combination Among all $\binom{n}{k}$ k-combinations of a given n-element set we pick one with the probability $1/\binom{n}{k}$.

Random bit sequence We obtain an arbitrary long sequence of 0 and 1 so that any subsequent bit is chosen with probability $\frac{1}{2}$ (does not depend on any previous bit, similarly to flipping a coin). Each subsequence of this random sequence is equally likely to occur.

Next lecture

Chapter 4. Counting and combinatorial identities

- inclusion exclusion principle
- double counting
- combinatorial identities
- proofs "by counting"

Discrete mathematics

Petr Kovář
petr.kovar@vsb.cz

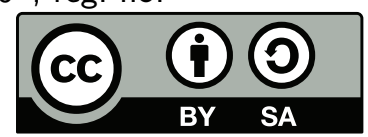
VŠB – Technical University of Ostrava

Winter term 2021/2022
DiM 470-2301/02, 470-2301/04, 470-2301/06

EUROPEAN UNION
European Structural and Investment Funds
Operational Programme Research,
Development and Education

MINISTRY OF EDUCATION,
YOUTH AND SPORTS

The translation was co-financed by the European Union and the Ministry of Education, Youth and Sports from the Operational Programme Research, Development and Education, project "Technology for the Future 2.0", reg. no. CZ.02.2.69/0.0/0.0/18_058/0010212.



About this file

This file is meant to be a guideline for the lecturer. Many important pieces of information are not in this file, they are to be delivered in the lecture: said, shown or drawn on board. The file is made available with the hope students will easier catch up with lectures they missed.

For study the following resources are better suitable:

- Meyer: Lecture notes and readings for an http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-042j-mathematics-for-computer-science-fall-2005/readings/" (weeks 1-5, 8-10, 12-13), MIT, 2005.
- Diestel: Graph theory http://diestel-graph-theory.com/ (chapters 1-6), Springer, 2010.

See also http://homel.vsb.cz/~kov16/predmety_dm.php

Chapter 4. More counting techiques

- inclusion/exclusion principle
- combinatorial identities
- binomial theorem
- pigeon-hole principle

Recapitulation

We introduced terms and symbols

- permutation $P(n)$
- k-combination with or without repetition, $C^*(n,k)$ or $C(n,k)$
- k-permutation with or without repetition, $P^*(n,k)$ or $P(n,k)$

We derived formulas for the number of various selections and arrangements.

But not all selections or arrangements can be counted as simple selections/arrangements. For example

- size of a union of sets
- number of bijections without fixed points
- number of decompositions of an n-element set to k disjoint subsets
- number of decompositions of n to k summands, while order of summands is irrelevant

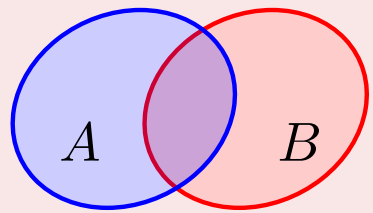
4.1. Inclusion exclusion principle

For small n we use it often intuitively:

Theorem

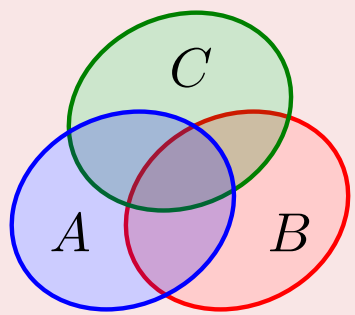
The number of elements in a union of two sets is:

$$|A \cup B| = |A| + |B| - |A \cap B|.$$



The number of elements in a union of three sets is:

$$|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |B \cap C| - |A \cap C| + |A \cap B \cap C|.$$



General form of the inclusion exclusion principle

The number of elements in a union of n sets is:

$$\left|\bigcup_{i=1}^{n} A_i\right| = \sum_{\substack{J\subseteq\{1,\ldots,n\}\\ J\neq\emptyset}} (-1)^{|J|-1} \cdot \left|\bigcap_{i\in J} A_i\right|.$$

To count the cardinality of a union, we

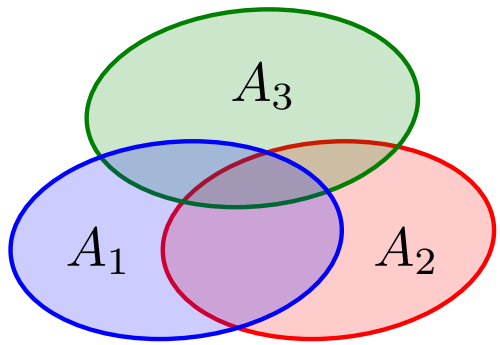
- sum the cardinalities of all sets,
- subtract the cardinalities of intersections of all pairs of sets,
- add the cardinalities of intersections of all triples of sets,
- subtract the cardinalities of intersections of all quadruples of sets,
- ...

Cardinality of union of three sets

For example for $n = 3$ we get

$$\left|\bigcup_{i=1}^{3} A_i\right| = \sum_{\substack{J\subseteq\{1,2,3\}\\ J\neq\emptyset}} (-1)^{|J|-1} \cdot \left|\bigcap_{i\in J} A_i\right| =$$

$$= |A_1| + |A_2| + |A_3| -$$
$$- |A_1 \cap A_2| - |A_1 \cap A_3| - |A_2 \cap A_3| +$$
$$+ |A_1 \cap A_2 \cap A_3|.$$

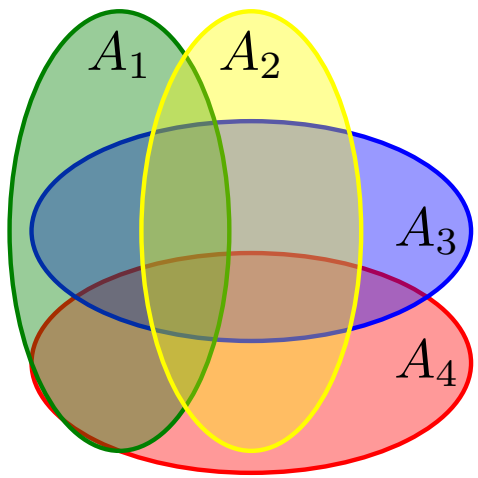


Cardinality of union of four sets

For example for $n = 4$ we get

$$\left|\bigcup_{i=1}^{4} A_i\right| = \sum_{\substack{J\subseteq\{1,2,3,4\}\\ J\neq\emptyset}} (-1)^{|J|-1} \cdot \left|\bigcap_{i\in J} A_i\right| =$$

$$\begin{aligned}
= \quad & |A_1| + |A_2| + |A_3| + |A_4| - \\
- \quad & |A_1 \cap A_2| - |A_1 \cap A_3| - |A_2 \cap A_3| - |A_1 \cap A_4| - |A_2 \cap A_4| - |A_3 \cap A_4| \\
+ \quad & |A_1 \cap A_2 \cap A_3| + |A_1 \cap A_2 \cap A_4| + |A_1 \cap A_3 \cap A_4| + |A_2 \cap A_3 \cap A_4| - \\
- \quad & |A_1 \cap A_2 \cap A_3 \cap A_4|.
\end{aligned}$$



Special case of inclusion exclusion principle

A simpler form (with fewer summands), if the intersections of i sets have always the same cardinality:

$$\left|\bigcup_{j=1}^{n} A_j\right| = \sum_{i=1}^{n} (-1)^{i-1} \cdot \binom{n}{i} \cdot \left|\bigcap_{j=1}^{i} A_j\right| .$$

To count the cardinality of a union, we

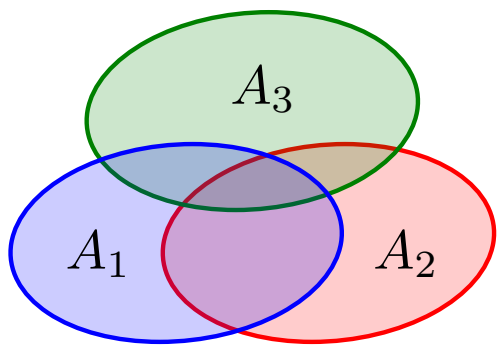
- take the number of one-element sets $\times$ size of A_1,
- subract number of two-element sets $\times$ size of pair-set intersections,
- add number of three-element sets $\times$ size of tripple-set intersections,
- subract number of four-element sets $\times$ size of quadruple-set intersections,
- ...

Cardinality of the union of three sets if each set and each intersection have the same cardinality

For $n = 3$ we have

$$\left|\bigcup_{i=1}^{3} A_i\right| = \sum_{k=1}^{3}(-1)^{k-1} \cdot \binom{3}{k} \cdot \left|\bigcap_{j=1}^{k} A_j\right| =$$

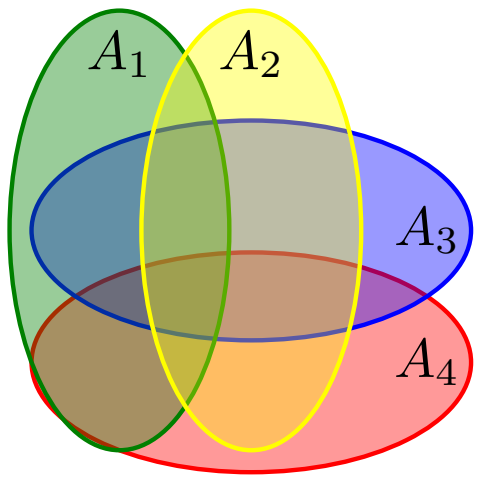
$$= \binom{3}{1} \cdot |A_1| - \binom{3}{2} \cdot |A_1 \cap A_2| + \binom{3}{3} \cdot |A_1 \cap A_2 \cap A_3|.$$



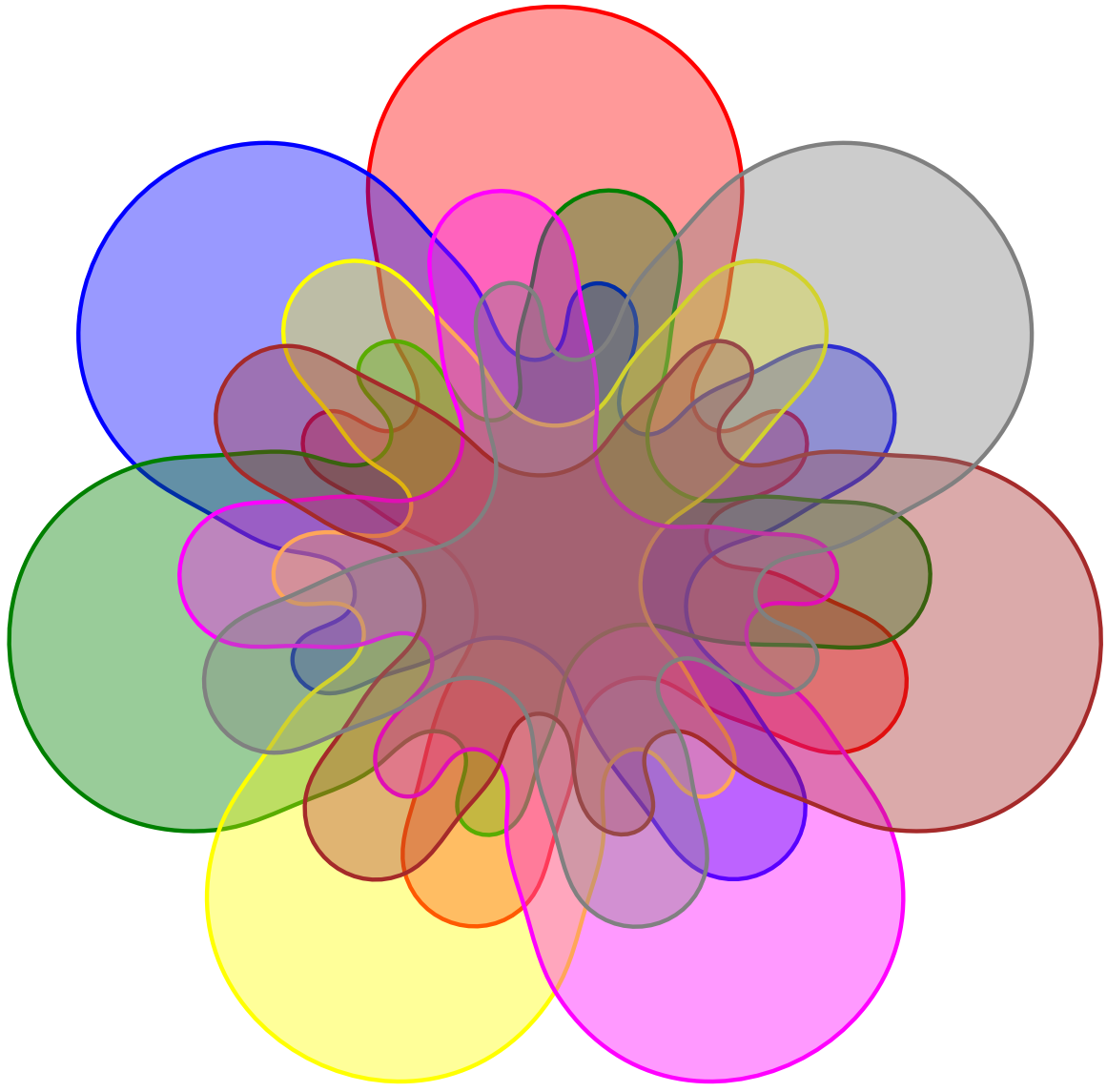
Cardinality of the union of four sets if each set and each intersection have the same cardinality

For $n = 4$ we have

$$\begin{aligned}\left|\bigcup_{i=1}^{4} A_i\right| &= \sum_{k=1}^{n}(-1)^{k-1} \cdot \binom{n}{k} \cdot \left|\bigcap_{j=1}^{k} A_j\right| = \\ &= \binom{4}{1} \cdot |A_1| - \binom{4}{2} \cdot |A_1 \cap A_2| + \\ &+ \binom{4}{3} \cdot |A_1 \cap A_2 \cap A_3| - \binom{4}{4} |A_1 \cap A_2 \cap A_3 \cap A_4|.\end{aligned}$$



Venn diagram for seven sets – Adelaide



Example

There are 25 students in a class. 17 study English and 10 German. 4 study English and German, 4 English and French, 2 German and French and one all three languages. How many students study only French?

We denote the sets by E, G a F. We know

$$|E| = 17,\ |G| = 10,\ |E \cap G| = |E \cap F| = 4,\ |G \cap F| = 2,\ |E \cap G \cap F| = 1$$

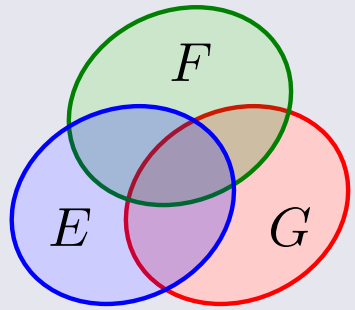
From the equation

$$|E \cup G \cup F| = |E| + |G| + |F| - |E \cap G| - |G \cap F| - |E \cap F| + |E \cap G \cap F|$$

it follows

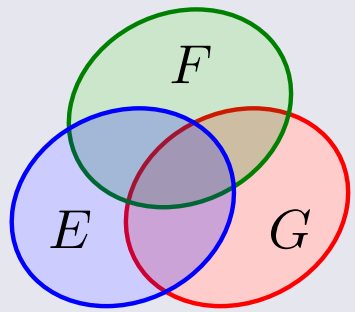
$$|F| = |E \cup G \cup F| - |E| - |G| + |E \cap G| + |G \cap F| + |E \cap F| - |E \cap G \cap F|$$

$$|F| = 25 - 17 - 10 + 4 + 4 + 2 - 1 = 7.$$



Example (continued)

But some of these 7 students study also other languages!



Just French

$$x = |F| - |E \cap F| - |G \cap F| + |E \cap G \cap F|$$
$$x = 7 - 4 - 2 + 1 = 2 \text{ students.}$$

2 students study just French.

Combinatorial identities

For binomial coefficients we can derive many interesting formulas. There is an entire part of Discrete mathematics dealing with them.

Lemma

For all $n \geq 0$ the following holds

$$\binom{n}{0} = \binom{n}{n} = 1.$$

Statement, proof of which is just a substitution and one or two simple steps we consider as obvious and their proof we do not write down.

Compare the number of corresponding subsets.

More combinatorial identities

Lemma

For all $n \geq k \geq 0$ the following holds

$$\binom{n}{k} = \binom{n}{n-k}.$$

If the proof requires some elaborate step, "trick", or genuine derivation, it is customary to give some explanation.

Compare the number of corresponding subsets.

Lemma

For all $n \geq k \geq 0$ the following holds

$$\binom{n}{k} + \binom{n}{k+1} = \binom{n+1}{k+1}.$$

Proof (direct by substitution and derivations)

$$\binom{n}{k} + \binom{n}{k+1} = \frac{n!}{k! \cdot (n-k)!} + \frac{n!}{(k+1)! \cdot (n-k-1)!} =$$

$$= \frac{n! \cdot (k+1) + n! \cdot (n-k)}{(k+1)! \cdot (n-k)!} = \frac{n! \cdot (n+1)}{(k+1)! \cdot (n-k)!} =$$

$$= \frac{(n+1)!}{(k+1)! \cdot ((n+1)-(k+1))!} = \binom{n+1}{k+1}.$$

□

Combinatorial proof is explanatory:
Compare the number of $(k+1)$-element subsets of some $(n+1)$-element set.

□

Notion of the binomial coefficient

These formulas are an *alternative definition of binomial coefficients*.

$$\binom{n}{0} = \binom{n}{n} = 1 \qquad \binom{n}{k} = \binom{n}{n-k} \qquad \binom{n}{k} + \binom{n}{k+1} = \binom{n+1}{k+1}.$$

The value of the binomial coefficient is uniquely determined by these equalities

- no need for factorials,
- each value can be (recursively) evaluated.

Pascal's triangle

$$\binom{0}{0}$$

$$\binom{1}{0} \quad \binom{1}{1}$$

$$\binom{2}{0} \quad \binom{2}{1} \quad \binom{2}{2}$$

$$\binom{3}{0} \quad \binom{3}{1} \quad \binom{3}{2} \quad \binom{3}{3}$$

$$\binom{4}{0} \quad \binom{4}{1} \quad \binom{4}{2} \quad \binom{4}{3} \quad \binom{4}{4}$$

$$\binom{5}{0} \quad \binom{5}{1} \quad \binom{5}{2} \quad \binom{5}{3} \quad \binom{5}{4} \quad \binom{5}{5}$$

. . . . . .

Pascal's triangle

$$\binom{0}{0} = 1$$

$$\binom{1}{0} = 1 \quad \binom{1}{1} = 1$$

$$\binom{2}{0} = 1 \quad \binom{2}{1} \quad \binom{2}{2} = 1$$

$$\binom{3}{0} = 1 \quad \binom{3}{1} \quad \binom{3}{2} \quad \binom{3}{3} = 1$$

$$\binom{4}{0} = 1 \quad \binom{4}{1} \quad \binom{4}{2} \quad \binom{4}{3} \quad \binom{4}{4} = 1$$

$$\binom{5}{0} = 1 \quad \binom{5}{1} \quad \binom{5}{2} \quad \binom{5}{3} \quad \binom{5}{4} \quad \binom{5}{5} = 1$$

All border elements are 1.

Pascal's triangle

$$\binom{0}{0} = 1$$

$$\binom{1}{0} = 1 \quad \binom{1}{1} = 1$$

$$\binom{2}{0} = 1 \quad \binom{2}{1} = 2 \quad \binom{2}{2} = 1$$

$$\binom{3}{0} = 1 \quad \binom{3}{1} = 3 \quad \binom{3}{2} = 3 \quad \binom{3}{3} = 1$$

$$\binom{4}{0} = 1 \quad \binom{4}{1} = 4 \quad \binom{4}{2} = 6 \quad \binom{4}{3} = 4 \quad \binom{4}{4} = 1$$

$$\binom{5}{0} = 1 \quad \binom{5}{1} = 5 \quad \binom{5}{2} = 10 \quad \binom{5}{3} = 10 \quad \binom{5}{4} = 5 \quad \binom{5}{5} = 1$$

All border elements are 1.

All inner elements equal the sum of two elements immediately above.

Binomial Theorem

Binomial Theorem

For all $n > 0$ the following holds

$$(1+x)^n = \binom{n}{0} + \binom{n}{1}x + \binom{n}{2}x^2 + \cdots + \binom{n}{n-1}x^{n-1} + \binom{n}{n}x^n.$$

Proof The proof can run by induction, but there is a nice argument. Multiplying through we use the rule "multiply each element with each other". Thus in $\underbrace{(1+x)(1+x)\ldots(1+x)}_{n}$ each product x^k appears as many times as there are k-element selections from n parentheses. There are $\binom{n}{k}$ such different k-element subsets. □

Combinatorial identities derived from Binomial Theorem

Binomial Theorem

For all $n > 0$ the following holds

$$(1+x)^n = \binom{n}{0} + \binom{n}{1}x + \binom{n}{2}x^2 + \cdots + \binom{n}{n-1}x^{n-1} + \binom{n}{n}x^n.$$

From the Binomial theorem follows for $n \geq 0$

$$\binom{n}{0} + \binom{n}{1} + \binom{n}{2} + \binom{n}{3} + \cdots + \binom{n}{n-1} + \binom{n}{n} = 2^n\,.$$

The number of all subsets on an n-element set is 2^n.

From the Binomial theorem follows for $n > 0$

$$\binom{n}{0} - \binom{n}{1} + \binom{n}{2} - \binom{n}{3} + \ldots - (-1)^n\binom{n}{n-1} + (-1)^n\binom{n}{n} = 0\,.$$

The number of odd-sized subsets on an n-element set is the same as he number of even-sized subsets.

Proofs "by counting"

Sometimes we have to show that there exists an element with a certain property, but we cannot find/construct one. Such proofs are called non-constructive.

Instead to "construct" a solution, we show by "counting" there has to be at least one.

The pigeon-hole principle (Dirichlet's principle)

When distributing $\ell + 1$ (or more) objects into ℓ boxes, there has to be a box with at least two objects.

Proofs by counting

The existence of a possibility will follow from the fact that there are too few cases in which the possibility does not occur.

Example

We see three cars entering a tunnel, but only two cars leaving the tunnel. This means there is one car left in the tunnel (though we do not see it).

Example

8 friends went on a 9 day vacation. Each day some triple of them went for a trip. Show, that at least one pair of friends didn't go together on a trip.

Proof Checking of all possibilities would take long...
The proof by counting is easy: In one triple there are 3 pairs, thus after 9 days there was *at most* $9 \cdot 3$ *pairs* on trips. But $9 \cdot 3 = 27 < \binom{8}{2} = 28$, thus at least one pair is missing.

Question

Are there two people on Earth with the same number of hair?

Example

In a drawer there are 30 pairs of black socks, 10 pairs of brown socks, and 3 pairs of white socks. How many socks we have to take (without light or looking) to guarantee, that we have at least one pair of the same color?

"Boxes" in the Pigeon-hole principle are the three colors. While taking four socks (not distinguishing the right or left sock), at least two of them have to be of the same color.

Question

We have four natural numbers. Show, that among them there are two numbers difference of which is divisible by 3.

Question

We have 3 natural numbers. Show, that among them there are two numbers difference of which is divisible by some prime.

Handshaking problem

There are n people in the room, some of them shook hands. Show that there are always at least two people who performed the same number of handshakes.

Example

We have five natural numbers. Show that there are always two among them, such that their sum is divisible by 9.

Proof (incorrect!) We have a total of 9 different classes modulo 9. Among five numbers we obtain 10 different sums. Surely, there has to be at least one sum in each class, in some class there will be at least two sums. Thus the pair which is in class "0", has its sum divisible by 9. □

Question

Why is the proof not correct?

Hint: try to verify the argument for the following set of five numbers: $\{0, 2, 4, 6, 8\}$.

Next lecture

Chapter 5. Recurrence relations

- motivation
- sequences given by recurrence relations
- methods of solving recurrence relations

Discrete mathematics

Petr Kovář
petr.kovar@vsb.cz

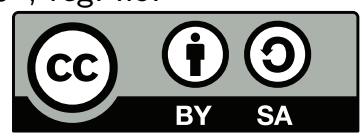
VŠB – Technical University of Ostrava

Winter term 2021/2022
DiM 470-2301/02, 470-2301/04, 470-2301/06

EUROPEAN UNION
European Structural and Investment Funds
Operational Programme Research,
Development and Education

MINISTRY OF EDUCATION,
YOUTH AND SPORTS

The translation was co-financed by the European Union and the Ministry of Education, Youth and Sports from the Operational Programme Research, Development and Education, project "Technology for the Future 2.0", reg. no. CZ.02.2.69/0.0/0.0/18_058/0010212.



About this file

This file is meant to be a guideline for the lecturer. Many important pieces of information are not in this file, they are to be delivered in the lecture: said, shown or drawn on board. The file is made available with the hope students will easier catch up with lectures they missed.

For study the following resources are better suitable:

- Meyer: Lecture notes and readings for an http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-042j-mathematics-for-computer-science-fall-2005/readings/" (weeks 1-5, 8-10, 12-13), MIT, 2005.
- Diestel: Graph theory http://diestel-graph-theory.com/ (chapters 1-6), Springer, 2010.

See also http://homel.vsb.cz/~kov16/predmety_dm.php

Lecture overview

Kapitola 5. Recurrence relations

- motivation
- sequences given by recurrences
- main problem
- methods of solving
- examples

5. Recurrence relations

Last chapter already mentioned than not all selections and arrangements can be expressed in simple "closed" formulas mentioned in Section 2.
Today we mention several typical problems that we encounter when using recursive algorithms.
We how, how the complexity of certain such algorithms can be expressed.

Typical examples of recursive algorithms or recursive approaches

- merge sort
- dynamic programming
- using n pairs of parentheses on $n + 1$ terms
- number of "ordered rooted trees" in chapter UTG 4

5.1. Motivation examples

The classic Fibonacci sequence is notoriously known.

Fibonacci sequence

A young pair of rabbits has been released on an island. The rabbits are mature at the age of two months, after that they they raise another pair of rabbits each month. What is the number f_n of pairs of rabbits after n months?

Clearly $f_1 = f_2 = 1$.
For $n \geq 3$ is the number of pairs given by

- the number of pairs in the previous months,
- the number of pairs of two months age f_{n-2}, that became mature and can breed.

Altogether we have $f_n = f_{n-1} + f_{n-2}$ pair, if dying of age is neglected.

The solution, i.e. the formula for f_n we derive at the end of the lecture.

Towers of Hanoi

We have three pegs and a set of discs of different sizes. All discs are on one peg arranged according their size. The task is to move all discs to another peg while

- always one discs is moved,
- never a larger disc can be on top of a smaller one.

What is the smallest number of moves H_n to move the entire tower of n discs?

Towers of Hanoi

To move the largest disc, $n-1$ smaller discs have to be moved to another peg using H_{n-1} moves.

We divide the total number of moves H_n into three parts

- first using H_{n-1} moves transfer $n-1$ smaller discs on the third peg,
- then using a single move transfer the largest disc to the desired peg,
- finally using H_{n-1} moves transfer $n-1$ smaller discs on top of the largest disc.

The total number of moves is given by the recurrence relation

$$H_n = 2H_{n-1} + 1,$$

while clearly $H_1 = 1$.

The solution, i.e. the formula for H_n we derive at the end of the lecture.

Bit strings without adjacent zeroes

Example

How many bit strings of length n are there, that have no two adjacent zeroes? (important in bar codes)

Denote the number of required bit strings with n bits by a_n.
We distinguish, if a string of n bits end with a 0 or a 1 (assume $n \geq 3$).

- if the last bit is 1, then there are precisely a_{n-1} such strings with an additional bit 1,
- if the last bit is 0, then the next-to-the-last bit has to be 1 and there are a_{n-2} such strings with additional bits 10 at the end.

These are all the options, therefore the total number of strings with n bits, where no two zeroes are adjacent, is

$$a_n = a_{n-1} + a_{n-2}.$$

It remains to figure out that $a_1 = 2$, $a_2 = 4 - 1 = 3$.

At the end of the lecture we derive the formula for a_n.
Notice: a_n is similar to, yet different from the Fibonacci sequence.

Code words with an even number of zeroes

Example

A computer system works with keywords made from digits 0, 1, . . . , 9. A valid code word has an even number of zeroes. How man such code words of length n exist?

Let x_n denote the number of such code words with n digits.
We distinguish if the n-th digit of a code word is 0 or no (suppose $n \geq 2$).

- code words with last digit not 0 are precisely $9x_{n-1}$, where the last digit 1,2,. . . , 9 was added to some of x_{n-1} code words of length $n-1$,
- code words with last digit 0, are precisely those the are not code words x_{n-1}.

These are all possibilities, therefore the total of code words of length n with an even number of zeroes is

$$x_n = 9x_{n-1} + (10^{n-1} - x_{n-1}) = 8x_{n-1} + 10^{n-1}.$$

It remains to notice that $x_1 = 9$. We search for the formula expressing x_n.

Number of ways to parenthesize $n+1$ terms with n parentheses

Example

We have an expression with $n+1$ terms, priority of operation $\oplus$ is given by n pairs of parentheses. Kolik existuje různých způsobů uzávorkování C_n?

$C_0 = 1$, since x_1 is unique.
$C_1 = 1$, since $(x_1 \oplus x_2)$ is unique.
$C_2 = 2$, since $((x_1 \oplus x_2) \oplus x_3)$, $(x_1 \oplus (x_2 \oplus x_3))$ are two possibilities.
$C_3 = 5$, there are 5 ways $(((x_1 \oplus x_2) \oplus x_3) \oplus x_4)$, $((x_1 \oplus (x_2 \oplus x_3)) \oplus x_4)$, $((x_1 \oplus x_2) \oplus (x_3 \oplus x_4))$, $(x_1 \oplus ((x_2 \oplus x_3) \oplus x_4))$, $(x_1 \oplus (x_2 \oplus (x_3 \oplus x_4)))$.

In general the most our parenthesis ha only one operator "$\oplus$". Notice the operation is between two smaller terms, there are n different numbers of terms to the left of the operator in the outer parenthesis.

$$C_n = \sum_{k=0}^{n-1} C_k C_{n-k-1}$$

Recurrence relation $C_n = \sum_{k=0}^{n-1} C_k C_{n-k-1}$ appears in a number of different real life problems, so called. Catalan numbers.

5.2. Sequences give by recurrence relations

Recall:

Sequences are given by

- a list of first elements: $1, 3, 7, 15, 31, \ldots$
- a recurrence relation: $a_n = 2a_{n-1} + 1$, $a_0 = 1$
- a formula for n-th term: $a_n = 2^n - 1$

Now we deal with recurrence relations, every subsequent term can be evaluated based on previous terms.

Main problem

Find the formula for the n-th term.

- if it exists,
- if it is possible,
- and if we can do so.

Linear homogeneous recurrence relations of order k with constant coefficients

Linear homogeneous recurrence relations of order k with constant coefficients is a sequence given by a recurrence relation of the form

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k},$$

where $c_1, c_2, \ldots, c_k$ are real numbers, $c_k \neq 0$.

Let us explore the definition

- it is linear, because it is a linear combination of the previous terms,
- it is homogeneous, because there is **no** term without a_i,
- it is of order k, because a_n is given by k previous terms,
- it has constant coefficients, because each coefficient is a constant independent on n.

For a unique description of the sequence given by a recurrence relation of order k we have to *provide k first terms*.

Fibonacci sequence

Fibonacci sequence $f_n = f_{n-1} + f_{n-2}$ is a linear homogeneous recurrence relation of second order with constant coefficients.
First two terms are $f_1 = 1$, $f_2 = 1$.

Bit strings without adjacent zeroes

Sequence of the number of bit strings of length n, which have no adjacent zeroes $a_n = a_{n-1} + a_{n-2}$, is a linear homogeneous recurrence relation of order 2 with constant koeficients.
First two terms are $a_1 = 2$, $a_2 = 3$.

Hanoi tower

Sequence of the number of steps H_n *necessary to move the entire tower of n discs* $H_n = 2H_{n-1} + 1$ is a linear recurrence relation of the first order with constant coefficients, which is not homogeneous.
First term is $H_1 = 1$.

Code words with an even number of zeroes

The number of codewords made from digits 0, 1, . . . , 9, where each codeword has an even number of zeroes is a linear recurrence relation of the first order $x_n = 8x_{n-1} + 10^{n-1}$. First term is $x_1 = 9$.
This relation is not homogeneous, since 10^{n-1} *is not* a coefficient at a_i.
This relation has not constant coefficients, since term 10^{n-1} *depends* on n.

Catalan numbers

The sequence of Catalan numbers $C_n = \sum_{k=0}^{n-1} C_k C_{n-k-1}$ is given by a homogeneous recurrence relation. First terms are $C_1 = 1$, $C_2 = 2$.
This recurrence relation is not linear, because we multiply terms C_k, C_{n-k} and has no fixed order, because the number of terms grows with n.

Example

Recurrence relation $a_n = a_{n-1} \cdot a_{n-2}$ is a homogeneous recurrence relation of second order with constant coefficients. First two terms are $a_1 = 1$, $a_2 = 2$.
This recurrence relation is not linear, since we multiply terms a_{n-1}, a_{n-2}.

5.3. Methods for solving recurrence relations

Main problem of solving recurrence relations

If a linear recurrence relation of (small) order k with constant coefficients is given by a recurrence relation and sufficient first terms, we can "solve" this recurrence relation. This means, we find a formula for the n-th term, which evaluates a_n *without the knowledge of previous terms*.

We provide a general framework:

- first we set up a so called characteristic equation,
- we find the roots of the characteristic equation,
- based on the roots we set up a general solution,
- based on the value of the given first terms of the sequence we evaluate coefficients of the general solution.

We start with simple examples.

Characteristic equation and its roots

One can show (e.g. using so called generating functions), that the solution of the linear homogeneous recurrence relations with constant coefficients will have the form $a_n = r^n$, where r is a constant.
Substituting into the recurrence relation we obtain

$$\begin{aligned} a_n &= c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k} \\ r^n &= c_1 r^{n-1} + c_2 r^{n-2} + \cdots + c_k r^{n-k} \\ r^k &= c_1 r^{k-1} + c_2 r^{k-2} + \cdots + c_k r^{k-k} \\ 0 &= r^k - c_1 r^{k-1} - c_2 r^{k-2} - \cdots - c_k. \end{aligned}$$

The last equation is the characteristic equation of the recurrence relation. Clearly, the solution of this equation in variable r are the roots r_i. We call them characteristic roots.

Generalization of the solution

We split the solution of linear homogeneous recurrence relation with constant coefficients into several steps.

The next step includes solutions to a larger family of recurrence relations:

- first we show how a general form of the solution of a linear homogeneous recurrence relation of order 2 with constant coefficients looks like,
  - if there are two distinct real characteristic roots,
  - if there are two identical real characteristic roots.
- Next we show a general form of the solution of a linear homogeneous recurrence relation of order k with constant coefficients.
- Finally we provide a general form of the solution of a linear recurrence relation of order k with constant coefficients.

Theorem

Let c_1, c_2 be two real numbers. If the characteristic equation $r^2 - c_1 r - c_2 = 0$ has two distinct (real) roots r_1, r_2, then the solution of the recurrence relation $a_n = c_1 a_{n-1} + c_2 a_{n-2}$ is of the form $a_n = \alpha_1 r_1^n + \alpha_2 r_2^n$, for $n = 0, 1, 2, \ldots$.

There is a stronger claim, which we omit here.

Example

Solve the recurrence relation $a_n = a_{n-1} + 2a_{n-2}$, where $a_0 = 2$, $a_1 = 7$.

We follow the steps suggester earlier:
We expect the solution of the form $a_n = r^n$. Substituting to the recurrence relation we get the characteristic equation

$$\begin{aligned} r^2 - r - 2 &= 0 \\ (r+1)(r-2) &= 0. \end{aligned}$$

Characteristic roots are $r_1 = 2$, $r_2 = -1$. The general solution has the form

$$a_n = \alpha_1 2^n + \alpha_2 (-1)^n.$$

Substituting a_0, a_1 we get two equations in two variables α_1, α_2.

$$\begin{aligned} a_0 = 2 &= \alpha_1 \cdot 1 + \alpha_2 \cdot 1 \\ a_1 = 7 &= \alpha_1 \cdot 2 + \alpha_2 \cdot (-1) \end{aligned}$$

Solving the equation yields $\alpha_1 = 3$, $\alpha_2 = -1$, thus the general solution is

$$a_n = 3 \cdot 2^n - 1 \cdot (-1)^n.$$

Indeed,

- the formula $a_n = 2 \cdot_1 2^n - 1(-1)^n$ for $n = 0, 1, 2, \ldots$
- the recurrence relation $a_n = a_{n-1} + 2a_{n-2}$, where $a_0 = 2$, $a_1 = 7$

describe the same sequence:

$$2, 7, 11, 25, 47, 97, 191, 385, 767, 1\,537, 3\,071, \ldots$$

Now we examine the case with two identical characteristic roots.

Theorem

Let c_1, c_2 be two real numbers, where $c_2 \neq 0$. If the characteristic equation $r^2 - c_1 r - c_2 = 0$ has a double (real) root r_0, then the solution of the recurrence relation $a_n = c_1 a_{n-1} + c_2 a_{n-2}$ is of the form $a_n = \alpha_1 r_0^n + \alpha_2 n r_0^n$, for $n = 0, 1, 2, \ldots$.

Example

Solve the recurrence relation $a_n = 10a_{n-1} - 25a_{n-2}$, where $a_1 = 3$, $a_2 = 5$.

Example

Solve the recurrence relation $a_n = 10a_{n-1} - 25a_{n-2}$, where $a_1 = 3$, $a_2 = 5$.

We follow the same steps:
We expect the solution of the form $a_n = r^n$. Substituting to the recurrence relation we get the characteristic equation

$$\begin{aligned} r^2 - 10r + 25 &= 0 \\ (r-5)(r-5) &= 0. \end{aligned}$$

Characteristic roots are $r_1 = r_2 = 5$, we denote $r_0 = 5$. The general solution has the form

$$a_n = \alpha_1 5^n + \alpha_2 n 5^n.$$

Substituting a_0, a_1 we get two equations in two variables α_1, α_2.

$$\begin{aligned} a_0 = 3 &= \alpha_1 \cdot 1 + 0 \\ a_1 = 5 &= \alpha_1 \cdot 5 + \alpha_2 \cdot 5 \end{aligned}$$

Solving the equation yields $\alpha_1 = 3$, $\alpha_2 = -2$, thus the general solution is

$$a_n = 3 \cdot 5^n - 2n5^n.$$

We found the formula for the n-th term

- $a_n = 3 \cdot 5^n - 2n5^n$ describes the same sequence as
- the recurrence relation $a_n = 10a_{n-1} - 25a_{n-2}$, where $a_1 = 3$, $a_2 = 5$.

Sequence

$$3, 5, -25, -375, -3\,125, -21\,875, -140\,625, \ldots$$

Solution of linear recurrence relations can be generalized to higher orders.

Theorem

Let $c_1, c_2, \ldots, c_k$ be k real numbers. If the characteristic equation $r^k - c_1 r^{k-1} - c_2 r^{k-2} - \cdots - c_k = 0$ has k distinct (real) roots $r_1, r_2, \ldots, r_k$, then the solution of the recurrence relation $a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k}$ is of the form $a_n = \alpha_1 r_1^n + \alpha_2 r_2^n + \cdots + \alpha_k r_k^n$, for $n = 0, 1, 2, \ldots$.

Example

Solve the recurrence relation $a_n = 4a_{n-1} - a_{n-2} - 6a_{n-3}$, where $a_0 = 6$ $a_1 = 5$, $a_2 = 13$.

We get the characteristic equation

$$r^3 - 4r^2 + r + 6 = 0.$$

Characteristic roots are $r_1 = -1$, $r_2 = 2$, $r_3 = 3$. The general solution has the form

$$a_n = \alpha_1(-1)^n + \alpha_2 2^n + \alpha_3 3^n.$$

Substituting a_0, a_1, a_2 we get three equations in three variables α_1, α_2, α_3.

$$\begin{aligned} a_0 = 6 &= \alpha_1 + \alpha_2 + \alpha_3 \\ a_1 = 5 &= -\alpha_1 + 2\alpha_2 + 3\alpha_3 \\ a_2 = 13 &= \alpha_1 + 4\alpha_2 + 9\alpha_3 \end{aligned}$$

Solving the equation yields $\alpha_1 = 2$, $\alpha_2 = 5$, $\alpha_3 = -1$, thus the general solution is

$$a_n = 2 \cdot (-1)^n + 5 \cdot 2^n - 3^n.$$

Solving general linear homogeneous recurrence relations with constant coefficients

Theorem

Let $c_1, c_2, \ldots, c_k$ be k real numbers. If the characteristic equation $r^k - c_1 r^{k-1} - c_2 r^{k-2} - \cdots - c_k = 0$ has t distinct roots $r_1, r_2, \ldots, r_t$ with multiplicities $m_1, m_2, \ldots, m_t$, then the solution of the recurrence relation $a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k}$ for $n = 0, 1, 2, \ldots$ has the form

$$\begin{aligned} a_n &= (\alpha_{1,1} + \alpha_{1,2} n + \cdots + \alpha_{1,m_1} n^{m_1-1}) r_1^n + \\ &\quad + (\alpha_{2,1} + \alpha_{2,2} n + \cdots + \alpha_{2,m_2} n^{m_2-1}) r_2^n + \\ &\quad + \cdots + (\alpha_{t,1} + \alpha_{t,2} n + \cdots + \alpha_{t,m_t} n^{m_t-1}) r_t^n \end{aligned}$$

To find the solution, we

1. get the characteristic equation,
2. find characteristic roots (if possible),
3. set up the general form of the solution with coefficients $\alpha_{i,j}$,
4. substitute k first (known) terms,
5. solve the system of equations with k variables,
6. set up the general solution.

Solving general linear non-homogeneous recurrence relations with constant coefficients

So far only homogeneous recurrence relations . . .

Solving non-homogeneous recurrence relations in two steps:

- general solution of the associated homogeneous recurrence relation,
- one particular solution of the non-homogeneous recurrence.

Theorem

Let $c_1, c_2, \ldots, c_k$ be k real numbers, let $F(n)$ be a function not identically zero.
If $a_n^{(p)}$ is a particular solution to the non-homogeneous linear recurrence relations with constant coefficients

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k} + F(n),$$

then every solution is of the form $a_n^{(p)} + a_n^{(h)}$, where $a_n^{(h)}$ is the general solution of the associated homogeneous recurrence relation

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k}.$$

Example

Show that $a_n^{(p)} = -n - 2$ is a (particular) solution of the recurrence relation $a_n = 2a_{n-1} + n$.

To verify a solution is easy: substitute and compare:

$$\begin{aligned} a_n &= 2a_{n-1} + n \\ -n - 2 &= 2\left(-(n-1) - 2\right) + n \\ -n - 2 &= -n - 2. \end{aligned}$$

Notice: the solution has the form $a_n^{(p)} = cn + d$.

Example

Show that $a_n^{(p)} = c \cdot 7^n$ is the form of a (particular) solution of the recurrence relation $a_n = 5a_{n-1} - 6a_{n-2} + 7^n$.

Again substitute and compare:

$$\begin{aligned} a_n &= 5a_{n-1} - 6a_{n-2} + 7^n \\ c \cdot 7^n &= 5c \cdot 7^{n-1} - 6c \cdot 7^{n-2} + 7^n \\ c &= \frac{49}{20}. \end{aligned}$$

Theorem

Let $c_1, c_2, \ldots, c_k$ be k real numbers, let $F(n)$ be a function not identically zero.
Suppose $a_n^{(p)}$ is a solution of the non-homogeneous linear recurrence relations with constant coefficients

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k} + F(n),$$

where $F(n) = (b_t n^t + b_{t-1} n^{t-1} + \cdots + b_1 n + b_0) s^n$.

1. When s is not a root of the characteristic equation of the associated homogeneous linear recurrence relations, then $a_n^{(p)}$ has the form

$$(p_t n^t + p_{t-1} n^{t-1} + \cdots + p_1 n + p_0) s^n.$$

2. When s is a root with multiplicity m of the characteristic equation of the associated homogeneous linear recurrence relations, then $a_n^{(p)}$ has the form

$$n^m (p_t n^t + p_{t-1} n^{t-1} + \cdots + p_1 n + p_0) s^n.$$

Example

Solve the recurrence relation $a_n = 2a_{n-1} + n$.

First we find the solution of the associated linear homogeneous recurrence relation

$$a_n = 2a_{n-1}.$$

The characteristic equation $r^n = 2r^{n-1}$ has a nonzero root $r = 2$.
Therefore the general solution has the form $a_n^{(h)} = \alpha 2^n$.

Substituting into the associated linear homogeneous recurrence relation we get

$$\alpha 2^n = 2 \cdot \alpha 2^{n-1}.$$

We evaluate $\alpha = 1$, therefore the solution of the associated linear homogeneous recurrence relation is $a_n^{(h)} = 1 \cdot 2^n = 2^n$.

Next we find a particular solution of the original non-homogeneous linear recurrence relation. By the previous theorem is

$$a_n^{(p)} = n(cn + d)2^n,$$

since base 2 is the root of the characteristic equation.

To find the constants we substitute the particular solution $a_n^{(p)} = n(cn+d)2^n$ into the recurrence relation $a_n = 2a_{n-1} + n2^n$.
We get

$$\begin{aligned} n(cn+d)2^n &= 2\cdot(n-1)(c(n-1)+d)2^{n-1} + n2^n \\ (cn^2+dn)2^n &= 2\cdot(c(n-1)^2+d(n-1))2^{n-1} + n2^n \\ (cn^2+dn)2^n &= (cn^2-2cn+c+dn-d+n)2^n \\ dn &= (-2c+d+1)n+(c-d). \end{aligned}$$

Comparing coefficients of the polynomials at n^1 and n^0 we get a system of linear equations

$$\begin{aligned} d &= -2c+d+1 \\ 0 &= c-d. \end{aligned}$$

The solution is $c = \frac{1}{2}$, $d = \frac{1}{2}$ and thus the particular solutions is $a_n^{(p)} = n(\frac{1}{2}n+\frac{1}{2})2^n = (n^2+n)2^{n-1}$.

The solution of the given recurrence relation is

$$a_n = a_n^{(h)} + a_n^{(p)} = 2^n + (n^2+n)2^{n-1} = (n^2+n+2)2^{n-1}.$$

5.4. Solving the motivation examples from the first section

Fibonacci sequence

Solve the recurrence relation $f_n = f_{n-1} + f_{n-2}$, where $f_0 = 0$, $f_1 = 1$.

We obtain the characteristic equation $r^2 - r - 1 = 0$.
Characteristic roots are $r_1 = (1 + \sqrt{5})/2$, $r_2 = (1 - \sqrt{5})/2$. The general solution has the form

$$f_n = \alpha_1 \left(\frac{1+\sqrt{5}}{2}\right)^n + \alpha_2 \left(\frac{1-\sqrt{5}}{2}\right)^n.$$

Substituting $f_0 = 0$, $f_1 = 1$ we get two equations in two variables α_1, α_2.

$$\begin{aligned} 0 &= \alpha_1 \cdot 1 + \alpha_2 \cdot 1 \\ 1 &= \alpha_1 \cdot \left(\frac{1+\sqrt{5}}{2}\right) + \alpha_2 \cdot \left(\frac{1-\sqrt{5}}{2}\right) \end{aligned}$$

Solving the system yields $\alpha_1 = \frac{\sqrt{5}}{5}$, $\alpha_2 = -\frac{\sqrt{5}}{5}$, thus, the general solution is

$$a_n = \frac{\sqrt{5}}{5} \cdot \left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{\sqrt{5}}{5} \cdot \left(\frac{1-\sqrt{5}}{2}\right)^n.$$

Towers of Hanoi

Solve the recurrence relation $H_n = 2H_{n-1} + 1$, where $H_1 = 1$.

It is not a homogeneous linear recurrence relation.
On the other hand it is a first order recurrence, we can obtain the solution differently.
Notice

$$\begin{aligned} H_n &= 2H_{n-1} + 1 \\ &= 2(2H_{n-2} + 1) + 1 = 2^2 H_{n-2} + 2 + 1 \\ &= 2^2(2H_{n-3} + 1) + 2 + 1 = 2^3 H_{n-2} + 2^2 + 2 + 1 \\ &\vdots \\ &= 2^{n-1} H_1 + 2^{n-2} + 2^{n-3} + \cdots + 2 + 1 \\ &= 2^{n-1} + 2^{n-2} + 2^{n-3} + \cdots + 2 + 1 \\ &= 2^n - 1. \end{aligned}$$

The solution of the linear non-homogeneous recurrence relation of the Towers of Hanoi is

$$H_n = 2^n - 1.$$

Bit strings with no adjacent zeroes

Solve the recurrence relation $a_n = a_{n-1} + a_{n-2}$, where $a_1 = 2$, $a_2 = 3$.

The characteristic equation is $r^2 - r - 1 = 0$.
Characteristic roots are $r_1 = (1 + \sqrt{5})/2$, $r_2 = (1 - \sqrt{5})/2$. The general solution has the form

$$a_n = \alpha_1 \left(\frac{1+\sqrt{5}}{2}\right)^n + \alpha_2 \left(\frac{1-\sqrt{5}}{2}\right)^n .$$

Substituting $a_1 = 2$, $a_2 = 3$ we get two equations in two variables α_1, α_2.

$$\begin{aligned} 2 &= \alpha_1 \cdot \left(\frac{1+\sqrt{5}}{2}\right) + \alpha_2 \cdot \left(\frac{1-\sqrt{5}}{2}\right) \\ 3 &= \alpha_1 \cdot \left(\frac{1+\sqrt{5}}{2}\right)^2 + \alpha_2 \cdot \left(\frac{1-\sqrt{5}}{2}\right)^2 \end{aligned}$$

Solving the system yields $\alpha_1 = \frac{5+\sqrt{5}}{10}$, $\alpha_2 = \frac{5-\sqrt{5}}{10}$, the general solution is

$$a_n = \frac{5+\sqrt{5}}{10} \cdot \left(\frac{1+\sqrt{5}}{2}\right)^n + \frac{5-\sqrt{5}}{10} \cdot \left(\frac{1-\sqrt{5}}{2}\right)^n .$$

Code words with an even number of zeroes

Solve the recurrence relation $x_n = 8x_{n-1} + 10^{n-1}$, where $x_1 = 9$.

This linear non-homogeneous recurrence relation we did not learn how to solve, its coefficients are not constant.

However, the solution is

$$a_n = \frac{1}{2} \cdot 8^n + \frac{1}{2} \cdot 10^n.$$

Merge sort

Merge sort is a well known algorithm for sorting a sequence of n numbers. It is a recursive algorithm.
Knowing the algorithm, it is easy to see, that the number of comparisons (and operations) M_n to sort a sequence of n terms can be bounded by a recurrence relation $M_n = 2M_{n-1} + n$, where $M_1 = 1$.

This non-homogeneous linear recurrence relation we cannot solve now, its coefficients are not constant.

