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
Johannes Lengler, Gleb Novikov, Chris Wendler, Ulysse Schaller

14. December 2020

Algorithms & Data Structures Homework 13 HS 20

Exercise Class (Room & TA): ____________________

Submitted by: ____________________

Peer Feedback by: ____________________

Points: _____

Submission: This exercise sheet is not to be turned in. The solutions will be published at the end of the week, before Christmas.

Exercise 13.1 *Shortest path with negative edge weights (part I).*

Let $G = (V, E, c)$ be a graph with edge weights $c : E \to \mathbb{Z} \setminus \{0\}$ and $c_{\min} = \min_{e \in E} c(e)$.

a) Since Dijkstra's algorithm must not be used whenever some edge weights are negative (i.e., $c_{\min} < 0$), one could come up with the idea of applying a transformation to the edge weight of every edge $e \in E$, namely $c'(e) = c(e) - c_{\min} + 1$, such that all weights become positive, and then find a shortest path P in G by running Dijkstra with these new edge weights c'.

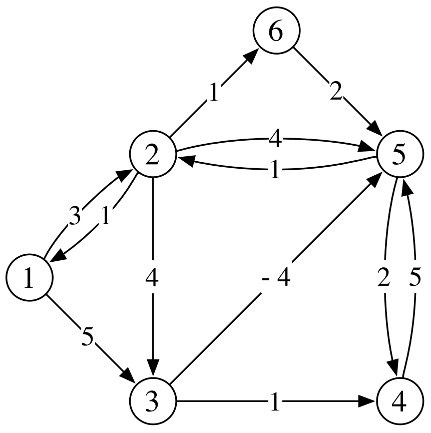
   Show that this is not a good idea by providing an example graph G with a weight function c, such that the above approach finds a path P that is not a shortest path in G. The example graph should have exactly 5 nodes and not all weights should be negative.

b) Now consider the problem of finding a minimum spanning tree of a connected undirected graph $G = (V, E, c)$. Show that if we add any number b to all weights of the edges, then (the edge sets of) minimum spanning trees of G do not change.

   So if we have an algorithm that finds a minimum spanning tree in any connected undirected graph with positive edge weights, we can also use it to find a minimum spanning tree in arbitrary connected undirected graph (by using new weights $c'(e) = c(e) - c_{\min} + 1$).

Exercise 13.2 *Shortest path with negative edge weights (part II).*

We consider the following graph:



1. What is the length of the shortest path from vertex 1 to vertex 6?
2. Consider Dijkstra's algorithm (that fails here, because the graph has negative edge weights). Which path length from vertex 1 to vertex 6 is Dijkstra computing? State the sets $S, V \setminus S$ immediately before Dijkstra is making its first error and explain in words what goes wrong.
3. Which efficient algorithm can be used to compute a shortest path from vertex 1 to vertex 6 in the given graph? What is the running time of this algorithm in general, expressed in n, the number of vertices, and m, the number of edges?
4. On the given graph, execute the algorithm by Floyd and Warshall to find *all* shortest paths. Express all entries of the $(6 \times 6 \times 7)$-table as 7 tables of size 6×6. (It is enough to state the path length in the entry without the predecessor vertex.) Mark the entries in the table in which one can see that the graph does not contain a negative cycle.

Exercise 13.3 *Variants of shortest-path-problems.*

Let $G = (V, E)$ be a directed weighted graph with positive edge weights and let $s, t \in V$ be two vertices.

Design an efficient algorithm for each of the following variants of the shortest-path problem and state the running time of your algorithm.

1. We would like to find the shortest path from s to t that passes through two additionally given vertices u and v.
2. Let $k \in \mathbb{N}$. Among all paths from s to t with at most k vertices in between (i.e., with at most $k+1$ edges) we would like to find the shortest such path.

Exercise 13.4 *Invariant and correctness of algortihm (**Exam exercise from January 2020**).*

Given is a weighted directed acyclic graph $G = (V, E, w)$, where $V = \{1, \ldots, n\}$. The goal is to find the length of the longest path in G.

Let's fix some topological ordering of G and consider the array top$[1, \ldots, n]$ such that top$[i]$ is a vertex that is on the i-th position in the topological ordering.

