Exercise Sheet 11

Handout: November 18th — Deadline: November 25th

Question 11.1 (0.25 marks)

Consider a modification of the rod-cutting problem where in addition to a price p_i for each rod, each cut incurs a fixed cost c. The revenue for a solution is now the sum of the prices of the individual pieces minus the costs of making the cuts. Give a Dynamic Programming Bottom-Up algorithm to solve this problem.

Question 11.2 (0.25 marks)

Modify Memoized-Cut-Rod so that it also returns the actual solution rather than just its value.

Question 11.3 (0.25 marks)

Provide the pseudo-code of an $O(n)$ time Dynamic Programming algorithm for calculating the n_{th} Fibonacci number. Draw the subproblem graph. How many vertices and edges does the subproblem graph contain?

Question 11.4 (1 mark)

After retiring from a large company, Mr Mortimer wonders how much money he could have made if he had invested his life savings in shares of his company. He looks back at the share prices over the n days of his employment and asks himself how much his investment could have returned if he had—somehow—known the best time to buy and sell his shares. Mortimer doesn't like trading shares, so he would only ever buy once and sell once and assume that his life savings is a fixed amount.

Let $a_1, \ldots, a_n$ describe the difference of share prices over time: a_i is the amount of money Mr Mortimer would have gained if he had held on to shares on day i. Note that a_i can be negative, in which case day i would have been a loss. Assume $a_i \neq 0$. Let $f(i, j) = \sum_{k=i}^{j} a_k$ be the money earned if Mortimer had bought shares at the start of day i of his employment and sold them at the end of day j. Mortimer wants to find the maximum return $\max\{f(i, j) \mid 1 \leq i \leq j \leq n\}$.

Example: for $a_1, \ldots, a_n = +5, -3, -4, +8, -1, +12, -6, +4, +4, -14, +2, +8$, Mortimer's maximum return would have been $f(4, 9) = 8 + (-1) + 12 + (-6) + 4 + 4 = 21$ pounds.

(a) Design an algorithm in pseudocode that computes the maximum return using dynamic programming in time $O(n)$, following the hints below. Explain your solution.

(b) Show that your algorithm runs in time $O(n)$.

Hints: Use a bottom-up approach, tabulating for each day k:

- the maximum return A_k for an investment up to day k (buying and selling up to day k, formally $\max\{f(i,j) \mid 1 \leq i \leq j \leq k\}$) and
- the maximum return B_k up to day k for an *ongoing* investment: still holding on to shares on day k (formally $\max\{f(i,k) \mid 1 \leq i \leq k\}$).

Work out how A_1 and B_1 can be initialised, and work out Bellman equations showing how, for $k \geq 2$, A_k and B_k can be computed based on the input and values of A and B that you have tabulated earlier. Pay attention to the order in which to tabulate values. Don't forget to state how to compute the final output from the tabulated values.

Question 11.5 (0.5 marks)

Implement the three problems "Longest common subsequence", "Stone Merging" and "Task arrangement" on the OJ system.