HOwever it can be shown, that the solution describing the number of steps of the Merge sort algorithm, is a function of complexity

$$M_n = O(n \log n).$$

Next lecture

Chapter 6. Congruences and modular arithmetics

- motivation
- division and divisibility
- linear congruences in one variable
- methods of solving
- examples and applications

Discrete mathematics

Petr Kovář
petr.kovar@vsb.cz

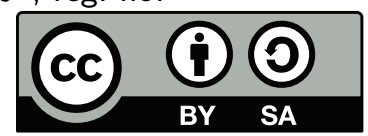
VŠB – Technical University of Ostrava

Winter term 2021/2022
DiM 470-2301/02, 470-2301/04, 470-2301/06

EUROPEAN UNION
European Structural and Investment Funds
Operational Programme Research,
Development and Education

MINISTRY OF EDUCATION,
YOUTH AND SPORTS

The translation was co-financed by the European Union and the Ministry of Education, Youth and Sports from the Operational Programme Research, Development and Education, project "Technology for the Future 2.0", reg. no. CZ.02.2.69/0.0/0.0/18_058/0010212.



About this file

This file is meant to be a guideline for the lecturer. Many important pieces of information are not in this file, they are to be delivered in the lecture: said, shown or drawn on board. The file is made available with the hope students will easier catch up with lectures they missed.

For study the following resources are better suitable:

- Meyer: Lecture notes and readings for an http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-042j-mathematics-for-computer-science-fall-2005/readings/" (weeks 1-5, 8-10, 12-13), MIT, 2005.
- Diestel: Graph theory http://diestel-graph-theory.com/ (chapters 1-6), Springer, 2010.

See also http://homel.vsb.cz/~kov16/predmety_dm.php

Lecture overview

Chapter 6. Congruences and modular arithmetics

- motivation
- division and divisibility
- linear congruences in one variable
- methods of solving
- examples and applications

6. Congruences and modular arithmetics

Modern electronic communication makes use of coding theory and cryptography.

- coding – storing or transfer of data with possible loss or disruption of the data; we require to minimize the possible loss
- cryptography – storing or transfer of data, which has to stay hidden from or unreadable for third parties

Using results of Number Theory and Grout Theory.

Exampkles

- CD storage format, mp3
- digital phone calls
- bar codes, ISBN
- RSA cryptosystem

Now follows a brief introduction to Number Theory used in later sections. We will be mostly using integers on a limited set (8, 16, 32 bits . . .).

Motivation examples

First we show/recall counting "mod n" and we learn to answer following questions:

Example

A devise read a UPC bar code. Is 041331021641 a valid UPC bar code?

Example

We wrote an ISBN book number 0-03-001559-5. Is this a valid ISBN code?

Later we show *some* errors can be detected and some can be corrected.

Example

We know the fourth digit of the UPC bar code 041331021641 is wrong, what is the correct digit?

Example

We know the ISBN book code 0-03-001559-5 is wrong. We know we often swap adjacent digits while writing. Can you derive the correct ISBN code?

Divisibility

Let a, b be two integers. We say a divides b, if there exists an integer k, such that $a \cdot k = b$, we write $a \mid b$.
If not, we say a does not divide b, we write $a \nmid b$.
Integer a is the divisor of b and b is a multiple of a.

Example

It holds $3 \mid 6, \quad 3 \mid 15$.
Also it holds $2 \mid -6, \quad 5 \mid -5 \quad$ a $\quad 7 \mid 0$.
However $6 \nmid 3, \quad 2 \nmid 5, \quad 4 \nmid -6, \quad 0 \nmid 1$.
It holds that $0 \mid 0$, while $0 = k0$, for an arbitrary $k \in \mathbb{Z}$.
We can't divide by zero, but zero can be a divisor, however a divisor of zero only.

Operation vs. relation

Division is an operation $\mathbb{Z} \times (\mathbb{Z} \setminus \{0\}) \to \mathbb{Q}$.
The result of the division of an arbitrary number by a nonzero number is the resulting (third) number.

Example

It holds $18 : 3 = 6, \quad 0 : 7 = 0, \quad 18 : 4 = \frac{9}{2}$.

Divisibility is a relation $|\subset \mathbb{Z} \times \mathbb{Z}$.
Two number (in the given order) are or are not related – one is the divisor of the second or not.

Example

3 divides 18.
7 divides 0.
4 does not divide 18.

Properties of divisibility

Theorem

Let b, c be integers and let a be a nonzero integer. Platí

- If $a \mid b$ and $a \mid c$, then $a \mid (b + c)$.
- If $a \mid b$ then $a \mid bc$ for all integer c.
- If $a \mid b$ and $b \mid c$, then $a \mid c$.

Examples

Because $3 \mid 12$, then $3 \mid 12c$ for every integer c.
(notice, not only $12c$, but also $12c + 3$, $12c + 6$, and $12c + 9$)

Corollary

Let b, c be integers and let a be a nonzero integer. If $a \mid b$ and $a \mid c$, then $a \mid rb + sc$ for arbitrary integers r, s.

The Quotient Remainder Theorem

The Quotient Remainder Theorem

For very integer a and every natural number b there exist unique integers q and r, such that $a = qb + r$, where $0 \leq r < m$.

Integer q is the quotient and non negative integer r is the remainder when dividing a by b.

Example

For $a = 111$ and $b = 9$ holds:

$$\begin{aligned} 111 &= 11 \cdot 9 + 12 \\ 111 &= 12 \cdot 9 + 3 \quad \text{must hold } 0 \leq r < 9 \\ 111 &= 13 \cdot 9 - 6 \end{aligned}$$

Example

It holds $7 \mid 21$, therefore $21 = 3 \cdot 7 + 0$, remainder is $r = 0$.
Since $8 \nmid 21$, therefore $21 = 2 \cdot 8 + 5$, remainder is $r = 5 \neq 0$.

Integer division with a remainder

By the Quotient Remainder Theorem we can to each pair of integers a, b ($b > 0$) assign an integer quotient and a remainder after integer division.

Operation integer division with a remainder

Let a, b be two integers.
In the equality $a = q \cdot b + r$ given gy the Quotient Remainder Theorem is a the dividend, b is the divisor, q is the integer quotient and r is the remainder after integer division of a by b.
The following notation is used to denote the two integer operations of quotient and remainder.

$$q = a \operatorname{div} b, \quad r = a \bmod b$$

Example

For $a = 111$ and $b = 9$ from the previous example holds:

$$111 \operatorname{div} 9 = 12, \qquad 11 \bmod 9 = 3$$

Notice, "mod" written inside a parenthesis has a different meaning (later).

Congruences

Let a, b be integers, let m be a positive integer. We say a, b are congruent modulo m, if both yield the same remainder after dividing by m. We write

$$a \equiv b \pmod{m}.$$

Otherwise we write

$$a \not\equiv b \pmod{m}.$$

Using the "mod" operation introduced earlier we can write

$$a \equiv b \pmod{m} \quad \Leftrightarrow \quad a \bmod m = b \bmod m.$$

Example

It holds $7 \equiv 1 \pmod{2}$, since 7 is odd.
It holds $12 \equiv 0 \pmod{2}$, since 12 is even.
It holds $61\,725 \equiv 0 \pmod{3}$, since 61 725 is a multiple of 3.

Equivalent formulations for congruence of two numbers

Lemma

Let a, b be integers, let m be a positive integer. Then $a \equiv b \pmod{m}$ if an only if $m \mid (b - a)$.

Example

It holds $8\,298 \equiv 8\,228 \pmod{7}$, because the difference $8\,298 - 8\,228 = 70$ is a multiple of 7.

Lemma

Let a, b be integers, let m be a positive integer. Then $a \equiv b \pmod{m}$ if and only if there exists an integer k such that $b = a + km$.

Example

Evaluate $748\,549 \bmod 7$ (the remainder after dividing 748 549 by 7).

It holds $748\,549 = 700\,000 + 48\,549 \equiv 48\,549 = 49\,000 - 451 \equiv -451 = -490 + 39 \equiv 39 = 35 + 4 \equiv 4 \pmod{7}$.

Properties of congruences

Congruences with the same modulus can be summed and multiplied.

Theorem

Let a, b, c, d be integers, let m be a positive integer.
If $a \equiv b \pmod{m}$, $c \equiv d \pmod{m}$, then also $a + c \equiv b + d \pmod{m}$, $ac \equiv bd \pmod{m}$.

Example

Because $7 \equiv 12 \pmod{5}$ a $-7 \equiv 3 \pmod{5}$, then also
$7 - 7 \equiv 12 + 3 \pmod{5}$ a $7 \cdot (-7) \equiv 12 \cdot 3 \pmod{5}$.

Notice, the reverse implication does not hold!

Example

It holds $3 + 6 \equiv 7 + 5 \pmod{3}$, but $3 \not\equiv 7 \pmod{3}$ nor $6 \not\equiv 5 \pmod{3}$.
Similarly $10 \cdot 6 \equiv 4 \cdot 15 \pmod{5}$, but $10 \not\equiv 4 \pmod{5}$ nor $6 \not\equiv 15 \pmod{5}$.

Modular arithmetics

We can sum and multiply remainders, when dividing by the same divisor (or modulus). The result is expressed again as a remainder modulo m.

Definition

Let a, b be integers. We introduce operations "$+_m$" and "$\cdot_m$" using the usual sum and product and the "mod" operation.

$$a +_m b = (a + b) \bmod m, \qquad a \cdot_m b = (a \cdot b) \bmod m.$$

This can also be introduces as counting with congruence classes modulo m.

Example

The sum of two even integers or the sum of odd two integers is an even integer.

$$a +_m b = (a + b) \bmod 2.$$

Taking two representatives 0 and 1 of the congruence classes S and L

$$S +_m S = S, \quad L +_m L = S, \quad L +_m S = L, \quad S +_m L = L.$$

Modular arithmetics - continued

Such operations have "nice" properties. They are

- closed on the set $\{0, 1 \ldots, m-1\}$ (under operation modulo m),
- commutative, $a +_m b = b +_m a, \qquad a \cdot_m b = b \cdot_m a,$
- asociative,

$$a +_m (b +_m c) = (a +_m b) +_m c, \qquad a \cdot_m (b \cdot_m c) = (a \cdot_m b) \cdot_m c,$$

- and distributive with respect to addition,

$$a \cdot_m (b +_m c) = a \cdot_m b +_m a \cdot_m c,$$

- there exist opposite numbers $-a = m - a$.
- However inverses might not exist!

Greatest common divisor and least common multiple

A prime is such positive integer that has *two* positive divisors: one and itself.

Definition

Let a, b be two integers. Greatest common divisor of a, b is such a positive common divisor m of a, b, which is divisible by each other common divisor. We denote it $\mathrm{GCD}(a, b)$ or simply (a, b). Moreover, if $\mathrm{GCD}(a, b) = 1$, we say a, b are coprime.

Examples

Numbers 91 and 77 are not coprime, $\mathrm{GCD}(91, 77) = 7$.
Numbers 92 and 77 are coprime, $\mathrm{GCD}(92, 77) = 1$.

We can have a set of mutually coprime numbers.

Finding a prime factorization, or verifying that a certain integer is a prime, is a difficult task.

Euclid's algorithm

IS an efficient way to find the GCD of two positive integers a, b.

- easy to implement,
- no need for prime factorization.

Euklidův algoritmus

Let a, b be two positive integers. We divide a by b in modular arithmetic with a remainder (using the Quotient Remainder Theorem) and proceed repeating this process, until the remainder is zero.

$$\begin{aligned} a &= bq_1 + r_1 \\ b &= r_1 q_2 + r_2 \\ r_1 &= r_2 q_3 + r_3 \\ &\vdots \\ r_{n-2} &= r_{n-1} q_{n-1} + r_n \\ r_{n-1} &= r_n q_n + 0 \end{aligned}$$

The last non-zero remainder r_n is the greatest common divisor.

Example

Find the greatest common divisor of 414 ans 662.

We denote $a = 414$, $b = 662$. (It is better to denote $a = 662$, $b = 414$.)

Proceed by the Euclid's algorithm:

$$\begin{aligned} 414 &= 662 \cdot 0 + 414 \\ 662 &= 414 \cdot 1 + 248 \\ 414 &= 248 \cdot 1 + 166 \\ 248 &= 166 \cdot 1 + 82 \\ 166 &= 82 \cdot 2 + 2 \\ 82 &= 2 \cdot 41 + 0 \end{aligned}$$

The greatest common divisor $(414, 662) = 2$.

Euclid's algorithm – implementation

Input are two positive integers a, b. Repeatedly we divide in modular arithmetic with a remainder.

Euclid's algorithm

```
int a,b;              // positive integers
x = a;                // dividend
y = b;                // divisor
while (y<>0) {
   r = x mod y;       // evaluate the remainder r
   x = y;             // the divisor becomes the dividend
   y = r;             // the remainder becomes the divisor
}
return x;             // (a,b) last non-zero remainder
```

Variable `x` holds the last *non-zero* remainder, which is the greatest common divisor of a, b.

Further use of Euclid's algorithm

Euclid's algorithm work not only for integers but on any set with two (nice) operations.

- dividing polynomials
- for so called Gaussian integers

The following theorem states that the greatest common divisor of a,b can be expressed as a linear combination of a, b.

Bézout's Theorem

Let a, b be positive integers. There exists integers r, s such that $\mathrm{GCD}(a, b) = ra + sb$.

Bézout's Theorem provides a nice tool to solve certain problems expressed by congruences.

Euclid's algorithm can be easily extend to evaluate the coefficients r, s in the Bézout's equality.

Example

We have shown that the greatest common divisor of 414 and 662 is 2. Find the Bézout's coefficients r, s, so that $2 = r \cdot 414 + s \cdot 662$.

Using the Euclid's algorithm we got:

$$\begin{aligned} 662 &= 414 \cdot 1 + 248 \\ 414 &= 248 \cdot 1 + 166 \\ 248 &= 166 \cdot 1 + 82 \\ 166 &= 82 \cdot 2 + 2 \\ 82 &= 2 \cdot 41 + 0 \end{aligned}$$

Now from the next-to-the last equation express $\mathrm{GCD}(414, 662)$ and backward substitutions.

$$\begin{aligned} 2 &= 166 - 2 \cdot 82 \\ 2 &= 166 - 2 \cdot (248 - 166 \cdot 1) = (-2) \cdot 248 + 3 \cdot 166 \\ 2 &= (-2) \cdot 248 + 3 \cdot (414 - 248 \cdot 1) = 3 \cdot 414 + (-5) \cdot 248 \\ 2 &= 3 \cdot 414 + (-5) \cdot (662 - 414 \cdot 1) = 8 \cdot 414 + (-5) \cdot 662 \end{aligned}$$

The coefficients are $r = 8$, $s = -5$.

The following statements follow by the Bézout's Theorem

Theorem

Let a, b, c be positive integers. If $a \mid bc$ and $(a, b) = 1$, then $a \mid c$.

In congruences we can cancel by numbers coprime with modulus m.

Theorem

Let a, b, c be integers and m a positive integer. If $ac \equiv bc \pmod{m}$ and $(m, c) = 1$, then $a \equiv b \pmod{m}$.

In congruences we can cancel by common divisors of both sides and the modulus m.

Theorem

Let a, b, c be integers and m a positive integer. If $ac \equiv bc \pmod{cm}$, then $a \equiv b \pmod{m}$.

We use these theorems in the last part of this lecture.

Linear congruences

Definition

Let a, b be integers, let m be a positive integer, and let x be a variable. Congruence

$$ax \equiv b \pmod{m}$$

is a linear congruence in one variable.

To solve a congruence is to find *all* values of x, for which the congruence holds.

Example

Solution of the congruence $x \equiv 1 \pmod{2}$ are (precisely) all odd integers.
Solution of the congruence $x \equiv 4 \pmod{7}$ are integers $x = 7k + 4$, $k \in \mathbb{Z}$.
Solution of the congruence $3x \equiv 0 \pmod{7}$ are integers $x = 7k$, $k \in \mathbb{Z}$.
Solution of the congruence $3x \equiv 4 \pmod{7}$ are integers $x = 7k + 6$, $k \in \mathbb{Z}$.
Congruence $3x \equiv 1 \pmod{6}$ has no solution.

Now we show how to find the solutions, provided it exists.

Solving linear congruences

For solving congruences we use (similarly as for equations) so called inverses modulo m.

Definition

Let a be an integer and let m be a positive integer, where $m > 1$. The integer $\overline{a}$ is the inverse to a modulo m, if $\overline{a} \cdot a \equiv 1 \pmod{m}$.

Example

Number 3 is inverse to 5 modulo 7, because $3 \cdot 5 \equiv 1 \pmod{7}$.
Number 3 is inverse to itself modulo 8, because $3 \cdot 3 \equiv 1 \pmod{8}$.
Number 3 is inverse to 7 modulo 10, because $3 \cdot 7 \equiv 1 \pmod{10}$.
Number 8 has no inverse modulo 10, since $8 \cdot x$ is even, $8 \cdot x \not\equiv 1 \pmod{10}$.

The following theorem shows, when inverses modulo m exist.

Theorem

Let a be an integer, let m be a positive integer, $m > 1$. If a, m are coprime, then there exists the inverse $\overline{a}$ of a modulo m and is unique modulo m.

The proof is constructive, it provides a way to find the inverse $\overline{a}$ to a.

Theorem

Let a be an integer, let m be a positive integer, where $m > 1$. If a, m are coprime, then there exists the inverse $\overline{a}$ of a modulo m and is unique modulo m.

Proof: Since a, m are coprime, then by Bézout's Theorem exist integers r, s, such that

$$r \cdot a + s \cdot m = 1.$$

This implies

$$\begin{aligned} r \cdot a + s \cdot m &\equiv 1 \pmod{m} \\ r \cdot a &\equiv 1 \pmod{m} \end{aligned}$$

Hence, r is the inverse $\overline{a}$ modulo m, thus $\overline{a} = r$.
We wont prove uniqueness here. □

A stronger claim holds also: if $\mathrm{GCD}(a, m) > 1$, then no inverse to a modulo m exists.

Example

Because $(3, 7) = 1$, we can write $1 = 5 \cdot 3 - 2 \cdot 7 \equiv 3 \cdot 5 \pmod 7$.
Number 5 is inverse to 3 modulo 7, thus $\overline{3} = 5$.

Notice, if a is not coprime to modulus m, no inverse can exist!

Example

Take 14, 6. Because $(14, 6) = 2$, by Bézout's Theorem follow
$2 = 1 \cdot 14 - 2 \cdot 6$ and no such smaller integer exist.
Therefore, for no number $\overline{a}$ can hold $14\overline{a} \equiv 1 \pmod 6$.

Solving linear congruences

Now we can solve linear congruences in one variable analogously to solving linear equations.

Example

Find all solutions of the linear congruence $3x \equiv 4 \pmod 7$.

First we find the inverse to 3 modulo 7. By previous example $\overline{3} = 5$.
We multiply both sides of the congruence by the inverse 5. We get

$$\begin{aligned} 5 \cdot 3x &\equiv 5 \cdot 4 \pmod 7 \\ x &\equiv 20 \pmod 7 \\ x &\equiv 6 \pmod 7 \end{aligned}$$

The solution are all integers, that have remainder 6 when dividing by 7.
The solution is $x = 7k + 6$, where $k \in \mathbb{Z}$.

The most toilsome part is to find the inverse modulo m.

Manipulation and simplification of congruences

Let a, b, c, d be integers and let m be a positive integer.
Let $a \equiv b \pmod{m}$, $c \equiv d \pmod{m}$ be congruences.

- We can add congruences with the same modulus.

$$a + c \equiv b + d \pmod{m}$$

- We can multiply congruences with the same modulus.

$$ac \equiv bd \pmod{m}$$

- We can multiply both sides of a congruence by the same integer c.

$$ac \equiv bc \pmod{m}$$

- We can cancel in congruences by c coprime with the modulus, thus for $(c, m) = 1$ is

$$ac \equiv bc \pmod{m} \quad \Rightarrow \quad a \equiv b \pmod{m}.$$

- We can cancel in congruences by $c = \mathrm{GCD}(a, b, m)$

$$ac \equiv bc \pmod{mc} \quad \Rightarrow \quad a \equiv b \pmod{m}.$$

These manipulations allow to simplify and solve linear congruences.

Example

Find the solution of the congruence $5x \equiv 2 \pmod{13}$ and verify it.

Using $\overline{5} = 8$ we get $x \equiv 16 \pmod{13}$, therefore $x = 13t + 3$, $t \in \mathbb{Z}$.

Another solution: we add modulus 13 to the right side $5x \equiv 2 + 13 \pmod{13}$. Cancelation by 5 yileds $x \equiv 3 \pmod{13}$, thus $x = 13t + 3$, $t \in \mathbb{Z}$.

Verification? Substitute $5(13t + 3) \equiv 5 \cdot 13t + 15 \equiv 0t + 2 \pmod{13}$.

Example

Find the solution of the congruence $3x \equiv 2 \pmod{15}$.

Congruence has no solution, because $(3, 15) = 3$, and can't cancel by 3.

Example

Find the solution of the congruence $3x \equiv 6 \pmod{15}$.

We cancel both sides and modulus by 3. We get $x \equiv 2 \pmod{5}$.
This conguence tha the solution $x = 5t + 2$, $t \in \mathbb{Z}$, because $(3, 15) = 3$.
We canceled both sides and the modulus by 3.

Chinese Remainder Theorem

The following theorem states, that there is a unique solution to a system of congruences with pairwise coprime moduli.

Chinese Remainder Theorem

Let $m_1, m_2, \ldots, m_n$ be coprime positive integers greater than one. Let $a_1, a_2, \ldots, a_n$ be integers. The system of congruences

$$\begin{aligned} x &\equiv a_1 \pmod{m_1} \\ x &\equiv a_2 \pmod{m_2} \\ &\vdots \\ x &\equiv a_n \pmod{m_n} \end{aligned}$$

Has a unique solution modulo $m_1 \cdot m_2 \cdots m_n$.

Due problems solved in ancient manuscripts is the theorem called "Chinese Remainder Theorem".

The proof of the theorem is constructive, however the solution can be found using manipulations backward substitution of congruences.

Example

Find the solution of the system of congruences

$$\begin{aligned} x &\equiv 1 \pmod 5 \\ x &\equiv 2 \pmod 6 \\ x &\equiv 3 \pmod 7 \end{aligned}$$

Based on the theorems above we get the solution of the first congruence

$$x \equiv 1 \pmod 5$$

to be $x = 1 + 5t$, where $t \in \mathbb{Z}$.
This solution we substitute the the second congruence

$$\begin{aligned} 1 + 5t &\equiv 2 \pmod 6 \\ 5t &\equiv 1 \pmod 6 \\ -t &\equiv 1 \pmod 6 \\ t &\equiv -1 = 5 \pmod 6 \end{aligned}$$

The solution of the congruence

$$t \equiv 5 \pmod 6$$

is $t = 6u + 5$, where $u \in \mathbb{Z}$.
Substituting do $x = 1 + 5t$ we get the solution of the first two congruences

$$x = 1 + 5(6u + 5) = 30u + 26, u \in \mathbb{Z},$$

which can be substituted to the third congruence. We get

$$\begin{aligned} 30u + 26 &\equiv 3 \pmod 7 \\ 2u - 2 &\equiv 3 \pmod 7 \\ 4 \cdot 2u &\equiv 4 \cdot 5 \pmod 7 \\ u &\equiv 6 \pmod 7. \end{aligned}$$

The solution is $u = 7v + 6$, where $v \in \mathbb{Z}$, which we substitute to the solution of the first two congruences.

$$x = 30u + 26 = 30(7v + 6) + 26 = 210v + 206, v \in \mathbb{Z}.$$

We get the solution of the system of three congruences.

Application of congruences – hash functions

When storing a large database we can add a new entry x to the end of the database. This is cumbersome or searching the database – we have to search the whole database.
Of there are m entries, we need $O(m)$ steps.

Hash function: We estimate the expected database size m and allocate the corresponding memory. New entry with key k we enter to position $h(k)$, where

$$h(k) = k \bmod m,$$

or the next appropriate free place after $h(k)$.
Instead of searching whole database, we start searching at position $h(k)$.

Example

Students at a university with 15 000 students, key is the social security number.

Possible hash function $h(k) = \mathit{security\_number} \bmod 15\ 000$.

Application of congruences – Pseudorandom numbers

A truly random number is computationally "expensive".
However, pseudorandom numbers we evaluate quickly

$$x_{n+1} = (ax_n + c) \bmod m,$$

where a, c, m are carefully selected integers. The value x_0 is the "seed".

Example

For example for $a = 7$, $b = 4$, $m = 9$, and $x_0 = 1$ we get

$$\begin{aligned} x_1 &= (7x_0 + 4) \bmod 9 = 11 \bmod 9 = 2 \\ x_2 &= (7x_1 + 4) \bmod 9 = 18 \bmod 9 = 0 \\ x_3 &= (7x_0 + 4) \bmod 9 = 4 \bmod 9 = 4 \\ x_4 &= (7x_0 + 4) \bmod 9 = 32 \bmod 9 = 5 \\ &\vdots \end{aligned}$$

This gives the sequence $1, 2, 0, 4, 5, 3, 7, 8, 6, 1, 2, 0, 4, \ldots$

A commonly used random generator: $a = 7^5$, $b = 0$, $m = 2^{31} - 1$.

Application of congruences – check sums

Parity sums

Let $x_1, x_2, \ldots, x_n$ be an n-bit word.
The sender adds another bit (bits), a parity check digit

$$x_{n+1} = x_1 + x_2 + \cdots + x_n \bmod 2.$$

If during the transfer *one* (or an odd number) errors occur, the recipient evaluates

$$x_1 + x_2 + + \cdots + x_n + x_{n+1} \not\equiv 0 \pmod 2$$

and can require ask for the message to be sent again.

Using several parity check digits, we can CORRECT certain errors without sending it again.
(example provided later)

Application of congruences – UPC bar codes

UPC bar code (Universal Product Code)

There are many variations on UPC bar codes, most of them make use of check sums.
UPC-A bar code has 12 digits. It satisfies

$$3x_1+x_2+3x_3+x_4+3x_5+x_6+3x_7+x_8+3x_9+x_{10}+3x_{11}+x_{12} \equiv 0 \pmod{10}$$

Example

Is 041331021641 a valid UPC code?

Since $0+4+3+3+9+1+0+2+3+6+12+1 = 44 \equiv 4 \pmod{10}$, therefore the code is invalid.

Application of solving linear congruences

Reconstruction of UPC bar codes

We know the UPC code 041331021641 is not valid. However it seems the fourth digit is damaged. What is the correct UPC code?

We know the digits of the UPC code 041?31021641 must satisfy

$$3x_1+x_2+3x_3+x_4+3x_5+x_6+3x_7+x_8+3x_9+x_{10}+3x_{11}+x_{12} \equiv 0 \pmod{10}$$

We set up a linear congruence.

$$\begin{aligned} 0+4+3+x_4+9+1+0+2+3+6+12+1 &\equiv 0 \pmod{10} \\ x_4+41 &\equiv 0 \pmod{10} \\ x &\equiv -1 \pmod{10} \\ x &\equiv 9 \pmod{10} \end{aligned}$$

The missing digit of the UPC bar code is 9.

It is easy to verify

$$3x_1 + x_2 + 3x_3 + x_4 + 3x_5 + x_6 + 3x_7 + x_8 + 3x_9 + x_{10} + 3x_{11} + x_{12} =$$
$$= 0+4+3+9+9+1+0+2+3+6+12+1 = 50 \equiv 0 \pmod{10}.$$

Further application of congruences

- ISBN/ISSN
- social security numbers (rodná čísla)
- banknote numbers
- bank account numbers
- simple ciphers

Application of solving linear congruences

Reconstruction of the social security number (rodné číslo)

An old lady forgot her social security number. She remembers her birthday and the last three digits. Thus, we can reconstruct the following parts of the number: 346509?248. What is the missing digit?

We know that the digits have to satisfy

$$x_1x_2 + x_3x_4 + x_5x_6 + x_7x_8 + x_9x_{10} \equiv 0 \pmod{11}.$$

We set up a linear congruence

$$\begin{aligned} 34 + 65 + 9 + (10x + 2) + 48 &\equiv 0 \pmod{11} \\ 1 - 1 + 0 + 10x + 4 &\equiv 0 \pmod{11} \\ 10x &\equiv -4 \pmod{11} \\ -x &\equiv -4 \pmod{11} \\ x &\equiv 4 \pmod{11} \end{aligned}$$

The missing digit is 4.

Next lecture

Algorithms for discrete structures

- types discrete structures
- implementation of sets
- generating selection and arrangements
- generating random numbers
- combinatorial explosion

Discrete mathematics

Petr Kovář
petr.kovar@vsb.cz

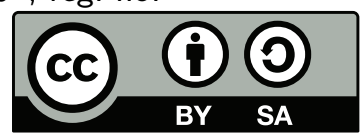
VŠB – Technical University of Ostrava

Winter term 2021/2022
DiM 470-2301/02, 470-2301/04, 470-2301/06

EUROPEAN UNION
European Structural and Investment Funds
Operational Programme Research,
Development and Education

MINISTRY OF EDUCATION,
YOUTH AND SPORTS

The translation was co-financed by the European Union and the Ministry of Education, Youth and Sports from the Operational Programme Research, Development and Education, project "Technology for the Future 2.0", reg. no. CZ.02.2.69/0.0/0.0/18_058/0010212.



About this file

This file is meant to be a guideline for the lecturer. Many important pieces of information are not in this file, they are to be delivered in the lecture: said, shown or drawn on board. The file is made available with the hope students will easier catch up with lectures they missed.

For study the following resources are better suitable:

- Meyer: Lecture notes and readings for an http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-042j-mathematics-for-computer-science-fall-2005/readings/" (weeks 1-5, 8-10, 12-13), MIT, 2005.
- Diestel: Graph theory http://diestel-graph-theory.com/ (chapters 1-6), Springer, 2010.

See also http://homel.vsb.cz/~kov16/predmety_dm.php

Lecture overview

Algorithms for discrete structures

- implementing basic structures
- implementing sets
- list of all selections or arrangements
- generating random numbers
- combinatorial explosion

7. Algorithms for discrete structures

In this chapter we describe how to implement some structures and algorithms introduced in Part I.

Some structures are easy to implement, some require a rather elaborate approach. They often differ in memory requirements or CPU time requirements.

Usually a general approach is more time/space consuming, on the other hand the generality must not necessarily be payed for by significantly slower algorithm.

This chapter is dedicated to selected implementations of structures and algorithms.

7.1. Implementing basic structures

- sequences
- mappings
- relations
- permutations

A (finite) sequence $(a_0, a_1, \ldots, a_{n-1})$
we implement as a one-dimensional field `a[ ]`, where `a[i]` = a_i.

Example

We have a (finite) sequence $(7, 5, 5, 7, 5, 6, 6)$.
We store the sequence in an array `p = [ 7 5 5 7 5 6 6 ]`.

Mappings

Mapping $f : A \to B$

Let us take a finite $A = \{a_0, a_1, \ldots, a_{n-1}\}$ and $B = \{b_0, b_1, \ldots, b_{m-1}\}$. Wework with subscripts only and implement the mapping as a sequence – field `f[ ]`, in which `f[i]=j` stands for $f(a_i) = b_j$.

This is particularly suitable when A and B are integer sets with small integers. For different sets we have to "translate" elements in A, B into their indices (usually CPU time consuming).

- for elements in A we can use hash tables
- for elements in B we use structured data types or pointers

Example

We have a mapping $f : [0, 5] \to [0, 5]$, where $f(0) = 4$, $f(1) = 5$, $f(2) = 3$, $f(3) = 3$, $f(4) = 2$, $f(5) = 2$.
We store the mapping in a field `f = [ 4 5 3 3 2 2 ]`.

Example

Take a mapping $f : \{A, B, C, D, E\} \to \{x, y, z, w\}$,
where $f(A) = w$, $f(B) = z$, $f(C) = w$, $f(D) = x$, $f(E) = w$.
Mapping is stored in a field `f = [ 3 2 3 0 3 ]`.
Potřebujeme pomocná pole `X = [ A B C D E ]`, `Y = [ x y z w ]`.

Example

Take a mapping $f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, where $f(x, y) = x^2 + 3y$.
Cannot be stored in a field! Not possible to store $\mathbb{R}$.

Example

Take a mapping $f : [-5 : 5] \times [-5 : 5] \to \mathbb{R}$, where $f(x, y) = x^2 + 3y$.
Mapping is stored in a two-dimensional field with 11×11 (approximate?) values.

Example

Take a mapping $f : [-5 : 5] \times [-5 : 5] \to \mathbb{R}$, where $f(x, y) = \sqrt{x + y}$.
Mapping is stored in a two-dimensional field with 11×11 (approximate!) values.

Binary relations

Binary relation R on the set A

For finite and small $A = \{a_0, a_1, \ldots, a_{n-1}\}$ we implement relation by a two-dimensional field (matrix) `r[ ][ ]`, in which
`r[i][j] = 0` when $(a_i, a_j) \notin R$ and
`r[i][j] = 1` when $(a_i, a_j) \in R$.

Example

We have a relation $R \subseteq [0,4]^2$, where $R = \{(0,0),(0,4),(1,3)(2,4)\}$.
Relation R can be stored in a two-dimensional field

$$\mathtt{R} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Properties of relations

Take a binary relation R on the set $\{0, 1, \ldots, n-1\}$ given by the field `r[][]`.

Check if the relation `r` is reflexive, $O(n)$

```
for (i=0; i<n; i++)
   if (!r[i][i]) {    // all ones?
      printf("Not reflexive!");
      return -1;
   }
```

Check if the relation `r` is symmetric, $O(n^2)$

```
for (i=0; i<n; i++)
   for (j=i+1; j<n; j++)
      if (r[i][j]!=r[j][i]) {     // a symmetric matrix?
         printf("Not symmetric!");
         return -1;
      }
```

Properties of relations (continued)

Is the relation `r` transitive? Verify for each tripple

$$\forall i, j, k : \; r[i][j] \; \wedge \; r[j][k] \; \Rightarrow \; r[i][k].$$

Check if the relation `r` is transitive, $O(n^3)$

```
for (i=0; i<n; i++)
   for (j=0; j<n; j++) {
      if (!r[i][j]) continue;          // skip
      for (k=0; k<n; k++) {
         if (!r[j][k]) continue;       // skip
         if (r[i][k]) continue;        // has to be!
         printf("Not transitive!");
         return -1;
      }
   }
```

Example

We have a relation $R \subseteq [0,4]^2$, where $R = \{(0,0),(0,4),(1,3)(2,4)\}$ stored in a two-dimensional field

$$\mathtt{R} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Relation is not reflexive.
Relation is not symmetric.
Relation IS transitive.

Question

How to test antisymmetry?
How to test linearity?
What is the time-complexity of these test?

Permutations

A permutation we implement as a bijective mapping $p : [0, n-1] \to [0, n-1]$.

Example

We have a permutation $\pi = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 4 & 2 & 1 & 3 & 0 & 5 \end{pmatrix}$, π can be stored in a field

$$\mathtt{p} = [4\ 2\ 1\ 3\ 0\ 5]$$

How to verify, the one dimensional field describes a permutation? It is enough to verify if p is onto (surjective).

Check whether p[ ] is a permutation, $O(n)$

```
for (i=0; i<n; i++)  u[i] = 0;          // an auxiliary field
for (i=0; i<n; i++)  if (p[i]>=0 && p[i]<n)  u[p[i]] = 1
   else printf("Not a permutation!");     // out of range
for (i=0; i<n; i++)
   if (u[i]!=1)
      printf("Not a permutation!");
```

Permutations (continued)

Composition of `p[ ]` and `q[ ]` is the permutation `r[ ]`, $O(n)$

```
for (i=0; i<n; i++)
     r[i] = q[p[i]];
```

We can obtain a list of all cycles by the code:

Cycle notation of an `n`-element permutation `p[ ]` of $[0, n-1]$, $O(n)$

```
for (i=0; i<n; i++)  u[i] = 0;          // an auxilliary field
for (i=0; i<n; i++) if (u[i]==0) {      // not used
   printf("\n(%d",i);  u[i] = 1;        // start cycle
   for (j=p[i]; j!=i; j=p[j])  {        // next in this cycle
      printf(",%d",j);  u[j] = 1;
   }
   printf(")");                         // close cycle
}
```

7.2. Set implementation

Set are not easy to implement. The problems include

- search for a particular element (not a specified list index),
- guarantee non-repetitive elements.

Characteristic function of a subset

The *universe* $\mathcal{U} = \{u_0, u_1, \ldots, u_{n-1}\}$, from which elements are taken, has to be known. Subsets $X \subseteq \mathcal{U}$ are implemented as fields `x[ ]`, where

$$\texttt{x[i]} = \begin{cases} 1 & \text{for } u_i \in X \\ 0 & \text{otherwise.} \end{cases}$$

Advantages: easy to search for a particular element, unions by using the *OR* function, intersection by *AND* function.

Disadvantage: suitable only for small universe $\mathcal{U}$!

Example

The set $A = \{2, 3, 5\}$ in a universe $U = [1, 10]$ is implemented using a characteristic function in a field A = [0 1 1 0 1 0 0 0 0 0].

Example

The set $A = \{2, 3, 5\}$ in a universe $U = [1, 100]$ is implemented using a characteristic function in a field A = [0 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0].
The set $B = \{1\}$, B = [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0].

To find the union $A \cup B$ on U use binary OR. To find the intersection $A \cap B$ use binary AND.

Example

The set $A = \{2, 3, 5\}$ in the universe $U = \mathbb{N}$ cannot be implemented.

List of elements

The set X is implemented as a list of elements. The list of k elements in X is stored in a field `x[ ]`, we can write

$$X = \{\mathtt{x}[1], \mathtt{x}[2], \ldots, \mathtt{x}[\mathtt{k}]\} \text{ for the field } \mathtt{x}[\ ] \text{ of length } k.$$

Advantage: work for big or even unspecified universe.

Instead of a field one can use a *dynamic linked list*, then it is easy to add or remove elements from the list.

Disadvantage: determining if a particular object is in the set (often used operation) is costly – one has to go through the entire list.

Example

The set $A = \{2, 3, 5\}$ in the universe $U = [-MAX\_INT, MAX\_INT]$ is implemeted as a field A = [2, 3, 5].

Example

The set $A = \{3, 5, 2\}$ in the universe $U = [-MAX\_INT, MAX\_INT]$ is implemeted as a field A = [3, 5, 2].

Test if element x is in the set a[ ] of size n, $O(n)$

```
for (i=0; i<n; i++) {              // traverse all field a[ ]
   if (a[i]==x) break;             // is x in a[ ]?
}
if (i<n) printf("Element x is in field a[ ]");   // found?
```

Union of two set in fields a[ ],b[ ] into the field c[ ], $O(n^2)$

```
for (i=0; i<m; i++)
   c[i] = a[i];                    // all m elements from a[ ]
for (i=0,k=m; i<n; i++) {          // next n elements from b[ ]
   for (j=0; j<m; j++)
      if (b[i]==a[j]) break;       // if b[i] in a[ ]
   if (j<m) continue;              // skip
   c[k++] = b[i];                  // or add it to c[ ]
}
```

Ordered list of elements

An easy modification of the previous implementation.
The elements in the list are ordered according some rule (length, size, lexicographic, etc.)

Advantage: on can use *binary search* in the set of elements by bisection in the list (see Example).

Example

The set $A = \{2, 3, 5\}$ in the universe $U = [-MAX\_INT, MAX\_INT]$ is implemented as a field A = [2, 3, 5].

Example

The set $A = \{3, 5, 2\}$ in the universe $U = [-MAX\_INT, MAX\_INT]$ is implemented as a field A = [2, 3, 5].

Binary search for k in an ordered field `p[ ]` of length `n`

```
int a = 0; b = n-1;
while (a<b && p[a]!=k) {          // k found?
   c = (a+b)/2;
   if (p[c]<k)  a = c+1;          // no, it will be bigger
   else         b = c;            // no, it will be smaller
}
if (p[a]!=k)  printf("The number k not in the list.");
```

Just $\lceil \log_2 n \rceil$ searching steps.

Adding a new element `x` to the set in a field `a[ ]` requires $O(n)$ operations:

- find the proper place, $O(\lceil \log_2 n \rceil)$
- copy or "shift" part of the field, $O(n)$

Similarly, when removing elements.

Union of two ordered list in fields a[ ], b[ ] of size m, n into an ordered field c[ ] of size l, $O(n+m)$

```
int i=0, j=0, k, l=0;
for (k=0; k < m+n; k++) {
   if (i >= m) {                    // if a[ ] exhausted
      c[l++] = b[j++];
      continue;
   }
   if (j >= n) {                    // if b[ ] exhausted
      c[l++] = a[i++];
      continue;
   }
   if (a[i] == b[j]) {              // just one copy
      j++;
      continue;
   }
   c[l++] = (a[i] < b[j]) ? a[i++] : b[j++];
}
```

Summary

- How large is the universe?
- Will we (and how often) modify the structure of the set?
- Will we (and how often) search the set?
- Will we (and how often) construct unions of sets?

... pick the approriate model.

7.3. Listing selection and arrangements

Often we have to traverse all selections or arrangements of a given type:

- different mappings,
- k-permutations,
- k-combinations without repetitions.

Simple traversing of all ordered pairs (triples, etc.)

All ordered pairs of indices `i,j` we traverse in a nested loop

2-permuations with repetition, $O(n^2)$

```
for (i=0; i<n; i++)                    // nested loop
   for (j=0; j<n; j++) {
      // process a particular ordered pair (i,j)
   }
```

All unordered pairs of indices `i,j` are traversed similarly

2-combinations, $O(n^2)$

```
for (i=0; i<n; i++)                 // just "above the diagonal"
   for (j=i+1; j<n; j++) {
      // process a particular unordered pair {i,j}
   }
```

Processing all permutations of an n-element set A in a[ ]

All $n!$ permutations are processed by a recursive algorithm (Heap 1963).

Heap's algorithm – permutations of n elements

```
int i, a[ ];
permutation(n, a[ ]) {
   if (n==1)
      // process permutation in a[ ]
   else
      for (i=0; i < n-1; i++) {
         permutation(n-1, a[ ]);
         if (n even)
            swap(a[i], a[n-1]);
         else
            swap(a[0], a[n-1]);
      }
      permutation(n-1, a[ ]);
}
```

Function `swap(x,y)` simply swaps the content of `x` and `y`.

Processing all mappings

All n^k mapping of a k-element set into and n-element set

$$\texttt{map} : \{0, 1, \ldots, k-1\} \rightarrow \{0, 1, \ldots, n-1\}$$

we traverse by the following code: k nested cycles not necessary!

k-permutations with repetition of n elements, $O(n^k)$

```
int i, map[k];
map[i = 0] = -1;
while (i>=0) {
   if (++map[i]>=n)              // increase by 1
     { i--; continue; }
   if (++i<k)                    // 'erase' next element
     { map[i] = -1; continue; }
   // process the mapping (map[0], ..., map[k-1])
   i--;
}
```

For each choice we verify

- if it exceeded n, then we return to the previous level,
- if this was the last choice for the k-th element, otherwise next level.

Processing all k-permutations (without repetitions) on n elements

k-permutations without repetition of n elements, $O(n^k)$

```
int i, j, arrange[k];
arrange[i = 0] = -1;
while (i>=0) {
   if (++arrange[i]>=n)        // increase by 1
      { i--; continue; }
   for (j=0; j<i; j++)         // does it repeat?
      if (arrange[i]==arrange[j])  break;
     if (j<i)  continue;       // skip repeated
   if (++i<k)                  // 'erase' next elements
      { arrange[i] = -1; continue; }
   // process k-permutation (arrange[0],...,arrange[k-1])
   i--;
}
```

For each choice we verify

- if it exceeded n, then we return to the previous level,
- if this is not a repeated element, then we skip it,
- if it was the last k, otherwise proceed by the next level.

Processing all k-combiations

We traverse all k-combinations (without repetition) on n elements.
It is similar to the previous case, but now we produce
ordered k-tuples. Hence every k-combination is obtained just once.

k-combinations (without repetition) of n elements, $O(n^k)$

```
int i, select[k];
select[i = 0] = -1;
while (i>=0) {
   if (++select[i]>=n)                  // increase by 1
      { i--; continue; }
   if (++i<k) {
      select[i] = select[i-1];          // sorted already!
      continue;
   }
   // process the k-combinaton (select[0],...,select[k-1])
   i--;
}
```

We do not have to check for repeated selections, since the elements of the selection are ordered.

7.4. Generating random numbers

We investigate really random sequence of bits in a computer.

Where do we require random numbers/bit sequences?

- generating random (large) private keys (for SSL certificates).
  Using random passwords for SSL encryption (if not random, it can be broken!).
- Resolving packet collisions on Ethernet by random pauses before next transmission.
- Used by *probability algorithms*, random bits can boost computation performance.
- In statistical analysis of real events, modelling real chaotic and physical experiments, etc.

Various random number generators

Elemental pseudorandom generators

Use various formulas as

$$x := (A \cdot x + B) \mod C.$$

We iterate this and certain bits x are used as the random sequence.

Disadvantage: *depends heavily* on previous iterations and *predictable*.

Pseudorandom generators with external input

Similar formulas as in the previous case with additional input from *external physical processes* (key press delays, disk reading delays, network statistics, etc.)

Problems: dependence on external conditions, *can be influenced* by the environment, each bit is "costly".

Hardware random generators

Based on *quantum noise* (in semiconductors).

Problems: translation into a uniform bit sequence, confidence in quantum mechanics.

7.5. Combinatorial explosion

In software based solution of problems in discrete mathematics we often require algorithms such as:

Traverse all cases.

Then we may encounter the phenomenon called *exponential combinatorial explosion*.

- fast growth of the factorial.
- tale about corn on chessboard fields

If the number of traversed cases grows exponentially, them even for an input increased by 1 the computational time increases many times.
In many situation in input of size 10 can be solved in seconds on a 386 processor, but the input of size 15 cannot be counted on most powerful machines in the world.

Remember this phenomenon when designing your algorithm with "brute force"!

By choosing an appropriate algorithm, data structure or input limitation we can achieve tremendous increase of performance.

Example

How many (non-isomorphic!) tournaments of n teams (disregards the round order, disregard the team numbers).

n=2 1 tournament

n=4 1 tournament

n=6 1 tournament

n=8 6 tournaments

n=10 396 tournaments

n=12 526 915 620 tournaments

n=14 1 132 835 421 602 062 347 tournaments

n=16 ?

If we distinguish the team numbers, then for $n = 14$ is 98 758 655 816 833 727 741 338 583 040 tournaments.

Next lecture

Part II Introduction to Graph Theory

Chapter 1. The graph

- motivation
- definition of a graph
- degree of a vertex

Discrete mathematics

Petr Kovář
petr.kovar@vsb.cz

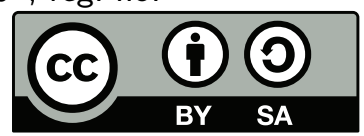
VŠB – Technical University of Ostrava

Winter term 2021/2022
DiM 470-2301/02, 470-2301/04, 470-2301/06

EUROPEAN UNION
European Structural and Investment Funds
Operational Programme Research,
Development and Education

MINISTRY OF EDUCATION,
YOUTH AND SPORTS

The translation was co-financed by the European Union and the Ministry of Education, Youth and Sports from the Operational Programme Research, Development and Education, project "Technology for the Future 2.0", reg. no. CZ.02.2.69/0.0/0.0/18_058/0010212.



About this file

This file is meant to be a guideline for the lecturer. Many important pieces of information are not in this file, they are to be delivered in the lecture: said, shown or drawn on board. The file is made available with the hope students will easier catch up with lectures they missed.

For study the following resources are better suitable:

- Meyer: Lecture notes and readings for an http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-042j-mathematics-for-computer-science-fall-2005/readings/" (weeks 1-5, 8-10, 12-13), MIT, 2005.
- Diestel: Graph theory http://diestel-graph-theory.com/ (chapters 1-6), Springer, 2010.

See also http://homel.vsb.cz/~kov16/predmety_dm.php

Part II Introduction to Graph Theory

Chapter 1. The graph

- motivation
- definition of a graph
- oriented graphs and multigraphs
- degree of a vertex
- subgraphs and isomorphisms
- implementation of graphs

Introduction to Graph Theory

Graph Theory originated rather recently

- L. Euler: Seven Bridges of Königsberg in 1736
- First monograph in 1936

Well known problem solved by Graph Theory:

- four color theorem
- shortest path in a graph
- maximum flow in a network

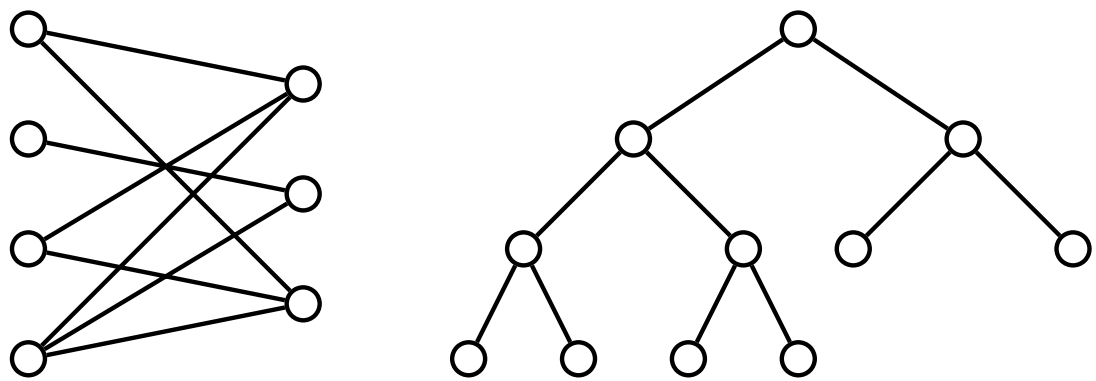
Motivation

Graph Theory is a very important discipline of Discrete Mathematics

- by graphs real life situations can be described easily
- intuitive interpretation
- easily implemented in computers

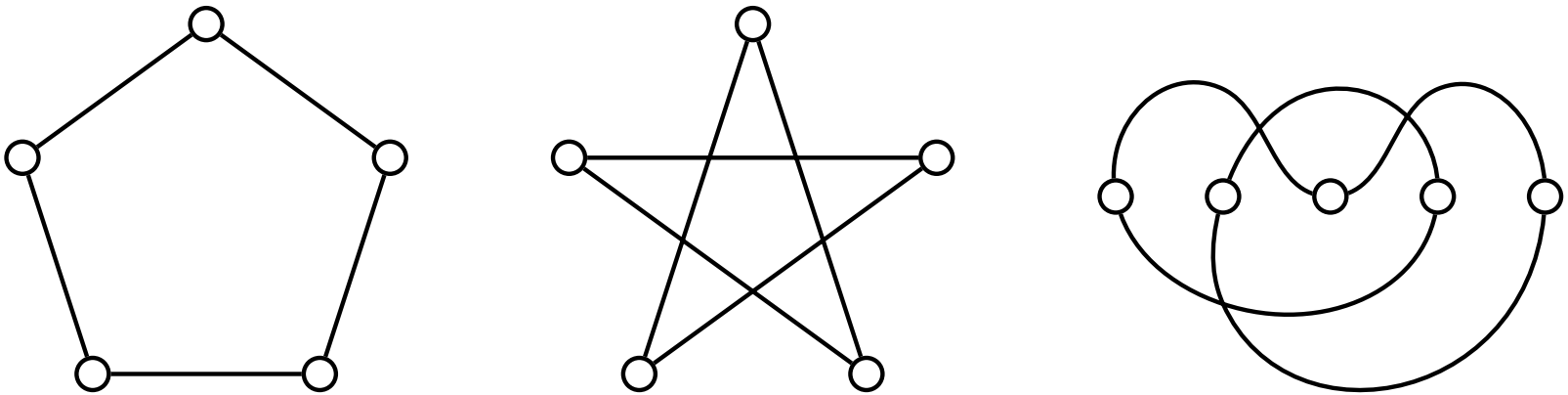
Informally: a graph contains

- vertices (nodes) – "dots" in the figure
- edges – "lines" joining two points in the drawing



Examples of graphs.

