University of Freiburg
Dept. of Computer Science
Prof. Dr. F. Kuhn
S. Faour, M. Fuchs, A. Mályusz

Algorithm Theory
Exercise Sheet 4

Due: Friday, 15th of November, 2024, 10:00 am

Remark: You are required to use the principle of *dynamic programming* in all of your algorithms. It is preferable if you write the algorithms in pseudocode.

Exercise 1: Paper Submissions *(6 Points)*

Professor Kuhn has to finish writing n different research papers p_i and would like to submit each one of them to a conference. However each paper p_i takes t_i time to be done and submitted by deadline d_i i.e. each deadline is according to the conference that the Professor wants to submit that paper in. Note that when we say by deadline d_i, we mean that at the latest the paper submission should be at time d_i. Moreover, writing any paper is available to be scheduled starting at time s.
Now, the task of writing the full paper p_i and then submitting it needs to be assigned a period from $s_i \geq s$ to $f_i = s_i + t_i$, and doing the same task for different papers should be assigned nonoverlapping intervals. Such an assignment of times will be called a *schedule.*
We consider the case in which writing each paper must either be finished and submitted by its deadline or not at all. We'll say that a subset P of the papers is *schedulable* if there is a schedule where the Professor is able to finish writing each paper in P and submit each of them by its deadline. Your problem is to select a schedulable subset of papers of maximum possible size and give a schedule for this subset that allows each paper to be fully written and submitted by its deadline.

Assume that all deadlines d_i and required times t_i are integers and $d_i \geq t_i$. Give an algorithm to find an optimal solution. Your algorithm should run in time polynomial in the number of papers that needs to be written and submitted n, and the maximum deadline $D = \max_i d_i$. Argue correctness and running time.
Hint: Prove that there is an optimal solution P (i.e., a schedulable set of maximum size) in which the papers in P are scheduled in increasing order of their deadlines.

Exercise 2: Generalized Max. Product Subarray *(7 Points)*

Given an integer array A of length n, and an integer m (where $m \leq n$). The *goal* is to find the maximum product that can be obtained by selecting exactly m non-overlapping contiguous subarrays. Note that each subarray should be non-empty. The following is an example:
Let $A = [1, -2, 3, -4, 5, -6, 0, 7, 8, 9, 0]$ and $m = 3$.
The maximum product can be achieved by choosing the subarrays $[3, -4, 5, -6], [7]$, and $[8, 9]$ with a total product of $3 \times -4 \times 5 \times -6 \times (7) \times (8 \times 9) = 181440$.

Give a polynomial time algorithm that achieves our goal. Argue correctness and run time.

Exercise 3: Longest Walk *(7 Points)*

Suppose you are given a graph $G = (V, E)$, where each node $v \in V$ is labeled with an elevation value $h(v) \in \mathbb{N}$. You can safely traverse an edge $(u, v) \in E$ from node u to node v if and only if the absolute difference between the elevation values of u and v is at most δ, that is, $|h(u) - h(v)| \leq \delta$, where $\delta > 0$ is an integer parameter.

Our goal now is to find a longest possible walk in the graph such that the walk starts at some node $s \in V$ and ends at another node $t \in V$. The walk should consist of two segments: an uphill segment, where the elevation must strictly increase in each step, followed by a downhill segment, where the elevation must strictly decrease in each step.

Note that a walk in a graph is path on which nodes are allowed to repeat. Note however that in the uphill segment, the elevation values are strictly increasing and in the downhill segment, the elevation values are strictly decreasing. Each node can therefore appear at most once in the uphill segment and at most once in the downhill segment. A node can however appear in the uphill and in the downhill segment.

Your input consists of the graph $G = (V, E)$, the elevation function $h : V \to \mathbb{N}$, the starting node s, the target node t, and the parameter $\delta > 0$. Your task is to find a longest possible walk as described above or determine that no such walk exists.

Give an algorithm that solves the problem in time $O(nm)$, where as usual $n = |V|$ and $m = |E|$. Argue correctness and run time.

Hint: It makes sense to first solve the following problem: For every $v \in V$, find the longest path from s to v on which the elevation values strictly increase or determine that no such path from s to v exists.