

# Homework 5: Dynamic Programming

Due: February 25, 2026 at 11:59 pm

**Problem 1.** You are given a roll (array) of  $n$  magical coins. Each coin has an initial value  $c_i$  which increases at each time step linearly. Therefore, at time step  $t_j$  the coin has a value of  $c_i j$ .

You want to sell the coins to maximize your total profit, however, you are only able to sell the coins one at a time. Furthermore, you are only able to sell the coins from either end of the roll of coins.

- (a) You initially spring for a greedy approach to sell your magical coins, i.e. at each time step you compare both ends of the array and sell the coin with least value. Is this strategy flawed? Explain your reasoning.
- (b) Design an algorithm which determines the optimal order to sell the magical coins to maximize profit, given that you know the starting value of each coin. Your algorithm should run in  $O(n^2)$  time. Provide a proof of correctness for your algorithm, and justify its runtime and memory utilization.

*Solution.*

□

**Problem 2.** You are organizing a film festival, and you must select a schedule of screenings that maximizes box office revenue. Each film has a scheduled start time  $startTime[i]$ , an end time  $endTime[i]$ , and an associated box office return  $profit[i]$ .

However, some films overlap in time, and viewers cannot attend two overlapping screenings. Screenings may be back-to-back: if one film ends at a time  $X$ , another may start exactly at time  $X$ .

- (a) Design a **dynamic programming** algorithm that determines the maximum total box office revenue you can achieve without overlapping screenings. Your algorithm should run in worst case  $\mathcal{O}(n^2)$  time (but could be faster ))
- (b) Provide a proof of correctness of your algorithm
- (c) State an upper bound to the worst-case running time and memory utilization. Provide proof for your statements.
- (d) Your festival has upgraded, and you may now host two movies at a time! How would your proposed solution change? Explain the modifications to your algorithm and informally argue why the modification is correct and what the new worst-case running time is.

**Problem 3.** You are an archaeologist who has discovered an ancient printing press in a hidden temple. This mystical device has a peculiar mechanism: in a single operation, it can print a sequence of one or more *identical* symbols starting at any position on a sacred scroll. However, when it prints, it completely overwrites whatever was previously in those positions.

You have been tasked with recreating a sacred inscription, represented as a string  $s$  of length  $n$  consisting of lowercase English letters. Starting with a blank scroll, you must use this ancient press to reproduce the inscription exactly.

For example, to print the inscription "aba":

- One approach uses 3 operations: print "a" at position 0, then "b" at position 1, then "a" at position 2.
  - A cleverer approach uses only 2 operations: print "aaa" across all positions first, then overwrite position 1 with "b".
- (a) Prove that there always exists an optimal solution in which the first printing operation begins at position 0 and prints the character  $s[0]$ .
- (b) Design a **dynamic programming algorithm** that determines the minimum number of printing operations required to produce the sacred inscription  $s$ . Your algorithms should run in worst case running time  $\mathcal{O}(n^3)$
- (c) Provide a proof of correctness for your algorithm.
- (c) Prove an upper bound to the worst-case running time of your algorithm and to its memory space utilization.