Figures of the same graph may be considerably different.



Different drawings of the same graph.

It may not be apparent if two different drawings represent the same graph (the same structure).

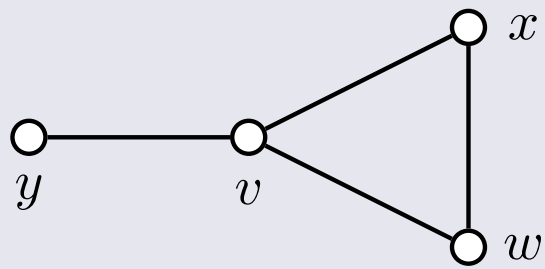
Definition of a graph

Definition

Graph G (*simple* graph) is an ordered pair $G = (V, E)$, where V is a nonempty set of *vertices* and E is the set of *edges* – the set of (some) two-element subsets of V.

Example

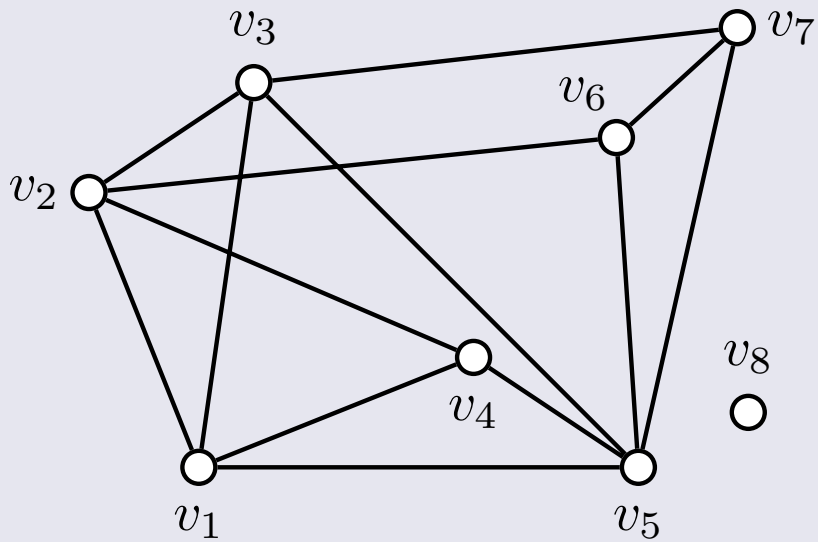
The graph $G = (V, E)$, where $V = \{v, w, x, y\}$ and $E = \big\{\{v, w\}, \{v, x\}, \{v, y\}, \{w, x\}\big\}$ we draw as follows:



The elements of V are called vertices, they are usually denoted by lower case letters u, v, ...
The elements of E are called edges. The edge between vertices u and v is the two-element subset $\{u, v\}$ of V, it is denoted by uv for short.

Example

Graph $G = (V, E)$, where $V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$ and $E = \{v_1v_2, v_1v_3, v_1v_4, v_1v_5, v_2v_3, v_2v_4, v_2v_6, v_3v_5, v_3v_7, v_4v_5, v_5v_6, v_5v_7, v_6v_7\}$.



If we are given a graph G, by $V(G)$ we understand the set of vertices of the graph G and by $E(G)$ the set of edges of G.

Note

Graph $G = (V, E)$ can be viewed as a special relation E on the set V, where E is *irreflexive* and *symmetric*.

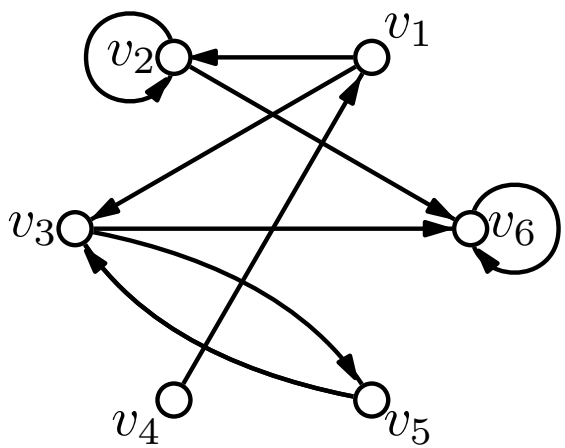
Oriented graphs and multigraphs

If a graph represents a transportation or road network, it is often necessary to impose orientation to edges, called arcs. We distinguish then the first (tail) and the end (head) vertex. The arc uv is not identical to the arc vu.

Definition

An oriented graph is the ordered pair $G = (V, E)$, where V is the set of *vertices* and a nonempty set of *arcs* is $E \subseteq V \times V$.

In a drawing we depict oriented edges by arrows.



An oriented graph.

Note

A simple oriented graph $G = (V, E)$ (without loops) can be considered as an irreflexive relation on a given set V.

Note

An oriented graph $G = (V, E)$ with loops can be considered as a (general) relation on a given set V.

Note

A multigraph is even more general than an oriented graph. Multiple edges, arcs and loops are allowed.

We focus on oriented graphs in the last chapter...

Common graph classes

Graph can be defined by giving

- sets V and E
- drawing
- name and parameter (parameters)

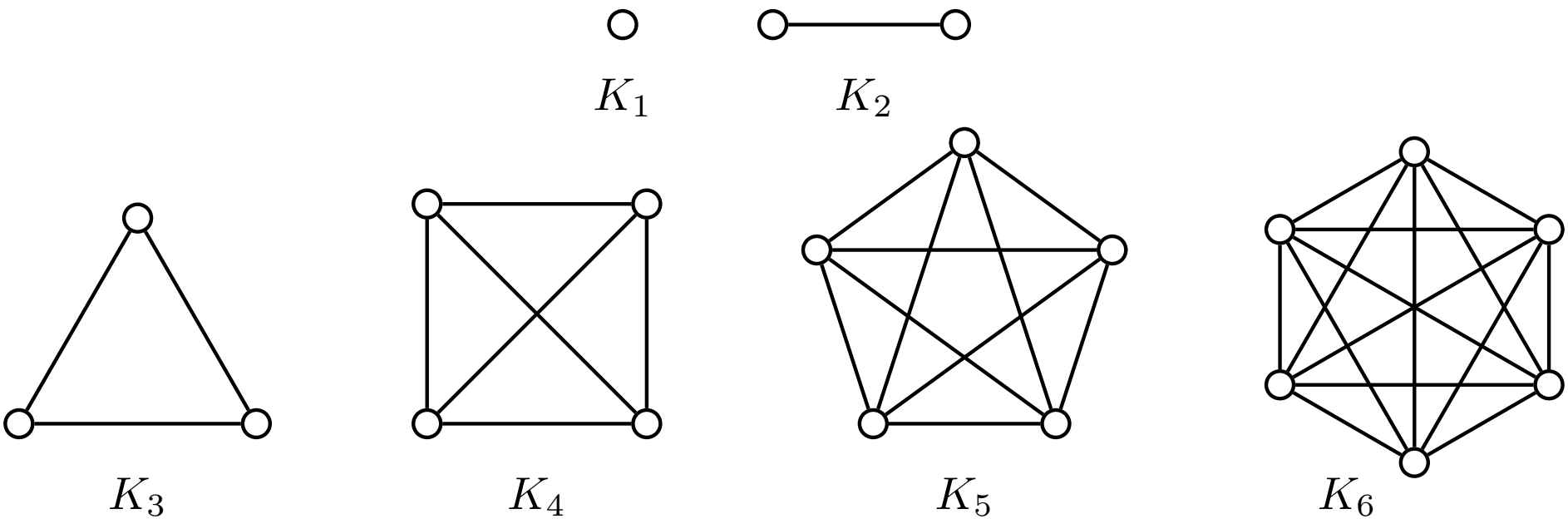
Some graph classes appear often and have their names

- paths
- trees
- caterpillars
- lollipops
- books
- ...

Complete graph K_n

The graph on n vertices ($n \geq 1$), which contains all $\binom{n}{2}$ edges is called complete graph and is denoted by K_n.

$$K_n = (V, E): \quad V = \{1, 2, \ldots, n\}, \quad E = \{ij : i, j = 1, 2, \ldots n \wedge i \neq j\}$$

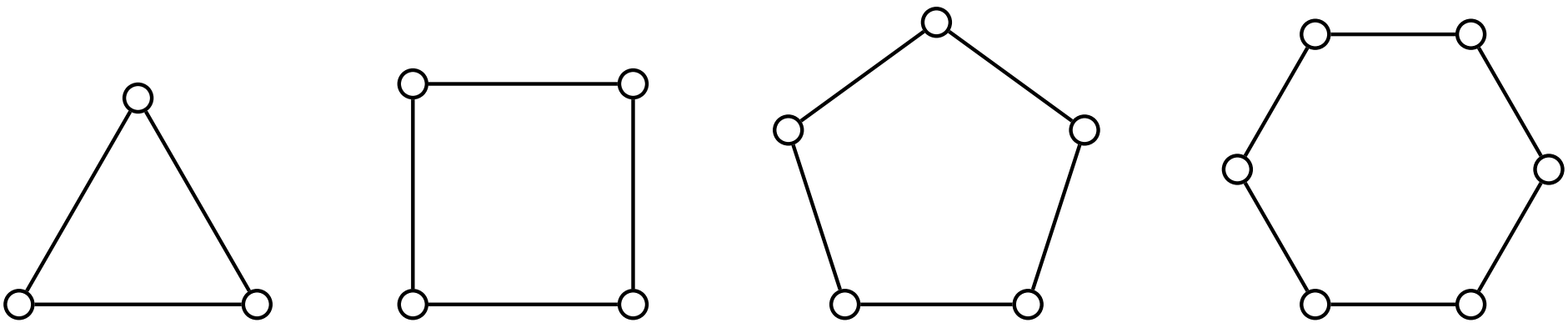


Trivial graph and complete graphs.

Cycle C_n

Graph on n vertices ($n \geq 3$), which are connected by edges into a single cycle is called a cycle on n vertices and denoted by C_n. The number n is the length of the cycle C_n.

$C_n = (V, E)$: $\quad V = \{1, 2, \ldots, n\}, \quad E = \{i(i+1) : i = 1, 2, \ldots n-1\} \cup \{1n\}$



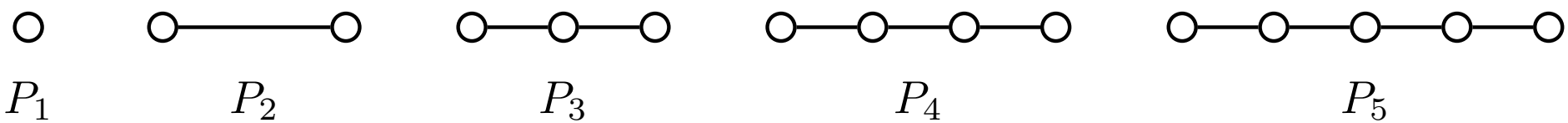
Cycles C_3, C_4, C_5 and C_6.

The cycle of length three is often called a triangle.

Path P_n

The graph on n vertices ($n > 0$), which are connected consecutively by $n-1$ edges is called a path and denoted by P_n.

$$P_n = (V, E): \quad V = \{1, 2, \ldots, n\}, \quad E = \{i(i+1): i = 1, 2, \ldots n-1\}$$



Paths P_1, P_2 P_3, P_4, and P_5.

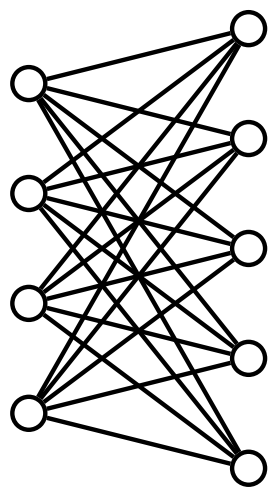
Notice, there is another notation (less common):
index = number of edges

Complete bipartite graph $K_{m,n}$

A graph, whose vertex set is partitioned into two disjoined nonempty subsets M and N ($|M| = m \geq 1$, $|N| = n \geq 1$) and which contains all $m \cdot n$ edges uv such that $u \in M$ and $v \in N$ is called complete bipartite graph and is denoted by $K_{m,n}$.

$$K_{m,n} = (M \cup N, E): \quad M = \{u_1, u_2, \ldots, u_m\}, \quad N = \{v_1, v_2, \ldots, v_n\},$$

$$M, N \neq \emptyset, \quad M \cap N = \emptyset, \quad E = \{u_i v_j : i = 1, 2, \ldots, m \wedge j = 1, 2, \ldots n\}.$$



Graph $K_{4,5}$.

Degree of a vertex in a graph

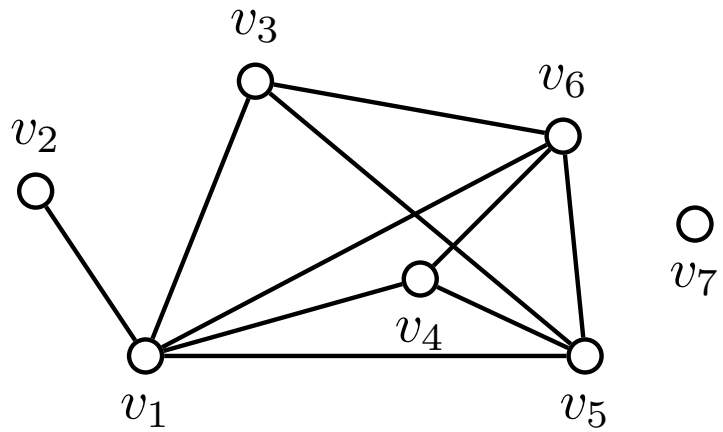
For a given edge uv, we call the vertices u and v the end-vertices of the edge uv.

We also say that u and v are incident with the edge uv ($u \in \{u, v\}$).

Two different end-vertices of the same edge we call neighbors. If such edge does not exist, we call the vertices independent.

Definition

Degree of a vertex in a graph G is the number of edges which are incident with the given vertex v.
The degree of a vertex v in a graph G is denoted by $\deg(v)$ (or $\deg_G(v)$).



Graph with degrees 5, 1, 3, 3, 4, 4, and 0.

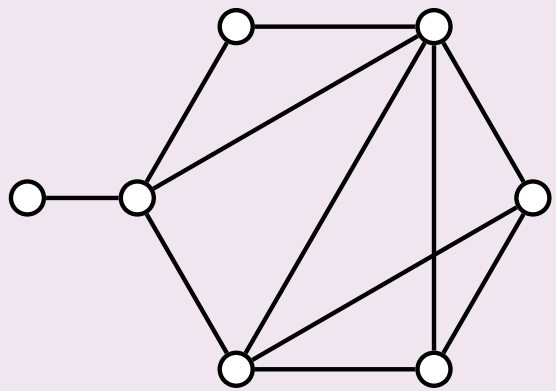
Theorem (Parity Principle)

The sum of all degrees in a graph is even and is equal to the double of edges.

$$\sum_{v \in V(G)} deg_G(v) = 2|E(G)|$$

Proof Both, the left and right side of the equation gives the count of end-vertices. Summing degrees every end-vertex contributes a 1 to a degree of some vertex. Moreover, every edge has two end-vertices, hence the sum of degrees is equal to the double of number of edges. □

Question



How many edges are in a graph with vertex degrees $5, 4, 4, 3, 3, 2, 1$*?*

Example

How many edges are in a graph G which has thirty vertices of degree 5 and five vertices of degree 4?

By Parity Principle the double of edges equals to the sum of vertex degrees.

$$\sum_{v \in V(G)} \deg_G(v) = 30 \cdot 5 + 5 \cdot 4 = 150 + 20 = 170$$

Thus, the graph G has $170/2 = 85$ edges. Does G exist? We show later.

Example

How many edges are in a graph G which has five vertices of degree 5 and thirty vertices of degree 4?

Similarly...

$$\sum_{v \in V(G)} \deg_G(v) = 5 \cdot 5 + 30 \cdot 4 = 145$$

By Parity Principle no such graph exists!

Example

Nine friends exchange Christmas presents. Everyone gave three presents to his friends. Show, that it is not possible that everyone receives presents from precisely those friends he gave his presents to. Hint: show that no graph model of such exchange can exist.

We set up a graph: vertices=friends, edges=pairs of exchanged presents.

$$\sum_{v \in V(G)} \deg_G(v) = 9 \cdot 3 = 27$$

By Parity Principle no such graph exists, thus there is no solution to the problem.

There is no exchange of presents possible such that each of the nine friends would receive presents from precisely those friends he/she gave his/her presents to.

Definition

Take a graph G with vertices $v_1, v_2, \ldots, v_n$. The sequence $(\deg(v_1), \deg(v_2), \ldots, \deg(v_n))$ is called the degree sequence of the graph G.

Not every sequence is a degree sequence of a graph. How to tell?

Theorem (Havel–Hakimi)

Let $(d_1 \geq d_2 \geq \cdots \geq d_n)$ be a sequence of natural numbers. Then a non-trivial graph on n vertices with the degree sequence

$$D = (d_1, d_2, \ldots, d_n)$$

exists if and only if there exists a graph on $n-1$ vertices with the degree sequence

$$D' = (d_2 - 1, d_3 - 1, \ldots, d_{d_1+1} - 1, d_{d_1+2}, d_{d_1+3}, \ldots, d_n).$$

Since we deal only with finite graphs, we can by recursion decide about every finite sequence if it is a degree sequence of some graph. Moreover we can construct example of such a graph (needs not to be unique!).

Example

Does there exist a graph with the degree sequence $(3, 3, 3, 1, 1)$?

No, by Parity Principle.

Example

Does there exist a graph with the degree sequence $(3, 3, 3, 1)$?

The Parity Principle? No such graph exists, we use Theorem H-H to prove it.

Examining the degree sequence $(3, 3, 3, 1)$ we get

$$(3, 3, 3, 1) \overset{HH}{\sim} (2, 2, 0) \overset{HH}{\sim} (1, -1)$$

which obviously is not a degree sequence.

Example

Does there exist a graph with the degree sequence $(6, 4, 4, 1, 1)$?

Obviously, no such graph exists.

If you do not know why, try using Theorem H-H.

Example

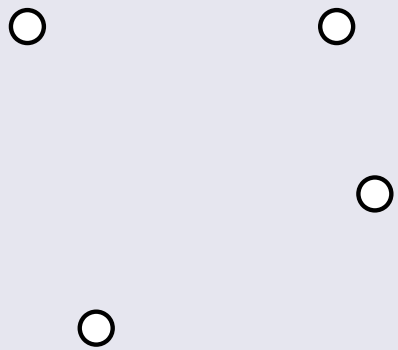
Does there exist a graph with the degree sequence $(6, 5, 4, 3, 3, 2, 1)$?

Yes, by Theorem H-H. Moreover, we construct such a graph.

Based on the degree sequence $(6, 5, 4, 3, 3, 2, 1)$ we get

$$(6, 5, 4, 3, 3, 2, 1) \overset{HH}{\sim} (4, 3, 2, 2, 1, 0) \overset{HH}{\sim} (2, 1, 1, 0, 0) \overset{HH}{\sim} (0, 0, 0, 0).$$

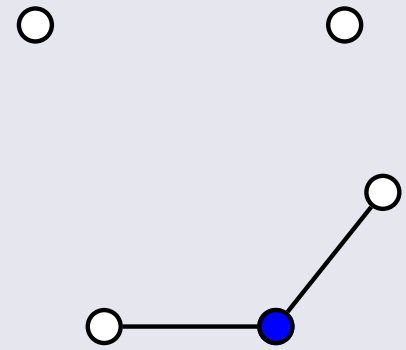
To construct the graph we add vertices.



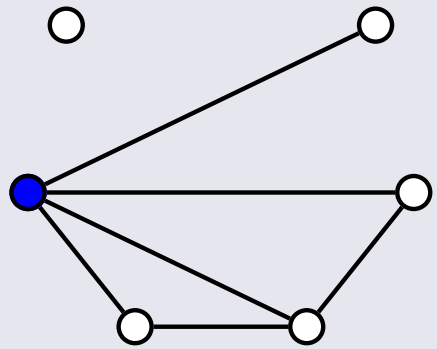
We start by drawing a graph with the degree sequence $(0, 0, 0, 0)$.

continued . . .

Then we add a vertex of degree 2 and connect it to two vertices of degree 0 (hereby we obtain vertices with degree 1).



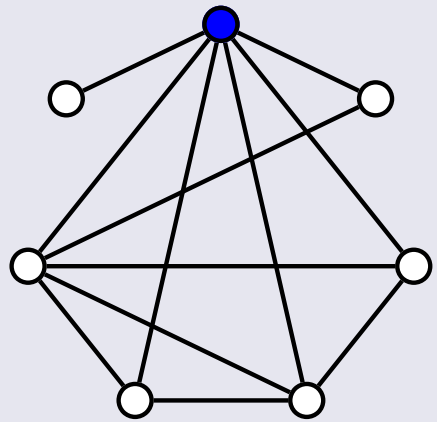
We obtain a graph with the degree sequence $(2, 1, 1, 0, 0)$.



To obtain a graph with the degree sequence $(4, 3, 2, 2, 1, 0)$, we add a vertex of degree 4 and connect it to vertices of degree 2, 1, 1 and 0.

continued . . .

Finally we add a vertex of degree 6 and connect it to all remaining vertices.



We get a graph with the degree sequence $(6, 5, 4, 3, 3, 2, 1)$.

Summary: using Havel-Hakimi Theorem we can

- decide, whether a given finite sequence of positive integers is a degree sequence of some graph,
- if yeas, the process yields an example of such a grah.

Definition

The highest degree of a vertex in a graph G we denote by $\Delta(G)$.
The smallest degree we denote by $\delta(G)$.

Questions

- Is every non-increasing sequence of natural numbers a degree sequence?
- How to tell if a particular sequence is a degree sequence of a graph?
- Is the graph uniquely determined by its degree sequence?
- How many graphs do exist with a particular degree sequence?

Subgraphs

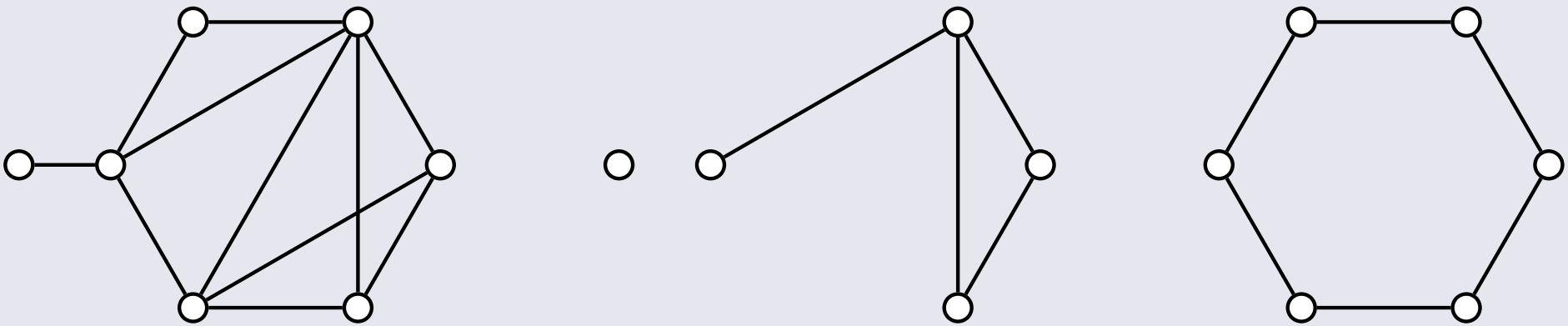
Often we consider graphs, that arose by deleting some vertices (and all incident edges) from a graph, or by deleting some edges or both. In this way we obtain a *subgraph*.

Definition

Graph H is called a subgraph of a graph G if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. We write $H \subseteq G$.

Notice: by deleting a vertex of G we have to delete all incident edges as well. The definition is correct, otherwise H would not be a graph.

Example

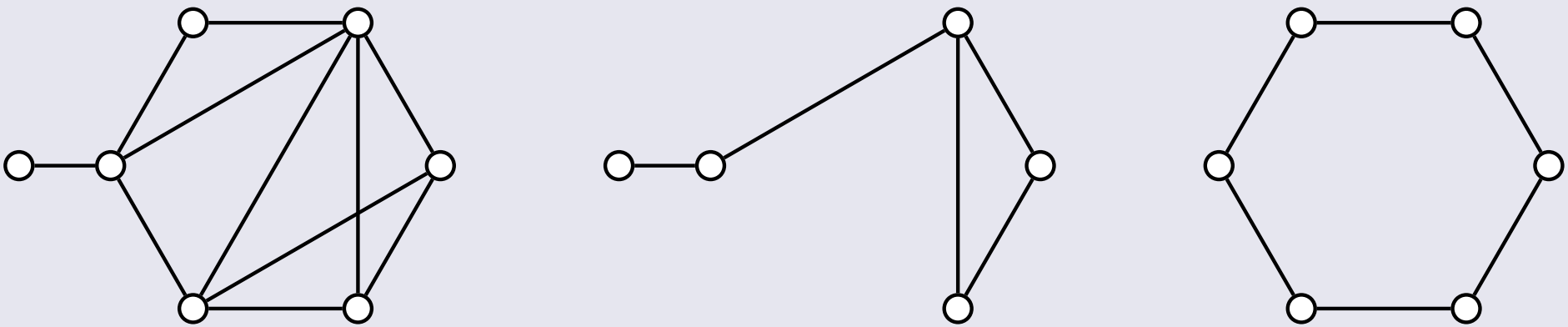


Graph G and its subgraphs H_1 and H_2.

Induced subgraph

A subgraph I of a graph G is called induced subgraph of G if $E(I)$ contains all edges of G, that are incident with vertices in $V(I)$ (we omit no other edges besides those incident with the deleted vertices).

Example



Graph G and subgraph H_1 (induced) and H_2 (not induced).

Factor of a graph

A subgraph F of a graph G is called a factor of G if $V(F) = V(G)$.

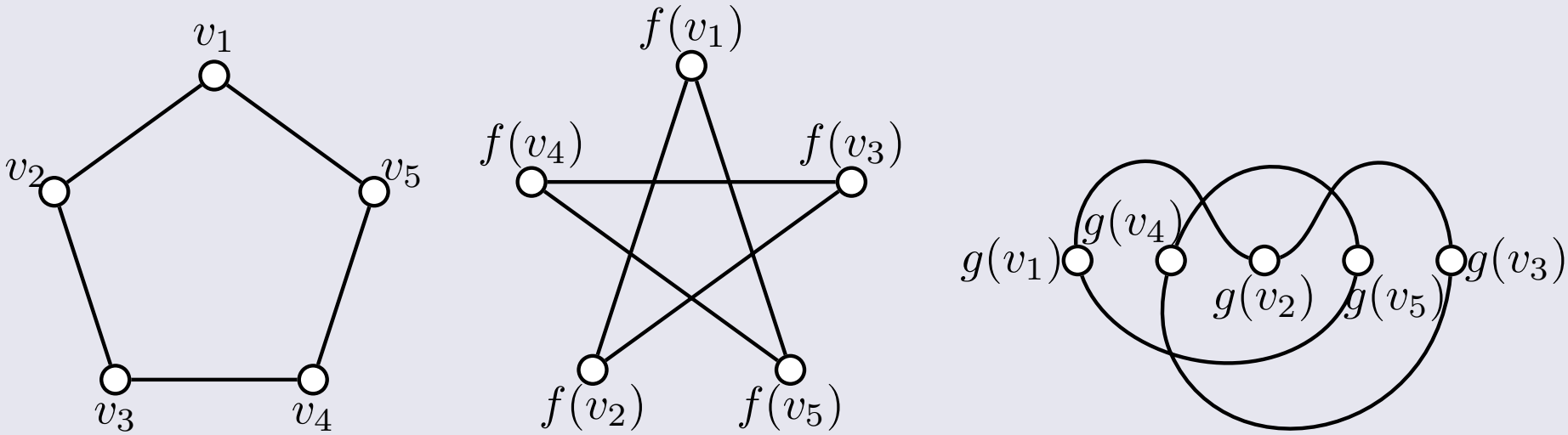
Graph isomorphisms

Definition

Isomorphism of graphs G and H is a bijective mapping $f : V(G) \to V(H)$, for which the following holds: every pair of vertices $u, v \in V(G)$ is joined by an edge in G if and only if $f(u), f(v)$ are joined by an edge in H.

For short $\forall u, v \in V(G) : uv \in E(G) \Leftrightarrow f(u)f(v) \in E(H)$.

Example



Isomorphic graphs.

Fact

Isomorphic graphs G and H have

- the same number of vertices
- the same number of edges
- the same highest degree $\Delta(G) = \Delta(H)$
- the same lowest degree $\delta(G) = \delta(H)$
- the same degree sequence
- each subgraph of G has to be a subgraph of H and vice versa
- ...

If some bijection f is an isomorphism, it has to map vertices to vertices of the same degree, e.g.

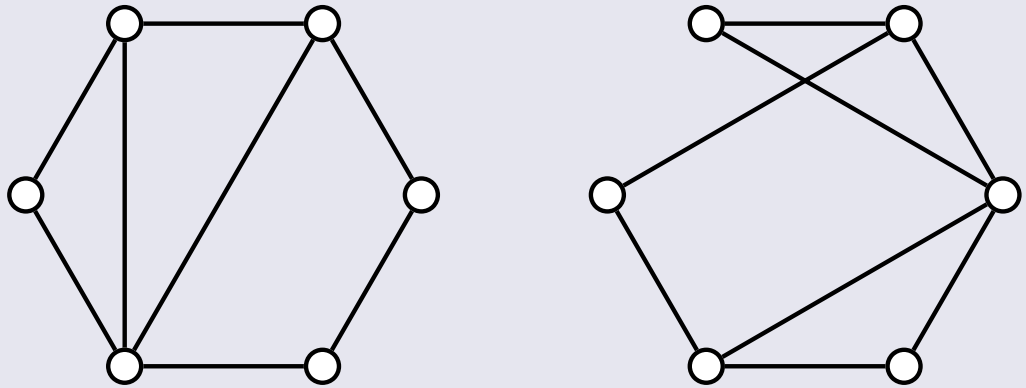
$$\deg_G(v) = \deg_H(f(v)).$$

This implication cannot be reversed!

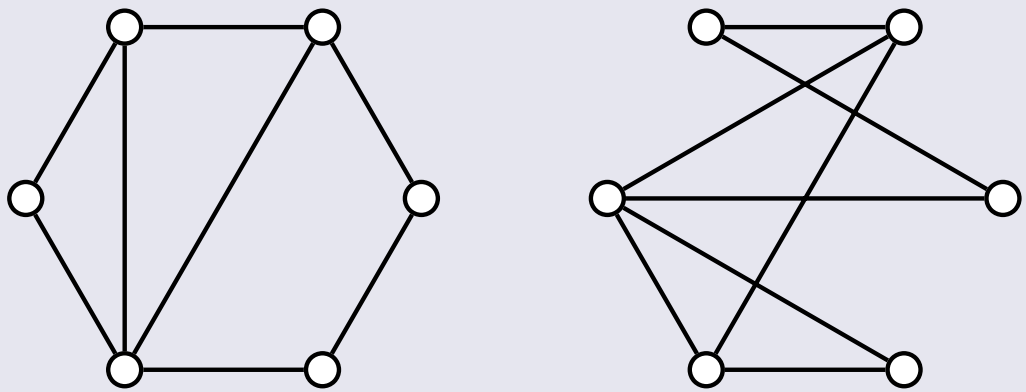
It is not enough to compare degree sequences!

Example

Are the following two graphs both with the degree sequence $(4, 3, 3, 2, 2, 2)$ isomorphic?

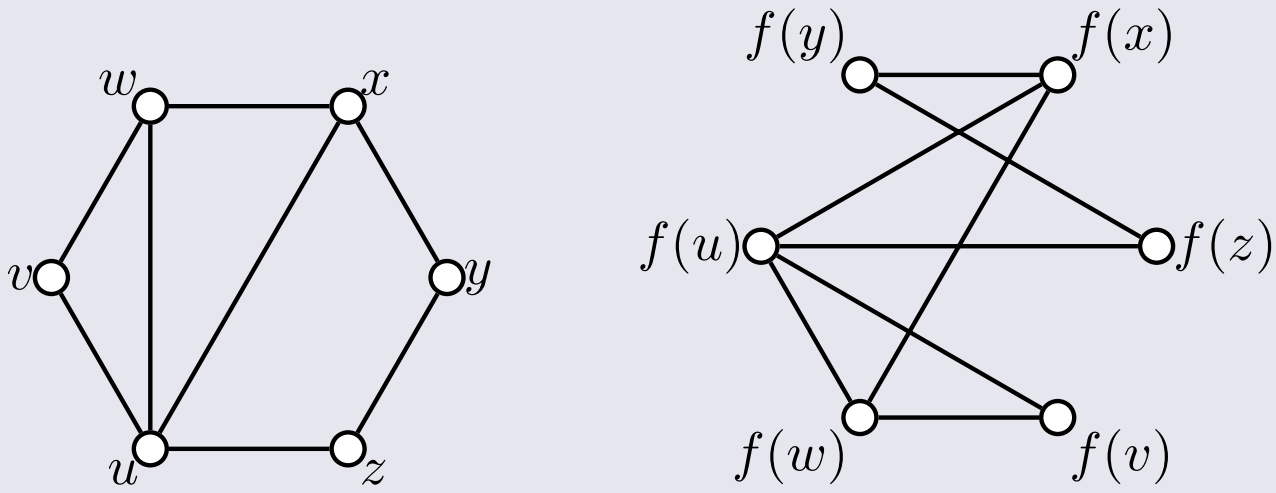


Are these two graphs isomorphic?



Are these two graphs isomorphic?

continued...



Isomorphism f between two given graphs.

Note

On the examples above we have practised some approaches used to determine, whether two given graphs are isomorphic or not.

No "fast and easy" universal algorithm for decision about being isomorphic is known!

The only universal algorithm for finding an isomorphism of two graphs or for deciding that such isomorphism does not exists is to *try all* feasible bijections between vertices (there can be up to $n!$ of them).

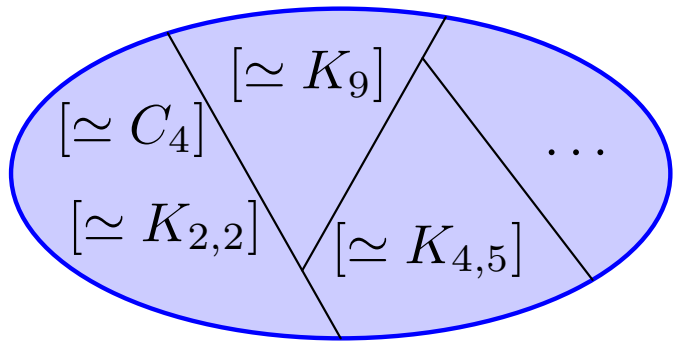
Theorem

Relation of "being isomorphic" $\simeq$ is an equivalence relation on the set of all graphs.

Proof Relation $\simeq$ is

- reflexive, since every graph is isomorphic to itself by identity,
- symmetric, since to every isomorphism (bijection) f there exists (a unique) f^{-1},
- transitive, since by composition of isomorphisms (mappings) we get again an isomorphism (mapping). □

When talking about a certain *graph*, we often address the entire isomorphism class, is does not depend on a particular representation, drawing or notation of a given graph.



Graph classes.

Implementation of graphs

We denote vertices of a graph G by natural numbers $0, 1, \ldots, n-1$.
Among the most common computer implementations of graphs are:

Adjacency matrix

Adjacency matrix of a graph G is a square matrix $A = (a_{ij})$ of order n, in which

$$a_{ij} = \begin{cases} 1 & \text{if } v_i v_j \in E(G) \\ 0 & \text{otherwise.} \end{cases}$$

It can be stored in a two-dimensional field `a[][]` where `a[i][j]=1` if the edge/arc ij is in the graph, otherwise `a[i][j]=0`.

The sum of the i-th row/column of the matrix equals the degree of vertex i.

Advantages:

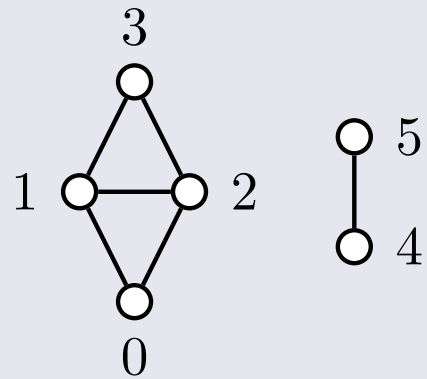
- quick to check the existence of and edge/arc uv,
- easy to add/remove an edge/arc ij.

Disadvantages:

- adjacency matrix is large, uses memory even for sparse graphs.

Example

Set up the adjacency matrix of the given graph.



The adjacency matrix is

$$A = \begin{array}{c} v\backslash v \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

Incidence matrix

Incidence matrix B of a graph G is a rectangular matrix with n rows and $m = |E(G)|$ columns. Each row corresponds to a given vertex of the matrix B and each column corresponds to a column of B.

Element b_{ij} of the matrix B equals 1, if vertex i is incident to the edge e_j, otherwise we set $b_{ij} = 0$.

Sum of every column is 2 and the sum of row i equals the degree of vertex i.

Advantages:

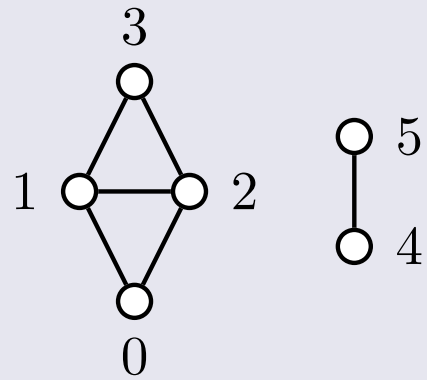
- easy to find end-vertices of a given edge,
- easy to "reattach" edge ij in a graph.

Disadvantages:

- expensive to add/remove edges to the incidence matrix,
- for large graphs is the incidence matrix large and sparse.

Example

Set up the incidence matrix of the given graph.



The incidence matrix is

$$B = \begin{array}{c} v\backslash e \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 01 & 02 & 12 & 13 & 23 & 45 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

Neighbor lists

For every vertex $i = 0, 1, \ldots, n-1$ of a given graph G we create an array (list) of vertices, say `neig[i][]`, which contains vertices adjacent to i.

Each array will have $\deg(i)$ entries, where $\deg(i)$ is the degree of vertex i, stored in `deg[]`.

Elements `neig[i][0], neig[i][1], ..., neig[i][deg[i]-1]` contain vertices (or their indices) adjacent to vertex i.

Example

`deg[] = [2, 3, 3, 2, 1, 1],`

$$
\begin{aligned}
\texttt{neig[0][]} &= [1, 2], \\
\texttt{neig[1][]} &= [0, 2, 3], \\
\texttt{neig[2][]} &= [0, 1, 3], \\
\texttt{neig[3][]} &= [1, 2], \\
\texttt{neig[4][]} &= [5], \\
\texttt{neig[5][]} &= [4].
\end{aligned}
$$

We have to keep the symmetry of edges ij and ji in mind!

Advantages:

- suitable for graphs with few edges, easy to evaluate $\deg(v)$,

Disadvantages:

- costly to alter the structure of a given graph,
- costly to find a given edge.

All given representations can be used for oriented graphs as well.
In general, for multigraphs one can use the incidence matrix or perhaps neighbor list.

More complex structures are needed for labeled (weighted) graphs (we assign labels/weights to vertices and/or edges).

We can use additional fields or structured data types...

Next lecture

Chapter 2. Connectivity of graphs

- motivation
- connectivity and components of a graph
- searching through a graph
- higher degrees of connectivity
- Eulerian graphs traversable in "one trail"

Discrete mathematics

Petr Kovář
petr.kovar@vsb.cz

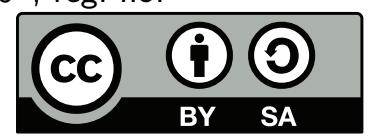
VŠB – Technical University of Ostrava

Winter term 2021/2022
DiM 470-2301/02, 470-2301/04, 470-2301/06

EUROPEAN UNION
European Structural and Investment Funds
Operational Programme Research,
Development and Education

MINISTRY OF EDUCATION,
YOUTH AND SPORTS

The translation was co-financed by the European Union and the Ministry of Education, Youth and Sports from the Operational Programme Research, Development and Education, project "Technology for the Future 2.0", reg. no. CZ.02.2.69/0.0/0.0/18_058/0010212.



About this file

This file is meant to be a guideline for the lecturer. Many important pieces of information are not in this file, they are to be delivered in the lecture: said, shown or drawn on board. The file is made available with the hope students will easier catch up with lectures they missed.

For study the following resources are better suitable:

- Meyer: Lecture notes and readings for an http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-042j-mathematics-for-computer-science-fall-2005/readings/" (weeks 1-5, 8-10, 12-13), MIT, 2005.
- Diestel: Graph theory http://diestel-graph-theory.com/ (chapters 1-6), Springer, 2010.

See also http://homel.vsb.cz/~kov16/predmety_dm.php

Lecture overview

Chapter Connectivity of graphs

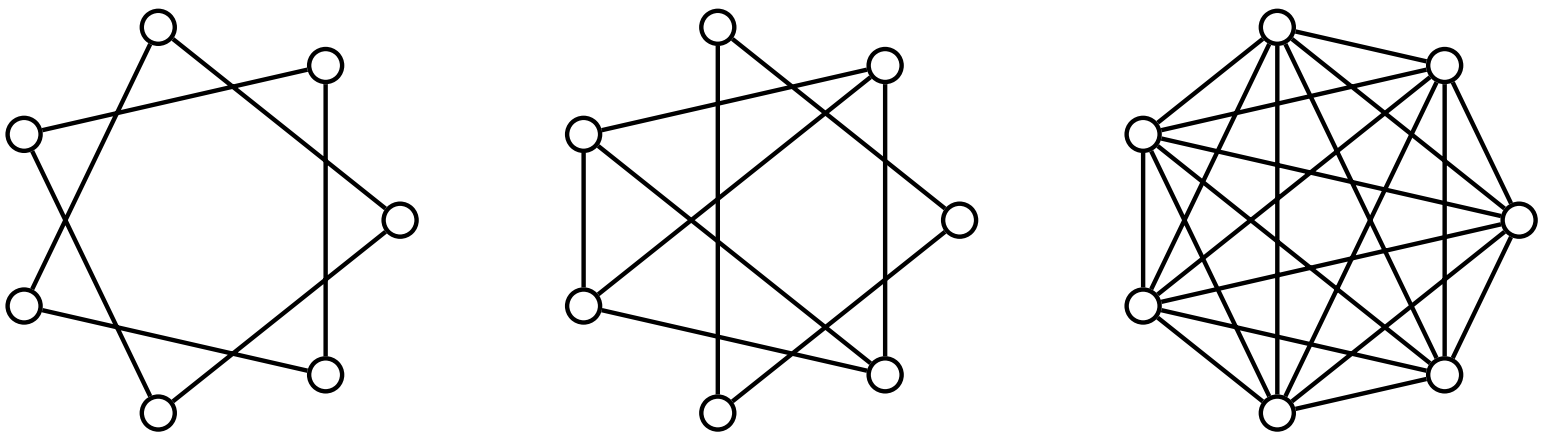
- motivation
- connectivity and components of a graph
- searching through a graph
- higher degrees of connectivity

Connectivity of graphs

If a graph represents a computer network or a road network, it is natural to examine, whether one can transmit a signal or send goods from vertex u to vertex v. This leads us to the notion of *connectivity of graphs*.
For similar reasons one can examine the robustness against local failures:

- vertex redundancy
- connectivity even in the case where several edges are cut

Hence we arrive at the notion of the *degree of vertex- and edge-connectivity*.



Connected and not connected graphs.

Connections in graphs, components

Informally: A graph is connected, if there exists a "connection" between every two vertices (not necessarily an edge).

Formally: we introduce the concept of a *walk*, *trail*, and *path* in s graph.

Definition

A v_0v_n walk in a graph G is such a sequence of vertices and edges

$$(v_0, e_1, v_1, e_2, v_2, \ldots, e_n, v_n),$$

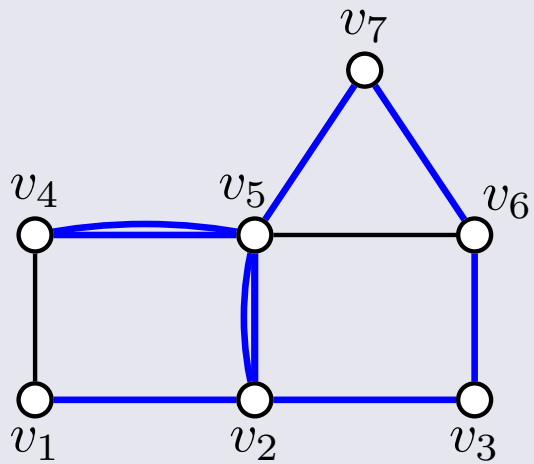
where v_i are vertices and e_i are edges of G such, that v_{i-1} and v_i are incident with e_i. The number of edges n is the length of the v_0v_n walk. v_0 is the starting vertex and v_n the end-vertex of the walk.

If there are no multiple edges, we can describe the walk by listing just a sequence of vertices.

$$(v_0, v_1, v_2, \ldots, v_n)$$

Alternatively we can omit the parentheses: $v_0, v_1, v_2, \ldots, v_n$.

Example

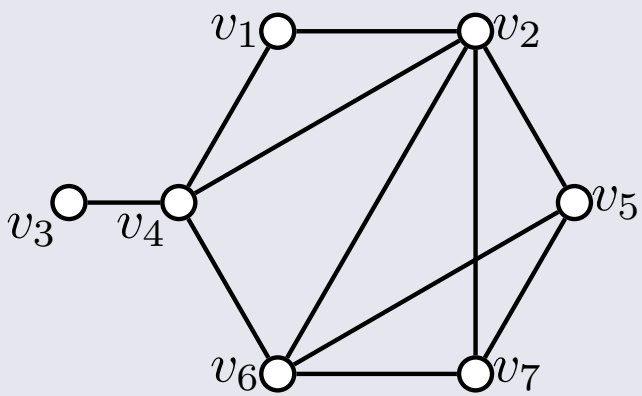


Walk $v_1, v_1v_2, v_2, v_2v_5, v_5, v_5v_7, v_7, v_7v_6, v_6, v_6v_3, v_3, v_3v_2, v_2, v_2v_5, v_5, v_5v_4, v_4, v_4v_5, v_5$ is highlighted in blue.

Briefly:

$v_1, v_2, v_5, v_7, v_6, v_3, v_2, v_5, v_4, v_5$.

Example



$v_1, v_2, v_6, v_7, v_2, v_1, v_2, v_3$ is not a walk

walk $v_1, v_2, v_6, v_7, v_2, v_1, v_2, v_4$

walk $v_1, v_2, v_7, v_5, v_6, v_4, v_3$

(trivial) walk v_4

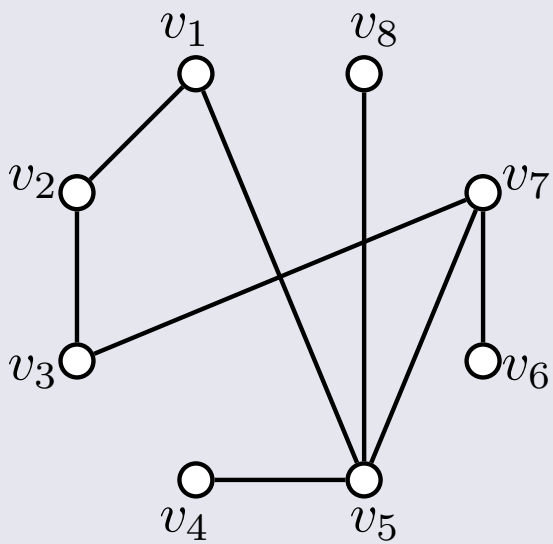
The notion of "connectivity" is based on the term "walk".

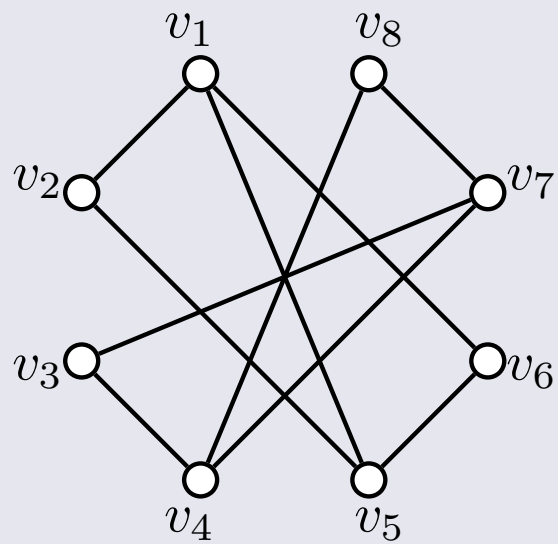
Definition

We say vertex v can be reached from vertex u, if there exists a uv walk in the given graph.
We say a graph is connected if for every pair of vertices u, v is vertex v reachable from vertex u. Otherwise the graph is not connected.

Example

Is each of the two graphs connected?





