Exercise Sheet 12 for Effiziente Algorithmen (Winter 2025/26)

Hand In: *Until 2026-01-30 18:00, on ILIAS.*

Problem 1

20 + 20 + 10 points

a) Show or refute: Let $G = (V, E)$ be a graph with edge weights $\ell_e \geq 0$. Furthermore, let P be a shortest path from s to t, i.e., a solution of the shortest-paths problem. Then P remains a solution of the problem if we replace all edge lengths ℓ_e by their squares ${\ell_e}^2$.

b) Let $G = (V, E)$ be a directed graph that contains no negative cycles. Show: If $(s = v_0, v_1, \ldots, v_k = t)$ is a shortest *(vertex-simple)* path from s to t, then every sub-path $(v_0, \ldots, v_l)$ for $l = 1, \ldots, k-1$ is also a shortest *(vertex-simple)* path from s to v_l.

c) Show that b) does *not* hold in general when the graph contains negative cycles.

Problem 2

30 points

A path graph is an undirected graph $P = (V, E)$, where

$$V = \{v_1, \ldots, v_n\}$$
$$E = \{\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{n-1}, v_n\}\}$$

Consider the following problem:

Given a path graph $P = (V, E)$ and an edge function $c : E \to \{\text{red}, \text{blue}\}$ that assigns a colour to each edge, determine the minimum number of edges that must be deleted from P so that no vertex is incident to both a red and a blue edge.

Develop a greedy algorithm that solves the problem in $\mathcal{O}(n)$ time. Prove that your algorithm is correct.

(Note: In the in-person exercise a possible greedy strategy is disproved.)

Problem 3

40 points

Let S be a finite set with $|S| \geq 2$. Let $S_1, \ldots, S_k$ be a partition of S and let

$$\mathcal{U} = \{A \subseteq S \mid |A \cap S_i| \leq 1,\ 1 \leq i \leq k\}$$

be a collection of subsets.

Show that $(S,\mathcal{U})$ is a matroid. Note that a *hereditary set system* does not have to be non-empty.

Problem 4

20 + 10 + 30 points

Consider the following problem:

Given a string s and a list of words w, check whether s can be transformed into a sequence of words from w by inserting spaces.

Example:
$s =$ `platzangstübersehen`
$w =$ [`angst`, `über`, `unter`, `weit`, `sehen`, `platz`, `platzangst`, `wir`, `übersehen`]

Possible solutions:

- `platz angst über sehen`
- `platzangst übersehen`

The following algorithm solves the problem recursively:

```
procedure WordRek(s, w):
    if s.isEmpty()
        return True
    for i := 1, ..., |s|
        pre := s.substring(0, i)
        suf := s.substring(i + 1, |s|)
        if w.contains(pre) ∧ WordRek(suf, w)
            return True
    return False
```

a) Assume simplistically that calls to *contains* and *isEmpty* each cost c, and that forming prefixes and suffixes incurs no additional cost. How can you estimate the complexity of the solution above?

b) The presented algorithm contains sub-problems that are recomputed repeatedly. Discuss which sub-problems these are.

c) Develop an algorithm based on dynamic programming that solves the problem in $\mathcal{O}(n^2)$ time and $\mathcal{O}(n)$ space. Use the **bottom-up** approach.