Consider the following pseudocode

Algorithm 1 Find-length-of-longest-path(G, top)

```
$L[1], \dots, L[n] \leftarrow 0, \dots, 0$
for $i = 1, \dots, n$ do
    $v \leftarrow \text{top}[i]$
    $L[v] \leftarrow \max_{(u,v)\in E} \{L[u] + w((u,v))\}$
return $\max_{1\leq i\leq n} L[i]$
```

Here we assume that maximum over the empty set is 0.

Show that the pseudocode above satisfies the following loop invariant INV(k) for $1 \leq k \leq n$: After k iterations of the for-loop, $L[\text{top}[j]]$ contains the length of the longest path that ends with top[j] for all $1 \leq j \leq k$.

Specifically, prove the following 3 assertions:

i) INV(1) holds.

ii) If INV(k) holds, then INV($k+1$) holds (for all $1 \leq k < n$).

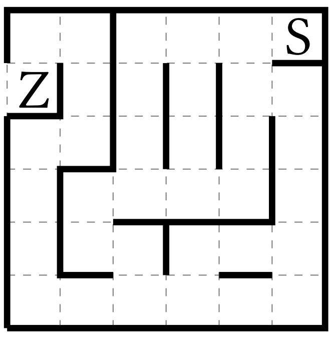
iii) INV(n) implies that the algorithm correctly computes the length of the longest path.

State the running time of the algorithm described above in Θ-notation in terms of $|V|$ and $|E|$. Justify your answer.

Exercise 13.5 *Minotaurus einsperren (**Klausuraufgabe vom Februar 2017**).*

King Minos instructs Daedalus to construct a maze to imprison the Minotaur. Daedalus presents his maze with n fields and a given starting field as a drawing on gridded paper. In the following figure you can see an example maze with $n = 36$ fields. The starting field is indicated by an S, and the target field at the exit with a Z. We want to determine how fast the Minotaur can escape from the given maze.

1. Model this problem as a shortest path problem:
   - Describe how the maze can be represented as a graph such that the following is true: The number of vertices on a shortest path between two vertices representing the starting and target field corresponds exactly to the smallest number of fields that have to be visited for reaching the target field Z.
   - Indicate how many vertices and edges your graph has in dependency of n.
   - Name an algorithm of the lecture that solves the shortest path problem for this graph as efficiently as possible. Also, provide the running time as concisely as possible in Θ notation.

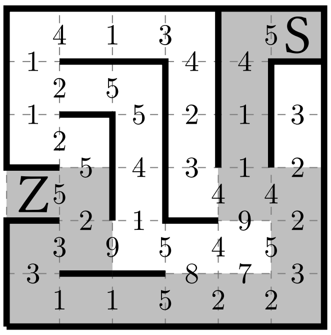


   Example: In the example on the right the Minotaur has to visit at least 21 fields (including the starting and the target field) to escape.

2. Various obstacles exist to complicate the escape. This means the time required to move from one field to another changes from obstacle to obstacle. For two adjacent fields that are not separated by a wall you are given the time that it takes to move from one field to another.

   How can the modeling from task 1. be adapted to compute, under consideration of the given times, a fastest route to escape from the maze?

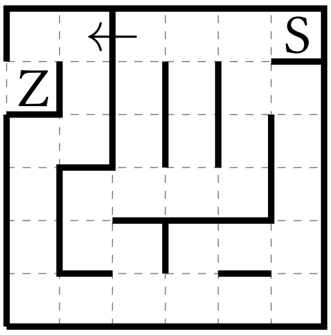
   - Describe how the maze can be represented as a graph such that the following is true: The length of a shortest path between two vertices representing the starting and target field corresponds exactly to the minimum time necessary for reaching the target field Z.
   - Indicate how many vertices and edges your graph has in dependency of n.
   - Name an algorithm of the lecture that solves the shortest path problem for this graph as efficiently as possible. Also, provide the running time as concisely as possible in Θ notation.



   Example: In the example on the right one needs at least 44 time units to escape. The fastest route is indicated in gray.

3. As another variant of task 1., the Minotaur has the force to destroy exactly one inner wall of the maze (i.e., one wall between two fields for which both these fields are inside the maze).

   Compute how many fields the Minotaur has to visit on his escape at least, by modeling the problem as a shortest path problem. Name an algorithm that solves this problem as efficiently as possible. Also, provide the running time as concisely as possible in Θ notation.



   Example: In the example on the right the Minotaur can destroy the wall that is marked with an arrow, and has to visit only 7 fields to escape (and not 21 as in task a)).