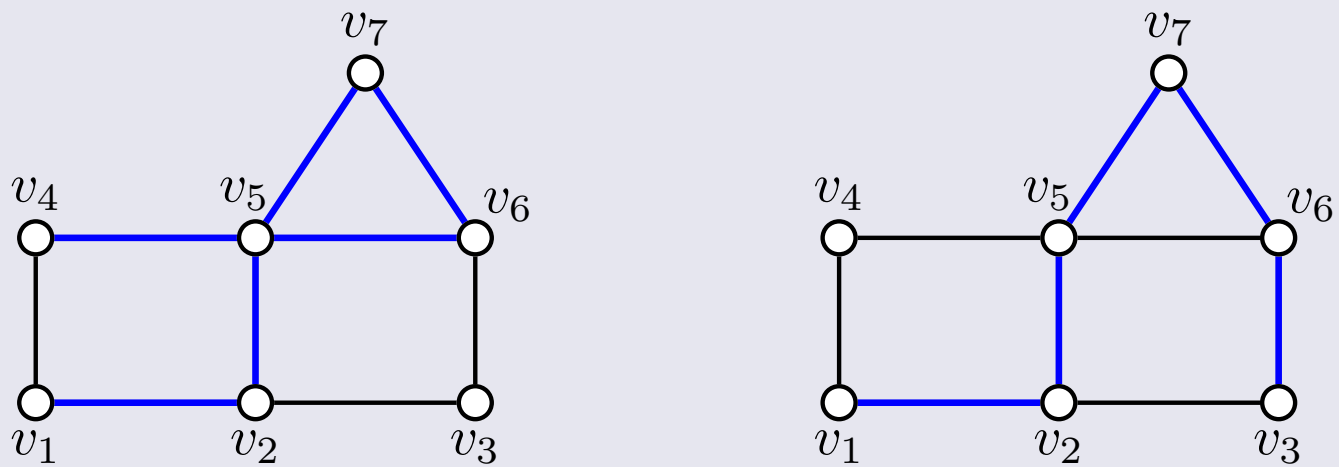
In some application the repetitions of edges or vertices is *not allowed* (pipes, traffic networks, electrical networks, . . .).

Definition

Trail is a walk with no repeated edges.
Path is a walk with no repeated vertices.

Terminology: we travel along trails, we draw "in one stroke".
Vertices and edges of a path in a graph form a subgraph that is a path.

Example



Trail $v_1, v_2, v_5, v_7, v_6, v_5, v_4$ *and a path* $v_1, v_2, v_5, v_7, v_6, v_3$.

Theorem

If there exists a uv walk in G, then there exists also a uv path in G.

Proof Let W be a uv walk $u = v_0, e_1, v_1, \ldots, e_n, v_n = v$ of length n in G. We want to find a uv walk P with no repeated vertex. If no vertex in W is repeated, then $P = W$ is the wanted path.
If a certain vertex v_i is repeated, we can omit the entire part of W between its first occurrence v_i and its last occurrence of, say v_k. We obtain a uv walk W', in which the vertex v_i occurs only once.
Now if no other vertex is repeated in W' we take $P = W'$. Otherwise repeat the process for the next repeated vertex.
The algorithm is deterministic, since there are only finitely many vertices in G. □

If there exists a uv walk in G, we can obtain a uv path by the proof of the theorem. We say that vertices u and v are joined by a path in G.

On the set of vertices of a given graph G we introduce the relation $\sim$. Two vertices $u, v \in V(G)$ are related in $\sim$ (we write $u \sim v$) if and only if there is uv walk in G.

We call $\sim$ a "relation of being reachable."

Lemma

The relation $\sim$ is an equivalence relation.

Proof

- Reflexivity follows from the existence of a trivial walk uu of length 0. For each vertex $u \in V(G)$ is $u \sim u$.
- Symmetry is obvious, since for each uv walk in G we can easily construct the vu walk in G by "reversing" the sequence of vertices and edges (in an unoriented graph). $\forall u, v \in V(G)$ is $u \sim v \Leftrightarrow v \sim u$.
- Transitivity follows from the fact, that by joining the walks $u, \ldots, v$ and $v, \ldots, w$ we obtain the walk $u, \ldots, w$. $\forall u, v, w \in V(G)$ is $u \sim v \wedge v \sim w \Rightarrow u \sim w$.

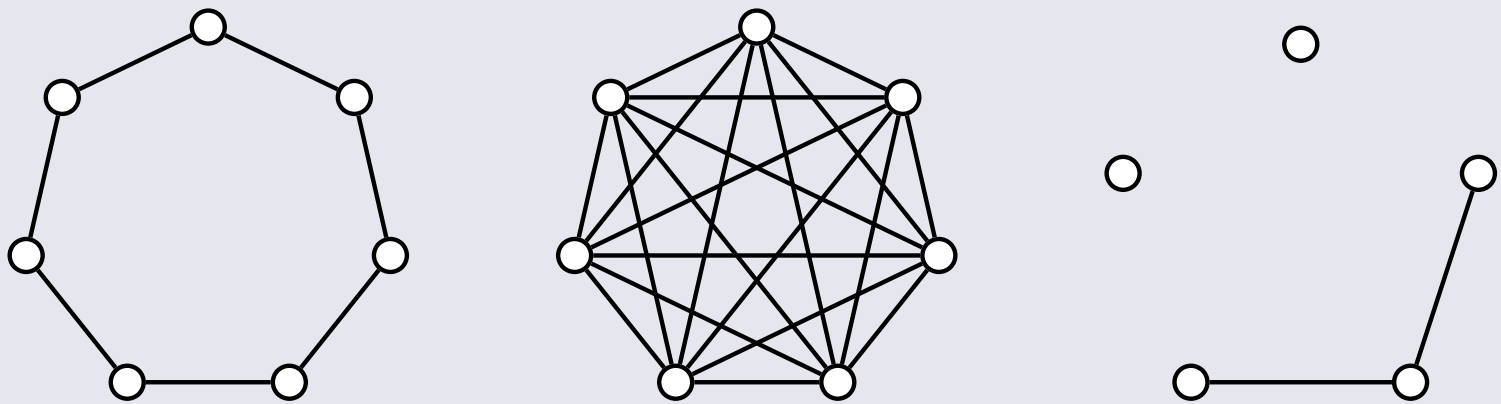
□

The relation $\sim$ from the Lemma above defines a partitioning of $V(G)$. Now we can define the following:

Definition

The equivalence classes of $\sim$ are subsets of $V(G)$ and subgraphs induced on these sets are called components of G.

Example



Examples of graphs with one and more components.

Two alternative definitions of connectivity follow.

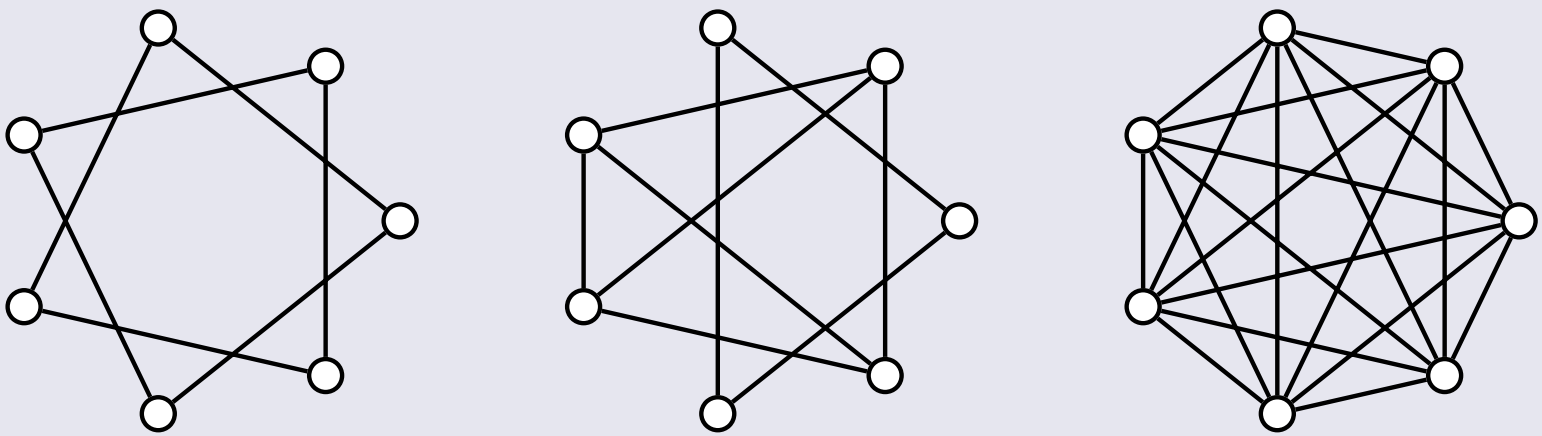
Definition

We say that a graph G is connected if it has only one component.

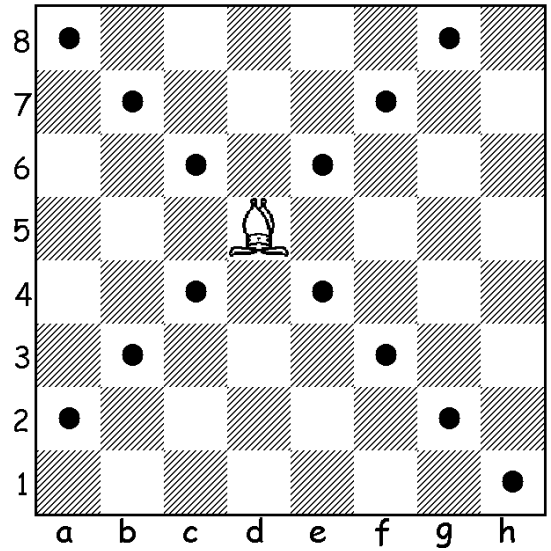
Definition

We say that G is connected if the $\sim$ relation on $V(G)$ is total.

Example



Examples of connected graphs and a not connected graph.



How bishop moves.

Example

Take the graph S, which describes all possible movements of a bishop on a chess board. (Recall that a bishop moves any number of vacant squares diagonally.)

Vertices correspond to squares and an edges joins two squares if and only if there is legal move of a bishop between them.

Graph S is not connected.

Example

Loyds' fifteen puzzle is a classic. The task is to shuffle the pieces numbered 1 through 15 so that they form an arithmetic progression 1 through 15 by rows.

We construct a graph of states: vertices are all possible arrangements of the pieces and an edge joined two arrangements if there is a single valid move from one to the other. One can show that such graph is not connected and thus there is no solution to the puzzle!

Towers of Hanoi

We have three pegs and a set of discs of different sizes. All discs are on one peg arranged according their size. The task is to move all discs to another peg while

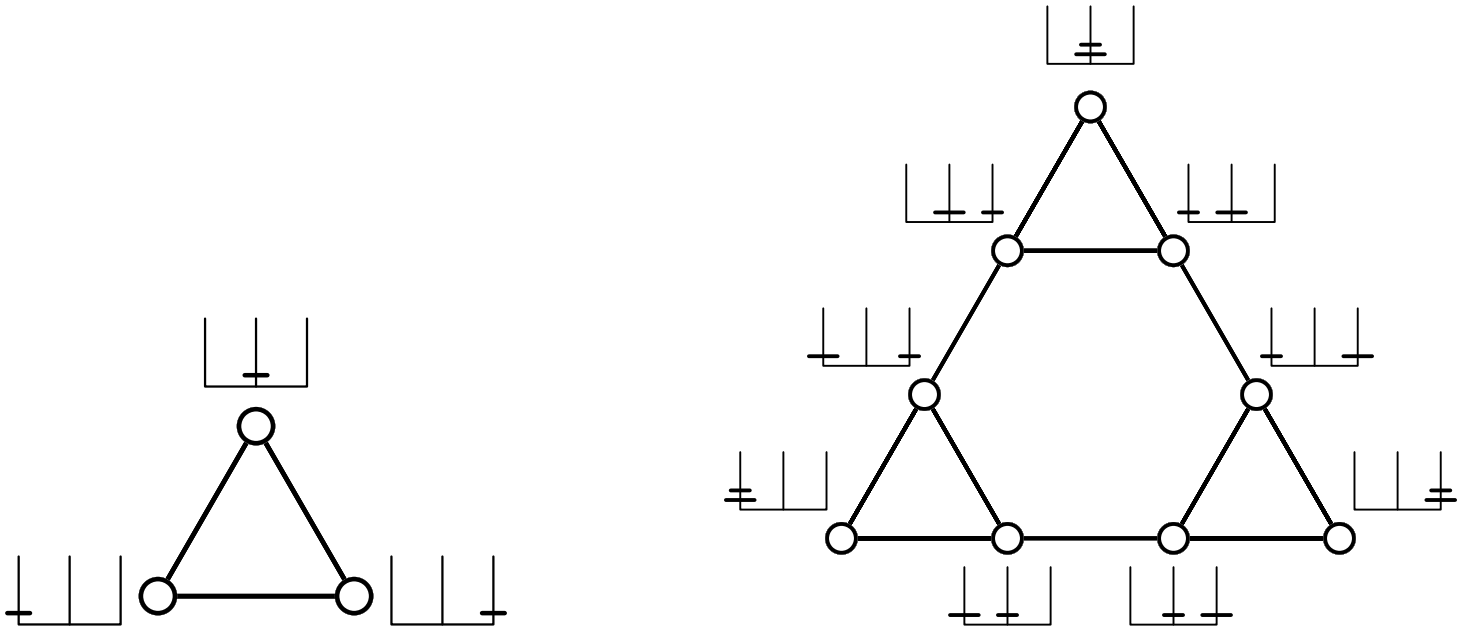
- always one discs is moved,
- never a larger disc can be on top of a smaller one.

Is it possible? What is the least number of moves required?

Graph formulation – state graph

For the solution we set up a state graph:

- vertices – each valid distribution of discs,
- edges – join two states with a valid move in between.

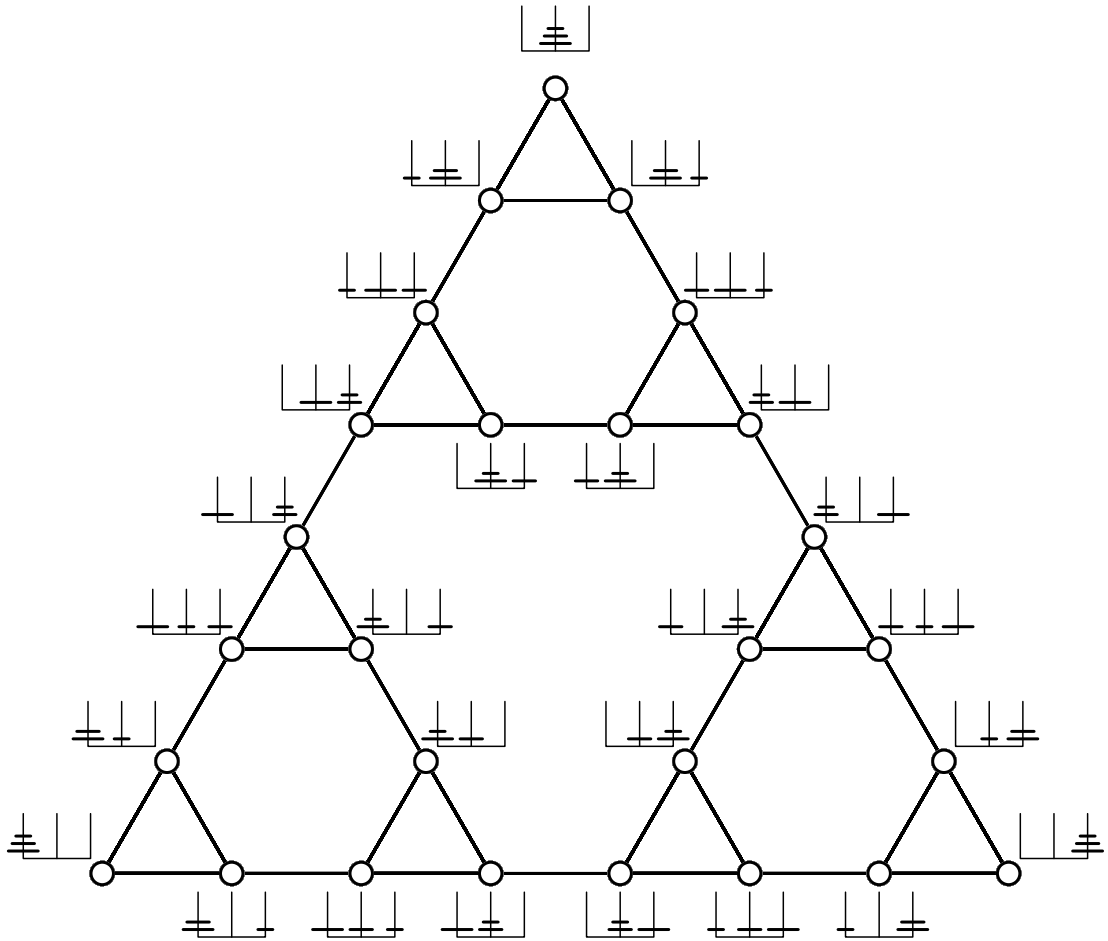


The puzzle with a single disc and with two discs.

For two discs we distinguish three cases where to put the larger discs.

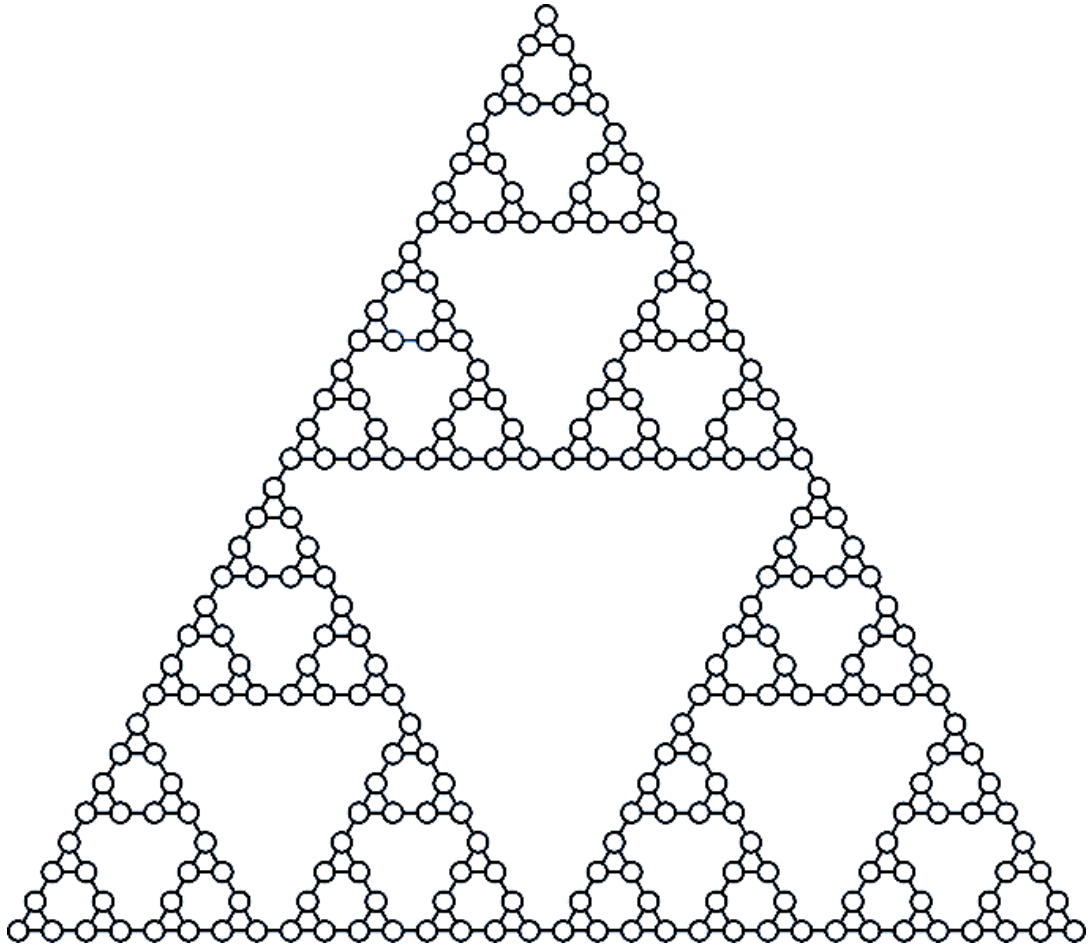
For each we take a copy of the state graph for one disc.
We add edges where the larger discs can be moved.

Graph formulation



For three discs similarly. . .

Graph formulation



. . . and for five discs.

Interpretation of the graph formulation

For n discs we have:

- 3^n different valid states $= 3^n$ vertices in the graph,
- all discs on one peg $=$ "tip" vertices,
- each state is reachable,
- to move all discs – at least $2^n - 1$ moves,
- fastest solution $=$ shortest path (next chapter).

Searching in a graph

For a general concept of "searching" in a graph we need to distinguish for each graph element few different states and one auxiliary structure:

Vertex can have the status ...

- initial – at the beginning,
- found – if it is found as an endpoint of an edge,
- processed – once all outgoing edges are processed.

Edge can have the status ...

- initial – at the beginning,
- processed – once it is processed from one end-point.

Depository as an auxiliary structure (sequence/set, see later),

- we store here all found and unprocessed vertices.

Based on how we pick the vertices in the depository, we obtain different variants of graph searching (depth-first/breadth-first search). For every vertex and edge we can implement an action to be performed – searching and processing a graph.

At the beginning

- pick an arbitrary vertex
- assign initial status to all vertices and edges

Algorithm of traversing all components

Traversing all connected components – we traverse each vertex and each edge.

```
// on the input is the graph G
input < graph G;
status(all vertices and edges of G) = initial;
depository U = arbitrary vertex u of G;
status(u) = found;
```

Now we traverse the graph...

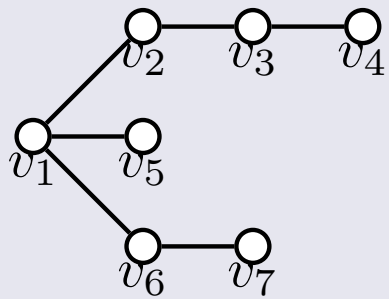
Algorithm of traversing all components (continued...)

```
// processing a component of G
while (U is not empty) {
   pick a vertex v from the depository U: U = U - {v};
   PROCESS(v);
   for (edges e incident with v)        // for all edges
      if (status(e) == initial)  PROCESS(e);
      w = other end-vertex of e = vw;  // known neighbor?
      if (status(w) == initial) {
         status(w) = found;
         add vertex w to depository U: U = U + {w};
      }
      status(e) = processed;
   }
   status(v) = processed;
   // check for additional components of G
   if (U is empty && G has additional vertices)
      U = {vertex u_1 from another component of G};
}
```

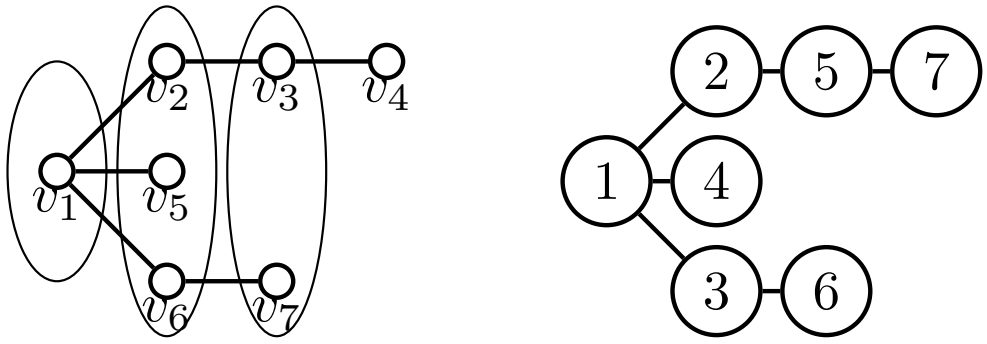
By various implementations of the *depository* we get various algorithms.

- *"Depth-first" search* – depository `U` is implemented as a stack, i.e. next processed vertex is the last found (and unprocessed).
- *"Breadth-first" search* – depository `U` is implemented as a queue, i.e. next processed vertex is the first found (and unprocessed).
- *Dijkstra algorithm* for shortest path – from the depository pick always the vertex closest to the initial vertex `v_0`; (work as breadth-first search when all edges are of "equal length").

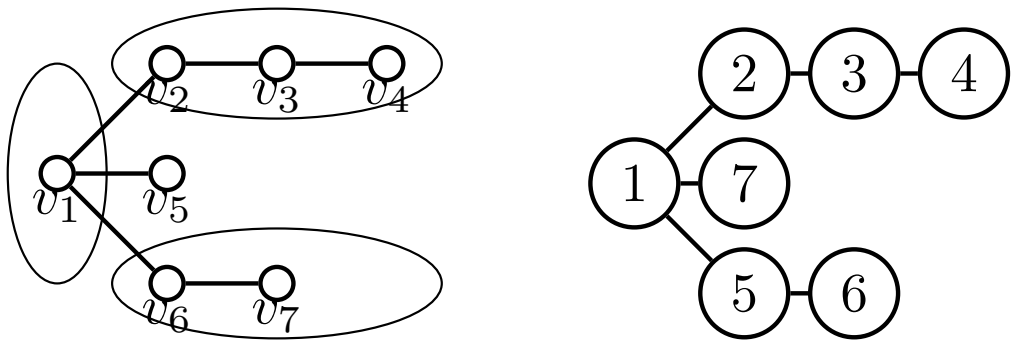
Example



Search through the graph using the depth-first and breadth-first search (starting at the vertex v_1).



Breadth-first search (starting at v_1).



Depth-first search (starting at v_1).

Note

The symbol $O(g(n))$ stands for all functions $f(n)$, for which there exist such positive constants c and n_0, that $\forall n > n_0$ is $0 \leq f(n) \leq c \cdot g(n)$.

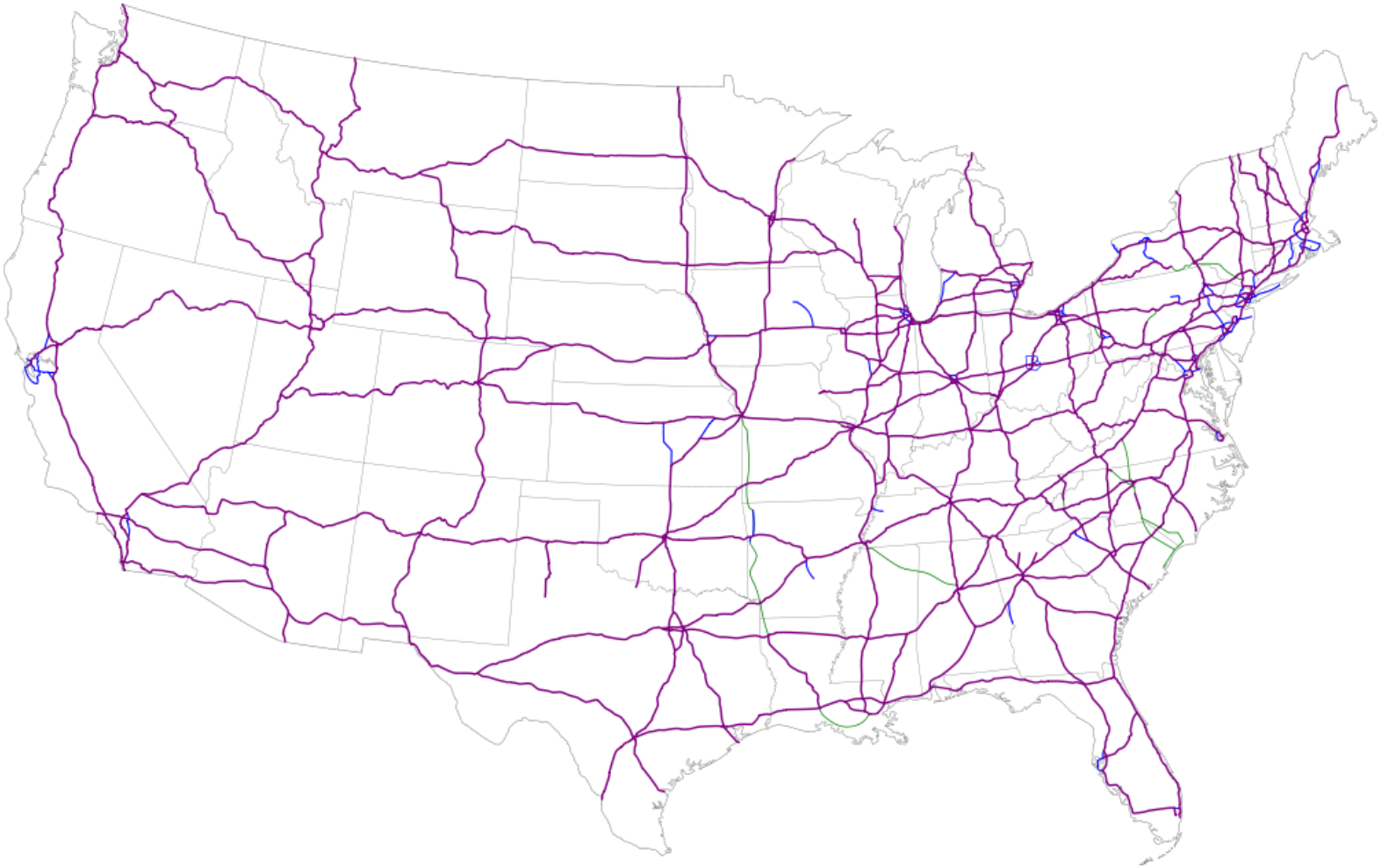
The algorithm described above is both easy and fast. The number of steps grows linearly with the number of vertices plus the number of edges of a given graph, the complexity is $O(n + m)$, where n is the number of vertices and m is the number of edges.

Questions

How to modify the algorithm to list all edges of a given graph?
How to modify the algorithm to check connectivity of a given graph?
How to modify the algorithm to find and distinguish all components of a given graph?

k-connectivity

Often we examine not only if there exists a connection between a two vertices in a graph, but also if there will be a loss of connectivity if the case of local failures (web, roads, electricity network).



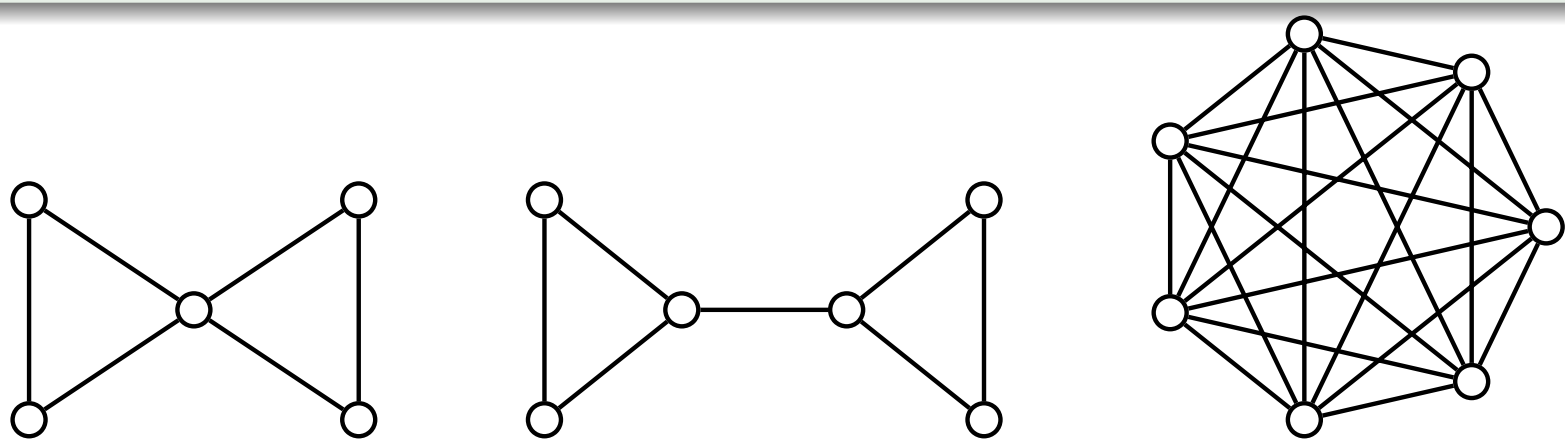
Eisenhower Interstate Highway System in the USA.

Definition

Graph G is edge k-connected if $k \geq 1$ and after removing any $k-1$ edges from G remains the resulting factor connected.
The edge-connectivity of G is such a highest number k that G is edge k-connected.

Definition

Graph G is vertex k-connected if $|V(G)| > k \geq 1$ and after removing any $k-1$ vertices from G remains the resulting induced subgraph connected.
The vertex-connectivity of G is such a highest number k that G is vertex k-connected.



Graphs with different edge/vertex k-connectivity.

We say that paths P and P' are:

- edge-disjoint if they share no edge,
- internally-disjoint if they share no internal vertex.

Theorem (Menger's theorem)

Graph G is edge k-connected if and only if there are at least k edge-disjoint paths between any two vertices (the paths can share vertices).
Graph G is vertex k-connected if and only if there are at least k internally-disjoint paths between any two vertices (the paths share only end-vertices).

Proof In the last Chapter. □

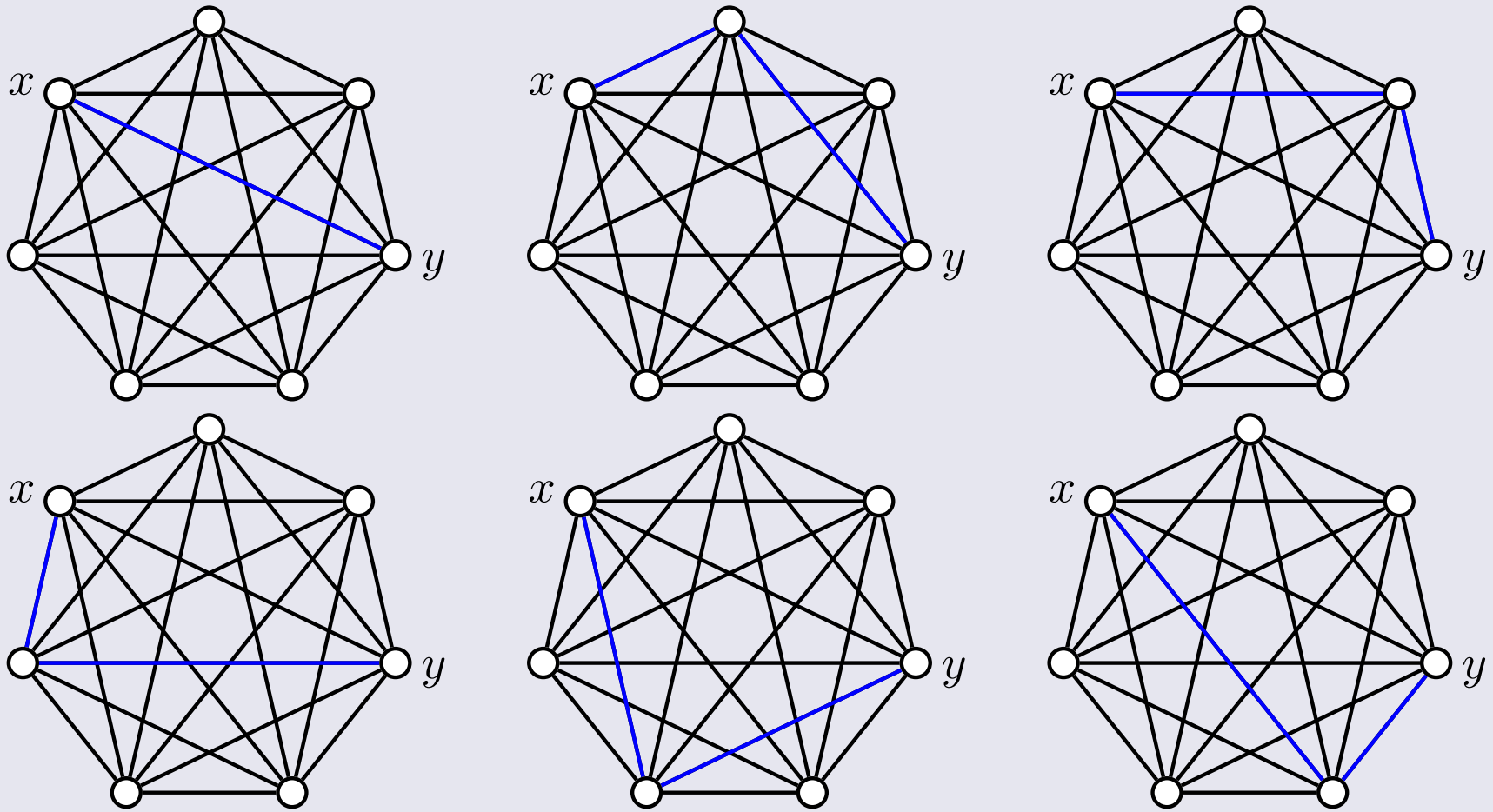
The following important theorem we state without a proof:

Theorem

In any graph G is the vertex-connectivity does not exceed edge-connectivity which then does not exceed the minimum vertex degree $\delta(G)$.

Example

The complete graph K_n is edge and vertex $(n-1)$-connected.



Different edge-/internally-disjoint paths between x and y in K_7.

Next lecture

Chapter Eulerian and hamiltonian graphs

- motivation
- eulerian graphs traversable in "one trail"
- hamiltonian graphs traversable in "one path"

Discrete mathematics

Petr Kovář
petr.kovar@vsb.cz

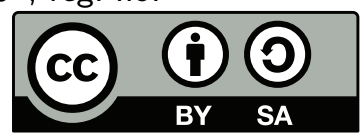
VŠB – Technical University of Ostrava

Winter term 2021/2022
DiM 470-2301/02, 470-2301/04, 470-2301/06

EUROPEAN UNION
European Structural and Investment Funds
Operational Programme Research,
Development and Education

MINISTRY OF EDUCATION,
YOUTH AND SPORTS

The translation was co-financed by the European Union and the Ministry of Education, Youth and Sports from the Operational Programme Research, Development and Education, project "Technology for the Future 2.0", reg. no. CZ.02.2.69/0.0/0.0/18_058/0010212.



About this file

This file is meant to be a guideline for the lecturer. Many important pieces of information are not in this file, they are to be delivered in the lecture: said, shown or drawn on board. The file is made available with the hope students will easier catch up with lectures they missed.

For study the following resources are better suitable:

- Meyer: Lecture notes and readings for an http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-042j-mathematics-for-computer-science-fall-2005/readings/" (weeks 1-5, 8-10, 12-13), MIT, 2005.
- Diestel: Graph theory http://diestel-graph-theory.com/ (chapters 1-6), Springer, 2010.

See also http://homel.vsb.cz/~kov16/predmety_dm.php

Lecture overview

Chapter Eulerian and hamiltonian graphs

- motivation
- eulerian graphs traversable in "one trail"
- hamiltonian graphs traversable in "one path"

Eulerian graphs

Historically first problem solved by graph theory approach in 1736: Seven bridges of Königsberg – search for a trail uv, such that it contains all edges of a given graph G.

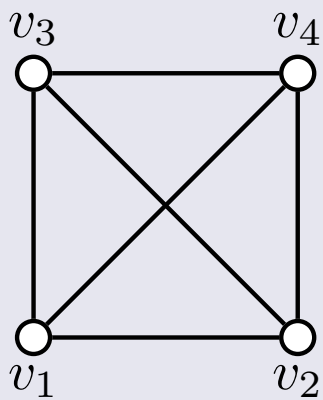
Example

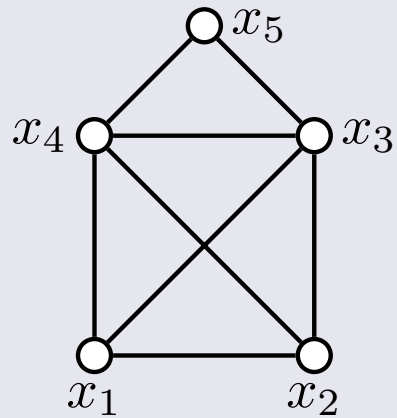
A postman has to deliver mail along each street in his district. Suppose he can traverse each street only once – he travels the shortest distance and delivers mail sooner.

Take a graph representing the district in which streets correspond to edges and junctions to vertices. On optimal solution to the postman problem corresponds to finding a trail that traverses each edge precisely once.

Similarly one can suggest an optimal route for snowplows, garbage cars, etc.

Example





Is it possible to draw the edges of G in one stroke?

Definition

A trail in a graph G which originates and stops in the same vertex is called a closed trail. Moreover, if it contains all edges of a connected graph G, it is a closed eulerian trail. A graph that allows a closed eulerian is called an eulerian graph.

A trail in a connected graph G which originates in one stops in another vertex and contains all edges of G is called an open eulerian trail.

We say that each such graph can be drawn in a single stroke.

Theorem

Graph G can be traversed by a single closed trail, if and only if G is *connected* and all vertices of G are of *even degree*.

Using an elegant argument one can show easily:

Corollary

Graph G can be traversed by a single open trail, if and only if G is *connected* and *precisely two* vertices of G are of *odd degree*.

Eulers' Theorem

Graph G can be traversed by a single closed trail, if and only if G is *connected* and all vertices of G are of *even degree*.

Proof By induction on the number of edges (just a sketch of "$\Leftarrow$").

Basis step:

We can start with the trivial graph. For non-trivial connected graph G is every vertex of degree at least 2. The smallest such graph is $G \simeq C_n$. Graph G contains a closed trail traversing all edges (why?).

Inductive step:

Suppose, that every connected graph with less than $|E|$ edges and with all vertices of even degree contains a closed trail traversing all edges. In G we take any cycle C (each vertex is of degree at least 2). In $G - C$ are vertices of even degree (or isolated vertices). If $G - C$ is not connected, each component contains by induction hypothesis a closed trail traversing all edges of the component.

Now we insert into the closed trail C a closed "sub-trail" at vertex v_i, one in each component. We obtain a closed trail traversing all edges of G.

The claim of "$\Leftarrow$" follows by (strong) mathematical induction. $\square$

Corollary

The edges of a graph G can be drawn in a single (open) stroke if and only if G is *connected* and *precisely two* of its vertices are of *odd degree*.

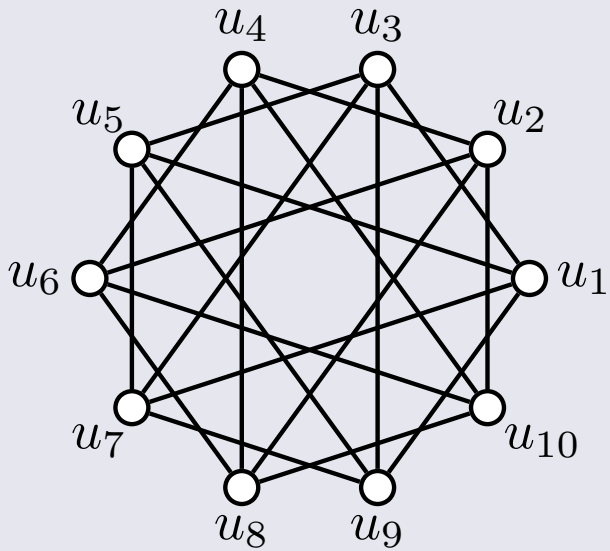
Proof

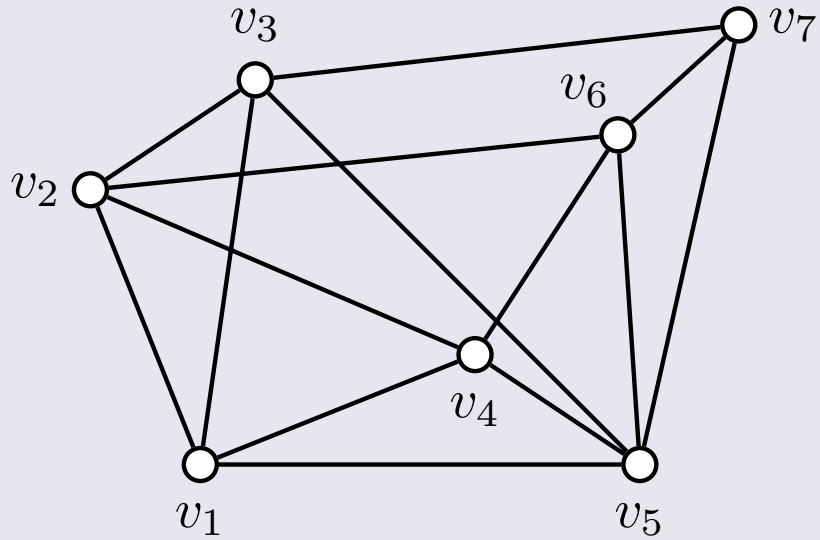
"$\Rightarrow$" Suppose the edges of a graph can be drawn in a single stroke (via an open eulerian trail). Then G is connected and all its vertices are of even degree with the exception of the first and the last vertex of the open eulerian trail.

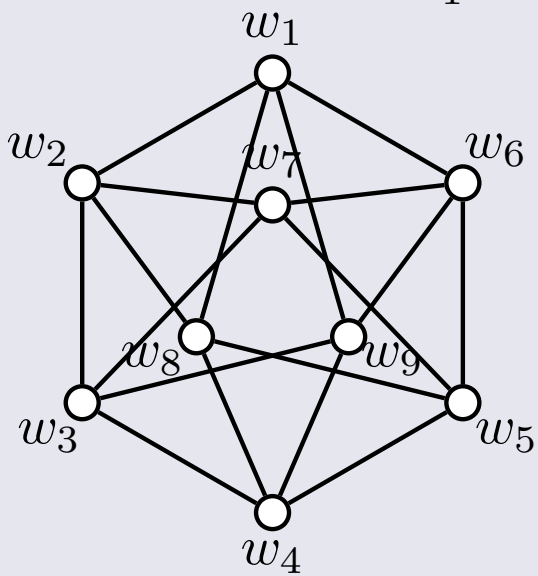
"$\Leftarrow$" Suppose G is connected wit precisely two vertices u and v of odd degree. We can add a new vertex x to G and join it with pair of edges to vertices u and v. We obtain a connected graph G' in which each vertex (u, v and x included) is of even degree.

By Eulers' Theorem there exists a *closed* eulerian trail T' in G'. After removing vertex x and both edges incident to x we obtain an open eulerian trail T from vertex u to v in G. $\square$

Examples







Which of these graphs are eulerian?

Eulerian trail can be used to solve other problem besides traveling.
One nice application of eulerian trails:

Example

The vertices of the state graph (corresponding to some system) represent states which may occur. We join two states by an edge if one state can lead to the other – e.g. Finite Automaton.

When designing a test of the system, we would like to check all states and all possible transitions. An optimal test may run along an eulerian trail.

Hamiltonian graphs

Hamiltonian cycle

(or hamiltonian circuit) in a given graph is a cycle that contains all vertices of G.
A graph for which a hamiltonian cycle exists is a hamiltonian graph.

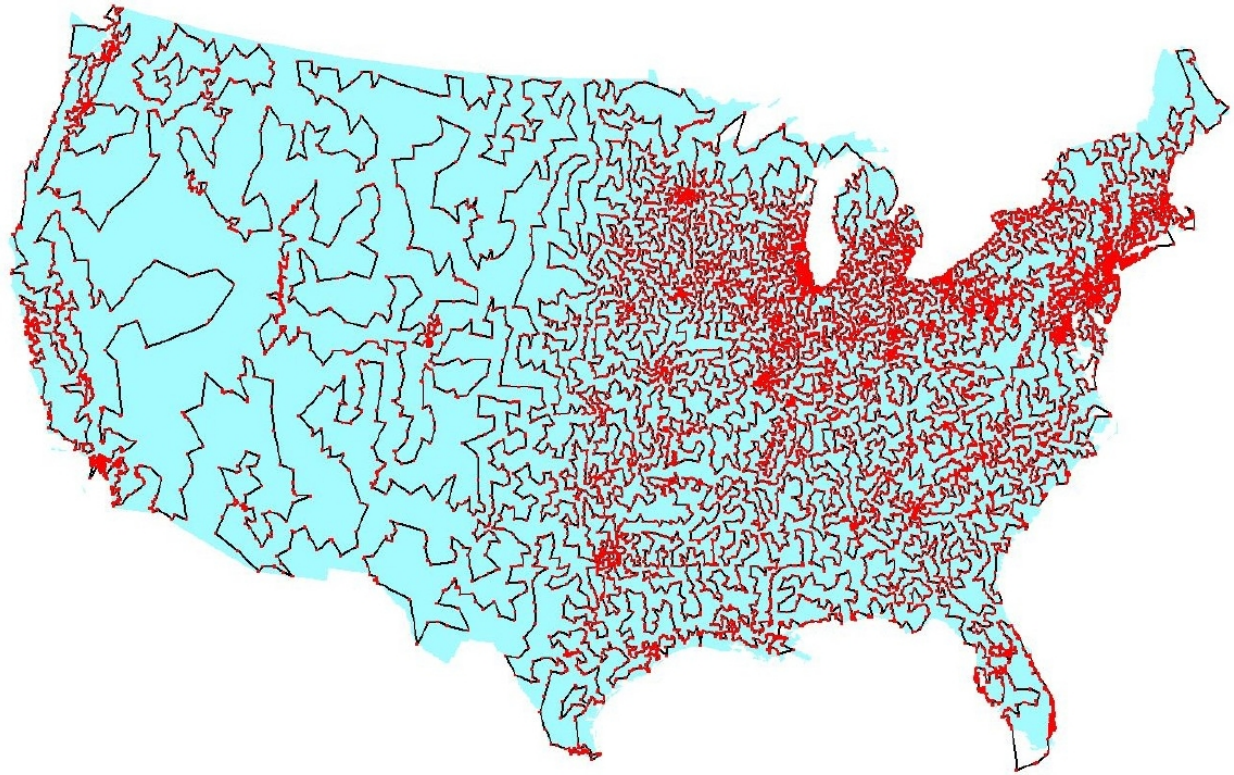
(Hamiltonian cycle visits every vertex of the graph.)

It may seem that constructing hamiltonian cycles is related to constructing eulerian trails. This is not the case!
While there is an easy necessary and sufficient condition of even degrees for the existence of eulerian trails in connected graphs, for the existence of hamiltonian cycles no such easy condition is known (some think it may not exists).

Corollary: it is not easy to decide whether a graph is hamiltonian or not.

Example

The traveling salesman problem is a well known motivation. The salesman wants to visit each city in his region, return to the starting city and travel the shortest possible distance during his travel.
Simplified version: can he visit every city wit at least 500 citizens precisely once and return back?



Optimal solution to the travelling salesman problem for 13 509 cities in the USA

Example

A postman in a village delivers mail each day only to some of the houses. Rather than traverse each street, he has to visit each address (house) to which a letter has to be delivered.

