

Departement of Computer Science
Markus Püschel, David Steurer
Johannes Lengler, Gleb Novikov, Chris Wendler

25. November 2019

Algorithms & Data Structures

Exercise sheet 10

HS 19

Exercise Class (Room & TA): _____

Submitted by: _____

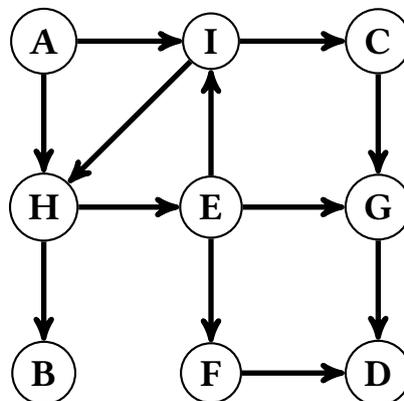
Peer Feedback by: _____

Points: _____

Submission: On Monday, 02 December 2019, hand in your solution to your TA *before* the exercise class starts. Exercises that are marked by * are challenge exercises. They do not count towards bonus points.

Exercise 10.1 *Depth-First Search and Breadth-First Search (1 point).*

Execute a depth-first search (Tiefensuche) on the following graph starting from vertex *A*. Use the algorithm presented in the lecture. When processing the neighbors of a vertex, process them in alphabetical order.



- Mark the edges that belong to the depth-first tree (Tiefensuchbaum) with a “T” (for tree edge).
- For each vertex, give its *pre*- and *post*-number.
- Give the vertex ordering that results from sorting the vertices by pre-number. Give the vertex ordering that results from sorting the vertices by post-number.
- Mark every forward edge (Vorwärtskante) with an “F”, every backward edge (Rückwärtskante) with an “B”, and every cross edge (Querkante) with a “C”.
- Does the above graph have a topological ordering? How can we use the above execution of depth-first search to find a directed cycle?

- f) Draw a scale from 1 to 18, and mark for every vertex v the interval I_v from pre-number to post-number of v . What does it mean if $I_u \subset I_v$ for two different vertices u and v ?
- g) Consider the graph above with the edge from I to H removed. How does the execution of depth-first search change? Which topological sorting does the depth-first search give? If you sort the vertices by *pre-number*, does this give a topological sorting?
- h) Consider the graph above with the edge from I to H removed. Give the order in which the vertices are enqueued by a breadth-first search starting in A . You should list vertices several times if they are enqueued more than once. When processing the neighbors of a vertex, process them in alphabetical order.

The BFS-order is the order in which the vertices are enqueued for the first time in a breadth-first search. What is the BFS-order of the above execution? Does it give a topological sorting?

Exercise 10.2 *Graph Coloring (1 point).*

A k -coloring of a graph $G = (V, E)$ is a mapping $c : V \rightarrow \{1, \dots, k\}$ that assigns a color to each vertex such that $c(v_i) \neq c(v_j)$ if $\{v_i, v_j\} \in E$. A two-coloring and three-coloring is a coloring with 2 and 3 colors, respectively.

For $n > 2$ consider the cycle graph G with $V = \{v_1, \dots, v_n\}$ and edges

$$E = \{\{v_i, v_{i+1}\}, \text{ for } i \in \{1, \dots, n-1\}\} \cup \{\{v_1, v_n\}\}.$$

- a) Show that for even $n > 2$ there exists a two-coloring of the cycle.
- b) Show that for odd $n > 2$ there does not exist a two-coloring of the cycle.
- c) Show that for arbitrary $n > 2$ there exists a three-coloring of the cycle.
- d) Come up with an example graph for which there does not exist a three-coloring.

Exercise 10.3 *Maze solver (1 point).*

You are given a maze that is described by a $n \times n$ grid of blocked and unblocked cells (see Figure 1). There is one start cell marked with 'S' and one target cell marked with 'T'. Starting from the start cell your algorithm may traverse the maze by moving from unblocked fields to adjacent unblocked fields. The goal of this exercise is to devise an algorithm that given a maze returns the best solution (traversal from 'S' to 'T') of the maze. The best solution is the one that requires the least moves between adjacent fields.

Hint: You may assume that there always exists at least one unblocked path from 'S' to 'T' in a maze.

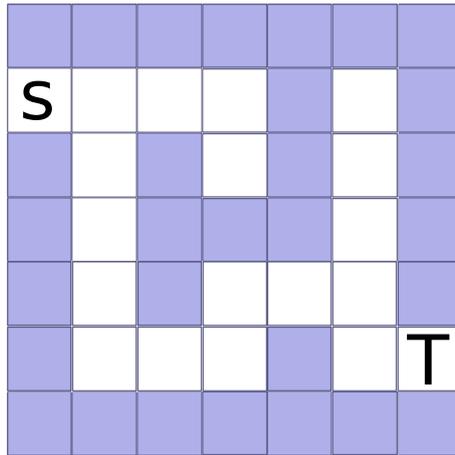


Figure 1: An example of 7×7 maze in which purple fields are blocked, white fields can be traversed (are unblocked). The start field is marked with 'S' and the target field with a 'T'.

- Model the problem as a graph problem. Describe the set of vertices V and the set of edges E in words. Reformulate the problem description as a graph problem on the resulting graphs.
- Choose a data structure to represent your maze-graphs and use an algorithm discussed in the lecture to solve the problem.
Hint: If there are multiple solutions of the same quality, return any one of them.
- Determine the running time and memory requirements of your algorithm in terms of n in Θ notation.

Exercise 10.4 Soapbox race.

You participate in a soapbox race. In contrast to a conventional race you are allowed to use the entire street network. The street network is modeled as a directed graph $G = (V, E)$. Each vertex $v \in V$ corresponds to a crossroad and there is an edge (v, w) in E if there is a street from v to w with no crossroads in between. Additionally, for each edge (v, w) we know that v 's altitude is strictly higher than w 's. Your soapbox is not motorized. Using the street from (v, w) requires $f((v, w))$ seconds, for each $(v, w) \in E$.

In this exercise you can assume that the directed graph is represented by a data structure that allows you to traverse the direct successors and direct predecessors of a vertex u in time $\mathcal{O}(\deg_+(u))$ and $\mathcal{O}(\deg_-(u))$ respectively, where $\deg_-(u)$ is the in-degree of vertex u and $\deg_+(u)$ is the out-degree of vertex u .

- Provide a *dynamic programming* algorithm that, given the graph G , the time function $f : E \rightarrow \mathbb{R}^+$, a start node s and a target node t , determines the time required by the fastest path from s to t . If there is no path from s to t , you can assume that the time is ∞ .
 - Dimension of the DP table:* What is the dimension of the table $DP[. . .]$? What range do you have in each dimension?
 - Definition of the DP table:* What is the meaning of each entry?
 - Computation of an entry:* How can an entry be computed from the values of other entries? Specify the base cases, i.e., the entries that do not depend on others.

- (d) *Calculation order*: In which order can entries be computed so that values needed for each entry have been determined in previous steps?
- (e) *Extracting the solution*: How can the final solution be extracted once the table has been filled?
- (f) *Running time*: What is the runtime of your solution?

Specifically, you can use the following scheme:

Dimensions of the table:

Meaning of a table entry (in words):

Computation of an entry (initialization and recursion):

Order of computation:

Computing the output:

Running time in \mathcal{O} -notation in terms of $|V|$ and $|E|$:

- b) Finally, assume you have already filled out the DP table. Describe how you can find the fastest path using the DP table.

***Hint:** If there are multiple fastest paths return only one of them. If there is no paths from s to t , you should output the message that there is no path from s to t .*