Practice Problems 5

Problem 1. You are given n intervals, where interval j has start time s_j, finish time f_j, and weight $w_j > 0$. Two intervals are *compatible* if they do not overlap (an interval finishing at time t is compatible with one starting at time t). Your goal is to choose a set of mutually compatible intervals with the maximum total weight.

Assume the intervals are sorted by finish time, $f_1 \leq f_2 \leq \cdots \leq f_n$, and let $p(j)$ be the largest index $i < j$ such that interval i is compatible with interval j (with $p(j) = 0$ if no such interval exists). Let $\mathrm{OPT}(j)$ denote the maximum total weight achievable using only intervals $1, \ldots, j$.

(a) Consider the following intervals, already sorted by finish time.

j	1	2	3	4	5	6	7	8
s_j	0	1	4	2	6	5	8	9
f_j	3	5	6	8	9	10	11	12
w_j	3	5	4	9	3	6	4	4

Compute $p(j)$ for every j. Then use the recurrence relation given in class to fill in the table $\mathrm{OPT}(0), \mathrm{OPT}(1), \ldots, \mathrm{OPT}(8)$, and trace back through the table to recover an optimal set of intervals.

(b) Recall Problem 1 from Practice Problems 4, where the goal was to maximize the total time of the scheduled intervals and greedy algorithms failed. Explain how to solve that problem optimally using dynamic programming.

(c) Now suppose you may schedule at most k intervals. Give a recurrence for $\mathrm{OPT}(j, \ell)$, the maximum total weight using only intervals $1, \ldots, j$ and at most ℓ of them. Explain the recurrence formula and state the running time of the resulting algorithm.

Problem 2. A *subsequence* of a string is obtained by deleting zero or more characters without changing the order of the remaining characters. Given strings $X = x_1 x_2 \cdots x_m$ and $Y = y_1 y_2 \cdots y_n$, the goal is to find a longest string that is a subsequence of both X and Y. Let $\mathrm{OPT}(i, j)$ denote the length of an LCS of $x_1 \cdots x_i$ and $y_1 \cdots y_j$.

(a) Let $X =$ `DYNAMIC` and $Y =$ `DRAMATIC`. Use the recurrence relation given in class to fill in the table $\mathrm{OPT}(i, j)$ for $0 \leq i \leq 7$ and $0 \leq j \leq 8$, and trace back through the table to recover an LCS of X and Y.

(b) A *palindrome* is a string that reads the same forwards and backwards, such as `RACECAR`. Given a string X of length n, we want the length of the longest subsequence of X that is a palindrome. Let X^R denote the reverse of X. Prove that the length of the longest palindromic subsequence of X equals the length of an LCS of X and X^R.

(c) A *substring* is a subsequence whose characters are contiguous in the original string. In the *longest common substring* problem, we want the longest string that is a substring of both X and Y. For example, the longest common substrings of `DYNAMIC` and `DRAMATIC`

are `AM` and `IC`. Give a recurrence for $S(i, j)$, the length of the longest common substring of X and Y that ends exactly at x_i and y_j. Explain the recurrence formula, explain how to obtain the final answer from the table, and state the running time of the resulting algorithm.

Problem 3. You are given the prices $p_1, p_2, \ldots, p_n$ of a stock over n days. You may buy one share on some day i and sell it on some later day $j \geq i$, for a profit of $p_j - p_i$. Your goal is to find the maximum possible profit. (Buying and selling on the same day gives profit 0, so the answer is never negative.) For example, if the prices are $8, 3, 6, 2, 5, 9, 4, 7$, the maximum profit is 7, obtained by buying at price 2 and selling at price 9.

(a) Give a divide and conquer algorithm for this problem that runs in $O(n \log n)$ time. Explain why it is correct, and give and solve the recurrence for its running time. **Hint: Split the days into two halves. Where can the best buy day and sell day be?**

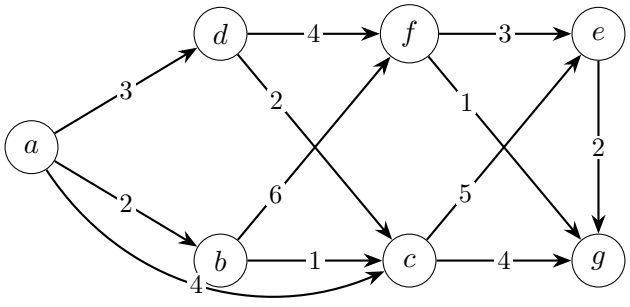
(b) Modify your algorithm from part (a) so that it runs in $O(n)$ time. **Hint: Have each recursive call return more than just the best profit.**

(c) Give a dynamic programming algorithm for this problem that runs in $O(n)$ time. Define your subproblem $\mathrm{OPT}(j)$, give a recurrence for it, and explain the recurrence formula.

Problem 4. You are given a directed acyclic graph (DAG) $G = (V, E)$ where each edge (u, v) has a weight $w(u, v)$. The length of a path is the sum of the weights of its edges, and a path may consist of a single vertex, which has length 0. Your goal is to find the longest path in G.

(a) Let $\mathrm{OPT}(v)$ denote the length of the longest path that ends at vertex v. Give a recurrence for $\mathrm{OPT}(v)$, and explain in what order the values $\mathrm{OPT}(v)$ should be computed. **Hint: A longest path ending at v is either just v itself, or ends with some edge (u, v). In the second case, what can you say about the part of the path before v?**

Then use your recurrence on the DAG below to compute the length of the longest path.



(b) In this part, you will show that LCS from Problem 2 is a special case of longest path in a DAG. Given strings $X = x_1 \cdots x_m$ and $Y = y_1 \cdots y_n$, construct a DAG with a vertex (i, j) for every $0 \leq i \leq m$ and $0 \leq j \leq n$, and with edges and weights chosen so that the longest path from $(0, 0)$ to (m, n) has length equal to the length of an LCS of X and Y. Explain why your graph is acyclic, and explain how your recurrence from part (a) on this graph corresponds to the LCS recurrence given in class.