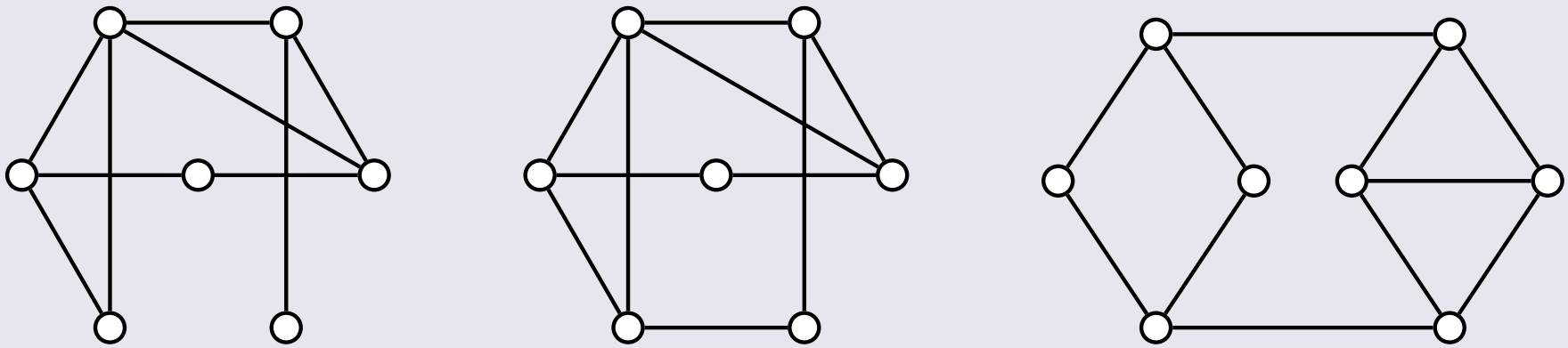
Example

In a warehouse when goods are to be deposited or picked up, the forklift (or pallet) has to visit each of the spot where a certain good is stored in pallets. The forklift has to visit all selected locations and travel the shortest distance possible.

All the problem mentioned above can be formulated as finding a hamiltonian cycle in a graph, or a shortest hamiltonian path, respectively.

Examples

Which of the following graph are/are not hamiltonian?



Hamiltonian an non-hamiltonian graphs.

Examples

More problem leading to hamiltonian cycles

- family travel plan for visiting several places of interest
- theater or circus tour through the country
- Hamilton game

Theorem (Dirac)

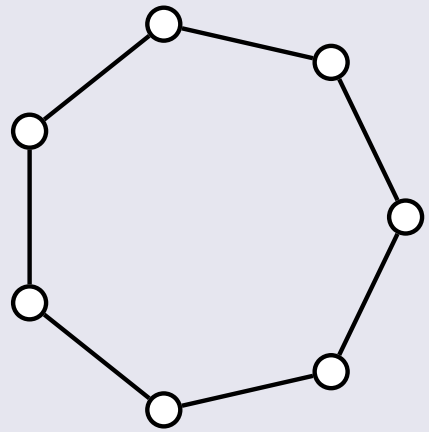
Let G be a graph on n vertices, where $n \geq 3$.
If the smallest degree is at least $n/2$, then G is hamiltonian.

Proof In another course "Teorie Grafů I".

Notice, the statement has the form of an implication, not an equivalence. Thus, each graph in which the smallest degree high enough is hamiltonian, but not each hamiltonian graph has to have a large small degree.

Example

A hamiltonian graph does not have to have "many" edges.



Cycle C_7.

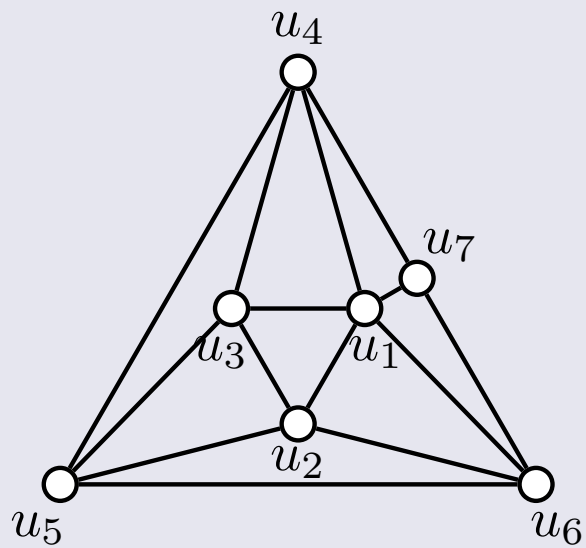
Theorem (Ore)

Let G be a graph on n vertices, where $n \geq 3$.
If for each independent (nonadjacent) vertices u and v in a graph G is $\deg(u) + \deg(v) \geq n$, then G is hamiltonian.

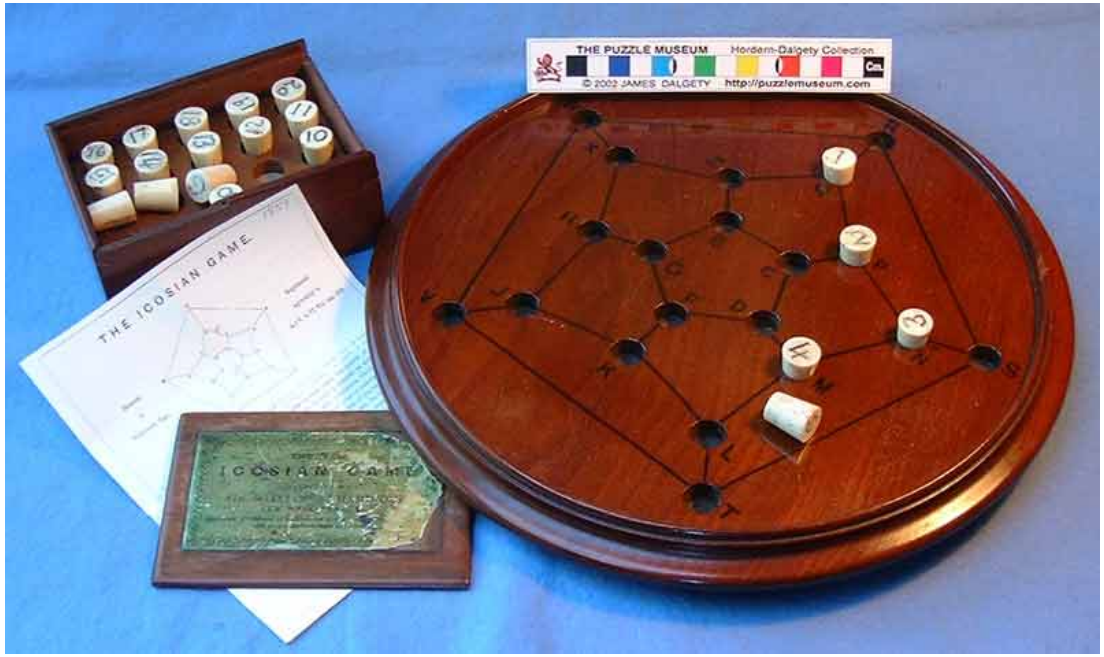
Diracs' Theorem is a special case of Ores' Theorem.

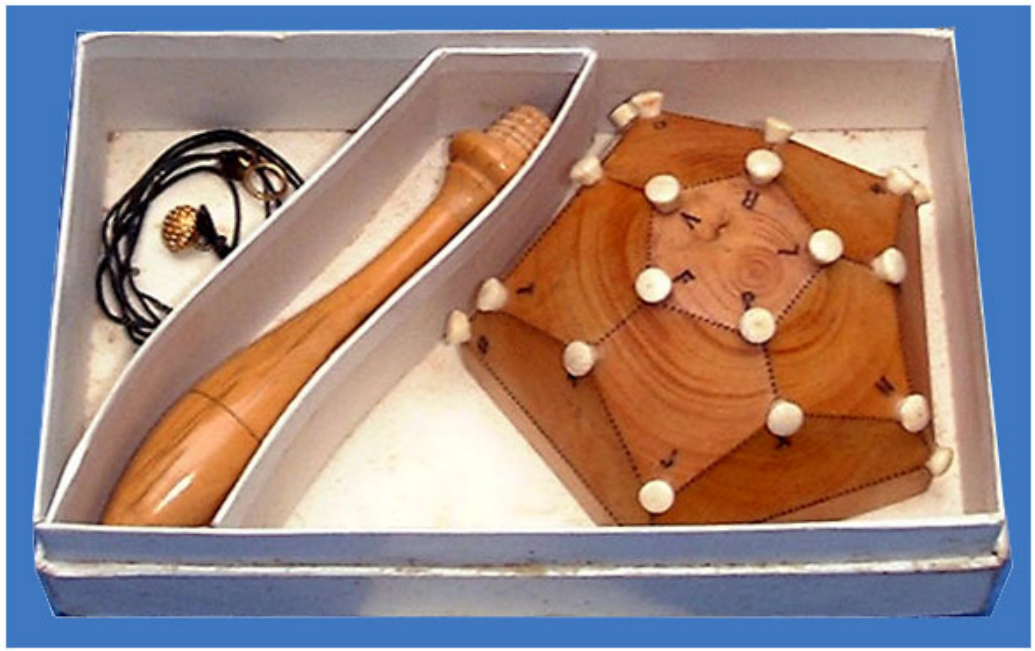
Example

Is this graph hamiltonian?



Why are the graphs called “hamiltonian”?





Next lecture

Chapter Distance and measuring in graphs

- motivation
- distance in graphs
- measuring in graphs
- weighted distance
- shortest path algorithm

Discrete mathematics

Petr Kovář
petr.kovar@vsb.cz

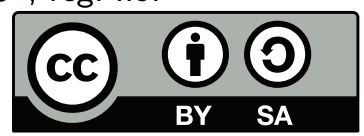
VŠB – Technical University of Ostrava

Winter term 2021/2022
DiM 470-2301/02, 470-2301/04, 470-2301/06

EUROPEAN UNION
European Structural and Investment Funds
Operational Programme Research,
Development and Education

MINISTRY OF EDUCATION,
YOUTH AND SPORTS

The translation was co-financed by the European Union and the Ministry of Education, Youth and Sports from the Operational Programme Research, Development and Education, project "Technology for the Future 2.0", reg. no. CZ.02.2.69/0.0/0.0/18_058/0010212.



About this file

This file is meant to be a guideline for the lecturer. Many important pieces of information are not in this file, they are to be delivered in the lecture: said, shown or drawn on board. The file is made available with the hope students will easier catch up with lectures they missed.

For study the following resources are better suitable:

- Meyer: Lecture notes and readings for an http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-042j-mathematics-for-computer-science-fall-2005/readings/" (weeks 1-5, 8-10, 12-13), MIT, 2005.
- Diestel: Graph theory http://diestel-graph-theory.com/ (chapters 1-6), Springer, 2010.

See also http://homel.vsb.cz/~kov16/predmety_dm.php

Lecture overview

Chapter Distance and measuring in graphs

- motivation
- distance in graphs
- measuring in graphs
- weighted distance
- shortest path algorithm

Motivation

In many real life applications of graphs we need to "measure" distances.

In a graph representing a road network it is natural to ask

"How far is it from vertex (place) u to vertex (place) v?"

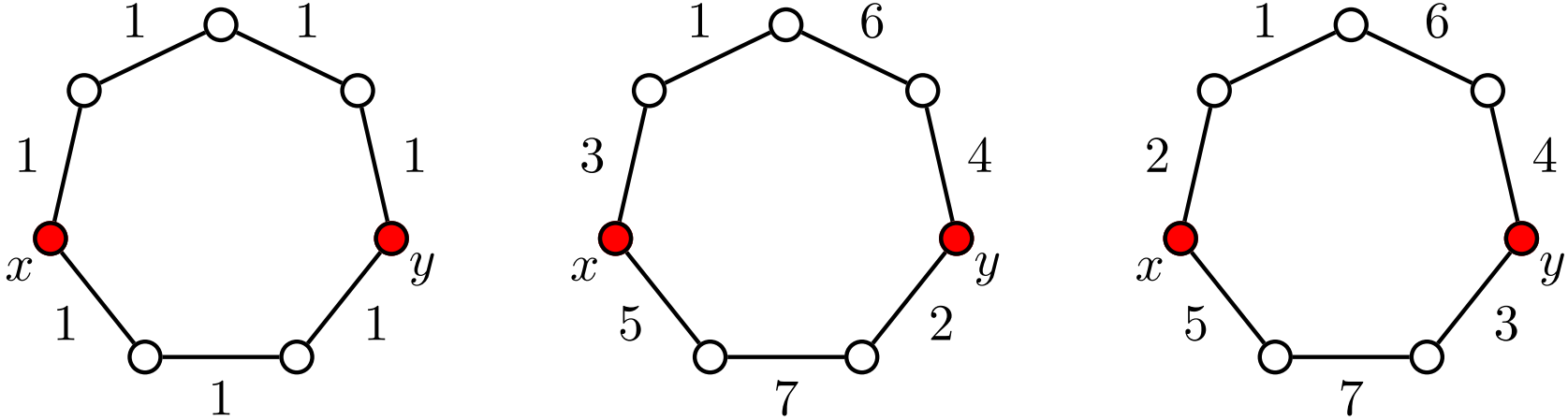
or

"How long does it tak to travel from vertex u to vertex v?"

The distance will not be just mere *number of edges* (number of roads traveled) but important will be their *length*. Notice that length have not been considered yet.

We will introduce the notion of labeling edges. The meaning of the labels may vary: length, width, capacity, color, ...

Usually, for labels one can use natural numbers only (well chosen scale).



Different distances between vertices u and v in graph C_7.

In the graph on the left

the distance between vertices x and y is $3 =$ number of edges of the shorter path (walk).

In the graph in the middle

the distance between vertices x and y is $14 = 3 + 1 + 6 + 4 = 5 + 7 + 2$.

In the graph on the right

the distance between vertices x and y is $13 = 2 + 1 + 6 + 4$.

Distance in graphs

First for unlabeled graphs, i.e. each edge has length 1.

Length of a walk is the number of edges in the sequence of vertices and edges in a a walk

$$v_0, e_1, v_1, e_2, v_2, \ldots, e_n, v_n,$$

where each edge e_i has end-vertices v_{i-1} and v_i.

Definition

Distance $\mathrm{dist}_G(u, v)$ between vertices u and v in a graph G is given by the length of the shortest walk between u and v in G. If no walk between u and v exists, we define the length to be $\mathrm{dist}_G(u, v) = \infty$.

Notice, that

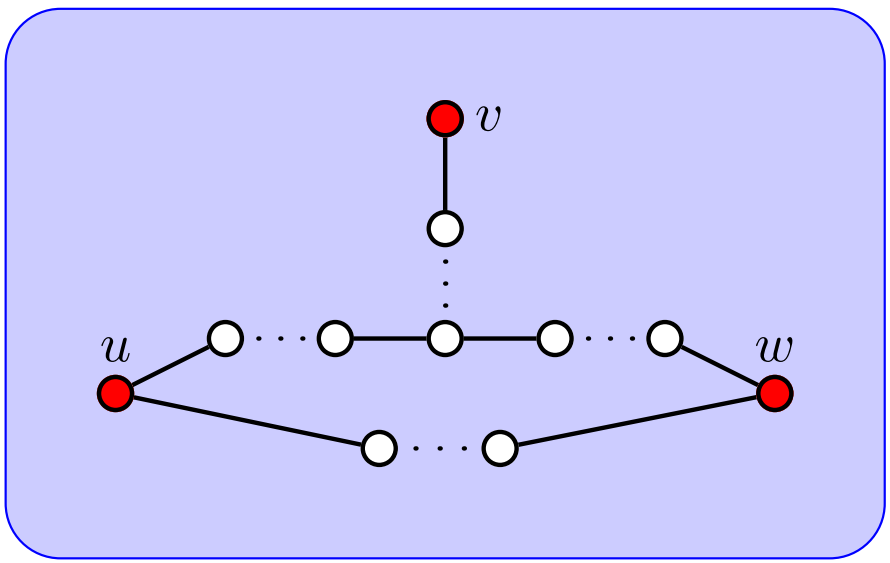
- the shortest walk (with the fewest edges) is always a path
- in unoriented graphs is $\mathrm{dist}_G(u, v) = \mathrm{dist}_G(v, u)$
- $\mathrm{dist}_G(u, u) = 0$
- if $\mathrm{dist}_G(u, v) = 1$, then edge $uv \in E(G)$

Lemma

Distance in a graph G satisfies the *triangle inequality*:

$$\forall u, v, w \in V(G)\colon\ \mathrm{dist}_G(u, w) \leq \mathrm{dist}_G(u, v) + \mathrm{dist}_G(v, w).$$

Proof The inequality follows from the observation, that the walk of length $\mathrm{dist}_G(u, v)$ between u, v joined with the walk of length $\mathrm{dist}_G(v, w)$ between v, w gives a walk of length $\mathrm{dist}_G(u, v) + \mathrm{dist}_G(v, w)$ between u, w. Never $\mathrm{dist}_G(u, w) > \mathrm{dist}_G(u, v) + \mathrm{dist}_G(v, w)$. Yet, a shorter walk from u to v can exist $\mathrm{dist}_G(u, w) \leq \mathrm{dist}_G(u, v) + \mathrm{dist}_G(v, w)$. □



Two walks u, v and v, w; a shorter walk between u, w.

Measuring in graphs (graph metrics)

When measuring distances one cannot simply choose among all possible paths.

Example

What is the number of all paths between u, v in a complete graph K_n.

1. If $u = v$, then there exists only one (trivial) path from u to v.
2. Of $u \neq v$ there exist $V(n-2,k) = \frac{(n-2)!}{(n-2-k)!}$ different paths from u to v with k internal vertices, $0 \leq k \leq n-2$.

The total number of different uv paths is $\sum_{k=0}^{n-2} \frac{(n-2)!}{(n-2-k)!}$.

There are $O((n-2)!)$ different paths ... too many possibilities.

There is a simple modification of the breadth-first search algorithm (depository implemented as a queue Q).
We determine lengths of the shortest paths form a given vertex to every other vertex.

Each newly found vertex w will be assigned the distance by one greater than the processed vertex v.
Distances are stored an a one-dimensional array `dist[]`.

Algorithm: Distances from a given vertex

```
// on the input is the graph G
input < graph G;
status(all vertices of G) = initial;
queue Q = a given vertex u of G;
status(u) = found;
dist(u) = 0;                                  // distance of u
```

Algorithm: Distances from a given vertex (continued)

```
// processing a selected component of G
while (Q is not empty) {
   pick a vertex v from the queue Q; Q = Q - v;
   for (edges e incident with v)        // for all edges
      w = other end-vertex of e = vw;   // known neighbor?
      if (status(w) == initial) {
         status(w) = found;
         add vertex w to queue: Q = Q + w;
         dist[w] = dist[v]+1;           // distance of w
      }
   }
   status(v) = processed;
}
// vertices in additional components are unreachable!
while (there are unprocessed vertices w in G) {
   dist[w] = MAX_INT;                  // infinity
   status(w) = processed;
}
```

Notice:

- The number of steps depends on the number of vertices and edges of the given graph.
  Complexity is $O(n+m)$, where n is the number of vertices and m is the number of edges.
  After the line `dist[w] = dist[v]+1;`
  add the line `pre[w] = v;`
- If we store for every vertex its *predecessor* on the shortest path, we can reconstruct the path:
  - the last vertex is w,
  - the next-to-the-last vertex is $pre[w]$,
  - the the next-to-the-next-to-the-last vertex is $pre[pre[w]]$,
  - ...
  - first (i.e. starting) vertex is $pre[\ldots pre[pre[w]]] = u$.

We assumed that vertices closer to u are processed before more distant vertices.
This can be proven and used to prove the validity of the algorithm.

Lemma

Let u, v, w be vertices of a connected graph G such, that $\mathrm{dist}_G(u, v) < \mathrm{dist}_G(u, w)$. In a *breadth-first* search in G starting at the vertex u the vertex v will always be found before the vertex w.

Proof By induction on $\mathrm{dist}_G(u, v)$.
Basis step: For $\mathrm{dist}_G(u, v) = 0$, i.e. $u = v$ the claim is obvious – the vertex u is found first.
Inductive step: Now for some $\mathrm{dist}_G(u, v) = d > 0$ we denote by v' the neighbor of v on the shortest walk u, v to u, obviously $d_G(u, v') = d - 1$. Similarly, by w' we denote the neighbor of w on the walk u, w to u, thus $\mathrm{dist}_G(u, v') < \mathrm{dist}_G(u, w')$.
By the induction hypothesis the vertex v' will be in a breadth-first search found before w'. This implies also, that v' will come to the queue of the depository before w', and thus the neighbors of v' (v is among them) will be found before the neighbors of w'. □

Corollary

The basic algorithm for breadth-first search can be used to count distances from the vertex u to all other vertices.

Proof is in the textbook.

Questions

Why the depth-first search cannot be used instead the breadth-first search? Which part of the algorithm would fail?

Evaluating the metrics

By a metrics we understand the distance between any pair of vertices in a given graph. We expect the metrics to satisfy "common properties".

Formally: the set of vertices along with the distance function for every pair of vertices forms a metric space.

Definition

Metrics ρ on a given set A is such a mapping $\rho : A \times A \to \mathbb{R}$, that $\forall x, y \in A$ the following holds

1. $\rho(x, y) \geq 0$ while $\rho(x, y) = 0$ only for $x = y$,
2. $\rho(x, y) = \rho(y, x)$,
3. $\rho(x, y) + \rho(y, z) \geq \rho(x, z)$.

Informally: The metrics in G is a matrix (two-dimensional field) `d[][]`, where `d[i][j]` gives the distance between vertices i and j (vertices are 0, 1, ..., $|V(G)| - 1$).

To find the metrics we can use the algorithm for measuring distances from a given starting vertex (repeat it for every starting vertex u).
There is a simpler algorithm:

Method: Counting the metric by joining paths

We denote the vertices of a graph by $0, 1, 2, \ldots, N-1$.

- Let `d[i][j]` equal 1 (optionaly to the length of edge ij), or ∞ if edge ij is not in the graph.
- After each iteration $t \geq 0$ contains `d[i][j]` the length of the shortest path between i, j which passes only through vertices in $\{0, 1, 2, \ldots, t\}$.
- During each iteration t we may modify the distance between every pair of vertices, there are two options:
  1) we find a shorter way through the newly added vertex t; we replace `d[i][j]` by a shorter length `d[i][t] + d[t][j]`, or
  2) adding the vertex t does not help to find a way shorter than `d[i][j]` obtained in the previous steps; then `d[i][j]` remains unaltered.

Floyd's Algorithm – shortest paths

```
input: adjacency matrix G[][] of a graph with N vertices,
       where G[i][j]=1 for edge ij and
             G[i][j]=0 otherwise;

// initialization (value MAX_INT/2 stands for "infinity")
for (i=0; i<N; i++)
     for (j=0; j<N; j++)
          d[i][j] = (i==j ? 0 : (G[i][j] ? 1 : MAX_INT/2));

// loop for every vertex t, index from [0,N-1]
for (t=0; t<N; t++)
   // traverse all pairs of vertices
     for (i=0; i<N; i++)
          for (j=0; j<N; j++)
          // is there a shorter path through t?
               d[i][j] = min(d[i][j], d[i][t]+d[t][j]);
```

In the computer we implement ∞ by a large constant, i.e. MAX_INT/2.

Advantages:

- easy implementation
- finds the distance between every pair of vertices

Disadvantages:

- even when searching only the distance of two vertices, we *have to* find the distance of every pair of vertices
- complexity of $O(n^3)$, where n is the number of vertices
- doesn't provide shortest paths, just distances
  (can't reconstruct the path based on the result only)

Weighted distance

We assign numbers to edges: length, width, capacity, color, ...

Definition

Labeling of a given graph G is a mapping $w : E(G) \to \mathbb{R}$, which assigns a real number $w(e)$ (called edge weight/label) to every edge of G. Weighted (or labeled) graph is a graph G along with a labeling.
In a positively weighted (labeled) graph G are all weights $w(e)$ positive ($w(e) > 0$ pro $\forall e \in E(G)$).

Edge weight – more commonly "labels".

In real life applications:

- labels are usually non-negative,
- we can use integers only when choosing a suitable scale (units).

Positively weighted (labeled) graph is a special case of a labeled graph.

Now we introduce distances in weighted graphs.

Definition

Let G be a weighted graph G with labeling w.
The length of a weighted walk $S = v_0, e_1, v_1, e_2, v_2, \ldots, e_n, v_n$ in G is the sum

$$d_G^w(S) = w(e_1) + w(e_2) + \cdots + w(e_n)$$

(each edge is counted as may times as it appears in the walk S).
(Weighted) distance between two vertices u, v in a weighted (positively labeled) graph (G, w) is

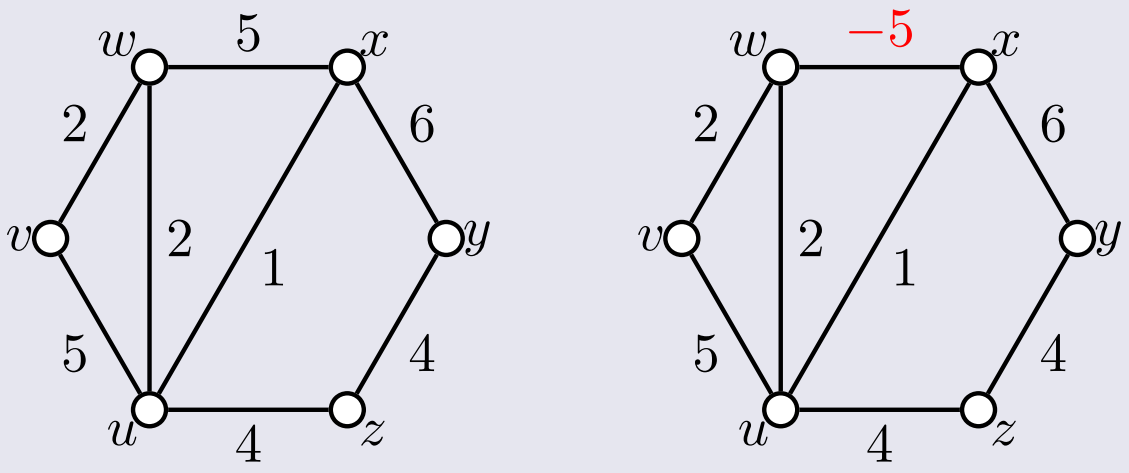
$$\mathrm{dist}_G^w(u, v) = \min\{d_G^w(S), \text{where } S \text{ is a } \textbf{path} \text{ between } u \text{ and } v\}.$$

If vertices u and v are unreachable, we set $\mathrm{dist}_G^w(u, v) = \infty$.

Lemma

Weighted distance in positively weighted graphs satisfies the *triangle inequality* $\quad \forall u, v, w \in V(G)\colon\ \mathrm{dist}_G^w(u, w) \leq \mathrm{dist}_G^w(u, v) + \mathrm{dist}_G^w(v, w)$.

Example



Two different labelings of G.

Questions

- What is the distance between v and y in the graph on the left? 13? 12? 11? 10?
- What is the distance between w and z?
  We do not allow negative weights, since then no shortest walk has to exists.
- What is the distance between v and y in the graph on the right? 3?, 0?, -1?, 10? $-n$?

Shortest path algorithm

For finding a shortest (weighted) path between two vertices of a positively weighted graph Dijkstra's algorithm is used.

- more complex than the algorithm above
- is *significantly faster*; finds the distance from a particular vertex to all other vertices, not between all pairs of vertices

Dijkstra's algorithm is used while searching connections in on-line search engines.

Dijkstra's algorithm

- is a modification of the breadth-first search algorithm – for each vertex v found we store the value of *distance* (length of the shortest u, v-path) from the vertex u, as well as the last vertex on this path.
- From the depository we always pick the vertex v *with the smallest distance* from u (no shorter u, v-path exists).
- After the search we have the distance form u to all vertices of the graph.

Dijkstra's algorithm (initialization)

Finds the shortest path between u and v of a positively weighted graph G (given by the incidence matrix).

```
input: graph on N vertices, in an incidence matrix neig[][]
       and w[][], where neig[i][0], ..., neig[i][deg[i]-1]}
       are neighbors of vertex i with degree deg[i] and edge
       from i to neig[i][k] has length w[i][k] > 0;
input: u,v, we search path from u to v;
// state[i] stores the state of vertex i:
//    0 ... initial
//    1 ... processed
// dist[i] gives the shotest (so far) distance to i
// pre[i]  contains the predecessor of i

// initialization
for (i=0; i<=N; i++)   // MAX_INT also to dist[N]!
     { dist[i] = MAX_INT; state[i] = initial; }
dist[u] = 0;
```

Dijkstra's algorithm (continued)

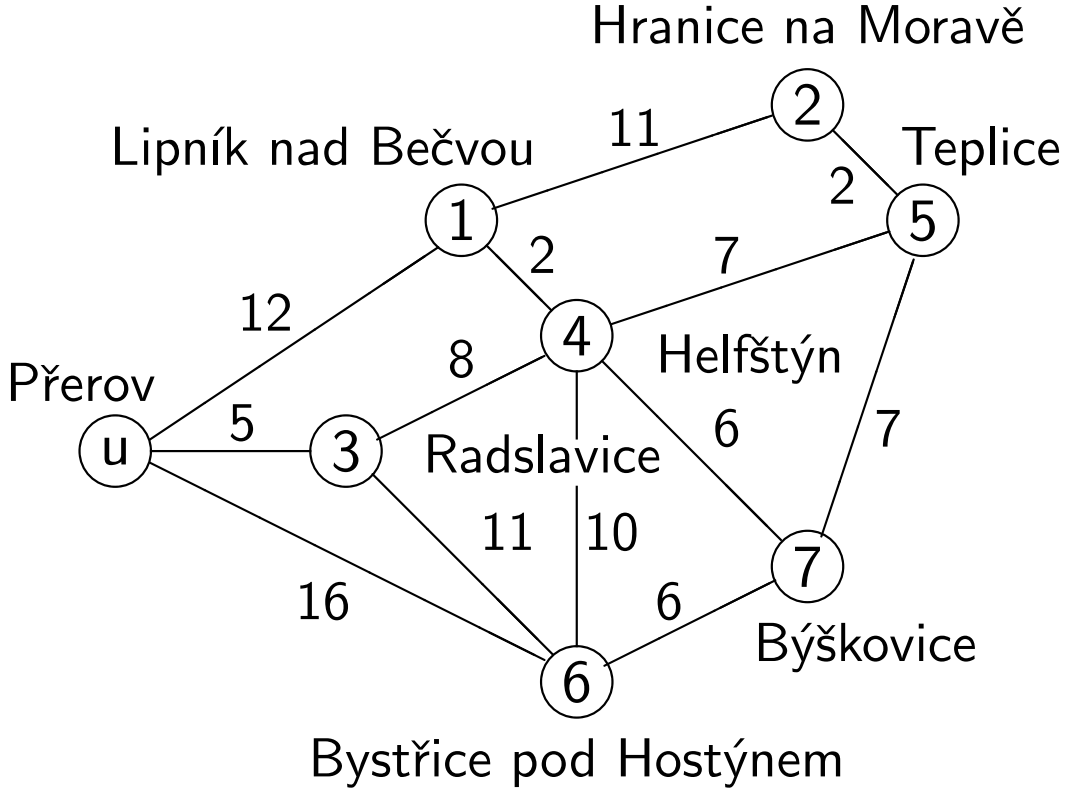
```
while (state[v] == initial) {
   for (i=0, j=N; i<N; i++)      // dist[N] = MAX_INT
      if (state[i] == initial && dist[i] < dist[j])
         j = i;
   // we have the closest unprocessed vertex j
   // process it
   if (dist[j] == MAX_INT) return NO_PATH;
   state[j] = processed;
   for (k=0; k<deg[j]; k++)
      if (dist[j]+w[j][k] < dist[neig[j][k]]) {
         dist[neig[j][k]] = dist[j]+w[j][k];
         pre[neig[j][k]] = j;
      }
      // field pre[] containfs information about
      // predecessors on the shortest path
}
output: Path of length dist[v] stored recursively in pre[];
```

Notes to Dijkstra's algorithm

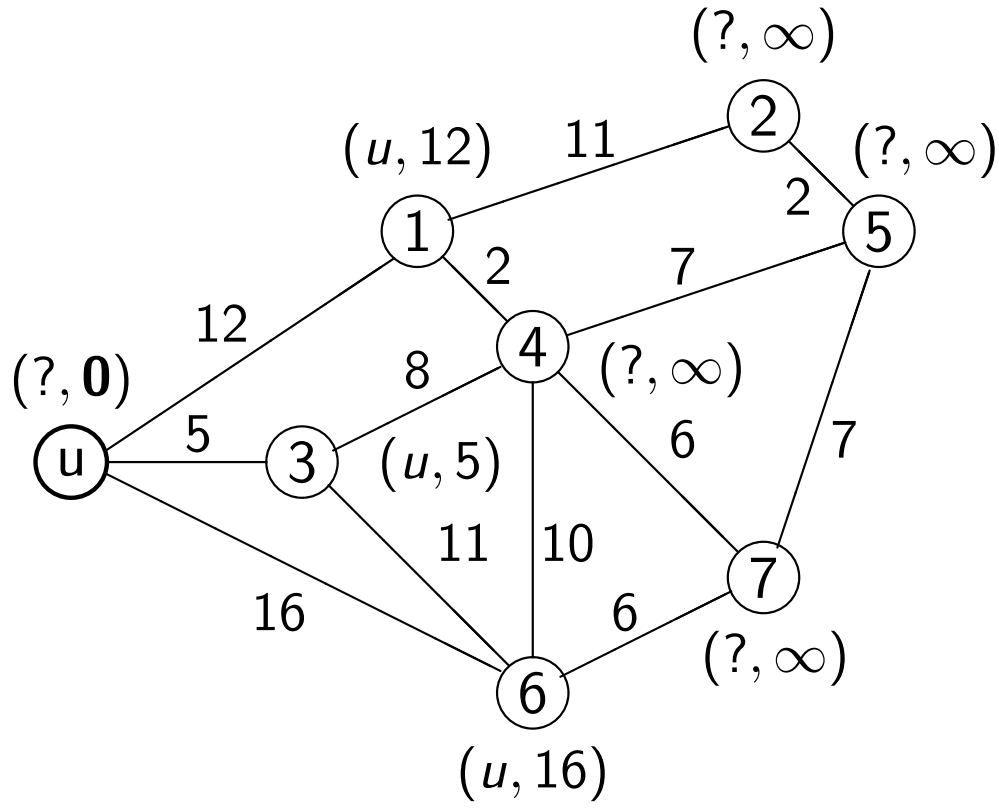
- Running the loop not with the condition `state[v] == initial`, but until all vertices are processed, the algorithm gives the shortest path and its length from u to all vertices. This information is stored in `dist[]` and `pre[]`.
- The total number of steps in Dijkstra's Algorithm for finding the shortest path from `u` to `v` is approximately N^2, where N in the number of vertices.
- Implementing the depository in a convenient way (e.g. heap with the distance as a key) an even faster implementation can be achieved on sparse graphs – running time is approximately the number of edges.
- Algorithm works also for *oriented graphs*.
- We can modify it easily also for *widest road*.

An example follows...

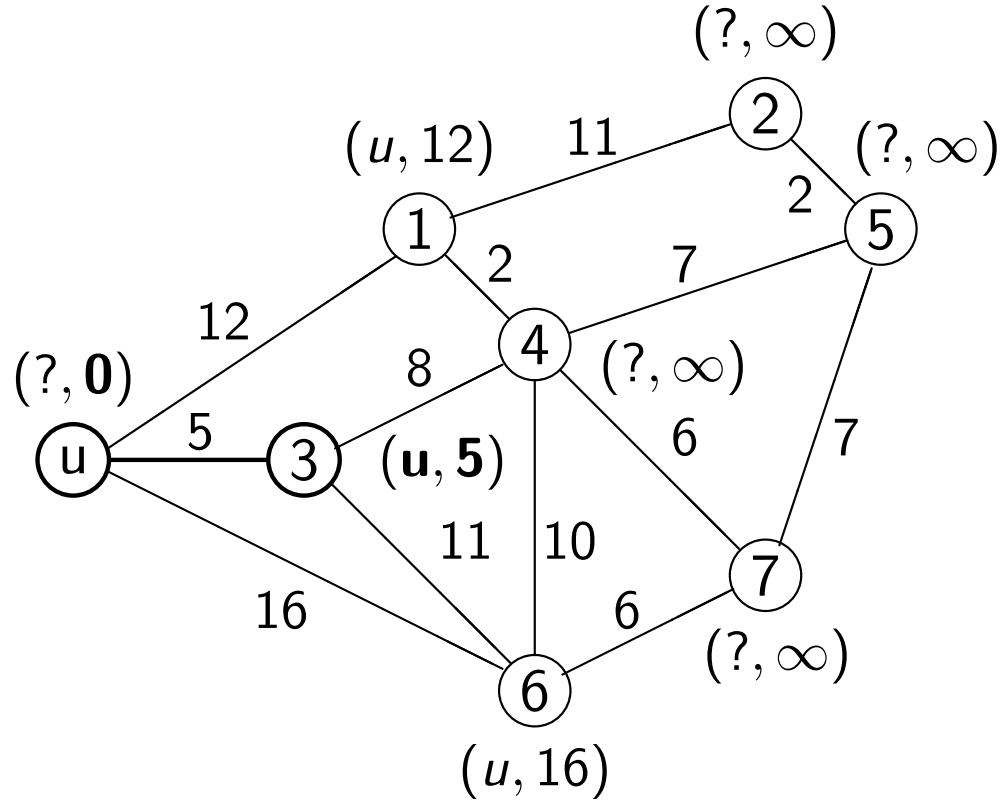
Take the road map close to Přerov. We search for distance from Přerov to all other places.



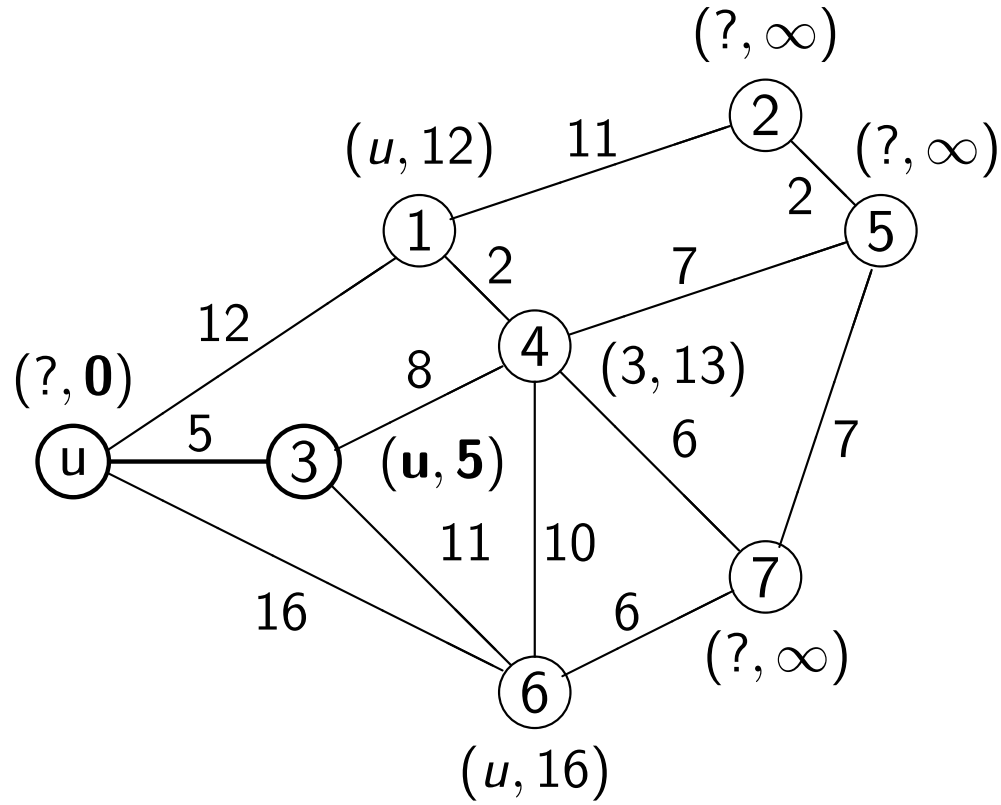
This is a graph representation of the road map. Edges in the graph are labeled by distances in kilometers. Vertices represent cities and roads are depicted by edged joining the corresponding vertices. Vertex i will be labeled by $(pre[i], dist[i])$.



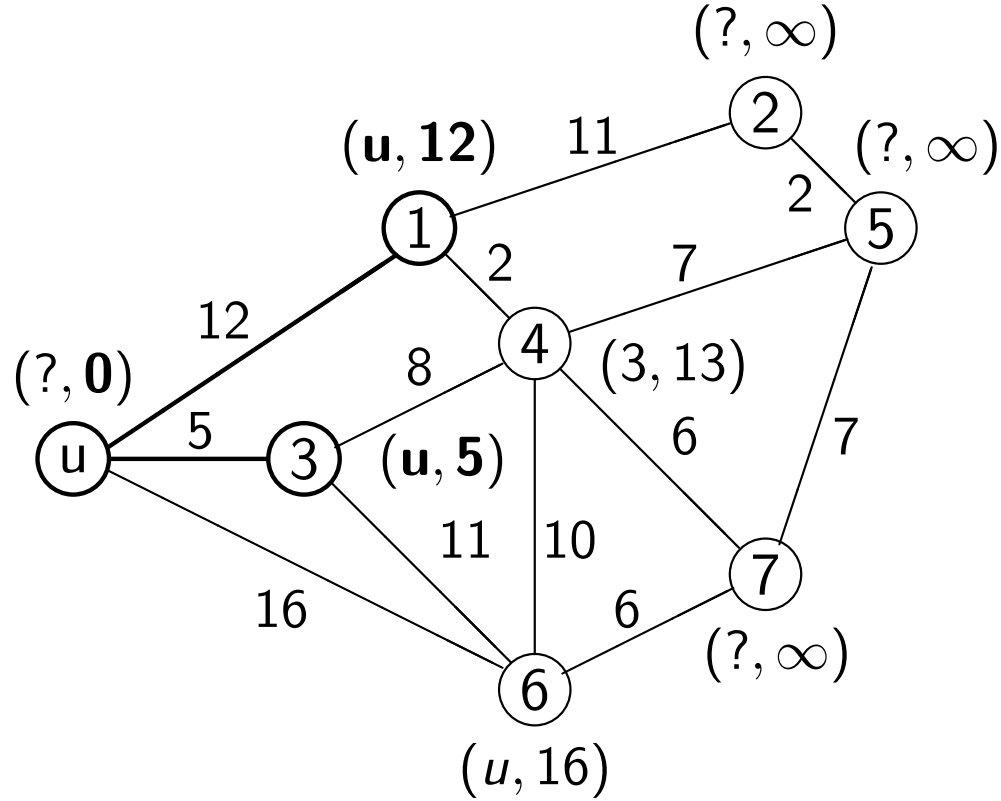
In the initial step of Dijkstra's algorithm each vertex will be in the state 0 (initial state). Only the starting vertex u will have distance 0, i.e. labeled by $(0, 0)$. All remaining vertices are labeled by $(?, \infty)$.
In the first step all vertices j, adjacent to u will be labeled by $(s, w[s][j])$.



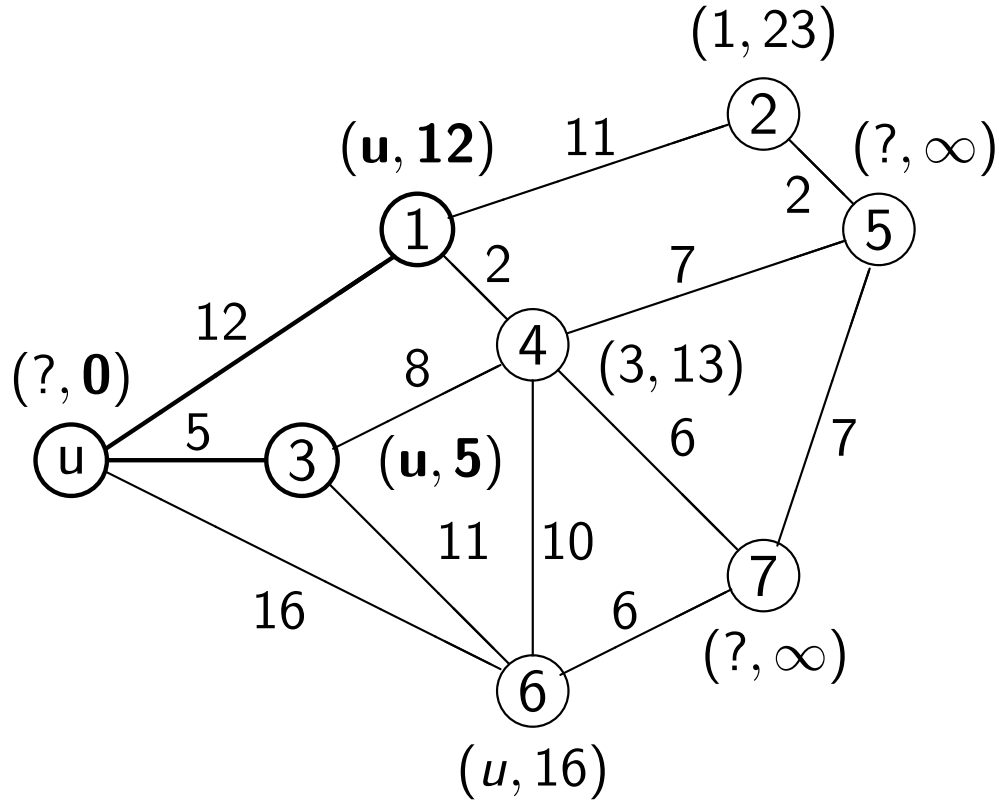
Next we choose the vertex j, which has from u the distance. This is the vertex 3.



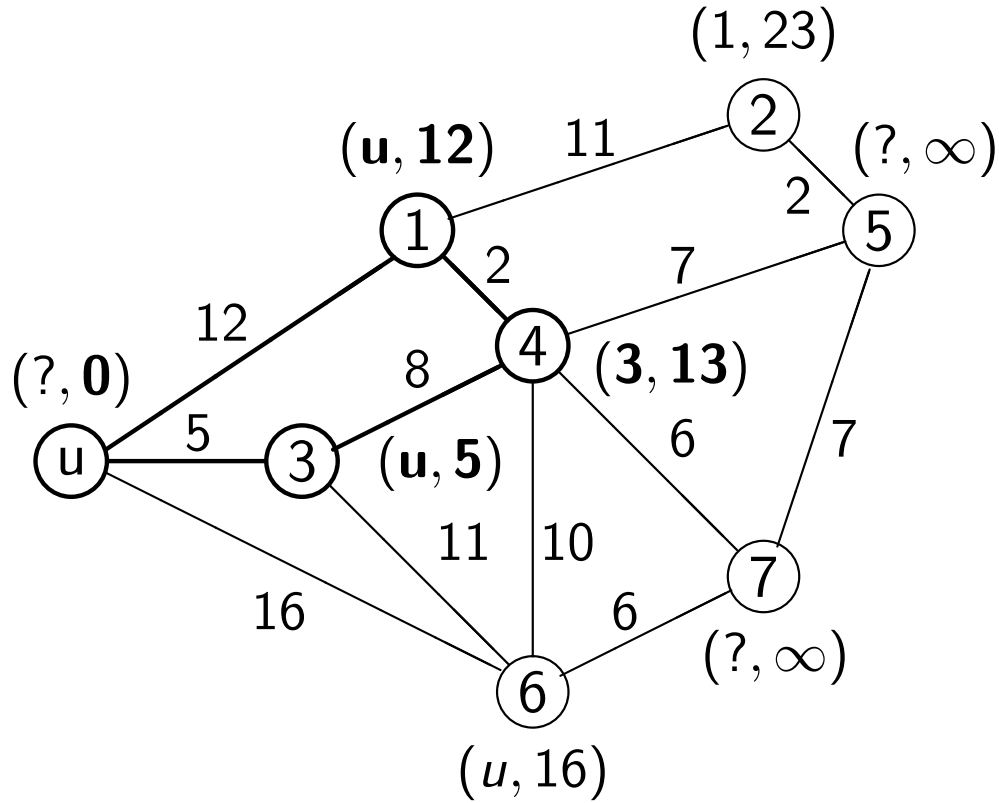
In the next step we modify the label of neighbors of 3 (the closest unprocessed vertex).
We modify the label of vertex 4. The new label of vertex 4 will be $(3, 13)$. The label of vertex 6 will not be changed. The vertex u is also adjacent to 3, but it is processed and its label will change no more.



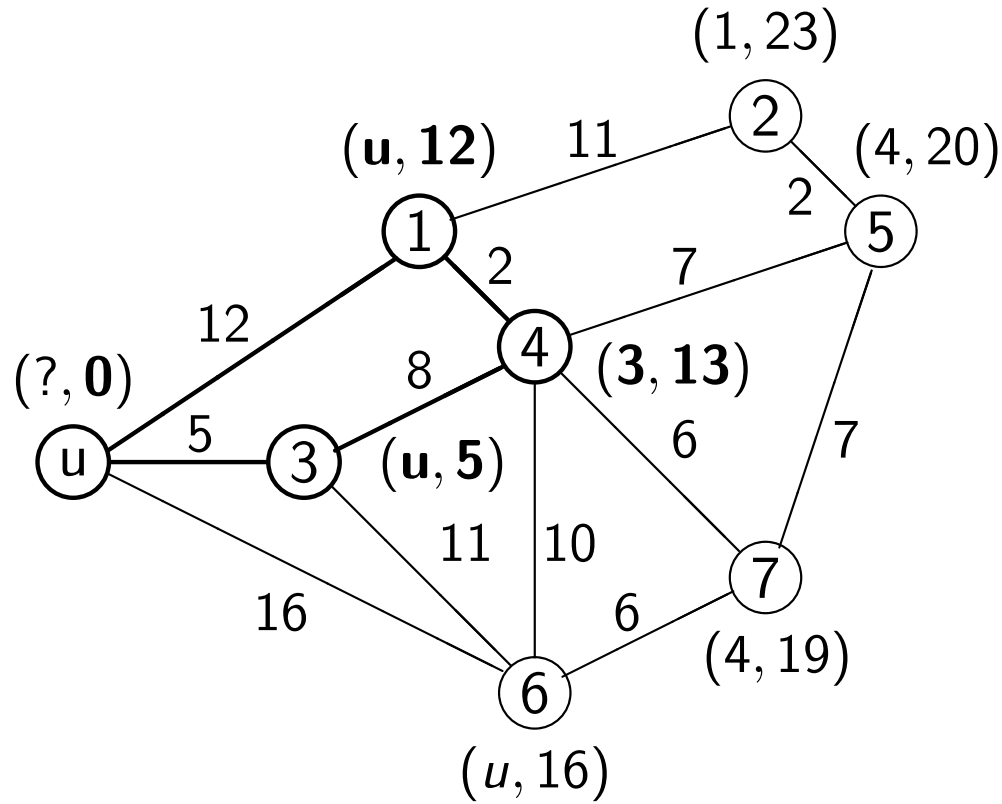
Next we pick the vertex j, with the closest distance from u.
This is the vertex 1 ($dist[1] = 12$).



