COL351: Analysis and Design of Algorithms Announced on: Sep 07, 2026 (Monday)

Tutorial Sheet 8

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 choose to be at either store A or store B. If you are currently at store A (respectively, B), moving to store B (respectively, A) the following day will incur a cost of C. For each day $i \in \{1, ..., n\}$, we are given the profits $P_A(i)$ (respectively, $P_B(i)$) that you will earn if you are at store A (respectively, B) on that day. Your goal is to create a schedule that specifies which store you should be at each day to maximize your overall earnings, calculated as total profit minus the costs of moving between stores.

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 the maximum-size subset of jobs with 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, and the term is nearing its end. Each course requires a final project that must still be completed. Each project will be graded on a scale from 1 to g (where $g \geqslant 1$), with higher numbers indicating better grades. Your goal, of course, is to maximize your average grade across the n projects.

   You have a total of $H > n$ hours available to work on all the projects combined, and you need to determine how to allocate your time effectively. For simplicity, let's assume that H is a positive integer and that you will spend an integer number of hours on each project. To aid in your decision, you have developed a set of functions $\{f_1, f_2, \ldots, f_n\}$ for each of your n courses. Each function is defined as follows: if you spend $h \leqslant H$ hours on the project for course i, you will receive a grade of $f_i(h)$.

   You can assume that the functions f_i are non-decreasing, meaning that if you spend more time on a project (i.e., if $h < h'$), you will receive a higher grade (i.e., $f_i(h) \leqslant f_i(h')$).

   Given these functions $f_1, f_2, \ldots, f_n$, your task is to decide how many hours to allocate to each project (only in integer values) in order to maximize your average grade as determined by the functions f_i. Furthermore, your algorithm should be efficient, with a running time that is polynomial in n, g, and H, without any of these quantities appearing as exponents in the 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). Design an $\mathcal{O}(n^2)$ algorithm that takes a sequence $x[1 \ldots n]$ and returns the longest palindromic subsequence as well as its length.