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

20. December 2021

# Algorithms & Data Structures — Homework 13 — HS 21

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).*

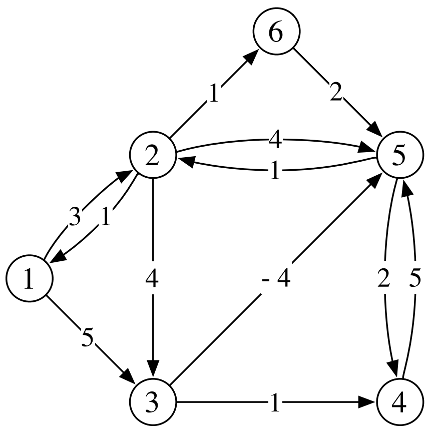
Let $G = (V, E, w)$ be a graph with edge weights $w : E \to \mathbb{Z} \setminus \{0\}$ and $w_{\min} = \min_{e \in E} w(e)$.

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

Show that this is not a good idea by providing an example graph $G$ with a weight function $w$, such that the above approach finds a path $P$ that is not a shortest path in $G$ (this path $P$ can start from the vertex of your choice). The example graph should have exactly 5 nodes and not all weights should be negative.

**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 $6 \times 6$. (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 *Invariant and correctness of algorithm (**This exercise is from the January 2020 exam**).*

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)

```
$L[1], \ldots, L[n] \leftarrow 0, \ldots, 0$
for $i = 1, \ldots, n$ 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 $\Theta$-notation in terms of $|V|$ and $|E|$. Justify your answer.

## Exercise 13.4 *Underground world (**This exercise is from the January 2021 exam**).*

a) Consider the following problem. The Swiss government is negotiating a deal with Elon Musk to build a tunnel system between all major Swiss cities. They put their faith into you and consult you. They present you with a map of Switzerland. For each pair of cities it depicts the cost of building a bidirectional tunnel between them. The Swiss government asks you to determine the cheapest possible tunnel system such that every city is reachable from every other city using the tunnel network (possibly by a tour that visits other cities on the way).

   i) Model the problem as a graph problem. Describe the set of vertices, the set of edges and the weights in words. What is the corresponding graph problem ?

   ii) Use an algorithm from the lecture to solve the graph problem. State the name of the algorithm and its running time in terms of $|V|$ and $|E|$ in $\Theta$-notation.

b) Now, the Swiss tunneling society contacts the government and proposes to build the tunnel between Basel and Geneva for half of Musk's cost. Thus, the government contacts you again. They want you to solve the following problem: Given the solution of the old problem in a) and an edge for which the cost is divided by two, design an algorithm that updates the solution such that the new edge cost is taken into account. *In order to achieve full points, your algorithm must run in time $O(|V|)$.*

   *Hint:* You are only allowed to use the *solution* from a), i.e. the set of tunnels in the chosen tunnel system. You are not allowed to use any intermediate computation results from your algorithm in a).

   i) Describe your algorithm (for example, via pseudocode). A high-level description is enough.

   ii) Prove the correctness of your algorithm and show that it runs in time $O(|V|)$.

## Exercise 13.5 *Cheap flights (**This exercise is from the January 2020 exam**).*

Suppose that there are $n$ airports in the country Examistan. Between some of them there are direct flights. For each airport there exists at least one direct flight from this airport to some other airport. Totally there are $m$ different direct flights between the airports of Examistan.

For each direct flight you know its cost. The cost of each flight is a strictly positive integer.

You can assume that each airport is represented by its number, i.e. the set of airports is $\{1, \ldots, n\}$.

a) Model these airports, direct flights and their costs as a directed graph: give a precise description of the vertices, the edges and the weights of the edges of the graph $G = (V, E, w)$ involved (if possible, in words and not formal).

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

b) Suppose that you are at the airport $S$ and you want to fill the array $d$ of minimal traveling costs to each airport. That is, for each airport $A$, $d[A]$ is a minimal cost that you must pay to travel from $S$ to $A$.

   Name the most efficient algorithm that was discussed in lectures which solves the corresponding graph problem. If several such algorithms were described in lectures (with the same running time), it is enough to name one of them. State the running time of this algorithm in $\Theta$-notation in terms of $n$ and $m$.

c) Now you want to know *how many* optimal routes there are to airport $T$. In other words, if $c_{\min}$ is the minimal cost from $S$ to $T$ then you want to compute *the number of routes from $S$ to $T$ of cost $c_{\min}$*.

   Assume that the array $d$ from b) is already filled. Provide an as efficient as possible *dynamic programming* algorithm that takes as input the graph $G$ from task a), the array $d$ from point b) and the airports $S$ and $T$, and outputs the number of routes from $S$ to $T$ of minimal cost.

   Address the following aspects in your solution and state the running time of your algorithm:

   1) *Definition of the DP table*: What are the dimensions of the table $DP[\ldots]$ ? What is the meaning of each entry ?

   2) *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.

   3) *Calculation order*: In which order can entries be computed so that values needed for each entry have been determined in previous steps ?

   4) *Extracting the solution*: How can the final solution be extracted once the table has been filled ?

   5) *Running time*: What is the running time of your algorithm ? Provide it in $\Theta$-notation in terms of $n$ and $m$, and justify your answer.