KØBENHAVNS UNIVERSITET

Introduktion til diskret matematik og algoritmer: Problem Set 4

Due: Wednesday March 25 at 12:59 CET.
Submission: Please submit your solutions via *Absalon* as a PDF file. State your name and e-mail address close to the top of the first page. Solutions should be written in LaTeX or some other math-aware typesetting system with reasonable margins on all sides (at least 2.5 cm). Please try to be precise and to the point in your solutions and refrain from vague statements. Never, ever just state the answer, but always make sure to explain your reasoning. *Write so that a fellow student of yours can read, understand, and verify your solutions.* In addition to what is stated below, the general rules for problem sets stated on *Absalon* always apply.
Collaboration: Discussions of ideas in groups of two to three people are allowed—and indeed, encouraged—but you should always write up your solutions completely on your own, from start to finish, and you should understand all aspects of them fully. It is not allowed to compose draft solutions together and then continue editing individually, or to share any text, formulas, or pseudocode. Also, no such material may be downloaded from or generated via the internet to be used in draft or final solutions. Submitted solutions will be checked for plagiarism.
Grading: A score of 120 points is guaranteed to be enough to pass this problem set.
Questions: Please do not hesitate to ask the instructor or TAs if any problem statement is unclear, but please make sure to send private messages—sometimes specific enough questions could give away the solution to your fellow students, and we want all of you to benefit from working on, and learning from, the problems. Good luck!

1 (90 p) When Jakob has international visitors, he needs to give them travel directions from Kastrup Airport to the Department of Computer Science (DIKU) at Universitetsparken. Jakob is aware of the following relevant public transport options in Copenhagen with travel times as stated (in either direction, and with time for switching transport mode included):

- Between Kastrup Airport and Kongens Nytorv by metro: 13 minutes.
- Between Kastrup Airport and København H by train: 20 minutes.
- Between København H and Nørreport by train: 3 minutes.
- Beween København H and Kongens Nytorv by metro: 4 minutes.
- Between Kongens Nytorv and Vibenshus Runddel by metro: 10 minutes.
- Between Vibenshus Runddel and Universitetsparken by foot: 9 minutes.
- Between Nørreport and Universitetsparken by bus: 8 minutes.

What is not so clear to Jakob is how he should use this information to find as fast a route as possible between the airport and DIKU to suggest to his visitors.

1a Help Jakob by modelling this problem as a graph. Explain what the vertices and edges represent and what other information you need to add to the graph. Make sure to show concretely what graph you obtain for Jakob's problem above.