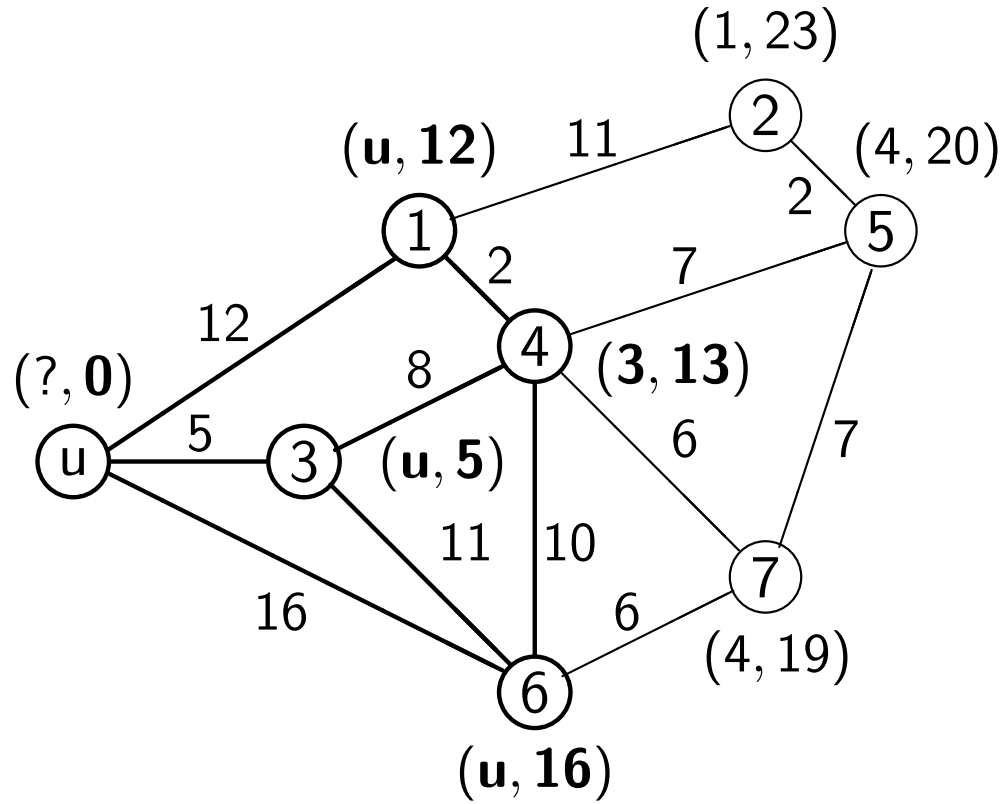
Now vertex 2 will be labeled $(1, 23)$, since $\infty > dist[1] + w[1][2]$. But the label of 4 will not be changed.



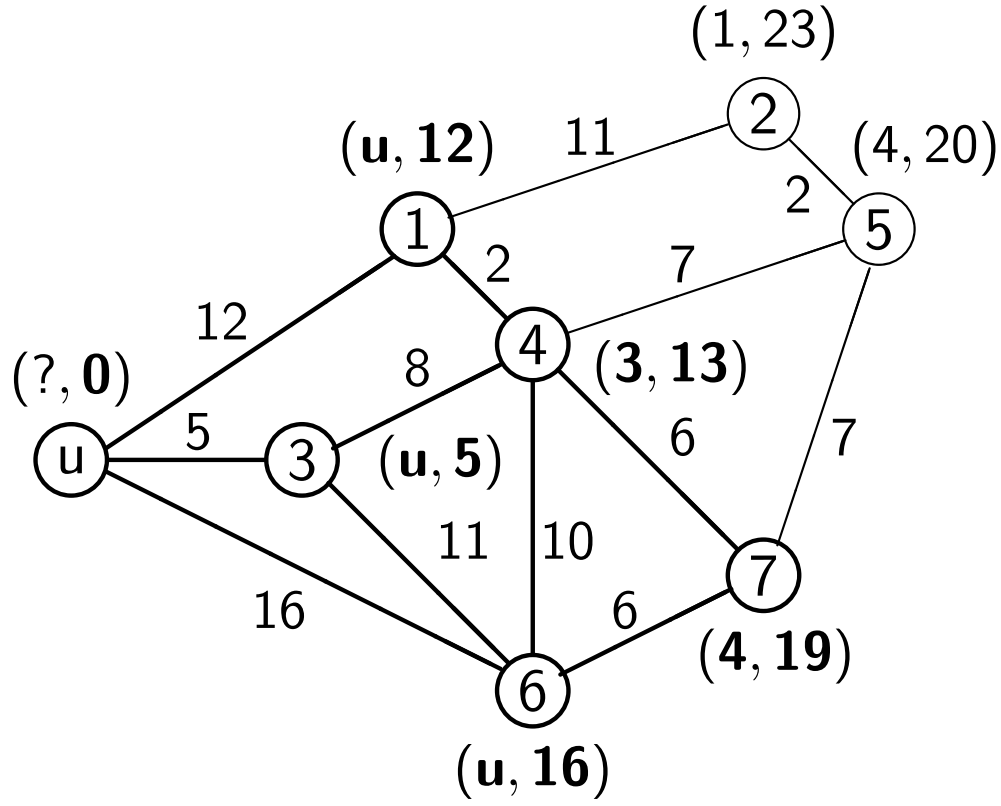
Vertex 4 is the closest to u, it will be processed next.



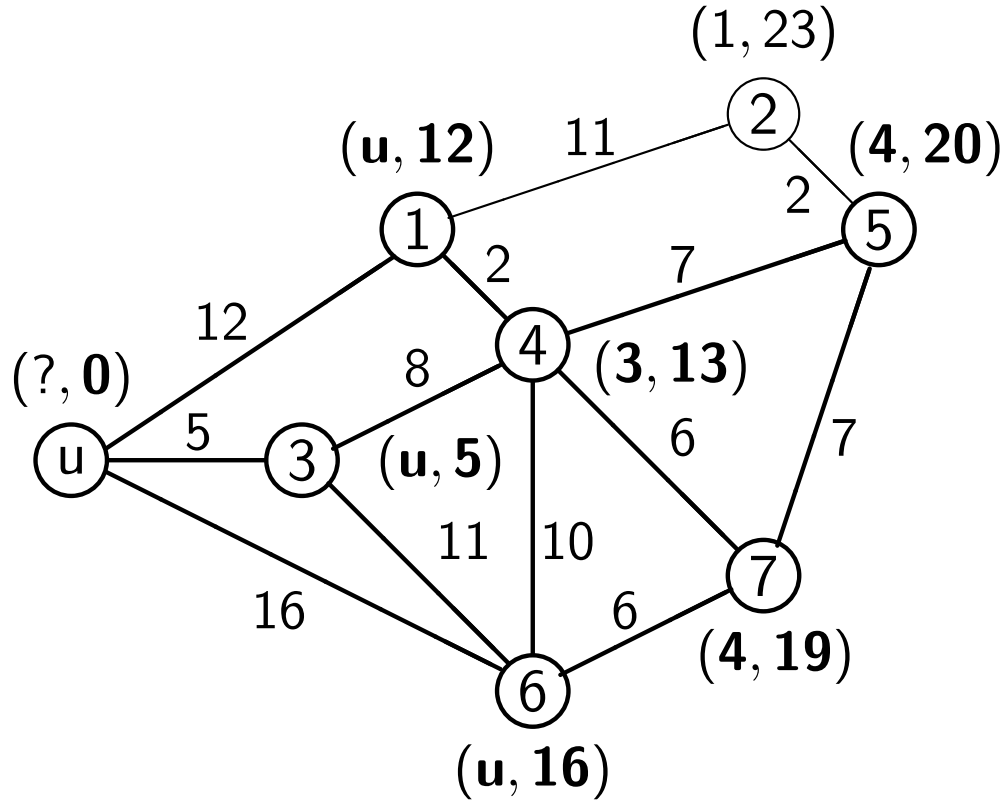
The unprocessed neighbors of vertex 4 are 5, 6 and 7. Since $dist[5] > dist[4] + w[4][5]$ $(\infty > 13 + 7)$, we label vertex 5 by $(4, 20)$. The label of vertex 6 will not change. The vertex 7 will be labeled by a new label $(4, 13 + 6)$.



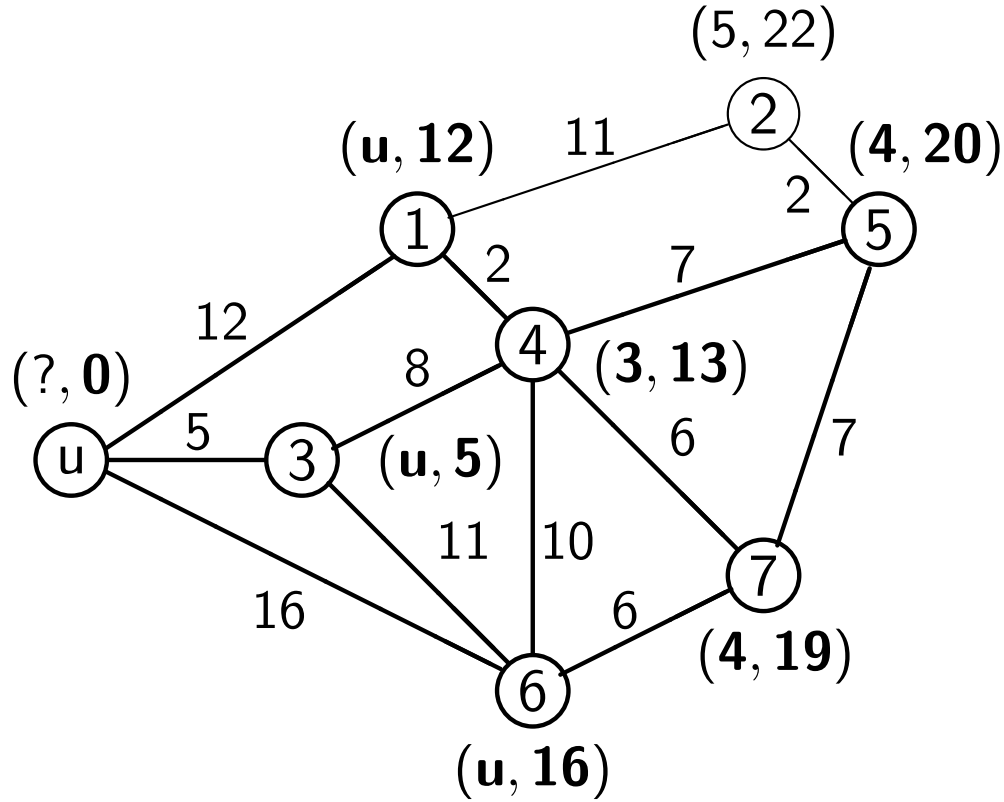
Closest to vertex u is now the vertex 6. The remaining unprocessed vertices 2, 5 and 7 have a higher $dist[i]$.
We will not modify any label, no distance can be improved!
Note: If there are more vertices with the same distance, we choose one arbitrarily.



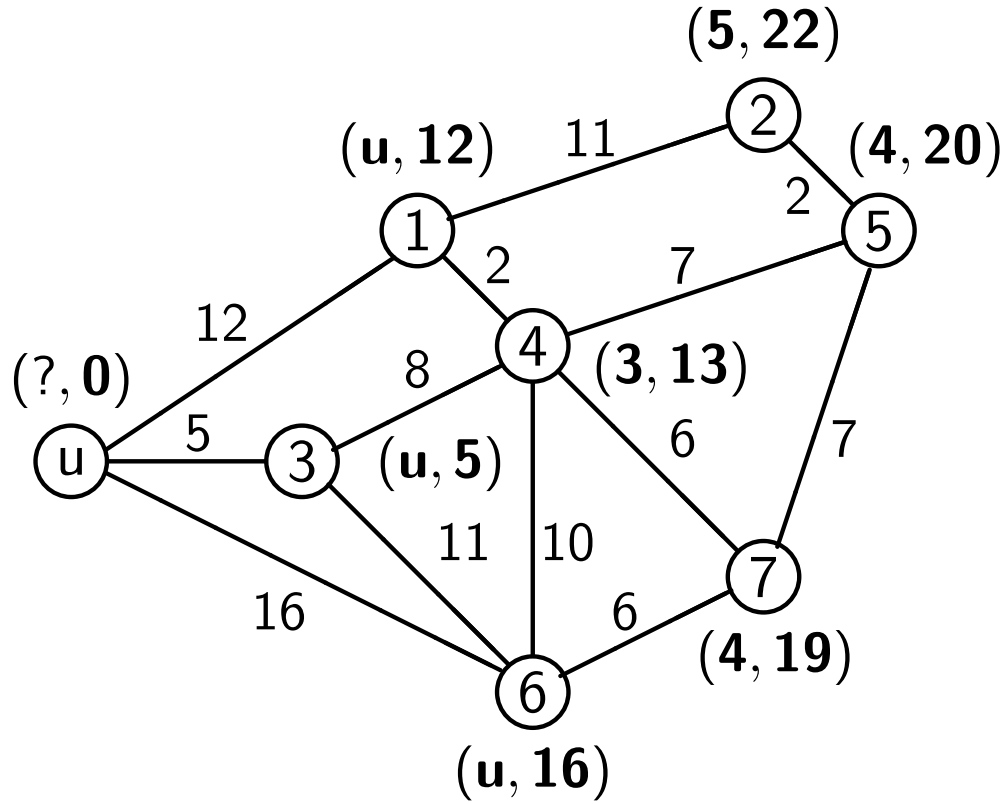
Closest to vertex u is now the vertex 7 ($dist[7] = 19$). Again no label will be modified.



Closest to vertex u is the vertex 5 since $dist[5] < dist[2]$ $(20 < 23)$. We process it.



Last unprocessed vertex is now vertex 2.
Since $dist[2] > dist[5] + w[5][2]$ $(23 > 22)$ the new label of vertex 2 will be $(5, 22)$.



Now (the only) vertex closest to u is vertex 2. We modify its state and the algorithm stops.
We have found the distance from u to all vertices in the graph.

Proof that Dijkstra's Algorithm works correct

Theorem

Let G be a (positively) weighted graph and let u and v be two vertices in G. Dijkstra's Algorithm finds the shortest path from vertex u to vertex v.

Proof

By S we denote the set of processed vertices.
Key observation is that after each iteration gives `dist[i]` the distance from `u` to `i` traversing only all *processed* vertices in S. These distances are the same when traversing any vertices in G.

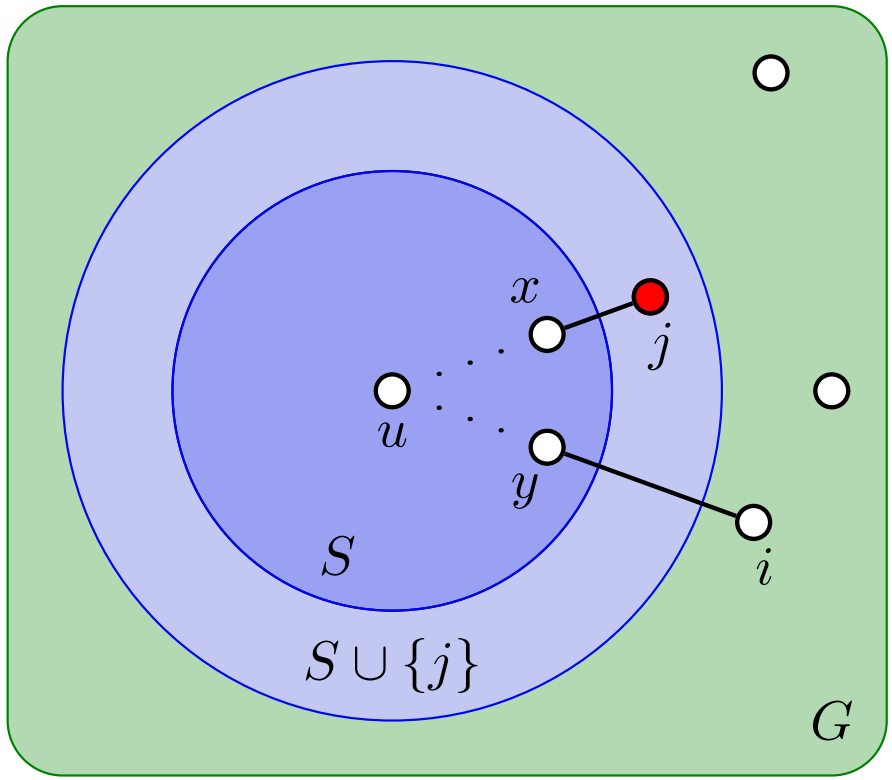
We proceed by induction on the number of iterations:

Basis step: In the first iteration of Dijkstra's Algorithm the only vertex in the depository is u. We process it and modify the distance to its neighbors based on edge weights adjacent to u.

The claim holds trivially, since after the iteration $S = \{u\}$ and all the distances through vertices in S only are minimal.

Proof (continued)

Inductive step: In every subsequent iteration we choose from the depository the vertex `j` with the distance to vertex `u`.
At the same time no shorter path to `j` exists, all paths through unprocessed vertices has to be longer, no shortcut through more distant vertices is not possible due the choice of `j`.



Here we use the that the weights `w[][]` are positive, through `i` the paths have to be longer than through `j`. The claim follows by induction. □

Next lecture

Chapter Trees and forest

- motivation
- basic tree properties
- rooted trees
- isomorphism of trees
- spanning trees of graphs

Discrete mathematics

Petr Kovář
petr.kovar@vsb.cz

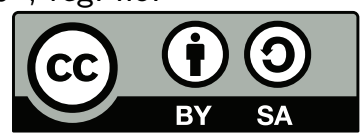
VŠB – Technical University of Ostrava

Winter term 2021/2022
DiM 470-2301/02, 470-2301/04, 470-2301/06

EUROPEAN UNION
European Structural and Investment Funds
Operational Programme Research,
Development and Education

MINISTRY OF EDUCATION,
YOUTH AND SPORTS

The translation was co-financed by the European Union and the Ministry of Education, Youth and Sports from the Operational Programme Research, Development and Education, project "Technology for the Future 2.0", reg. no. CZ.02.2.69/0.0/0.0/18_058/0010212.



About this file

This file is meant to be a guideline for the lecturer. Many important pieces of information are not in this file, they are to be delivered in the lecture: said, shown or drawn on board. The file is made available with the hope students will easier catch up with lectures they missed.

For study the following resources are better suitable:

- Meyer: Lecture notes and readings for an http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-042j-mathematics-for-computer-science-fall-2005/readings/" (weeks 1-5, 8-10, 12-13), MIT, 2005.
- Diestel: Graph theory http://diestel-graph-theory.com/ (chapters 1-6), Springer, 2010.

See also http://homel.vsb.cz/~kov16/predmety_dm.php

Lecture overview

Chapter Trees

- motivation
- basic tree properties
- rooted trees
- isomorphism of trees
- spanning trees of graphs

Chapter Trees

Motivation

Among the most common structures in both nature and mathematics are trees (objects with "tree" structure).
There exist a vast amount of objects, that can be described by a "tree".

- genealogy trees
- evolutionary tree
- electrical circuits
- hierarchical structure (chief and subordinated)
- branching in a search

Common property: no "cycle" in the structure.

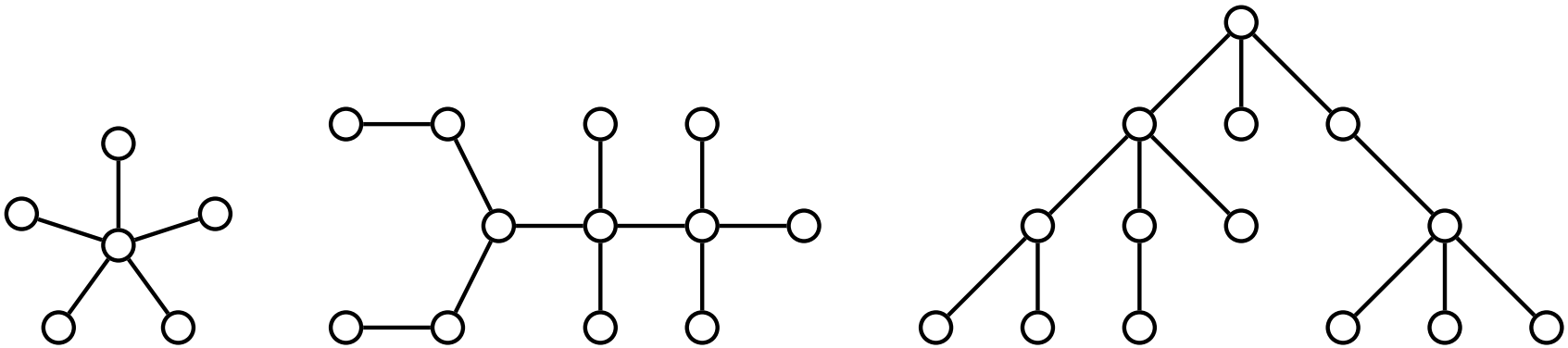
Elementary properties of trees

We say, that a graph is acyclic, if it does not contain a cycle. thus if no its subgraph is isomorphic to a cycle.

Definition

A simple connected graph that acyclis is a tree.

A forest is a graph, whose components are trees.



Note to terminology

- Forest is a (finite simple) acyclic graph.
- Tree is a connected forest.

Seems awkward...

Vertices of degree 1 are called leaves.
All other vertices are non-leaf vertices.

Lemma

A tree with more than one vertex contains at least one leaf.

Proof Connected graph with more than one vertex cannot have a vertex of degree 0. Let us take any tree T and some vertex v. Now we construct a longest possible trail S in T starting at v. S starts with any edge from v (such an edge exists, why?). In every consequent vertex u of the trail S

- either u is of degree at least 2 and we can extend S by another edge (notice: if some vertex would repeat in trail S, then S would contain a cycle, that would be a subgraph of tree T, which contradicts the definition of a tree),
- or u is the last vertex of the trail (u is of degree 1),

Since T is finite, we surely find such vertex of degree 1 in any tree T. □

Note

It's easy to prove, that every nontrivial tree T contains at least two leaves.

Questions

- How is called a tree with, precisely two vertices of degree 2 and no vertex with larger degree?
- Does there exist a tree with a vertex of degree k and less than k vertices of degree 1?
- Can you prove the previous assertion?
- How many edges have to be removed from K_n to obtain a tree?

Theorem

A tree on n vertices has precisely $n-1$ edges.

Proof

We proceed by induction on n.

Basis step: A (trivial) tree with one vertex has $n-1=0$ edges.

Inductive step: Let T be any non-trivial tree on $n>1$ vertices. By induction hypothesis every tree with less than n vertices has one edge less than vertices.

By the previous lemma T has a vertex of degree 1. By $T'=T-v$ we denote the graph, which arises from T by removing vertex v ("shaving").

- After removing a leaf the graph remains connected (no path between two vertices different from v does not pass through a vertex of degree 1), T' is connected.
- Removing a vertex/an edge no cycle arises, T' is also acyclic.

By induction hypothesis T' has one edge less then vertices, thus T' has $(n-1)-1$ edges. Hence the original tree T has one edge more, i.e. $(n-1)-1+1=n-1$ edges.

The claim follows by induction. □

Example

In the database there are 12 objects and 34 relations between the objects. We want the structure of objects draw as a graph in which objects correspond to vertices and relations to edges.
a) Will the resulting graph be a tree?
b) Will the resulting graph always be connected?

a) The resulting graph cannot be a tree, it has to contain cycles. A tree on 12 vertices has precisely 11 edges (relations).
Even if there were 11 relations, we cannot guarantee the resulting graph to be a tree, why?

b) The resulting graph can, but does not have to be connected. Connectivity depends on the stored structure.
E.g. it could be a graph with one component close to K_9 and three isolated vertices (K_9 has $\binom{9}{2} = 36$ edges, we can remove any 2 edges).

If the graph has 12 vertices and more than 55 edges, the resulting graph has to be connected. K_{12} has 66 edges and is edge 11-connected. After removing any less than $66 - 55 = 11$ edges the graph remains connected.

on proving theorems of the form $A \Rightarrow B$

Suppose A is the premise and B is the conclusion of the theorem.

Direct proof consists of a sequence of valid implications.

$$A = A_0 \Rightarrow A_1 \Rightarrow A_2 \Rightarrow \cdots \Rightarrow A_n = B$$

Indirect proof is a direct proof of the theorem $\neg B \Rightarrow \neg A$, which has the same truth value table as $A \Rightarrow B$.

$$\neg B = A_0 \Rightarrow A_1 \Rightarrow A_2 \Rightarrow \cdots \Rightarrow A_n = \neg A$$

In a proof by contradiction we assume that both the premise A and the negation of the conclusion $\neg B$ are true. By a sequence of valid implications we obtain a contradiction. By a contradiction we mean that both V and its negation $\neg V$ are true simultaneously, which is not possible.

$$A \wedge \neg B \Rightarrow \cdots \Rightarrow V \wedge \neg V$$

Assuming $\neg B$ leads to a contradiction, thus B holds.

Theorem

In a tree there exists exactly one path between every pair of vertices.

Proof By contradiction.
We assume the premise (T is a tree) and the negation of conclusion (there exists a pair of vertices T connected by none or at least two different paths).

Since T is connected (by definition of a tree), there is a path between any pair of vertices u, v. Now if u, v are connected by two different paths, their union is a walk in T and after "deleting" all repeated vertices in this walk we obtain a cycle in T, which contradicts the premise that there is no cycle in T.

The negation of the conclusion leads to a contradiction, thus thus there is precisely one path between any u and v. □

Note

Paths from u to v and the "reversed" path from v to u we consider a single path between u, v.

We know already, that a tree on n vertices contains $n-1$ edges. Adding one new edge

- does not violate connectivity,
- violates the state of being acyclic.

A tree with one additional edge contains a cycle, we show that there is just one such cycle.

Corollary

By adding one (new) edge to a tree (on at least three vertices) we obtain a graph with a single cycle.

Proof Suppose there is no edge uv in a tree T.
By adding edge uv precisely one cycle arises by joining uv and a unique path between u and$v$ in T (unique by previous theorem). □

Note

By adding at least two edges to a tree, the number of emerging cycles depends on where we add the edges.

Given a connected graph G and determining k-connectivity we asked how many edges at least have to be removed from G to obtain a disconnected graph. Now we ask

- how many edges at most can be removed from G to obtain a connected graph or, conversely
- how many edges at least have to remain in G for G to remain connected.

Trees are graphs that are both connected and no edge can be removed without loosing connectivity.

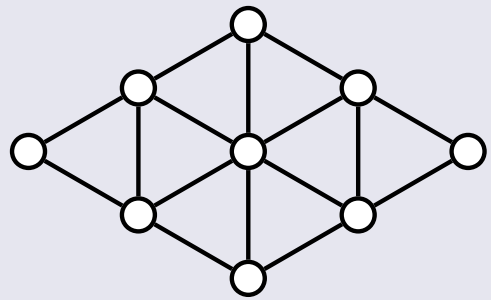
Theorem

A tree is the *minimum connected* graph (on a given set of vertices).

Proof A tree is connected by definition. If a graph contains a cycle, it remains connected also after removing any edge of this cycle. Thus the minimum connected graph is a tree.
Conversely, if after removing edge uv form a tree T the resulting graph remains connected, then between u, v in T would exist two paths: u, v-path in $T \setminus uv$ and edge uv. This contradicts previous theorem. Thus, tree is the minimum connected graph on a given set of vertices. □

Example

At most how many edges can be removed from the graph G so that the graph remains connected?



Graph G.

By the previous theorem the resulting graph has to be a tree.
The graph G has 9 vertices and 16 edges, thus by the theorem on the number of edges in a tree at most 8 edges can be removed.

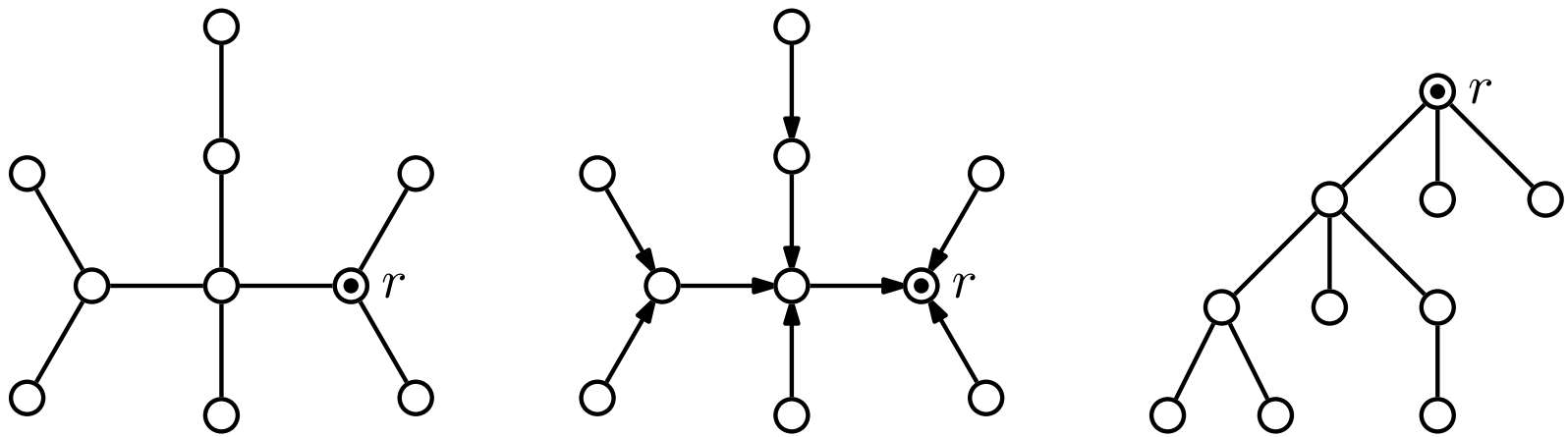
Moreover, one can remove such 8 edges that both removed edges and the remaining edges for a connected factor. Can you find such 8 edges?

Rooted trees

Is certain instances of "trees" it is convenient to select a vertex, called root, (as the "start" of data). Rooted trees have their origins also in family trees, (Tiggers "family tree") which implied terminology.

Definition

A rooted tree is a tree T along with one significant root vertex $r \in V(T)$, denoted by (T, r), we say tree T with root r.

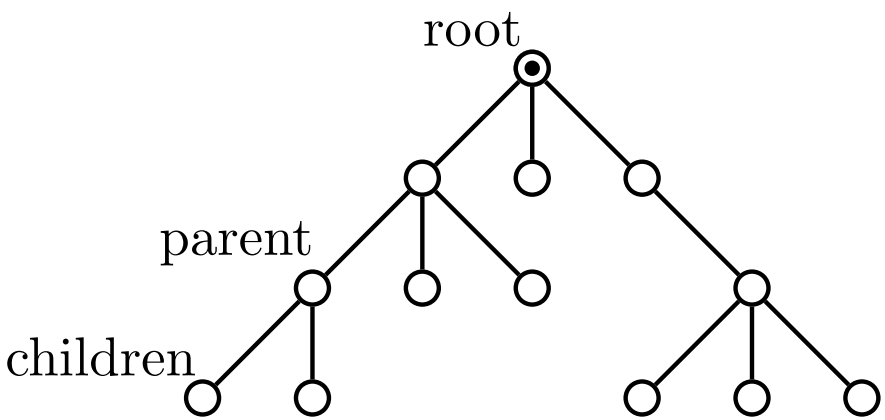


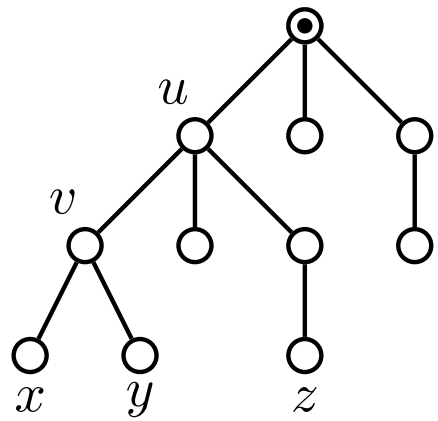
There is a difference between a „tree“ and a „rooted tree“, which has some extra information.

Root will be drawn on top.

Definition

Take a rooted tree (T, r) and a pair of adjacent vertices u, v, such that u is the neighbor of v on the path to r. Then u is called the parent of v and v is called the child of u.





Sometimes we will use other terms as „grandfather“, „sibling“, . . .

Notice: by choosing a different root the parent-child relationship can swap.

Definition

Vertices without children in non-trivial graphs are called bottom vertices.

Notice the bottom vertices are leaves, but not all leaves are necessarily bottom vertices.

Definition

Center of a tree T is the vertex or edge in T determined by the following algorithm:

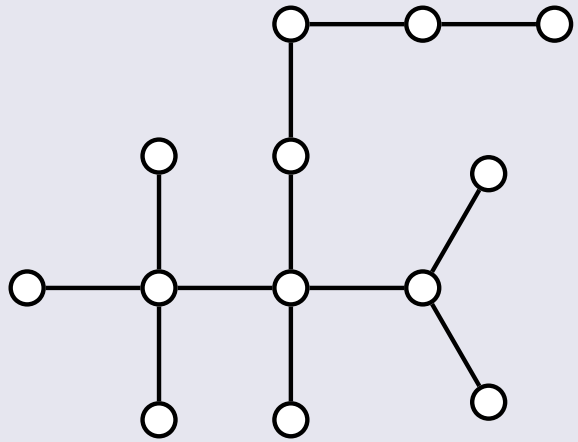
1. If a tree T has a unique vertex v, then v is the center of T.
   If a tree T has two vertices, its center is the edge joining the two vertices.
2. Otherwise we create a (smaller) tree $T' \subset T$ by deleting all leaves of T (shaving). It is obvious, that T' is not empty; proceed by step 1.

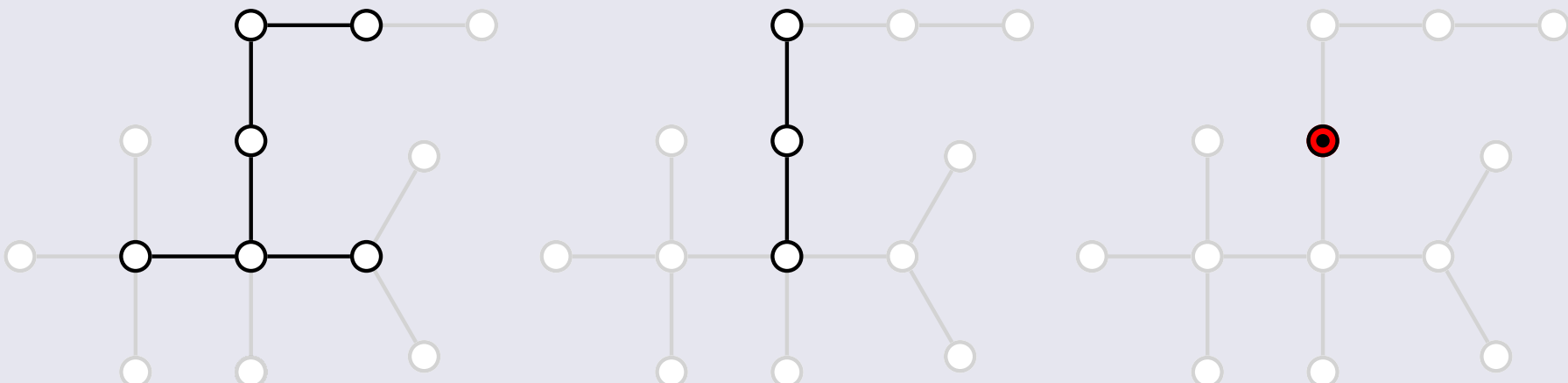
The center of T' obtained by recursion is also the center of T.

The process of removing leaves is called shaving.

Example

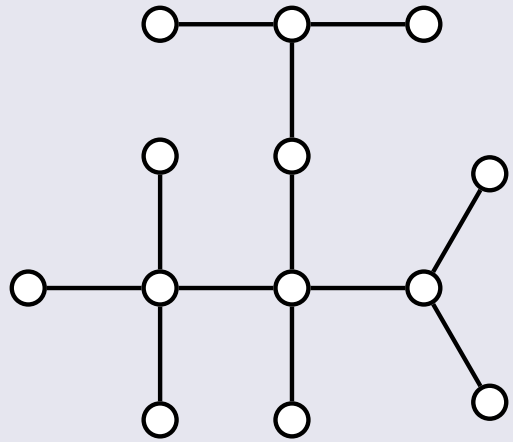
Find the center of tree T_1.

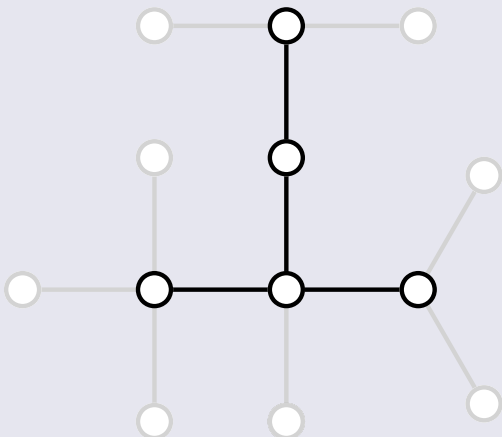


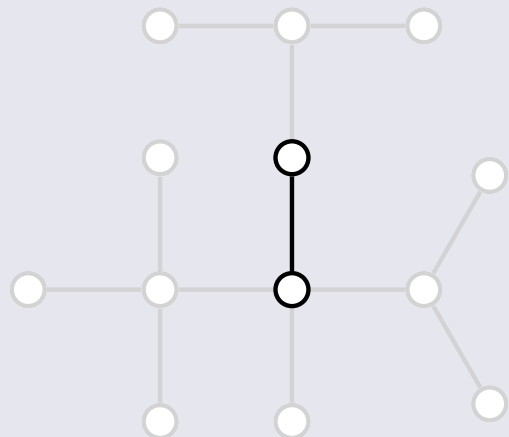


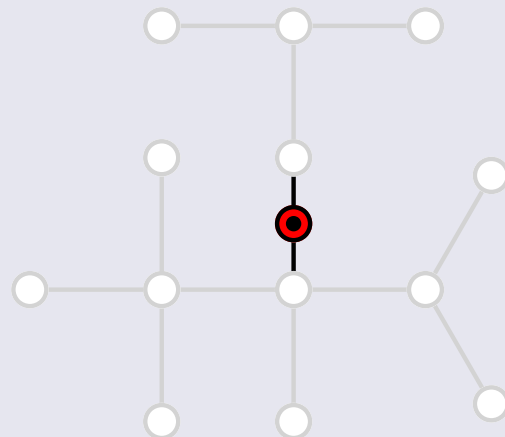
Example

Find the center of tree T_2.









Notice: we added a new vertex (and two edges instead of one edge).

The root and the center

The root can be any vertex in a given tree; the root does not have to be the center.

If the root of a tree has to be determined *uniquely*, then *center* is the best candidate (it is determined uniquely).

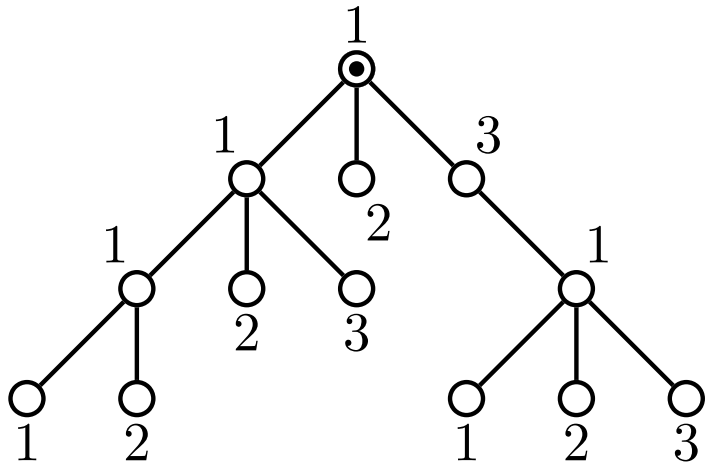
If the center is an edge add a new vertex onto the central edge so that it "splits" the edge into two.

Ordered rooted trees

Another information assigned to rooted trees is the ordering of children of every vertex (ordering the ancestors in one generation by their birth date).

Definition

The rooted tree (T, r) is ordered, if for every vertex is the order of its children determined uniquely ("from left to right").



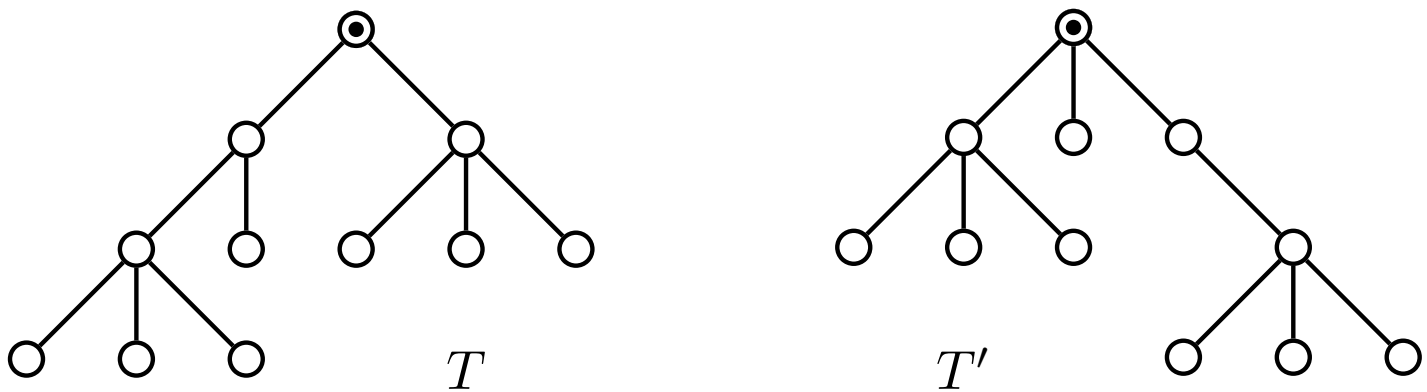
Formally: ordered rooted tree is (T, r, f), where T is a tree and r its root. Function $f : V(T) \to \mathbb{N}$ assigns each vertex it order among siblings $1, 2, \ldots, k$.

Isomorphism of trees

The notion of isomorphism of trees is a special case of isomorphism of graphs. Two trees are isomorphic, if they are isomorphic as graphs.
Recall that no fast algorithm for deciding whether two general graphs are isomorphic is known. Trees are such a special class of graphs that, for trees such algorithm does exist!
Before we give the algorithm, we have to introduce several terms.

Definition

Two rooted trees (T, r) and (T', r') are isomorphic if there exists an isomorphism of T and T', that takes root r onto root r'.

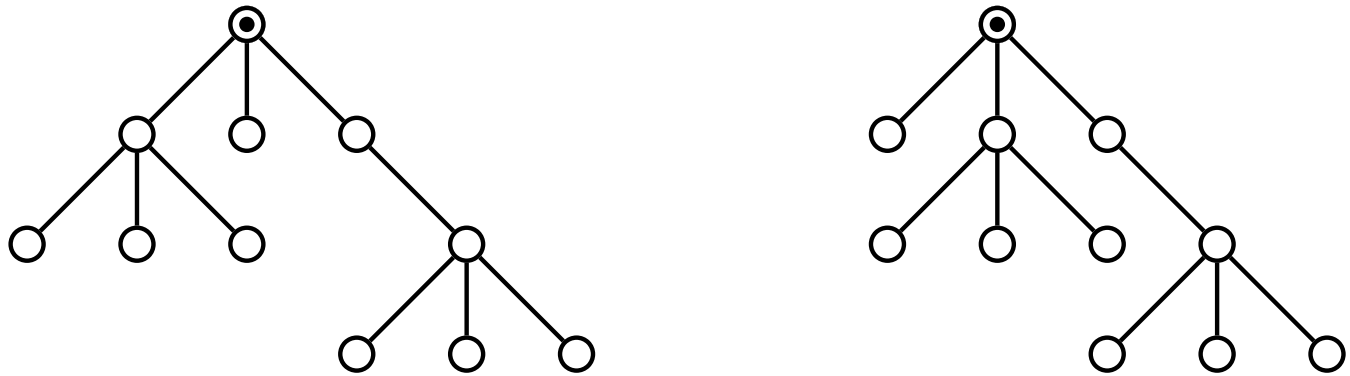


Two isomorphic trees that are not isomorphic as rooted trees.

Notice: the root is different in T and T'.

Definition

Two ordered rooted trees are isomorphic, if there exists an isomorphism of rooted trees, such that it *preserves the order of children* of every vertex.



Two isomorphic rooted trees that are not isomorphic as ordered rooted trees.

Notice: the order of children of the root differs.

Definition

Subtree of a vertex u of a given rooted tree (T, r) is each component of the graph $T - u$, which contains some child x of u.

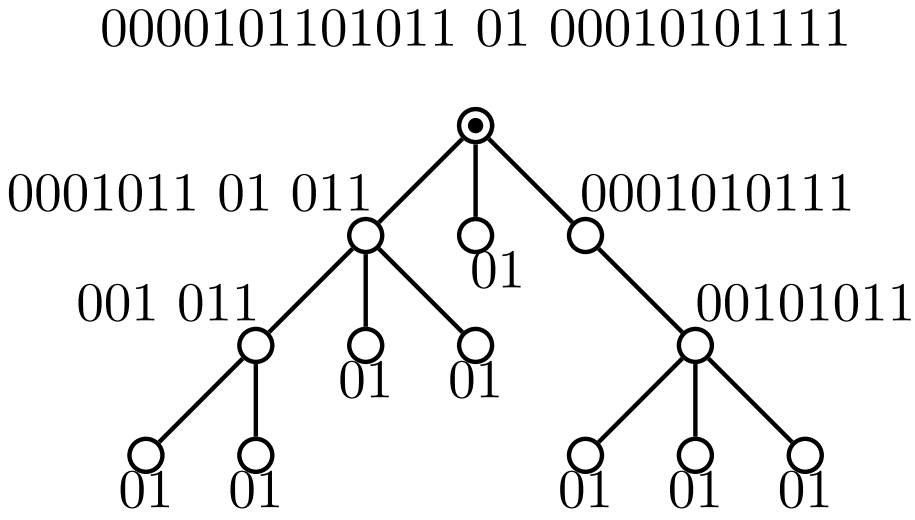
Each subtree of vertex u is again a (rooted) tree.

Encoding ordered rooted trees

To every ordered rooted tree we can easily assign a string of 0 and 1 which uniquely determine the tree.

Definition

Code of an ordered rooted tree is constructed recursively by joining codes of all subtrees of the root, ordered in a particular (uniquely chosen) ordering and enclosed in a pair of 0 and 1 (see figure).



Coding a rooted tree.

Note

Instead of “0” and “1” one can use, e.g. “(” and “)” or „A“, „B“.

Ordered rooted trees given by their code

We described how to obtain a code for a given rooted tree tree.
Now we show the reverse: how to draw a rooted tree given by its code.

Lemma

Take the code of an ordered rooted tree. The corresponding tree can be drawn by the following algorithm:

- when reading "0" at the beginning put the pen on the paper, draw the root vertex,
- when reading another "0" draw an edge to a child vertex of the current vertex,
- when reading "1" return to the parent of the current vertex or lift the pen if the current vertex is the root.

Notice: not every sequence of 0 and 1 is a code of some tree (see discussion).

Minimum code

We can consider tree codes as strings and we can order these strings uniquely, e.g. lexicographically.

Suppose the symbol 0 precedes symbol 1 in the dictionary.
E.g. the string 000111 precedes codes 001011, 0011, and 01.

One has to distinguish *code of an ordered rooted tree* and *minimum code of a rooted tree*:

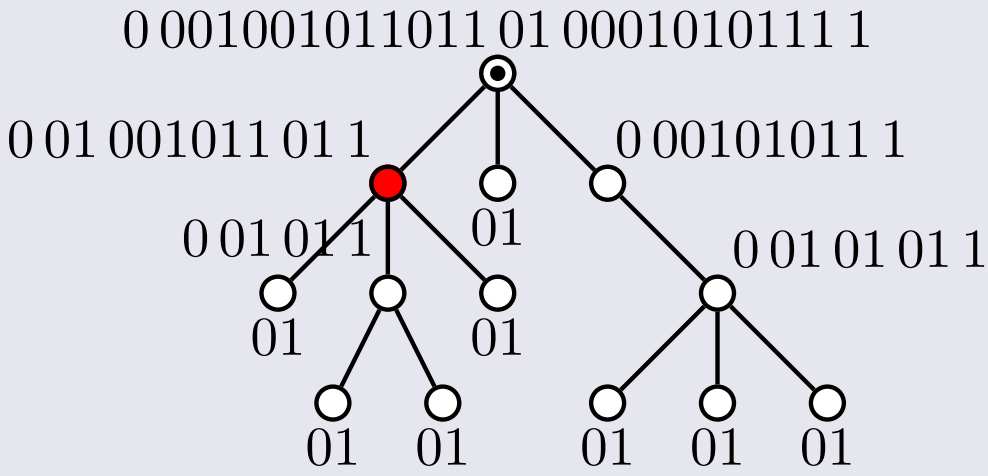
- drawing a tree given by the code of an ordered rooted tree (T, r), we obtain (T, r) again,
- drawing a tree given by the minimum code, the order of children can differ from the order in (T, r).

We say the rooted tree (T, r) „was reordered“.

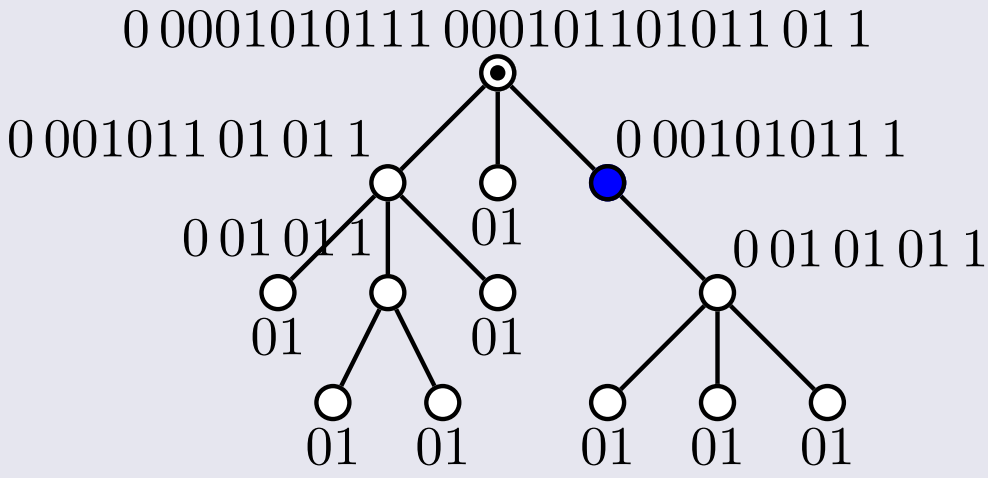
Note

The upper bound on time complexity for finding a minimum code of one tree is $O(n^3)$.

Example



The code of an ordered rooted tree.



The minimum code of an ordered rooted tree.

When determining isomorphism of two arbitrary trees we

- find the center of each tree,
- the center of each tree we choose as the root,
- we find the minimum codes (we order the codes of the children lexicographically by their codes)
- we use the following Lemma which guarantees the uniqueness of each code.

Lemma

Two ordered rooted trees are isomorphic if and only if their codes, obtained as described above, are the same strings.

The process results in the algorithm described below.

