Homework 6

CS6033 Design and Analysis of Algorithms I

Fall 2023

(Sec. B, Prof. Yi-Jen Chiang)

Due: Wed. 12/6 by 1pm
(submit online on NYU Brightspace; one submission per group)
Maximum Score: 105 points

Note: This assignment has 2 pages.

1. (25 points)
There is one machine and a set of n jobs $a_1, a_2, \cdots, a_n$ to be processed on that machine. The machine is available to start processing any job at time 0. Each job a_i has a processing time t_i, a profit p_i, and a deadline d_i, where $t_i \leq d_i$. The machine can process only one job at a time, and for each $i = 1, 2, \cdots, n$, job a_i must run uninterruptedly for t_i consecutive time units. If job a_i is completed by its deadline d_i, we get a profit p_i, but if it is completed after its deadline, we get a profit of 0. Each processing time t_i is a positive integer, and $\sum_{i=1}^{n} t_i = T$. Design and analyze a dynamic programming algorithm to find a schedule that achieves the maximum amount of profit. Your algorithm should run in $O(nT)$ worst-case time.

2. (25 points)
A certain string-processing language offers a primitive operation that splits a string into two pieces. Since this operation involves copying the original string, it takes n units of time for a string of length n, regardless of the location of the cut. Suppose that we want to break a string into many pieces. The order in which the breaks are made can affect the total running time. For example, if we want to cut a 20-character string at positions 3 and 10, then making the first cut at position 3 has a cost of $20 + 17 = 37$, while cutting at position 10 first has a cost of $20 + 10 = 30$.

Given a string X, its length $|X| = n$ and the location of m cuts that will eventually break X into $m + 1$ pieces $X_1, X_2, \cdots, X_{m+1}$ (where $X = X_1 X_2 \cdots X_{m+1}$), design and analyze a dynamic programming algorithm to find the minimum cost of breaking X into such $m + 1$ pieces $X_1, \cdots, X_{m+1}$. Your algorithm should run in $O(m^3)$ worst-case time.
(**Hint:** It might be helpful to first obtain $|X_i|$ (the length of X_i) for each $i = 1, 2, \cdots, m + 1$.)

3. (20 points)
Given a directed acyclic graph (DAG) $G = (V, E)$ where each edge e has a weight $w(e)$ ($w(e)$ can be > 0 or < 0) and a vertex $s \in V$, design and analyze a dynamic programming algorithm to compute, for each vertex $v \in V$, the **longest path** from s to v. Your algorithm should output, for each $v \in V$, two entries:
(1) $d(v)$, denoting the length of the longest path from s to v (or "undefined" if v cannot be reached from s), and
(2) $prev(v)$, denoting the **immediate predecessor vertex** of v in the longest path from s to v (or "undefined" if v cannot be reached from s; we let $prev(s) = \texttt{NULL}$).

Your algorithm should run in $O(V + E)$ worst-case time.

4. (35 points)
A ski rental agency has m pairs of skis, where the height of the i-th pair of skis is s_i. There are n skiers who wish to rent skis, where the height of the i-th skier is h_i. Ideally, each skier should obtain a pair of skis whose height matches the skier's own height as closely as possible. Your task here is to design an efficient algorithm to assign skis to skiers so that the sum of the absolute differences of the heights of each skier and their skis is minimized.
(**Note:** In the following, even if you cannot prove **part (a)**, you can still use the property in **part (a)** to solve the problems in **parts (b)** and (**c**).)

(**a**) Prove that there is no advantage to "cross match" — reverse the height order of skiers and skis. That is, if $s_1 < s_2$ and $h_1 < h_2$, matching $s_1 \leftrightarrow h_1$ and $s_2 \leftrightarrow h_2$ is always better than, or at least as good as, matching $s_1 \leftrightarrow h_2$ and $s_2 \leftrightarrow h_1$.
(**Hint:** Consider all possible relationships of the two intervals $[s_1, s_2]$ and $[h_1, h_2]$. Without loss of generality, you can assume that $s_1 \leq h_1$.) **(9 points)**

(**b**) Consider the special case of the problem where $m = n$. Use the property in **part (a)** and give an $O(n \log n)$ worst-case time algorithm. **(6 points)**

(**c**) Now consider the general case of the problem where $m > n$ and assume that $m = \Theta(n)$. Note that every skier should get one pair of skis but not every pair of skis needs to be rented out. Use the property in **part (a)** to establish the property of optimal substructure, and design and analyze a dynamic programming algorithm that runs in $O(mn)$ worst-case time.
(**Hint:** Re-arrange both the m pairs of skis and the n skiers appropriately, and define suitable sub-problems in dynamic programming.) **(20 points)**