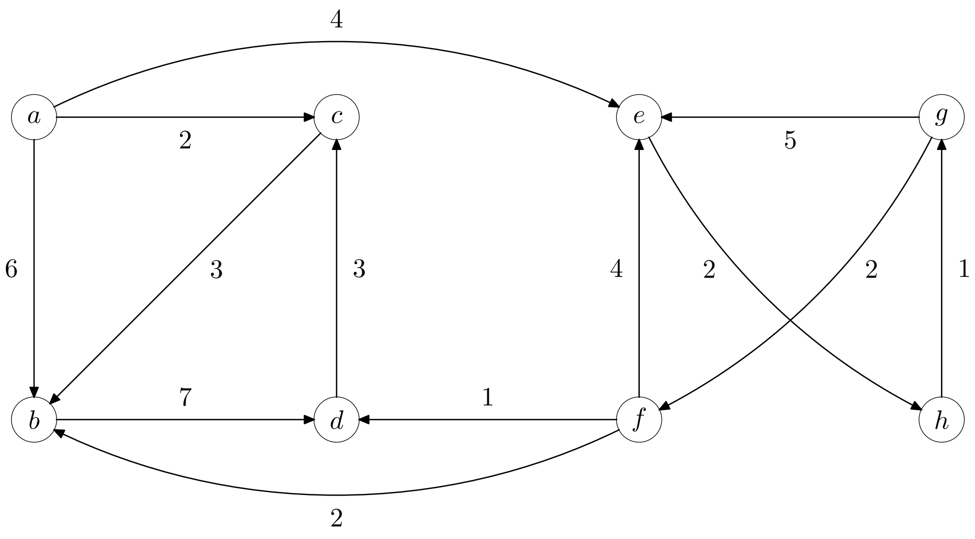


Figure 1: Graph for Problem 2

1b Propose a suitable graph algorithm to solve Jakob's problem. Explain what this algorithm is and why it is the right choice for this problem. Make a dry-run of the algorithm and explain the relevant steps in the execution (similarly to what has been done in class and in the lecture notes).

If your algorithm uses any auxiliary data structures, then explain in detail for the first two vertices processed how these data structures change. For the rest of the algorithm execution, just report what the relevant outputs of the data structures are without going into any details.

What travel directions for Jakob's visitors does your algorithm produce?

2 (120 p) Consider the graph in Figure 1 and the following ordered sequences listing the vertices in this graph:

1. a, c, e, b, h, g, f, d.
2. a, b, c, e, d, h, g, f.
3. a, c, b, e, h, g, f, d.
4. a, b, d, c, e, h, g, f.

For each of the sequences above, determine whether it can be the result of:

(a) a breadth-first search with vertices listed in order of visits;
(b) a depth-first search with vertices listed in order of discovery;
(c) a shortest-path computation with vertices listed in the order they are removed from the priority queue.

Each sequence above is the output of at most one of the algorithms, but since there are four sequences there could be a sequence that cannot be produced by any of the algorithms.

Partial credit is given for matching algorithms and vertex sequences correctly. For full credit, you need to give an overview of how and why the algorithms process the vertices in the given order (such as explaining the order of recursive calls, edges processed, or similar), or why none of the proposed algorithms could yield the sequence in question, but you do not have to provide detailed information about any auxiliary data structures used in the algorithms.

3 (50 p) Back in 2021 during the pandemic, all classes and exams were held virtually, but the Copenhagen Tivoli Gardens amusement park was open for physical visits by the general public. This meant that students frustrated by Zoom lectures and *Digital eksamen* exams were able to head to the *bumper cars* (*radiobiler*) and let go of some steam by trying to bump into as many other cars (or as few cars) as possible.

Suppose that there are $n \geq 2$ bumper cars sharing an area of $m > n^2/2$ square meters.[1] An amazing mathematical fact is that after every bumper car round has finished, there are two different bumper cars that have bumped with exactly the same number of other (distinct) cars. Prove this!