Algorithm Determining isomorphism of trees

Algorithm determines if two trees T and U are isomorphic ($T \simeq^{?} U$)

```
// Let T, U be two trees with the same number of vertices.
Input < trees T and U;
for (X=T,U) {
   // find the centers of U, T
   x = center(X);
   if (x is one vertex)
      r = x;
   else
     add new vertex r, replace edge x=uv by edges ru, rv;
   k[X] = minimum_code(X,r);
}
if ((|V(T)|==|V(U)|) && (k[T]==k[U] as strings))
   print("Trees T and U are isomorphic.");
else
   print("Trees T and U are not isomorphic.");
exit;
```

Algorithm ... continued (finding the minimum code)

Function `minimum_code(X,r)` finds for tree X with root r (lexicographic) minimum code.

```
// the input is a rooted tree (or a subtree)
input < rooted tree (X,r);

function  minimum_code(tree X, vertex r) {
   if (X has one vertex)
      return "01";
   Y[1...d] = {connected components X-r, subtrees without r};
   s[1...d] = {roots of subtrees Y[] in corresponding order};
               // roots are the children of r
   for (i=1,...,d)
       k[i] = minimum_code(Y[i],s[i]);
   sort lexicographic so that k[1] <= k[2] <= ... <= k[d];
   return "0"+k[1]+...+k[d]+"1";
}
```

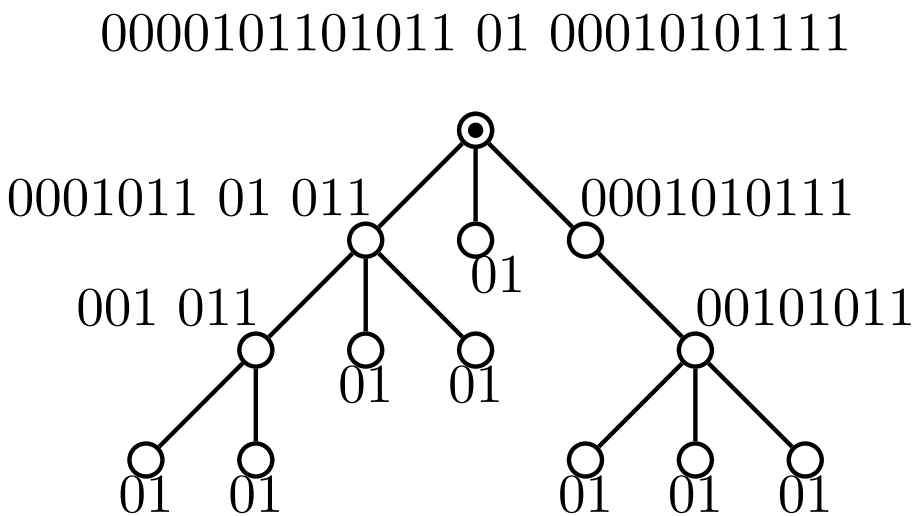
Functions recursively constructs the minimum code of tree X.

Note

Notice that in the Algorithm we check whether both have the same number of vertices.
For example paths P_{2n} and P_{2n+1} are not isomorphic, but since the center of path P_{2n} is the „middle“ edge, by finding the center we add a new vertex to this edge and obtain a second path P_{2n+1}. Without checking the number of vertices the algorithm could give a wrong answer.

Questions

- Is the following code a minimum code? Why?
- How would a minimum code look like?



Spanning tree

Recall the following terms:

- subgraph, factor
- connected and disconnected graph
- labeled graph, graph labeling

Definition

A spanning tree of a connected graph G is such a factor of G, which is a tree. (Factor of G is a subgraph, which contains all vertices of G.)
Weight of a spanning tree in a labeled graph G is the sum of labels of all edges of the spanning tree.

We say "edge label" and "spanning tree weight".

The importance of "spanning trees" lies in the minimality with respect to number of edges, while connectivity is preserved.
(we have a Theorem about the minimum connected subgraph)

The label of every edge can differ, we obtain:

Minimum spanning tree problem (MST)

Given a connected labeled graph G with non-negative edge labels w. The task is to find such a spanning tree T of G, which has among all spanning trees the minimum weight. Formally

$$MST = \min_{\text{span.tree}\, T \subseteq G} \left(\sum_{e \in E(T)} w(e) \right).$$

Questions

- How many edges has a spanning tree of a connected graph with n vertices?
- Is it possible to find a minimum spanning tree in a graph with negative labels?
- Is every connected factor with minimum edge labels a spanning tree?

We present several algorithms for finding a minimum spanning tree in a non-negatively labeled graph.

Algorithm Greedy minimum spanning tree algorithm

We have a labeled graphs G with non-negative labels w of edges. By m we denote the number of edges of G.

- We order the edges of G in an non-decreasing order according their labels:

$$w(e_1) \leq w(e_2) \leq \cdots \leq w(e_m).$$

- We start with an empty set of edges $T = \emptyset$ for the spanning tree.
- For $i = 1, 2, \ldots, m$ we take the edge e_i and if by adding it to the set T no cycle (induced by $T \cup \{e_i\}$) originates, we include e_i into T. Otherwise we "discard" e_i.
- At the end T contains all edges of a minimum spanning tree of graph G with labels w.

We show that the algorithm works correctly.

Theorem

The greedy algorithm finds a minimum spanning tree of a connected graph.

Proof By contradiction. Let T be the set of edges obtained by the Algorithm. Suppose that $w(e_1) \leq w(e_2) \leq \cdots \leq w(e_m)$. Let T_0 by the set of edges of such a minimum spanning tree (multiple spanning trees can have the same weight), which matches T in the most first edges. If $T_0 = T$, algorithm works correctly.

Suppose now that $T_0 \neq T$ and we obtain a contradiction. Thus we show that $T_0 \neq T$ cannot occur.

By $j > 0$ denote such an index, that the sets T_0 and T match in the first $j-1$ edges $e_1, \ldots, e_{j-1}$, but they do not match in edge e_j. Thus $e_j \in T$, but $e_j \notin T_0$. (According the algorithm $e_j \notin T$ and $e_j \in T_0$ cannot happen.) graph $T_0 \cup \{e_j\}$ contains the graph with edges precisely one cycle C. Cycle C cannot be a subgraph of the spanning tree T, thus there is an edge e_k in C, such that $e_k \notin T$ and $k > j$. Since $w(e_k) \geq w(e_j)$, the spanning tree with edges $T' = \big(T_0 \setminus \{e_k\}\big) \cup \{e_j\}$ (swapping edges e_k and e_j) does not have higher weight than T_0, but it matches T in more first edges! This is a contradiction with the choice of T_0. □

This greedy algorithm was introduced first by Kruskal in 1956. But it is known that Kruskal continued the work of a Czech mathematician Otakar Borůvka.
Already in 1926 solved Borůvka the question of building an optimal electrical network in southern Moravia and described a very similar algorithm in great detail using matrices.

Algorithm Borůvka's minimum spanning tree algorithm

Suppose G is a positively weighted graph with edges labeled by pairwise different labels.
At the beginning we order the edges according their increasing labels $w(e_1) < w(e_2) < \ldots < w(e_m)$.
The spanning tree we construct by adding edge e_i (for $i = 1, 2, \ldots, n$), if no cycle originates by adding e_i.

In response to Borůvka's work Vojtěch Jarník designed in 1929 a similar algorithm.

The Jarník's algorithm is known as Prim's algorithm (1957).

Algorithm Jarník's minimum spanning tree algorithm

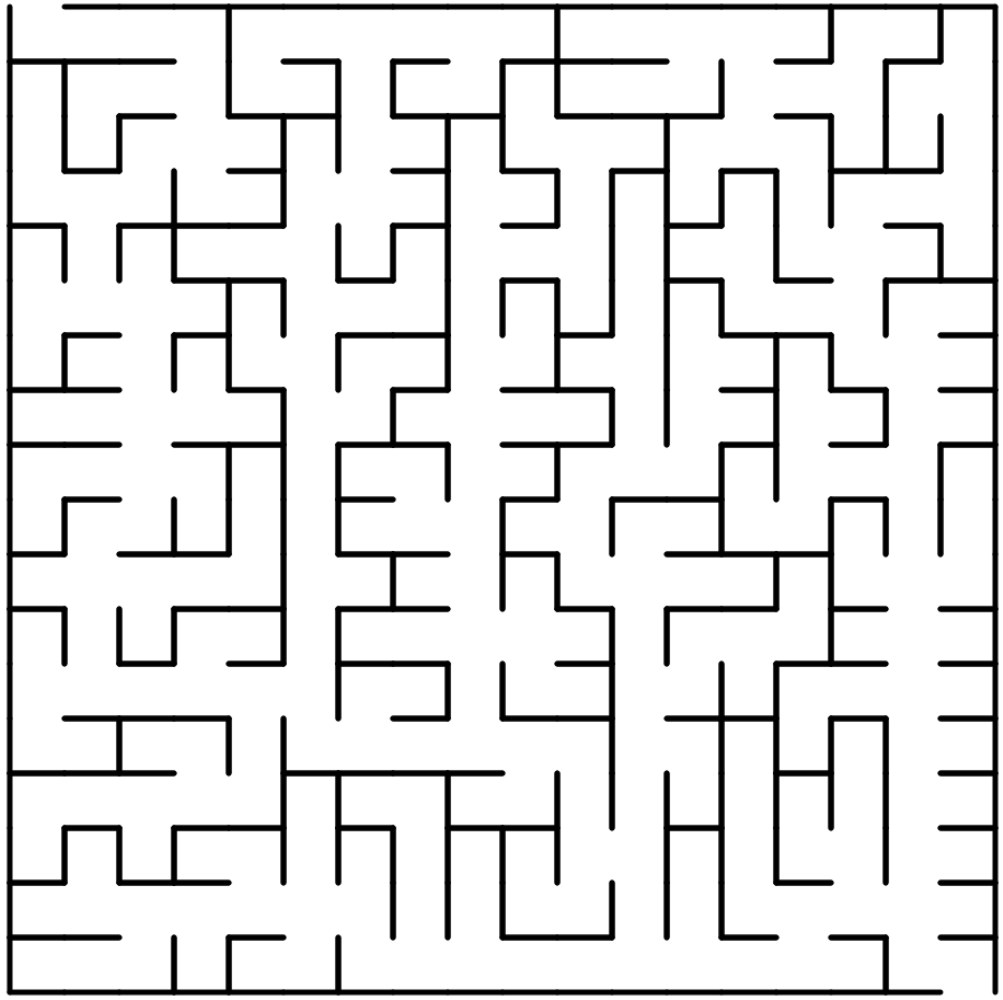
We do not order the edges. We construct the minimum spanning tree starting from any vertex. In every step we choose the edge with the smallest label with one end-vertex among the vertices in the already constructed subgraph and the other end-vertex among remaining vertices.

Notice:

- in Jarník's algorithm we need not to sort the edges,
- we do not need to check whether adding an edge produces a cycle, we save time.

Examples of the Greedy algorithm and Jarník's algorithm are given at `http://homel.vsb.cz/~kov16/predmety_dm.php`

MSP algorithms can be used for constructing labyrinths.



Labyrinth constructed using Jarník's algorithm.

For details see textbook „Úvod do teorie grafů“ (in Czech) or on-line.

Next lecture

Chapter Graph colorings and graph drawing

- motivation
- graph coloring
- drawing graphs in a plane
- recognizing planar graphs
- map coloring and planar graphs coloring

Discrete mathematics

Petr Kovář
petr.kovar@vsb.cz

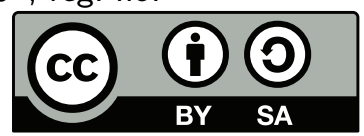
VŠB – Technical University of Ostrava

Winter term 2021/2022
DiM 470-2301/02, 470-2301/04, 470-2301/06

EUROPEAN UNION
European Structural and Investment Funds
Operational Programme Research,
Development and Education

MINISTRY OF EDUCATION,
YOUTH AND SPORTS

The translation was co-financed by the European Union and the Ministry of Education, Youth and Sports from the Operational Programme Research, Development and Education, project "Technology for the Future 2.0", reg. no. CZ.02.2.69/0.0/0.0/18_058/0010212.



About this file

This file is meant to be a guideline for the lecturer. Many important pieces of information are not in this file, they are to be delivered in the lecture: said, shown or drawn on board. The file is made available with the hope students will easier catch up with lectures they missed.

For study the following resources are better suitable:

- Meyer: Lecture notes and readings for an http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-042j-mathematics-for-computer-science-fall-2005/readings/" (weeks 1-5, 8-10, 12-13), MIT, 2005.
- Diestel: Graph theory http://diestel-graph-theory.com/ (chapters 1-6), Springer, 2010.

See also http://homel.vsb.cz/~kov16/predmety_dm.php

Lecture overview

Chapter Graph colorings and graph drawing

- motivation
- graph coloring
- drawing graphs in the plane
- recognizing planar graphs
- map coloring and planar graphs coloring

Graph colorings

We mention two problems that can be solved naturally using graph colorings.

Storing goods

There are many different food products in a storehouse. By regulations several of them certain have to be stored separately. E.g. fruit salads cannot be in the same department as raw eggs or salami cannot share department with raw meat.
What is the smallest number of departments necessary?

Set up a graph whose vertices represent stored goods and an edges joins two vertices whenever the corresponding two commodities have to be stored separately. Compartments are distinguished by colors.

Question

What is the least number of different colors necessary to color the vertices of the graph so that any two adjacent vertices have the different colors?

Optimization of traffic lights

A crossing has several corridors for both cars and pedestrians. Corridors (even of different directions) may not interfere and the traffic can flow simultaneously. On the other hand corridors that do interfere with each other have to have green within non-overlapping time slots. Time slots are distinguished by colors.
What is the least number of time intervals necessary in one "cycle" of the traffic lights?

In the graph model the vertices will represent corridors and edges will join vertices that correspond to corridors that do interfere.

Question

What is the least number of colors necessary to color the vertices of the graph so that any two adjacent vertices have the different colors?

We show some special cases and prove a couple of simpler theorems. First we introduce several definitions.

Definition

Graph coloring of G by k colors is such a mapping

$$c : V(G) \to \{1, 2, \ldots, k\},$$

that any two adjacent vertices have different colors, i.e. $c(u) \neq c(v)$ for every edge $uv \in E(G)$.

Note

Graph coloring is called also a proper vertex coloring of a graph.

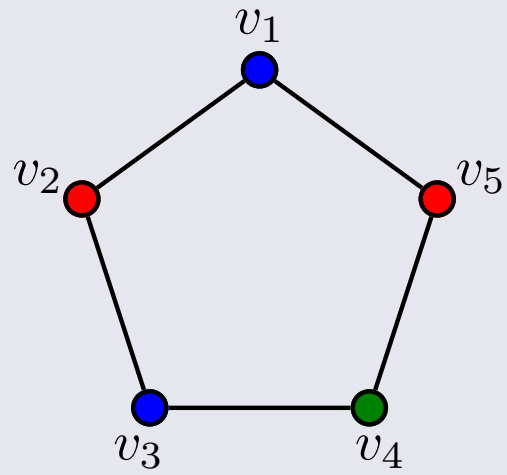
There exists a proper coloring of every graph by $|V(G)|$ colors. We are interested in the lowest possible number of colors, for which a graph coloring of G exists.

Definition

The chromatic number $\chi(G)$ of G is the least k, such that there exists a proper coloring of G by $\chi(G)$ colors.

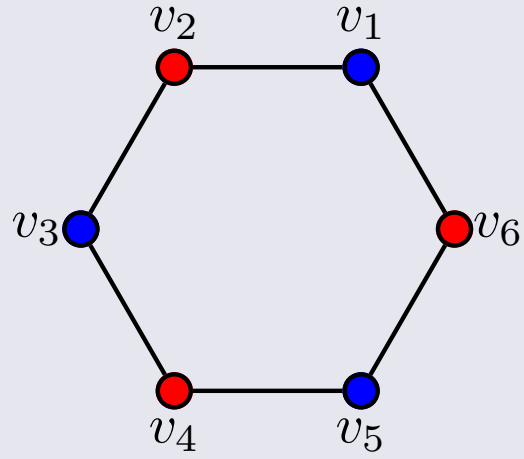
Example

What is the chromatic number of C_5?



Example

What is the chromatic number of C_6?



Upper bounds on the number of colors

Lemma

Let G by a simple graph on n vertices. Then $\chi(G) \leq n$. Equality holds if and only if G is a complete graph.

Proof In any graph G with n vertices it is enough to color every vertex by a different color and we get a proper vertex coloring of G by n different colors. Thus, $\chi(G) \leq n$.

If $G \simeq K_n$, then no two adjacent vertices can have the same color Thus, $\chi(K_n) = n$.

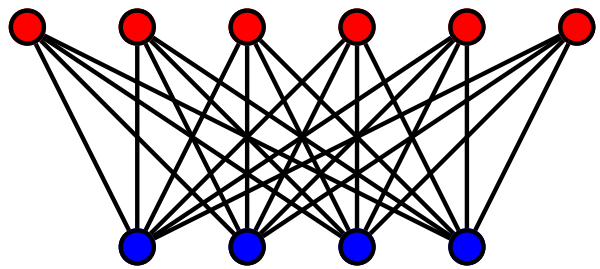
If some edge uv is missing in G, we can color $c(u) = c(v) = 1$ and color the remaining vertices by colors $2, 3, \ldots, n-1$. We obtain a proper vertex coloring by less than n colors, thus $\chi(G) < n$. □

Brook's Theorem

For every graph G with n vertices different from K_n and different from odd cycles C_n is $\chi(G) \leq \Delta(G)$.

Proof is beyond the level of this course, you can find it in the textbook „Teorie grafů“ (in Czech) or on-line.

Notice, not in every graph G as many as $\Delta(G)$ color have to be used. For example to color the vertices of a complete bipartite graph only two colors are necessary.



Algorithms for finding a proper vertex coloring by the least number of colors are complex and are not included in this text. For general graphs there are algorithms with complexity $O(n2^n)$, where n is the number of vertices.

Lower bounds on the number of colors

Brooks Theorem says *at most* how many colors are necessary to color the graph properly. Now we show a couple of simple bounds on how many colors are necessary *at least* for a proper edge coloring.

Theorem

Graph G has chromatic number 1 if and only if it has no edge.

Proof If a graph has no edge, we color all vertices by color 1. If all vertices have the same color, no edge can be in the graph. □

The next theorem we mention without proving it.

Theorem

If there is a complete subgraph on k vertices in a given graph G, then any proper vertex coloring of G requires at least k colors.

We prove one particular case of the theorem.

Theorem

Graph G has chromatic number 2 if and only if it contains no cycle of odd length (as a subgraph).

Proof (idea) An odd cycle cannot be properly colored by two colors. We choose any vertex v in G and color it by color 1. Vertices in odd distance from v we color by color 2. Vertices in even distance from v we color by color 1.
If any two vertices x, y in even distance from v are joined be edge xy, then $v, \ldots x, y, \ldots v$ is a walk of odd length. From the walk we obtain an odd cycle be deleting repeated parts which contradicts the premise. For vertices in odd distance from v we reason similarly. Thus, in this coloring no adjacent vertices have the same color and we have a proper coloring by two colors. □

Graphs without cycles of odd lengths are bipartite. The vertices of each such graph can be partitioned into two independent (partite) sets.
In each partite set it is enough to use only one color for all the vertices in the set.

How to determine chromatic number

To determine the chromatic number of a graph means to find the smallest number of colors required for a proper vertex coloring.

There is no theorem that would yield the chromatic number "easily".

The chromatic number can be found by algorithms with complexity $O(n2^n)$, where n is the number of vertices of the given graph.

Here we have shown

- upper and lower bound of the chromatic number,
- applications (warehouse problem, scheduling).

Drawing graphs in the plane

In some cases it is important how the drawing looks like. Printed circuit boards can be represented as graphs and when designing the board crossings have to be avoided.

Question: „Is it possible to draw a given graph without crossing edges?“

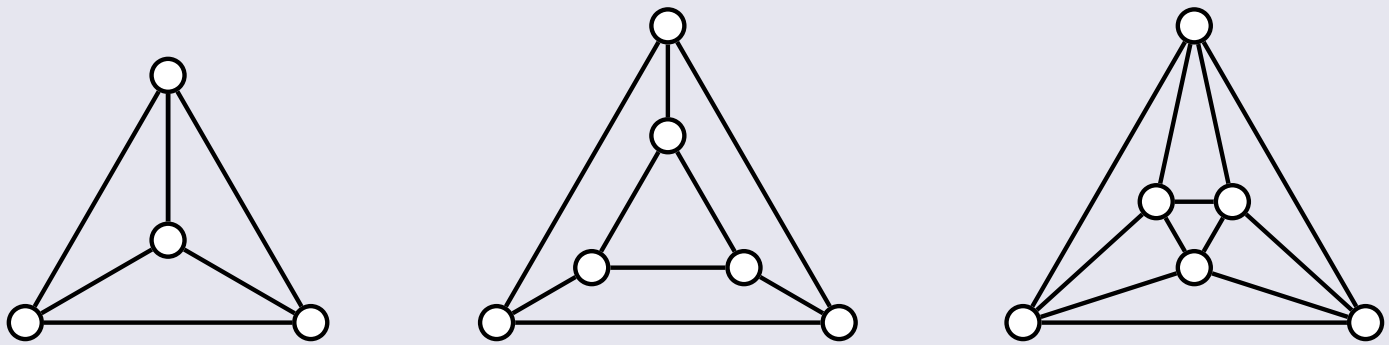
Definition

Planar drawing of a graph G is such a drawing in the plane, in which vertices are different points and edges are lines connecting the points of their end-vertices and no two edges intersect save their end-points.
We say a graph is planar if there exists its planar drawing.

Not every graph has a planar drawing!

Examples

Examples of planar graphs are graphs of polyhedrons (tetrahedron, cube, octahedron, dodecahedron, prisms, . . .)

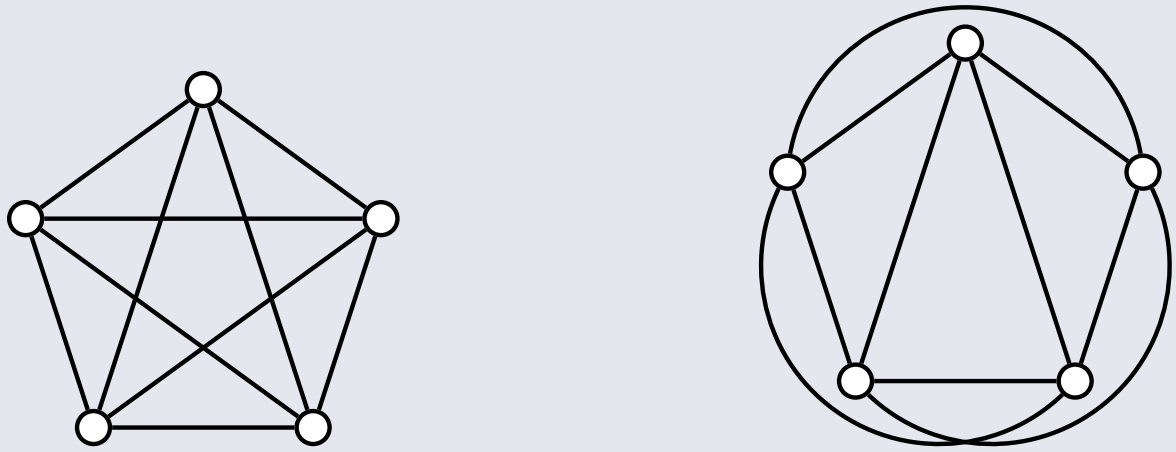


All graphs of polyhedrons are planar and (at least) 3-connected.

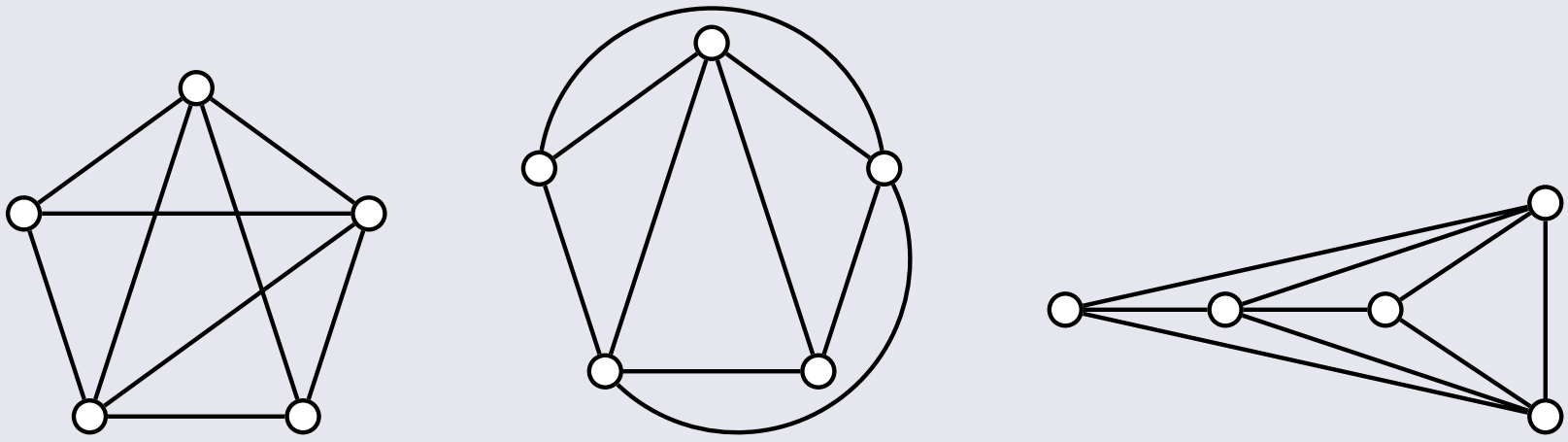
On contrary every planar 3-connected simple graph is a graph of some polyhedron.

Example

Are the graphs a) K_5, b) $K_5 - e$ planar (drawn without crossing edges)?



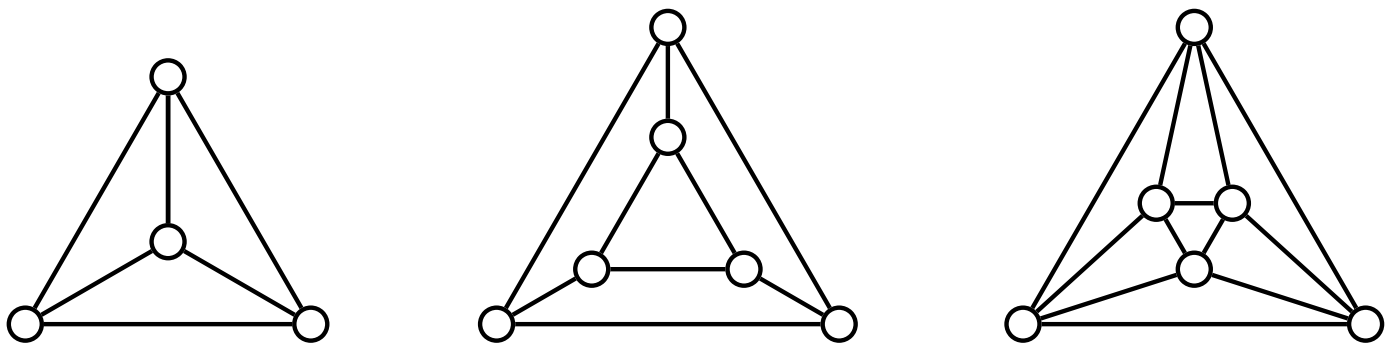
Graph K_5 and its drawing with a single crossing of edges.



Graph K_5 with an edge removed and two its planar drawings.

Definition

Faces in a planar drawing of a graph are connected regions in a plane bounded by edges and points of the drawing.



Faces in a planar drawing.

We show an important formula that counts graph elements of a planar graph: Euler's formula.

Theorem Euler's formula

A planar drawing of a *nonempty* connected graph G has f faces. The the following holds

$$v + f - e = 2.$$

Proof Proof goes by induction on the number of edges e.

Basis step: If G is a tree, it contains no cycle and the planar drawing has only one face. By a theorem a tree has $e = v - 1$ edges and be evaluate that $v + f - e = v + 1 - (v - 1) = 2$.

Inductive step: Suppose the claim holds for all graphs with $e - 1$ edges. If G contains a cycle C, then by omitting one edge uv of cycle C the number of edges decreases by 1. At the same time the number of faces decreases by 1, since the edge uv separated two faces (neighboring to uv) and by deleting uv these faces merge. The number of vertices remains the same.

By the induction hypothesis is $v + (f - 1) - (e - 1) = 2$, thus also $v + f - e = 2$. □

Note

Euler's formula is independent of a particular drawing, only on the graph structure.

Though it is a simple formula it has many applications and corollaries.

Corollary

A simple planar graph on $v \geq 3$ vertices has at most $3v - 6$ edges.

Proof Suppose we have a connected graph G, otherwise we can add more edges. By v we denote the number of vertices in G, by f the number of faces and by e the number of edges.
Since there are no loops or multiple edges, each face of G (in any planar drawing) is bounded by at least three edges. Each edge is counted at most twice (for both neighboring faces). Thus $2e \geq 3f$, from which follows $\frac{2}{3}e \geq f$. Substituting into the Euler's formula we get

$$2 = v + f - e \leq v + \frac{2}{3}e - e = v - \frac{1}{3}e$$

$$e \leq 3(v - 2) = 3v - 6.$$

□

If there are no faces with only three edges in G (a triangle-free graph) the number of edges is even smaller.

Corollary

A simple triangle-free planar graph on $v \geq 3$ vertices has at most $2v - 4$ edges.

Proof The proof is similar. By v we denote the number of vertices in G, by f the number of faces and by e the number of edges. Now we know there are no triangles in G, thus each face is bounded by at least four edges. Thus $2e \geq 4f$, which implies $\frac{2}{4}e \geq f$. Substituting into the Euler's formula we get

$$2 = v + f - e \leq v + \frac{2}{4}e - e = v - \frac{1}{2}e$$

$$e \leq 2(v - 2) = 2v - 4.$$

□

We can also bound the smallest degree of a planar graph!

Corollary

Every planar graph has a vertex of degree at most 5.
Every triangle-free planar graph has a vertex of degree at most 3.

Proof By contradiction. If all vertices would be of degree at least 6, there would be at least $\frac{1}{2} \cdot 6v = 3v$ edges in a planar graph, which contradicts previous Corollary. Thus there has to be a vertex of degree smaller than 6.

Similarly, if in a triangle-free graph all vertices would be of degree at least 4, there would be at least $\frac{1}{2} \cdot 4v = 2v$ edges in the graph, which contradicts previous Corollary. Thus there has to be a vertex of degree smaller than 4. □

Notice, that a planar graph can have vertices of high degree, but not all of them. There has to be some vertex of small degree as well.

Recognizing planar graphs

To "be planar", or "non-planar" is an important property of a graph with many applications. Among the most important are

- printed circuit boards of single layer (the circuits form a graph, need solder wires?)
- well drawn graphs (no unnecessary crossings)

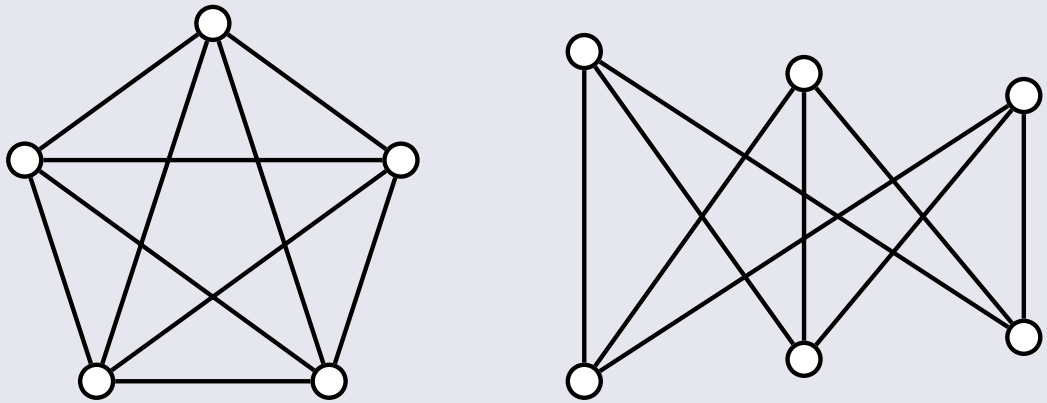
We show that Euler's formula and its corollaries can help when determining whether a graph is or is not planer.

In comparison to Hamiltonian cycles or graph colorings there are relatively fast algorithms.
We focus only on small graphs, the algorithm mentioned above go beyond the scope of this course.

We show two important graphs, that are not planar.

Example

Graphs K_5 and $K_{3,3}$ are non planar (are non-planar).



Graphs K_5 and $K_{3,3}$.

Proof We use the Corollary on the number of edges.
Notice that K_5 has 5 vertices and 10 edges. But by the Corollary a planar graph on five vertices has at most $e \leq 3 \cdot 5 - 6 = 9$ edges, hence K_5 is non-planar.

Similarly $K_{3,3}$ has 6 vertices and 9 edges. Moreover, it is triangle-free. But by the Corollary a triangle-free planar graph on six vertices has at most $2 \cdot 6 - 4 = 8$ edges, thus $K_{3,3}$ is non-planar. □

Corollary

Graphs K_5 and $K_{3,3}$ are not planar.

It can be shown that both K_5 and $K_{3,3}$ are special among all non-planar graphs. Their structure does not allow their planar drawing.

Moreover, no other such structure exists.

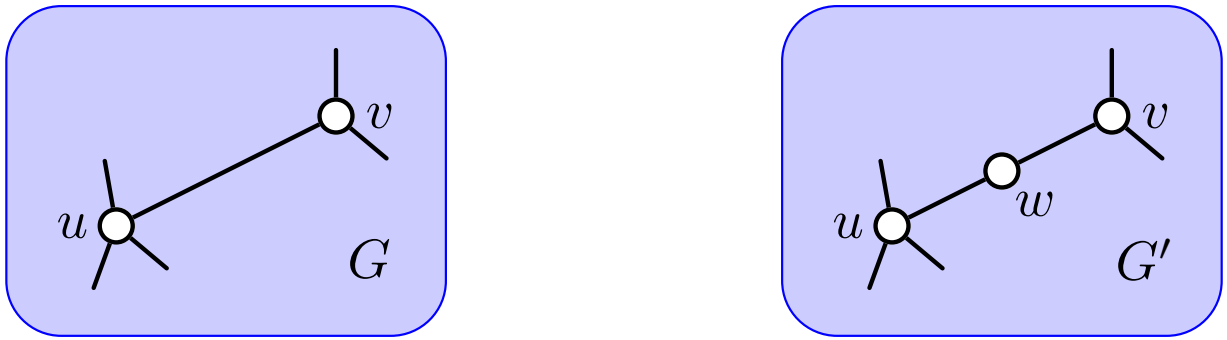
We introduce the notion of subdivision of a graph, that is a graph with similar structure, with some vertices of degree 2 added.

Definition

A subdivision of a graph G is a graphs that is obtained by replacing some edges by internally-disjoint paths.

We replace the edge uv of a graph G by a pair of edges uw and wv. We obtain a new graph G', which is a *subdivision* of the original graph G.

$$G' = (V(G) \cup \{w\}, (E(G) \setminus \{uv\}) \cup \{uw, wv\})$$

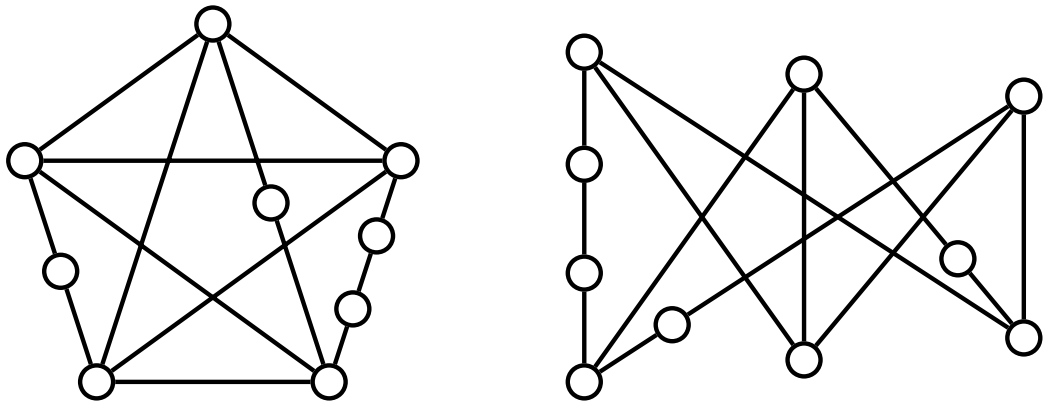


Graph G with a selected edge uv and a subdivision G' of graph G.

In 1930 K. Kuratowski proved the following simple theorem.

Theorem

Graph G is planar if and only if it does not contain a subgraph isomorphic to a subdivion of K_5 or $K_{3,3}$.



A subdivision of graphs K_5 and $K_{3,3}$.

It can be shown, that there exists a "nice" drawing of every planar graph:

Theorem

Every simple planar graph can be drawn in a plane without crossing edges so that all edges are straight lines.

Graph colorings and graph drawing

One of the best known problems in graph theory is the Four color theorem. Though the formulation is easy, correct solution took more than 100 years.

Four color problem

Given any political map, the regions may be colored using no more than four colors in such a way that no two adjacent (sharing a borderline) regions receive the same color.

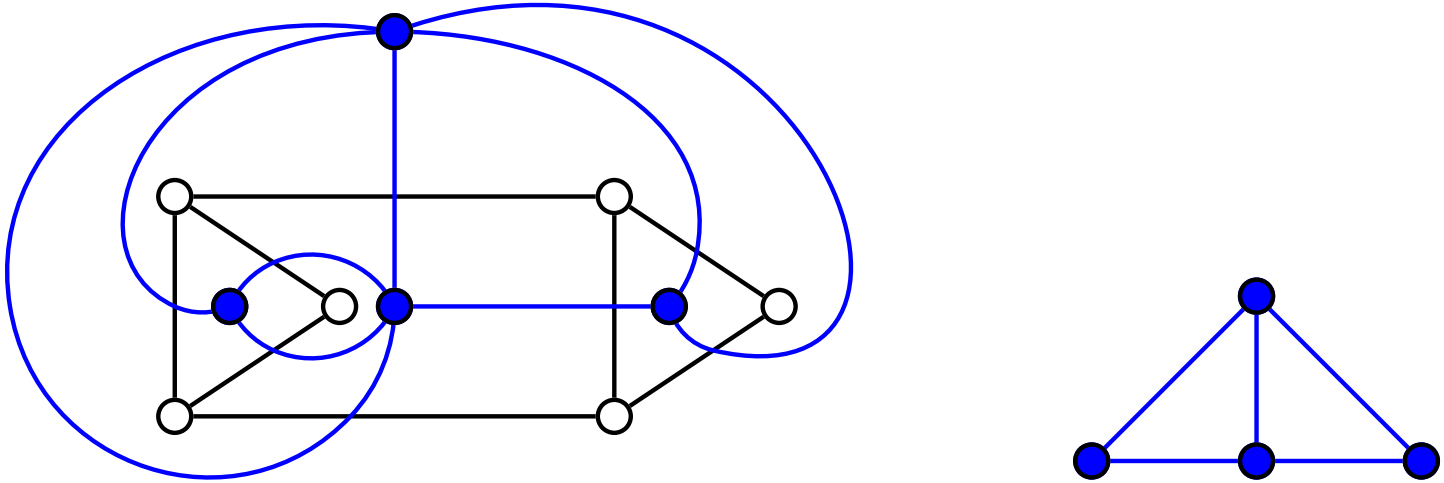
The solution required besides substantial theoretical work also a large scale computer search.

Example

A coloring of a political map can be translated into a proper vertex coloring of a graph.
Each region becomes a vertex (the capital).
Two vertices are joined by an edge if the corresponding states are neighboring.

Definition

Dual graph of a planar graph G we obtain by replacing every region by a vertex. Two vertices of the new graph are connected by an edge if and only if the corresponding regions share an edge.



Graph G with blue dual multigraph and a redrawn dual graph.

It can be shown, that the dual graph to a planar graph is again planar.

The process of transforming a political map into a graph is similar to constructing a dual graph.

In 1976 Appel and Haken, and later in 1993 again Robertson, Seymour, Sanders, and Thomas proved the theorem, which solved the four color problem. It is one of the most famous results in discrete mathematics.

Theorem Four Color Theorem

Every planar graph without loops has a proper coloring by at most 4 colors.

Proof . . . definitely exceed the scope of this course :-) □

But easily we can show a weaker result for 6 colors.

Theorem

Every planar graph can be properly colored by at most 6 color.
Every triangle-free planar graph can be properly colored by at most 4 colors.

Proof We show the second part, the first part is shown in the textbook.

We proceed by induction on the number of vertices of G.

Basis step: The trivial graph with one vertex is surely planar and can be colored by one color.

Inductive step: We have a planar graph with at least two vertices. Suppose all smaller planer graphs we can color by at most four colors. By a previous corollary we find in G a vertex v of degree at most 3. The graph $G - v$ is again planar and triangle-free. By the induction hypothesis we can color the graph $G - v$ by at most four colors. At most three of them will be used to color the neighbors of v, thus *always* there is a fourth color available to color v. □

Notice, the proof is constructive – we can find such coloring.

Next chapter

Chapter Flow in a network

- motivation
- definition of a network
- maximal flow algorithm
- network generalization
- further applications

Discrete mathematics

Petr Kovář
petr.kovar@vsb.cz

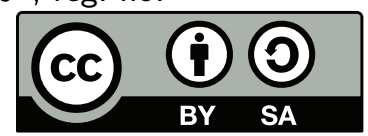
VŠB – Technical University of Ostrava

Winter term 2021/2022
DiM 470-2301/02, 470-2301/04, 470-2301/06

EUROPEAN UNION
European Structural and Investment Funds
Operational Programme Research,
Development and Education

MINISTRY OF EDUCATION,
YOUTH AND SPORTS

The translation was co-financed by the European Union and the Ministry of Education, Youth and Sports from the Operational Programme Research, Development and Education, project "Technology for the Future 2.0", reg. no. CZ.02.2.69/0.0/0.0/18_058/0010212.



About this file

This file is meant to be a guideline for the lecturer. Many important pieces of information are not in this file, they are to be delivered in the lecture: said, shown or drawn on board. The file is made available with the hope students will easier catch up with lectures they missed.

For study the following resources are better suitable:

- Meyer: Lecture notes and readings for an http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-042j-mathematics-for-computer-science-fall-2005/readings/" (weeks 1-5, 8-10, 12-13), MIT, 2005.
- Diestel: Graph theory http://diestel-graph-theory.com/ (chapters 1-6), Springer, 2010.

See also http://homel.vsb.cz/~kov16/predmety_dm.php

Lecture overview

Chapter Flow in a network

- motivation
- definition of a network
- maximum flow algorithm
- network generalization
- further applications

Flow in a network

Motivation Graph theory solves many problems on networks. We are given a network (computer network, pipelines, . . .), where edges represent connections and vertices form crossings or routers.
It is natural to give a bound on capacity of each edge (capacity = number).

Question

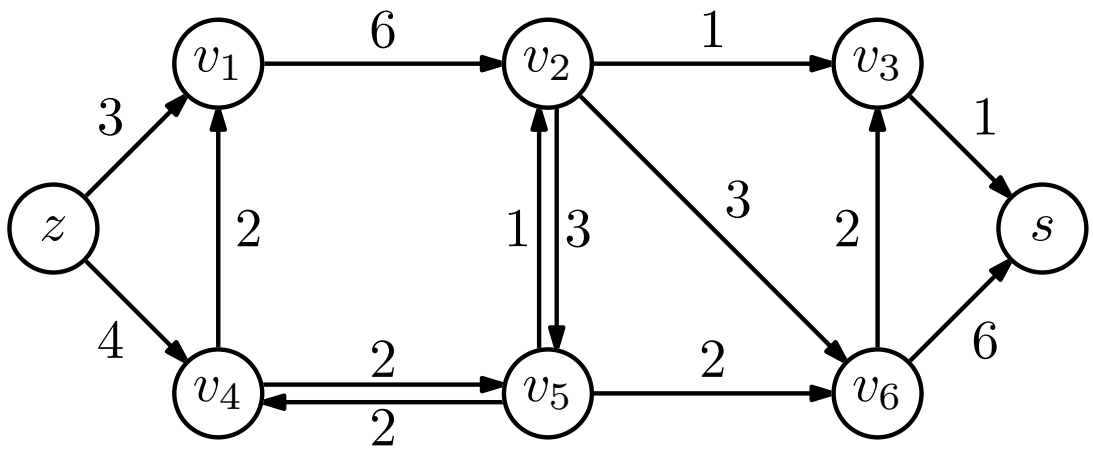
What is the largest possible number of units that can be transfered through the network (with given constraints) from z (source) to s (sink).

A network is a graph in which the capacities, the source and sink are given.

Definition

Network is a four-tuple $S = (G, z, s, w)$, where

- G is an oriented graph,
- vertices $z \in V(G)$, $s \in V(G)$ are the source and sink,
- $w : E(G) \to \mathbb{R}^+$ is a positive labeling of edges, called edge capacity.



Question

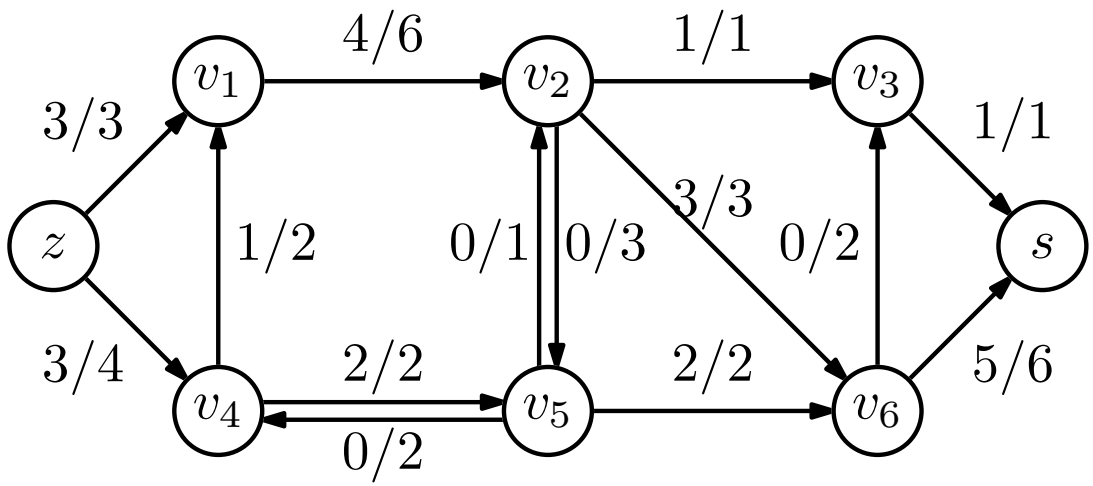
What is the largest possible number of units that can be transfered through the network G from the source z to the place of consumption s (sink). Of course obeying the *maximal capacity* of each edge.

A more complex problem is a network with

- given capacities of vertices
- multiple sources and sinks
- more products to be transfered in a network

In some cases this complex problem can be translated into to basic network defined above. We show how.

Notice that even for a maximum flow through a network the full capacities of all edges do not have to be achieved. This is when we define the flow correctly (flow/capacity).



Note

The edge capacity does not really have to be a capacity (can represent width, automobiles per minute, thickness of a pipe, resistance. . .) We will use terminology from liquid flows: amount „entering“ and „leaving“ a certain vertex.

By $e \to v$ we denote *incoming* edges to v, by $e \leftarrow v$ *outgoing* edges.

Definition

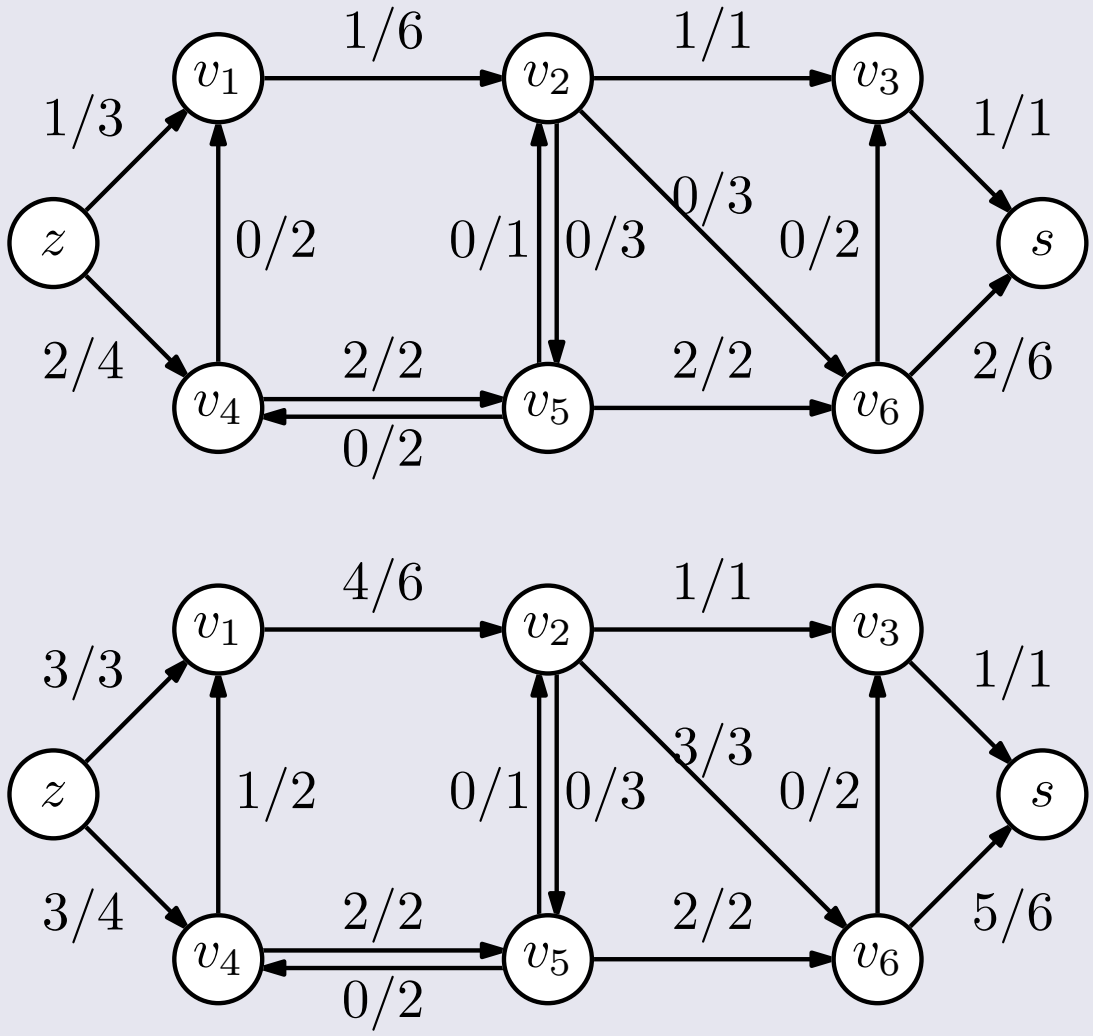
A flow in network $S = (G, z, s, w)$ is a function $f : E(G) \to \mathbb{R}_0^+$, where

- no edge capacity is exceeded: $\forall e \in E(G) : 0 \leq f(e) \leq w(e)$,
- the conservation-of-flow equation hold: $\forall v \in V(G), v \neq z, s : \sum_{e \to v} f(e) = \sum_{e \leftarrow v} f(e)$.

