COL351: Analysis and Design of Algorithms **Fall 2024**

Tutorial Sheet 8

Announced on: Sept 19 (Thurs)

- This tutorial sheet contains problems in dynamic programming (DP). When presenting your solutions, please define the DP table clearly. Don't forget to write the base case. The correctness and running time arguments can be brief (1-2 sentences).
- Problems marked with ($\star$) will not be asked in the tutorial quiz.

1. Suppose you own two stores, A and B. Each day, you can be either at A or B. If you are currently at store A (respectively, B) then moving to store B (respectively, A) the next day will cost C amount of money. For each day $i \in \{1, ..., n\}$, we are also given the profits $P_A(i)$ (respectively, $P_B(i)$) that you will make if you are at store A (respectively, B) on day i. Give a schedule which tells where you should be on each day so that the overall money earned, i.e., the profit minus the cost of moving between the stores, is maximized.

2. Given a tree T where the vertices have weights, an *independent set* is a subset of vertices such that there is no edge joining any two vertices in this set. Give an efficient algorithm to find an independent set of maximum total weight.

3. (At most one of the two parts should be asked in the tutorial quiz.)

   Recall the following problem from Tutorial Sheet 6:

   You are given as input n jobs, each with a start time s_j and a finish time t_j. Two jobs *conflict* if they overlap in time—if one of them starts between the start and finish times of the other. The goal is to select a maximum-size subset of jobs that have no conflicts. (For example, given three jobs consuming the intervals $[0, 3]$, $[2, 5]$, and $[4, 7]$, the optimal solution consists of the first and third jobs.)

   (a) Design an efficient dynamic programming algorithm for this problem.

   (b) Consider a generalization of the above problem wherein job j has a non-negative weight w_j. Design an efficient dynamic programming algorithm to select a maximum weight subset of mutually non-conflicting jobs.

4. You are given n boxes, where box i has height h_i, width w_i and length ℓ_i. Box i can be stacked on top of box j if $w_i < w_j$ and $\ell_i < \ell_j$. Give an algorithm for finding a stacking of a subset of boxes of maximum total height.

5. Suppose you are taking n courses this semester. It is nearing the end of the semester and each course has a final project that still needs to be completed (started?). Each project will be graded on the following scale: It will be assigned an integer number on a scale of 1 to $g \geqslant 1$, higher numbers denoting better grades. Your goal, of course, is to maximize your average grade on the n projects. You have a total of $H > n$ hours in which to work on the n projects cumulatively, and you want to decide how to divide up this time. For simplicity, assume H is a positive integer, and you will spend an integer number of hours on each project. To figure out how best to divide up your time, you have come up with a set of functions $\{f_1, f_2, \ldots, f_n\}$ (rough estimates, of course) for each of your n courses, defined as follows: If you spend $h \leqslant H$ hours on the project for course i, you'll get a grade of $f_i(h)$. You may assume that the functions f_i are non-decreasing, i.e., if $h < h'$, then $f_i(h) \leqslant f_i(h')$.

   Given these functions $f_1, f_2, \ldots, f_n$, decide how many hours to spend on each project (in integer values only) so that your average grade, as computed according to f_i, is as large as possible. In order to be efficient, the running time of your algorithm should be polynomial in n, g, and H. None of these quantities should appear as an exponent in your running time.

6. Recall the following problem from Tutorial Sheet 6:

   Imagine you have a set of n course assignments given to you today. For each assignment i, you know its deadline d_i and the time ℓ_i it takes to finish it. With so many assignments, it may not be possible to finish all of them on time. If you finish an assignment after its deadline, you get zero marks. Therefore, you must either complete the assignment by the deadline or not at all. How can you determine the maximum number of assignments you can complete within their deadlines?

   Design an efficient dynamic programming algorithm for this problem.

7. Given integers n and k, along with $p_1, \ldots, p_n \in [0, 1]$, you want to determine the probability of obtaining exactly k heads when n biased coins are tossed independently at random, where p_i is the probability that the i^{th} coin comes up heads. Give an $\mathcal{O}(n^2)$ algorithm for this task. Assume you can multiply and add two numbers in $[0, 1]$ in $\mathcal{O}(1)$ time.

8. ($\star$) A subsequence is *palindromic* if it is the same whether read left to right or right to left. For instance, the sequence

   $$A, C, G, T, G, T, C, A, A, A, A, T, C, G$$

   has many palindromic subsequences, including A, C, G, C, A and A, A, A, A (on the other hand, the subsequence A, C, T is not palindromic). Devise an algorithm that takes a sequence $x[1 \ldots n]$ and returns the longest palindromic subsequence as well as its length. Its running time should be $\mathcal{O}(n^2)$.