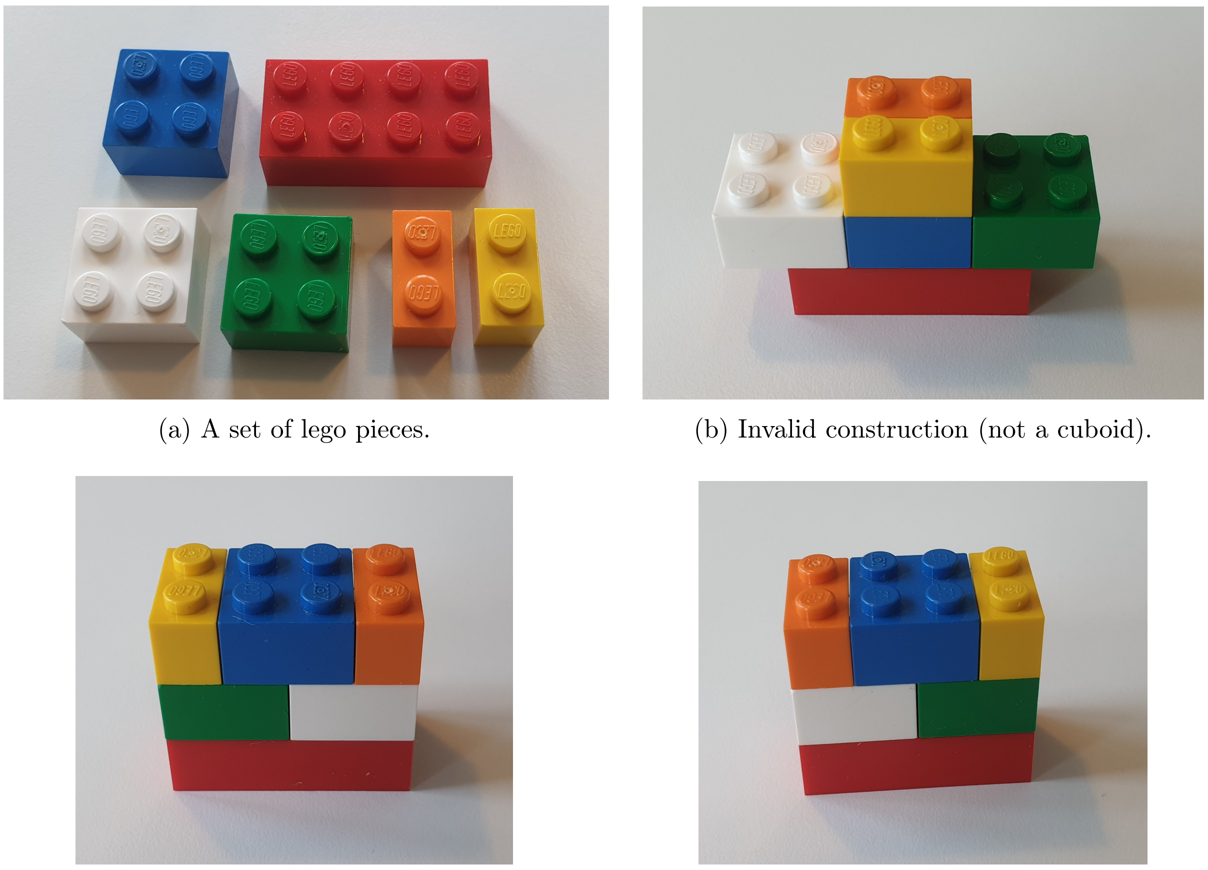
4 (60 p) Another way in which Danish computer science students were able to entertain themselves during the lock-down was to play with legos (or maybe not, but we need a story for this problem, so let us assume they did). In order to make this kind of activity more relevant from the point of view of discrete mathematics, rumour has it that the students often considered different sets of lego pieces (like the one in Figure 2a) and investigated in how many different ways cuboids (rectangular parallelepipeds) can be built from such pieces.

Inspired by this, let us consider the lego pieces in Figure 2a, which as we observe all have different colours. In this particular problem we want to build cuboids that are three layers high. The pieces should be put together in the standard lego way, with the lego studs pointing upwards and being fitted into the holes of any pieces above. All pieces should be used. Constructions like the one in Figure 2b are not valid—this particular construction does have three layers, and the studs point upwards and are fitted into the holes above when possible, but the resulting geometric shape is not a cuboid.

Two cuboids are considered to be the same if they can be rotated so that they become similar. Thus, the constructions in Figures 2c and 2d are both valid, but are considered to be the same cuboid, since rotating Figure 2c yields Figure 2d.

How many *different* cuboids can you build with the lego pieces in Figure 2a satisfying the restrictions described above? Please make sure to explain clearly how you reason to reach your answer.

[1] In reality, there are 18 bumper cars in the Copenhagen Tivoli Gardens sharing an area of 280 square meters, but this is not too relevant for this problem.



(a) A set of lego pieces.

(b) Invalid construction (not a cuboid).

(c) Valid construction of 3-layer cuboid.

(d) Symmetric version of the same cuboid.

Figure 2: Lego pieces and constructions (and non-constructions) of cuboids for Problem 4.