The value of a flow f is

$$\|f\| = \sum_{e \leftarrow z} f(e) - \sum_{e \to z} f(e).$$

Example



Flow and a maximum flow in a network (G, z, s, w).

Source and sink are exceptional vertices in the network.

The conservation-of-flow equations do not hold for them!

- from the source "issues" more than comes in
- the sink "drains" more than comes out

The difference for both these vertices is the same.

Lemma

By f_z we denote the sum of flows on outgoing edges minus the sum of flows on the incoming edges to the source z. By f_s we denote the sum of flows on outgoing edges minus the sum of flows on the incoming edges to s. Then $f_z = -f_s$ holds.

Proof

$$0 = \sum_e (f(e) - f(e)) = \sum_v \sum_{e \leftarrow v} f(e) - \sum_v \sum_{e \rightarrow v} f(e) = \sum_{v \in \{z,s\}} \left(\sum_{e \leftarrow v} f(e) - \sum_{e \rightarrow v} f(e) \right).$$

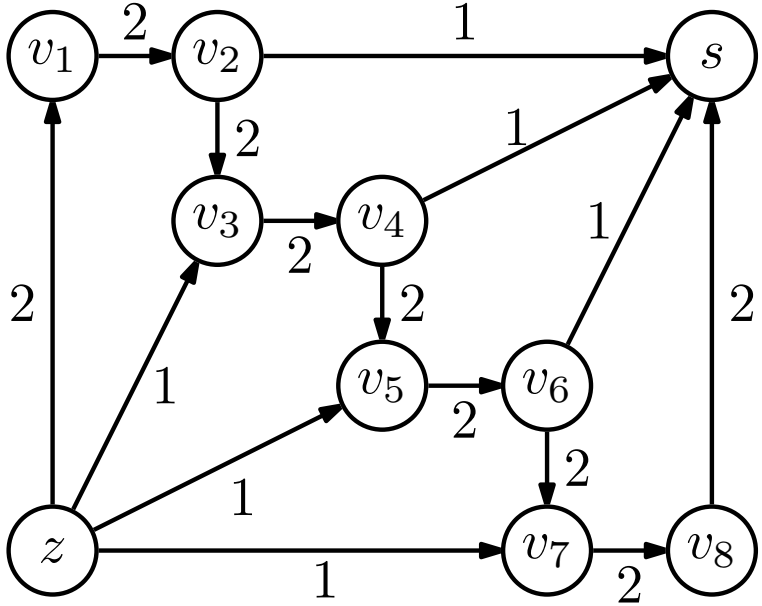
Double sums cancel out for all vertices in the network except z and s.

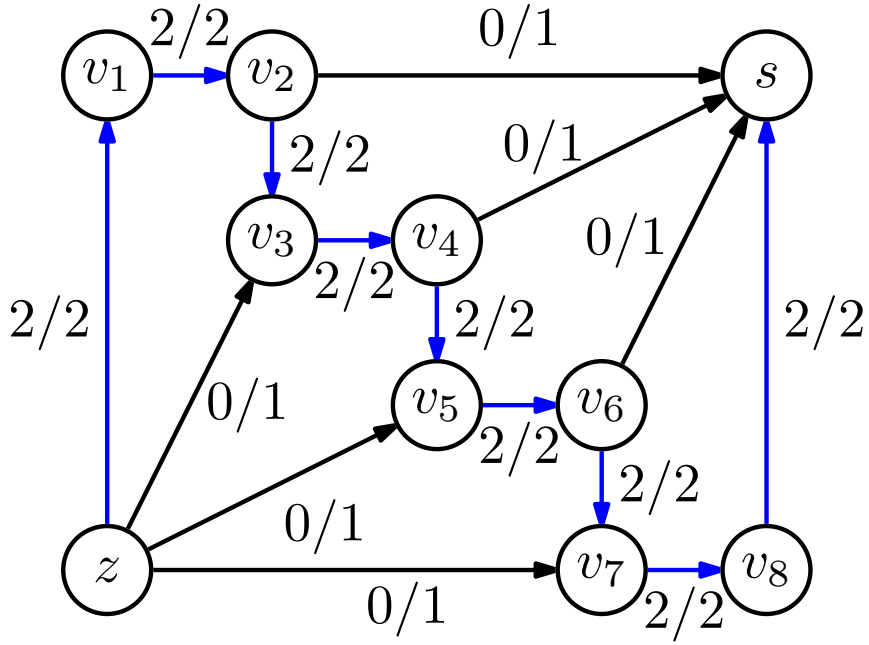
$$\left(\sum_{e \leftarrow z} f(e) - \sum_{e \rightarrow z} f(e) \right) = - \left(\sum_{e \leftarrow s} f(e) - \sum_{e \rightarrow s} f(e) \right). \qquad \square$$

Maximum flow algorithm

Our goal is to find the maximum possible flow from source z to sink s in a network (G) with capacities w.

A greedy algorithm does not give the maximum flow! (see figure)





A greedy algorithm yields a total flow of value 2.

The flow cannot be increased by simply adding some path with non-zero flow, but this is not the maximum flow. There exists a flow of value 5.

Definition

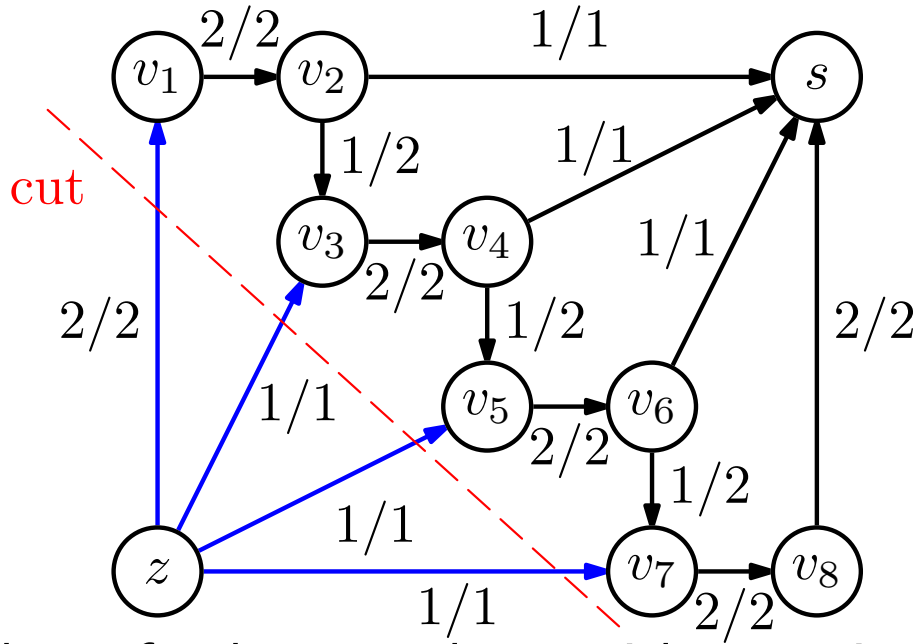
A cut in a network $S = (G, z, s, w)$ is such a subset of edges $C \subseteq E(G)$, that in $G - C$ (G with edges of C deleted) no oriented path from z to s remains.

Capacity of the cut C is the sum of capacities of edges in C, i.e. $\|C\| = \sum_{e \in C} w(e)$.

Theorem

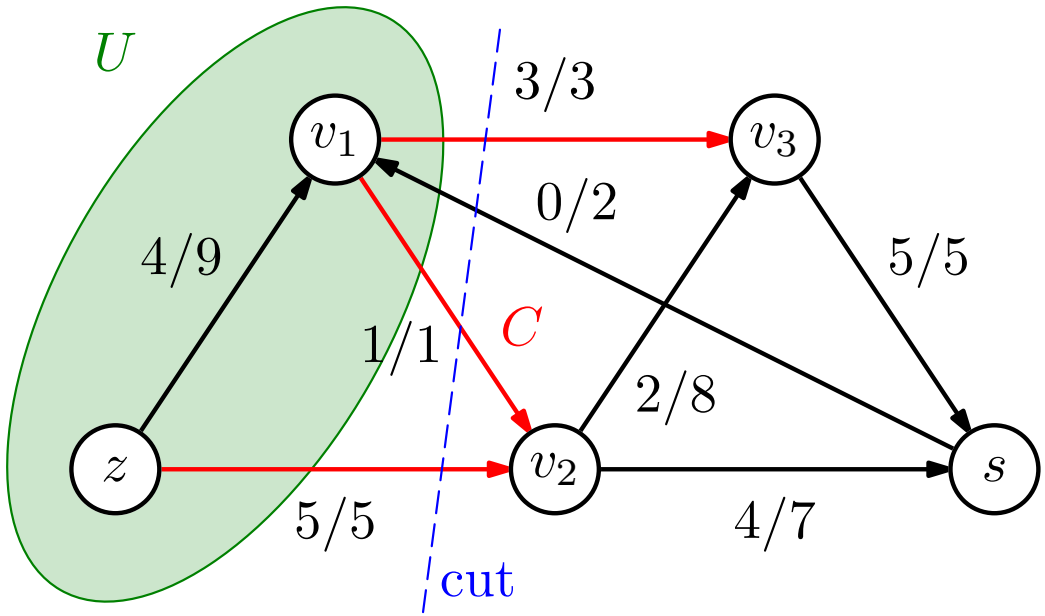
Value of the maximum flow equals the capacity of the minimum cut.

Proof later...



Flow of value 5 *and cut with capacity* 5.

Beware, one has to know what is a cut!



A cut contains only edges leading out of set U.

The theorem characterizes nicely the maximum flow:

The flow of value x is maximum, if there is a cut of capacity x.

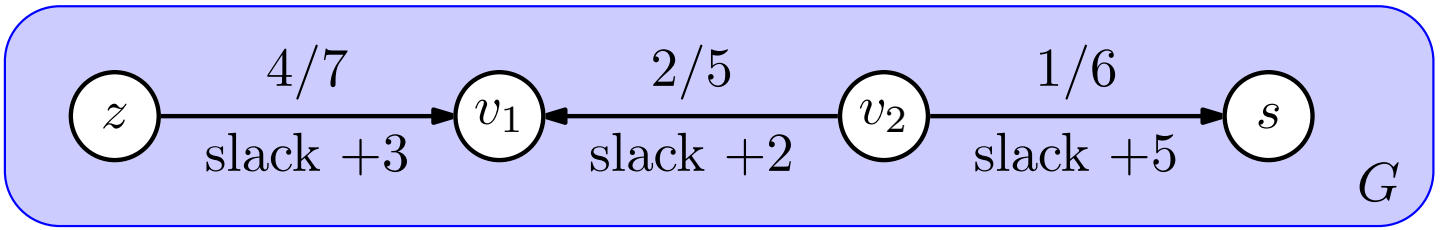
Definition

Let S be a network and let f be a flow in this network. An unsaturated path in S is an unoriented path $e_1, e_2, \ldots, e_m$ in G from vertex u to vertex v (usually from z to s), where

- $f(e_i) < w(e_i)$ for e_i oriented "along" the path from u to v,
- $f(e_i) > 0$ for e_i oriented the opposite way.

The value $w(e_i) - f(e_i)$ for edges e_i oriented from u to v and the value $f(e_i)$ for edges e_i in the opposite way is called the slack of the edge e_i.

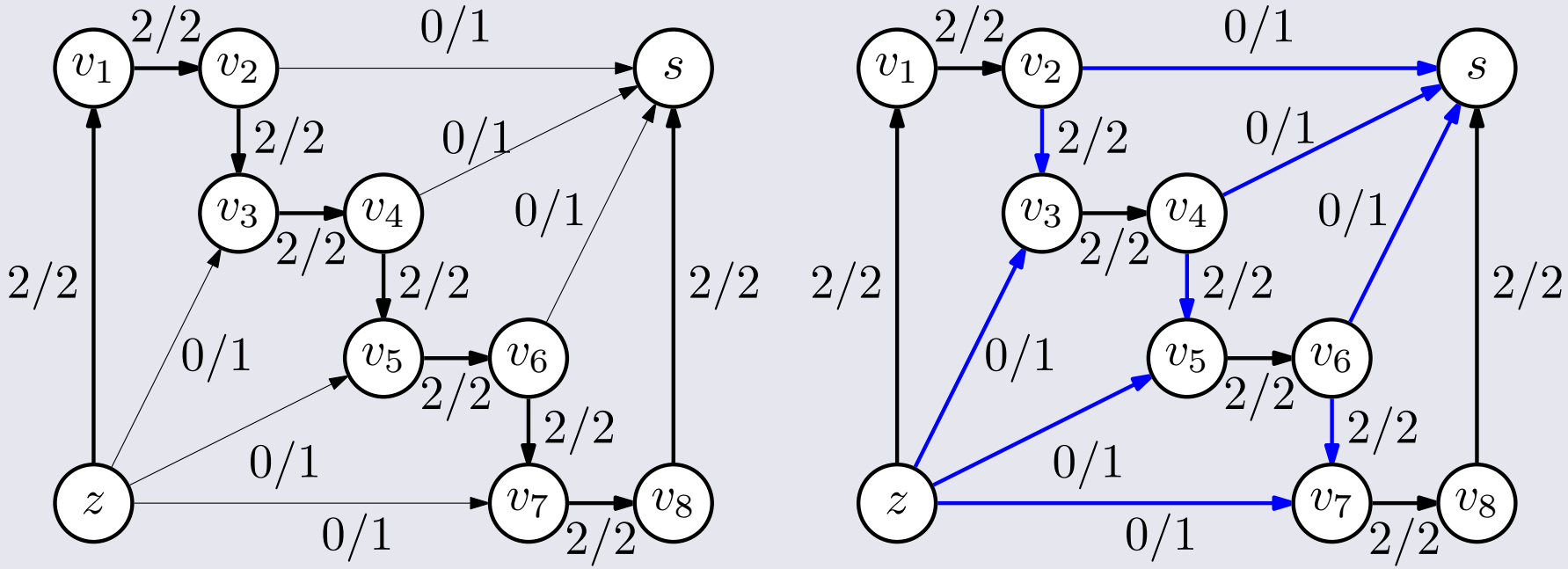
An unsaturated path has positive slacks δ on all edges.



Path with slack 2.

Example

Find several unsaturated paths in the given network.



Example of three unsaturated paths.

Algorithm Ford–Fulkerson's algorithm

```
input:  network S = (G,z,s,w);
initial flow is zero on every edge;

do {
    searching through G find the set U of all vertices
    reachable from z on unsaturated paths in G;
    if (s in U) {
       P = unsaturated path (found above) in S from z to s;
       increase the flow f by the slack of P;
} while (s in U);
output:  print the maximum flow f;
output:  print the min. cut as all edges from U to V(G)-U.
```

Proof We give a direct proof.

Obviously holds $\|f\| \leq \|C\|$.
If at the end of the algorithm we have a flow f and in network S there is a cut of the same capacity $\|C\| = \|f\|$, obviously we have the maximum possible flow in the network S. At the same time we prove Theorem on maximum flow.
It is enough to show that at the end of the algorithm the equality $\|f\| = \|C\|$ holds, where C is the cut between U and the remaining vertices in G.

Suppose we have a flow f in S and no unsaturated path from z to s exists. Thus the set U from the algorithm does not contain s (it is not reachable via unsaturated paths).

Since from U lead no unsaturated paths (nor edges), has every edge $e \leftarrow U$ (outgoing from U) full capacity $f(e) = w(e)$ and each edge $e \rightarrow U$ (incoming to U) flow $f(e) = 0$. The value of the flow f from z to s is

$$\|f\| = \sum_{e \leftarrow U} f(e) - \sum_{e \rightarrow U} f(e) = \sum_{e \leftarrow U} f(e) - 0 = \sum_{e \in C} w(e) = \|C\|.$$

This completes the proof. □

The algorithm finds the maximum flow with value $\|f\|$. Moreover, after the algorithm stops, one can easily find the minimum cut with value $\|C\|$. First we give the following observation:

Corollary

If all capacities in the network S have nonnegative integer values, has the maximum flow also integer value.

The maximum flow always fully saturates the edges of some edge cut, thus the value of such flow equals the sum of capacities of this edge cut. Especially, if all weighs are integer values, so will be the maximum flow.

Note

We point out that omitting the requirement on integer capacities one can come up with examples of simple networks with irrational capacities for which Ford-Fulkerson algorithm does not stop after finitely many steps.

Moreover, the iterated flow does not have to converge toward the maximum flow.

Generalizations of networks and their applications

The method shown above we can generalize to

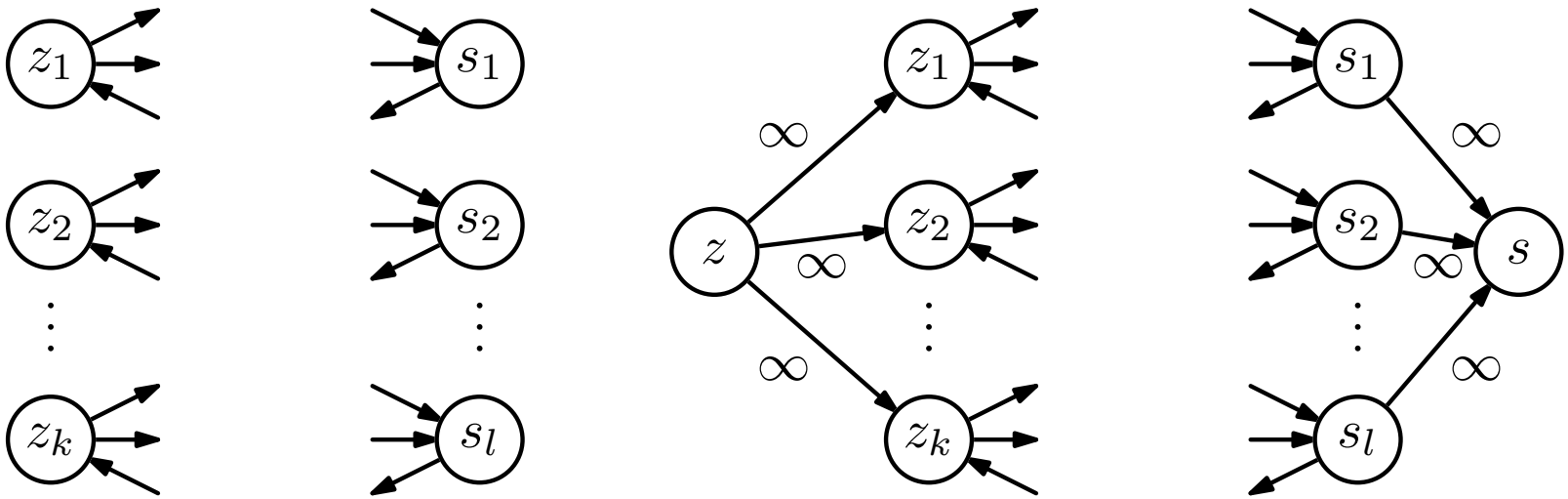
1. multiple sources and sinks,
2. allow unoriented edges in a network,
3. introduce capacities of vertices,
4. transport several products in one network,
5. pose minimal capacities on edges (something has to flow).

Instead of providing new of modifying the previous algorithms for each of the problems, we show how to modify the network.

The more complex problems will be transformed to the basic problem for which Ford-Fulkerson's algorithm can be used.

1) Multiple sources and sinks

If there are multiple sources or sinks in the network, we can transform the problem easily to a single source/sink problem.

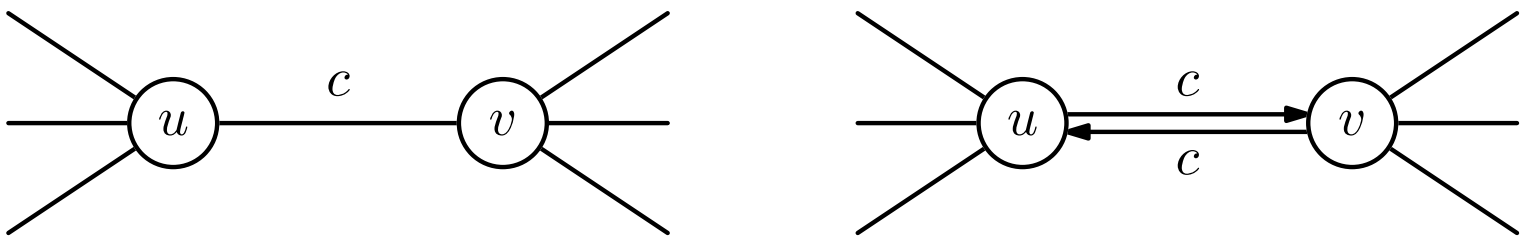


One can also introduce the source/sink capacities by taking the corresponding edge capacities instead of ∞.

2) Unoriented edges

In some real life application the orientation of edges is given (e.g. in sewage system, traffic corridors in road networks, . . .) In other applications the orientation of edges can be arbitrary (information or computer networks). Ford-Fulkerson algorithm is designed for oriented graphs.

Unoriented edges can be simply represented by a pair of edges with the same capacity and opposite orientation.

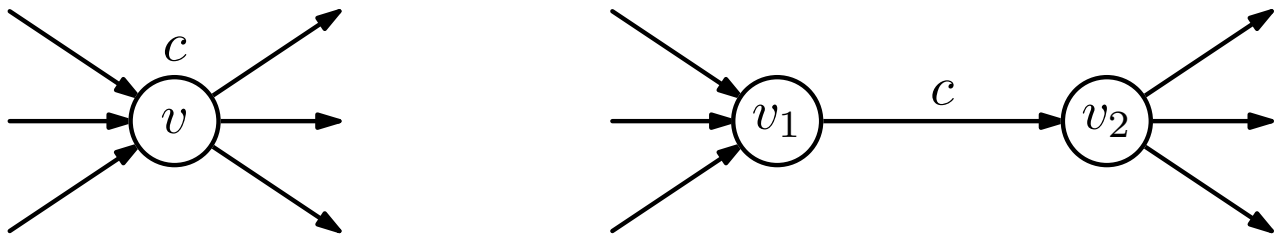


Yet, once the algorithm stops, the flow on the unoriented edge is given by the difference of flows on both oriented edges.

3) Vertex capacities

Naturally, constraints can arise not only for vertices, but also for vertices (crossings, nodes).

A network with vertex capacities can easily be translated into a network where just edges have capacities.



Each vertex with a given capacity c we replace by a pair of vertices joined by an edge with capacity c (we double the vertex).

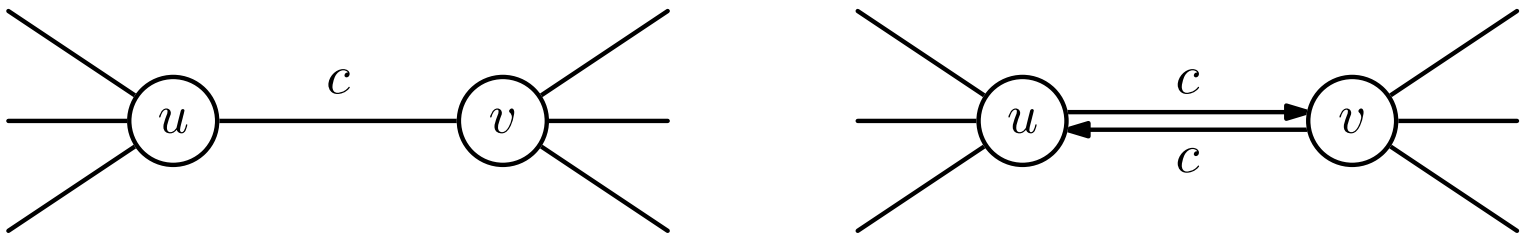
Arcs incoming to v will come into v_1 and arcs outgoing from v will go out from v_2.

4) Several products in one network

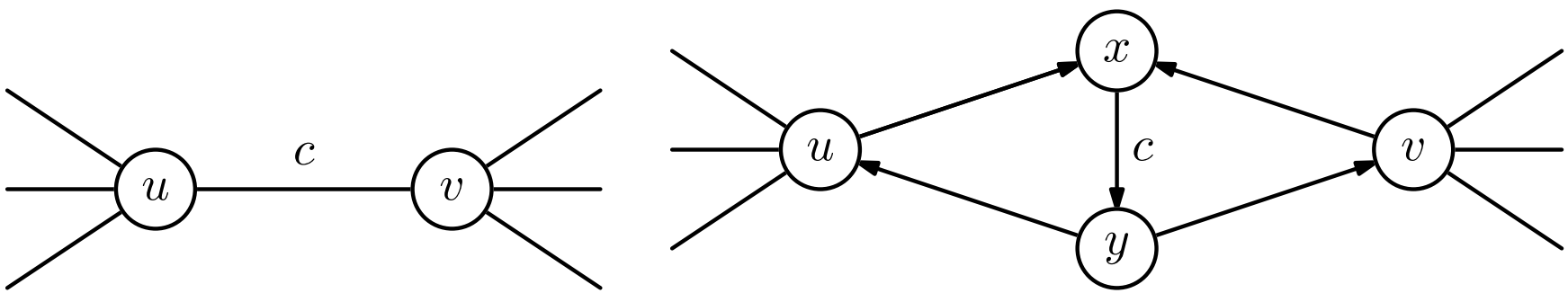
For multiple products transfered in one network the problem is complex. The algorithm for a maximum multi-product flow is beyond this course.

We show, how to translate an unoriented graph into an oriented network with given capacities.

Beware, it is not sufficient to take two opposite arcs:



The sum of flows transfered in one direction and another product transfered in the opposite direction must not exceed the capacity.



This conversion guarantees the total capacity for several products.

5) Minimal capacities of edges

If there are besides the maximal capacities also minimal capacities given, i.e. there is a nonzero flow required, the solution does not need to exist.

This problem is beyond the scope of this course.

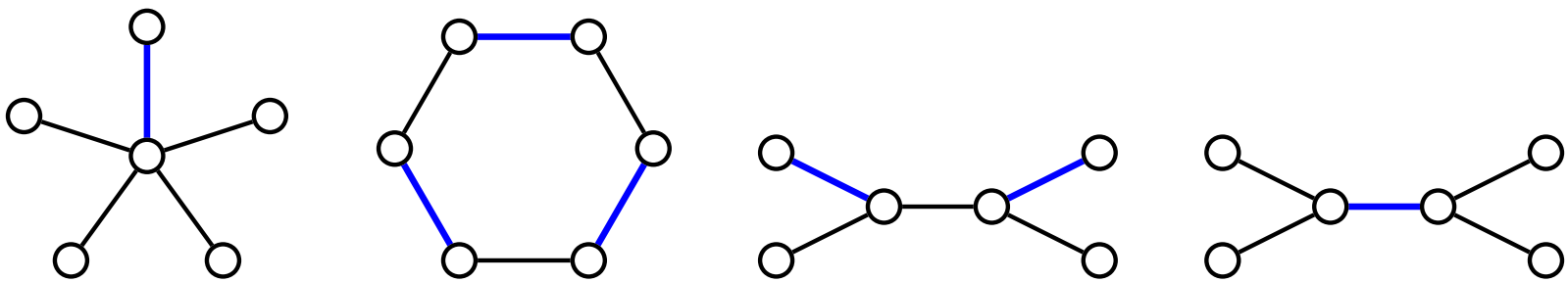
FYI: J. Demel, Grafy, SNTL, Praha, (1989).

Matching in bipartite graphs

There is a surprising variety of applications of the maximum flow algorithm. We show how to translate the search for maximum matching into the search for maximum flow.

Definition

Matching in (bipartite) graph G is a subset of independent edges $M \subseteq E(G)$ (no two edges in M share a vertex).



There are two graphs on six vertices. In the graph on the left the matching has at most one edge, while in the second graph there is a matching that covers all vertices.

Algorithm Maximum matching in a bipartite graph

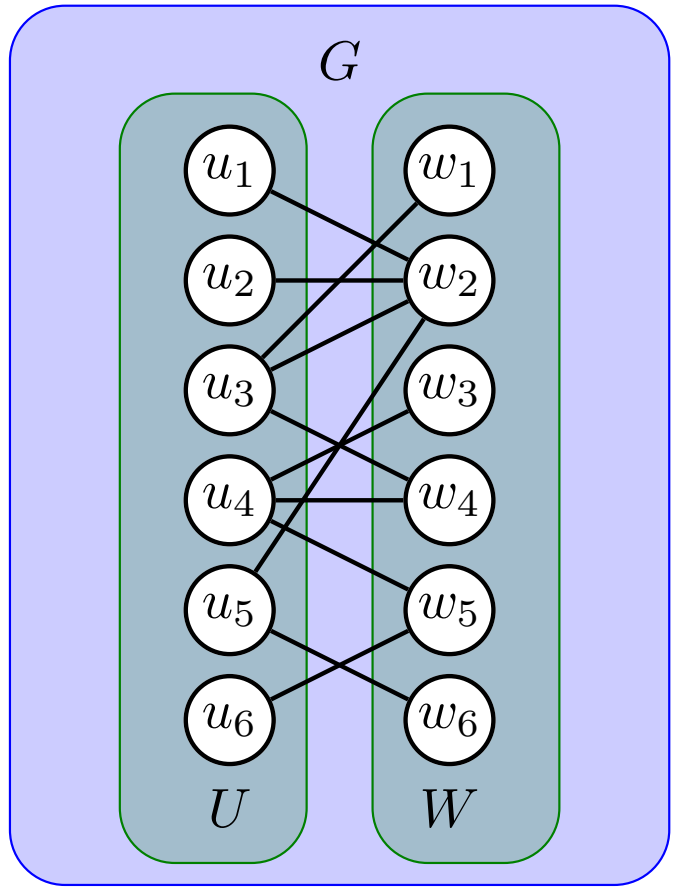
Let G be a bipartite graph vertex set split into two partite sets U and W:

1. we construct a network S: we add the source z and sink s, all vertices in U we join to z and all vertices in W to s, all edges are oriented from the source to the sink; their capacity is 1;
2. we find a (integer) maximum flow in S using previous algorithm;
3. maximum matching in G contains edges with non-zero flow.

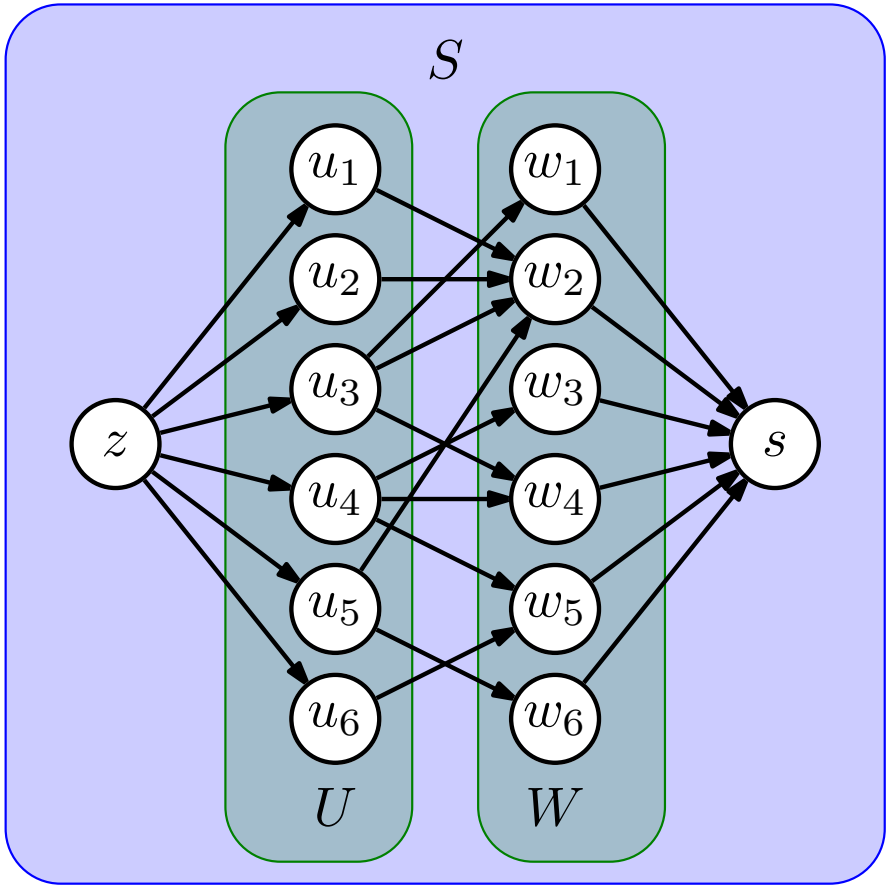
Proof By a corollary the maximum flow will have integer flow, the flow on each edge is either 0 or 1.

Each vertex in U is the end-vertex of precisely one edge with capacity 1, thus each vertex will be in at most one edge of the matching. Similarly for vertices in W. Thus the edges with non-zero flow form a matching, they share no end-vertex.

The matching is maximum. A matching with more edges would correspond to a flow in S with higher value, which leads to a contradiction. □

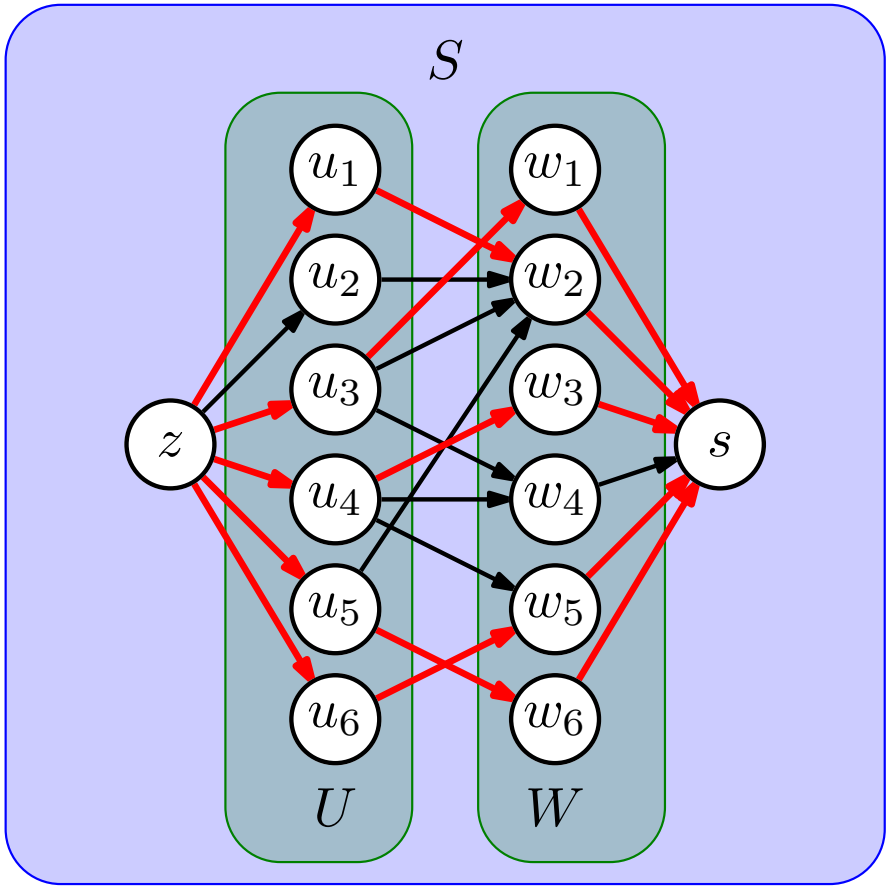


We are given a bipartite graph G.

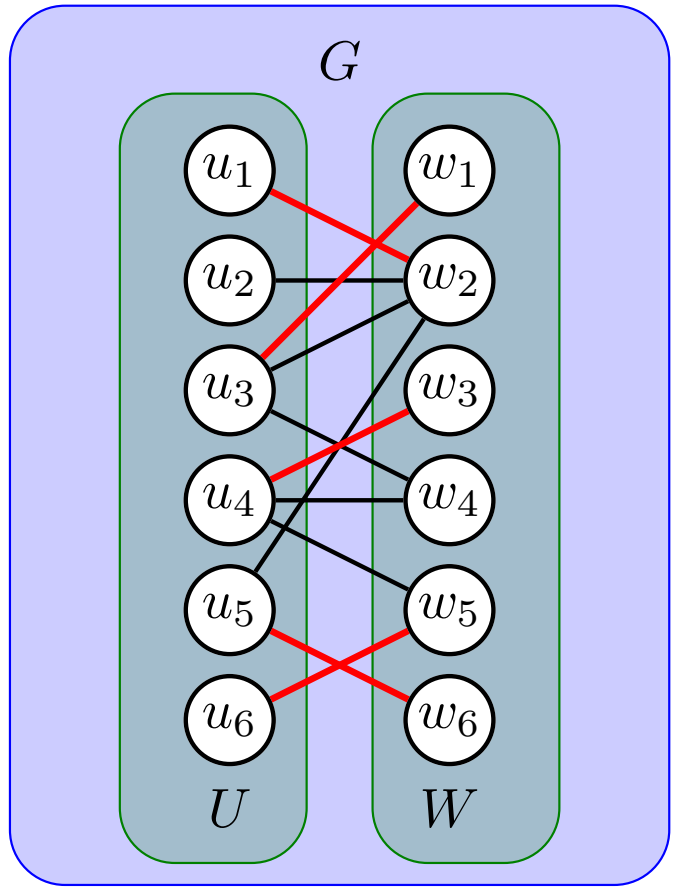


Let us construct the corresponding network S:

- we add a source vertex z and a sink vertex s,
- all vertices in U we join with z and all vertices in W we join with s,
- all edges of the network S are oriented from z to s; all capacities are 1.



Using the Algorithm we find the maximum flow in the network S. By Corollary all values of the maximum flow are integers.



The maximum matching contains just those edges of G with non-zero flow.

k-connectivity

Earlier we defined k-connectivity of graphs. Without proof we stater Menger's Theorem:

Theorem (Menger's theorem)

Graph G is k-edge connected if and only if there are at least k edge-disjoint paths between any two vertices (the paths can share vertices).
Graph G is k-vertex connected if and only if there are at least k internally-disjoint paths between any two vertices (the paths share only end-vertices).

Now we prove the theorem using the algorithm for maximum flow in a network.

First we notice that by definition of (vertex) k-connectivity holds:

Lemma

Let u, v be two vertices in G and $k > 0$ a natural number. Then between u and v there exist in G at least k edge-disjoint paths if and only if after removing any $k-1$ edges remain u and v in the same component.

Proof

"$\Rightarrow$" Follows by the definition of edge k-connectivity of a graph.

"$\Leftarrow$" Let G be a graph and let u and v be any pair of vertices in G. Let vertex u be the source and v the sink, we assign capacity 1 to each edge. Using Ford-Fulkerson's algorithm find the maximum flow from u to v.

The value of the flow is at least k, otherwise the value of the minimum cut is smaller than k. By removing the cut-edges we get a disconnected graph, while there are less than k removed edges which is not possible.

Thus, the edges with flow 1 form different (edge-disjoint) paths from u to v. (The second Mengers Theorem is proven similarly.) □

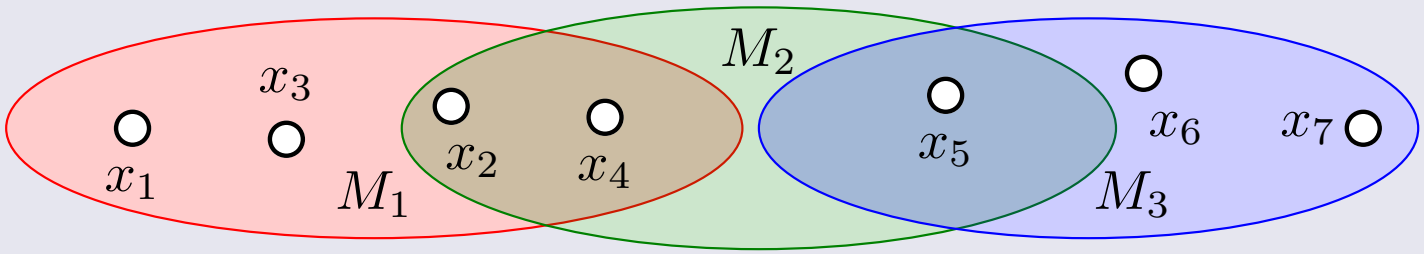
System of distinct representatives

Definition

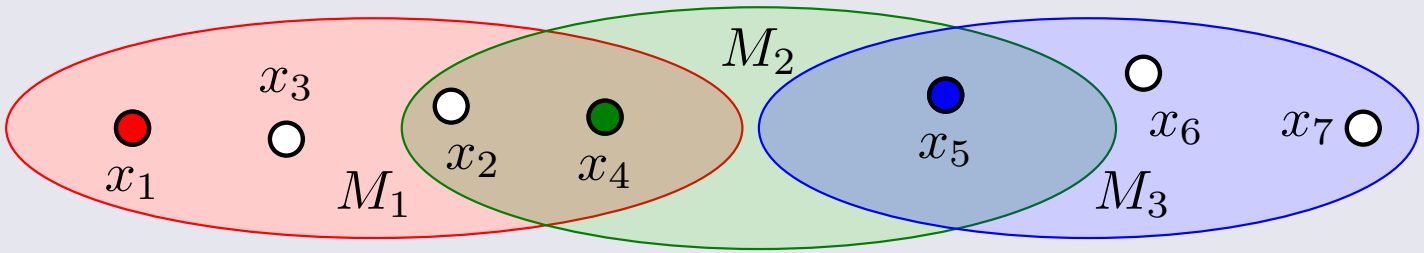
Let $M_1, M_2, \ldots, M_k$ be non-empty sets. System of distinct representatives for $M_1, M_2, \ldots, M_k$ is a sequence of *distinct* elements $(m_1, m_2, \ldots, m_k)$, such that $m_i \in M_i$ for $i = 1, 2, \ldots, k$.

Example

Find the distinct representatives for the given system.



One of the solutions.



An important and well known result is the following theorem

Marriage Theorem

Let $M_1, M_2, \ldots, M_k$ for $k > 0$ be non-empty sets. There exists a system of distinct representatives for these sets if and only if

$$\forall J \subset \{1, 2, \ldots, k\} : \left|\bigcup_{j \in J} M_j\right| \geq |J|,$$

meaning the union of and j sets in this system has at least j elements.

Marriage Theorem gives a sufficient and necessary condition for a set if distinct representatives to exist for a given set-system.

The find the system of representatives is difficult, yet sometimes by choosing a certain collection of sets one can disprove the existence of distinct representatives.

Note

Marriage Theorem is in some literature called „Hall's Theorem".

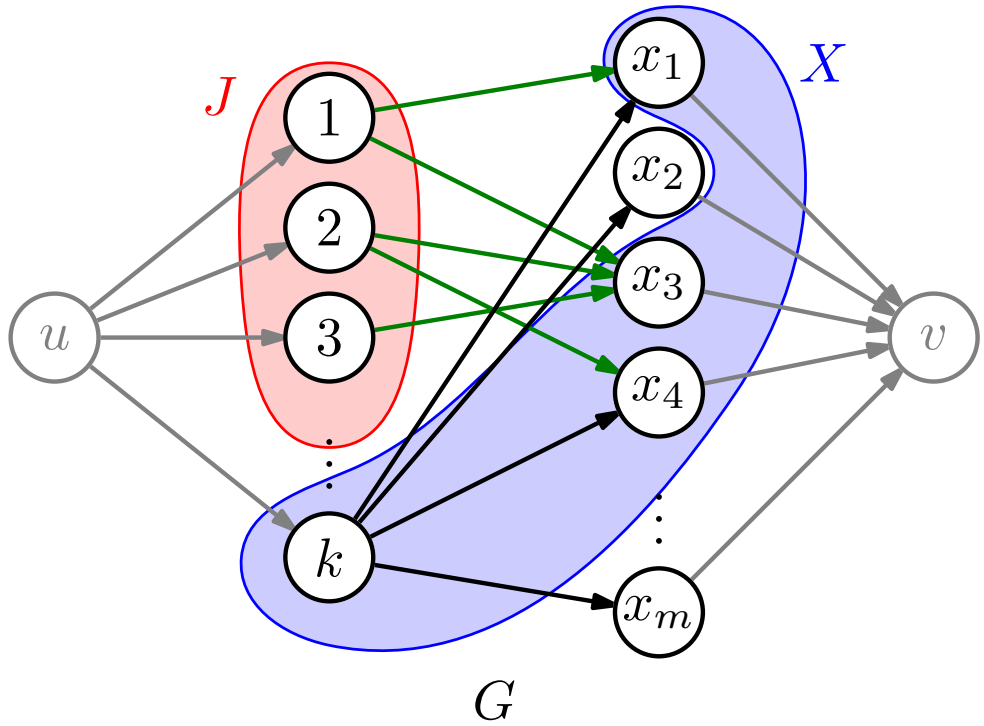
Proof

Denote by $x_1, x_2, \ldots, x_m$ all vertices in the union $M_1 \cup M_2 \cup \cdots \cup M_k$. We define a network S on the vertices $\{1, 2, \ldots, k\} \cup \{x_1, x_2, \ldots, x_m\} \cup \{u, v\}$. Moreover we add edges $\{u, i\}$ for $i = 1, 2, \ldots, k$, $\{x_j, v\}$ for $j = 1, 2, \ldots, m$ and edges $\{i, x_j\}$ for $x_j \in M_i$.

The construction of the network S is analogous as in Algorithm for maximum matching.
Each path from u to v is of the form u, i, x_j, v. It describes each representative $x_j \in M_i$ uniquely. The system of distinct representatives correspond to k vertex-disjoint paths from u to v.

Let X be any minimal subset of vertices in G, such that removing all vertices of X from G no path from u to v remains. By a lemma have all such sets a system of distinct representatives if and only if each such separating set X has et least k elements.

We define $J = \{1, 2, \ldots, k\} \setminus X$.



Now each edge leaving J (besides u) goes to vertices in $X \cap \{x_1, \ldots, x_m\}$, because no path from u to v exists. Thus

$$\left|\bigcup_{j \in J} M_j\right| = |X \cap \{x_1, \ldots, x_m\}| = |X| - |X \cap \{1, \ldots, k\}| = |X| - k + |J|.$$

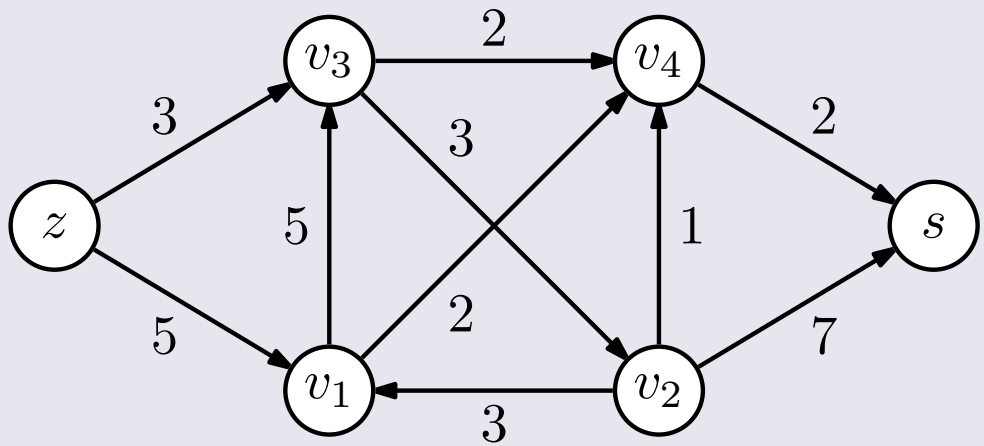
From this $|X| \geq k$ for all (minimal) separating sets X if and only if $\left|\bigcup_{j \in J} M_j\right| \geq |J|$ for all J. This completes the proof. $\square$

Example

We finish the lecture by providing a couple of examples on the maximum flow and minimum cut algorithm.

Example

What is the maximum flow in this network (G, z, s, w)?
And where is the minimum cut in the network?

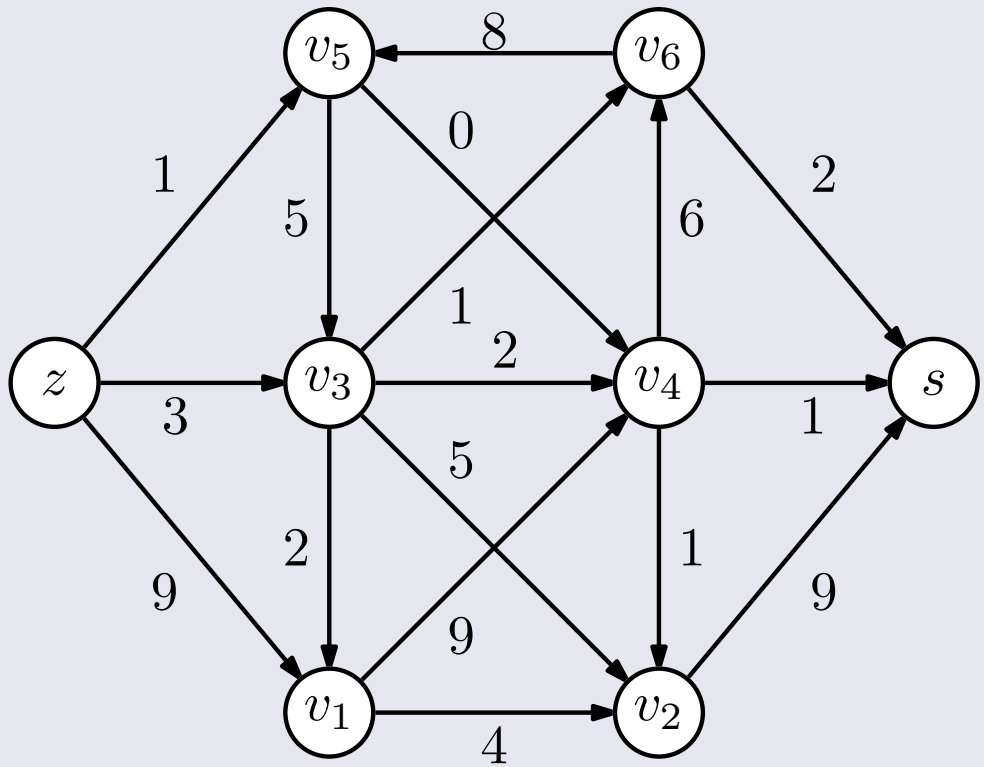


Network (G, z, s, w).

Last example

Example

What is the maximum flow in this network (G, z, s, w)?
And where is the minimum cut in the network?



Network (G, z, s, w).

The end

Thank you for your attention

Please do not forget about the evaluation.
Especially comments.

Exam dates

- “early” exam (?)

Good luck with your exam!