Eidgenössische Technische Hochschule Zürich
Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Federal Institute of Technology at Zurich

Departement of Computer Science
Markus Püschel, David Steurer
Gleb Novikov, Tommaso d'Orsi, Ulysse Schaller, Rajai Nasser

29. November 2021

Algorithms & Data Structures — Exercise sheet 10 — HS 21

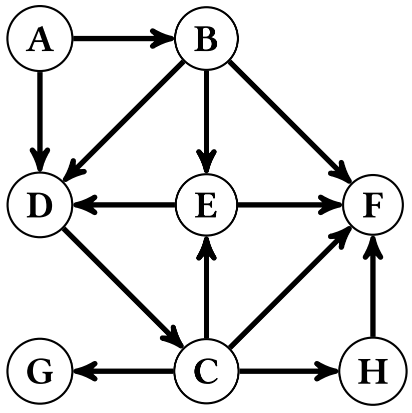
Exercise Class (Room & TA):	
Submitted by:	
Peer Feedback by:	
Points:	

Submission: On Monday, 6 December 2021, 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 *Breadth-First Search* **(1 point)**.

Execute a breadth-first search (Breitensuche) on the following (directed) graph starting from vertex A. Use the algorithm presented in the lecture.

When processing the neighbors of a vertex, process them in alphabetical order.



a) Provide the BFS order of visit of the nodes.

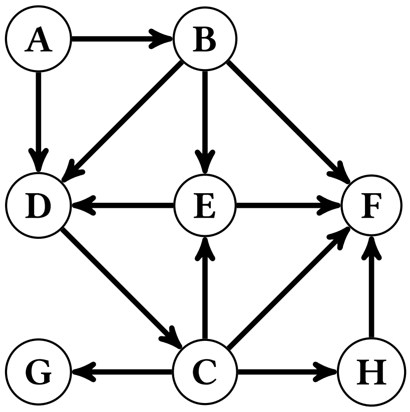
b) Provide the enter and leave time of each node.

c) Indicate the shortest-path-tree that is obtained by BFS.

d) Determine the distance from A to every node in the graph.

Exercise 10.2 *Depth-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.



a) Mark the edges that belong to the depth-first tree (Tiefensuchbaum) with a “T” (for tree edge).

b) For each vertex, give its *pre*- and *post*-number.

c) 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.

d) Mark every forward edge (Vorwärtskante) with an “F”, every backward edge (Rückwärtskante) with an “B”, and every cross edge (Querkante) with a “C”.

e) 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 16, 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 E to D 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?

Exercise 10.3 *Data structures for graphs.*

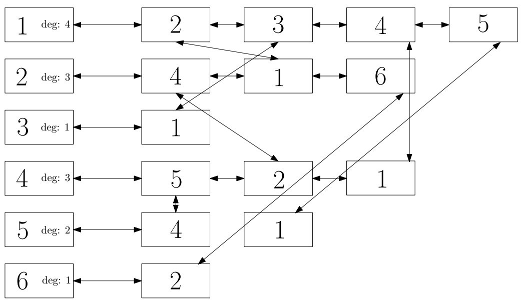
Consider three types of data structures for storing a graph G with n vertices and m edges:

a) Adjacency matrix.

b) Adjacency lists:

$$\begin{tikzcd}
\boxed{1} \arrow[r] & \boxed{2} \arrow[r] & \boxed{3} \arrow[r] & \boxed{4} \arrow[r] & \boxed{5} \\
\boxed{2} \arrow[r] & \boxed{4} \arrow[r] & \boxed{1} \arrow[r] & \boxed{6} & \\
\boxed{3} \arrow[r] & \boxed{1} & & & \\
\boxed{4} \arrow[r] & \boxed{5} \arrow[r] & \boxed{2} \arrow[r] & \boxed{1} & \\
\boxed{5} \arrow[r] & \boxed{4} \arrow[r] & \boxed{1} & & \\
\boxed{6} \arrow[r] & \boxed{2} & & &
\end{tikzcd}$$

c) Adjacency lists, and additionally we store the degree of each node, and there are pointers between the two occurences of each edge. (An edge appears in the adjacency list of each endpoint).



For each of the above data structures, what is the required memory (in Θ-Notation)?

Which runtime (worst case, in Θ-Notation) do we have for the following queries? Give your answer depending on n, m, and/or $\deg(u)$ and $\deg(v)$ (if applicable).

(i) Input: A vertex $v \in V$. Find $\deg(v)$.

(ii) Input: A vertex $v \in V$. Find a neighbour of v (if a neighbour exists).

(iii) Input: Two vertices $u, v \in V$. Decide whether u and v are adjacent.

(iv) Input: Two adjacent vertices $u, v \in V$. Delete the edge $e = \{u, v\}$ from the graph.

(v) Input: A vertex $u \in V$. Find a neighbor $v \in V$ of u and delete the edge $\{u, v\}$ from the graph.

(vi) Input: Two vertices $u, v \in V$ with $u \neq v$. Insert an edge $\{u, v\}$ into the graph if it does not exist yet. Otherwise do nothing.

(vii) Input: A vertex $v \in V$. Delete v and all incident edges from the graph.

For the last two queries, describe your algorithm.

Exercise 10.4 *Maze solver.*

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.*

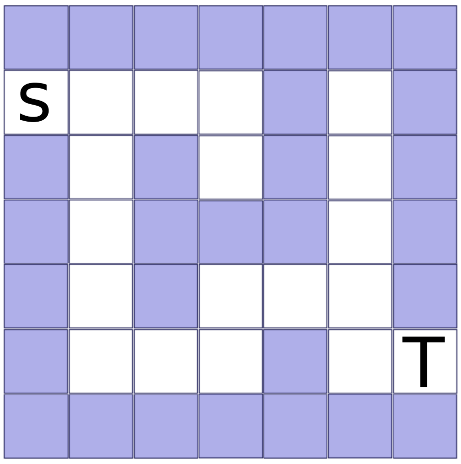


Figure 1: An example of 7×7 maze in which purple fields are blocked, white fields can be traversed (are unblocked). The start field is marked with 'S' and the target field with a 'T'.

a) 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 graph.

b) 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.*

c) Determine the running time and memory requirements of your algorithm in terms of n in Θ notation.

Exercise 10.5 *Driving on highways* **(1 point)**.

In order to encourage the use of train for long-distance traveling, the Swiss government has decided to make all the m highways between the n major cities of Switzerland one-way only. In other words, for any two of these major cities C_1 and C_2, if there is a highway connecting them it is either from C_1 to C_2 or from C_2 to C_1, but not both. The government claims that it is however still possible to drive from any major city to any other major city using highways only, despite these one-way restrictions.

a) 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 graph.

b) Describe an algorithm that verifies the correctness of the claim in time $O(n + m)$.

   Hint: *You can make use of an algorithm from the lecture. However, you might need to modify the graph described in part (a) and run the algorithm on some modified graph.*