CS 3000: Algorithms & Data — Spring '20 — Jonathan Ullman

Homework 4
Due Friday Feb 7 at 11:59pm via Gradescope

Name:
Collaborators:

- Make sure to put your name on the first page. If you are using the LaTeX template we provided, then you can make sure it appears by filling in the `yourname` command.
- This assignment is due Friday Feb 7 at 11:59pm via Gradescope. No late assignments will be accepted. Make sure to submit something before the deadline.
- Solutions must be typeset in LaTeX. If you need to draw any diagrams, you may draw them by hand as long as they are embedded in the PDF. I recommend using the source file for this assignment to get started.
- I encourage you to work with your classmates on the homework problems. *If you do collaborate, you must write all solutions by yourself, in your own words.* Do not submit anything you cannot explain. Please list all your collaborators in your solution for each problem by filling in the `yourcollaborators` command.
- Finding solutions to homework problems on the web, or by asking students not enrolled in the class is strictly forbidden.

Problem 1. *Road Trip*

Note, there has been a small correction to this problem. You are going on a long road trip down Interstate 90, driving all the way from the start at mile marker 0 to the end at mile marker M. Along the way there are n hotels you can stop at for the night, located at mile posts $m_1 < m_2 < \ldots, m_n$ (each m_i represents the number of miles from the starting point 0). Ideally, you'd like to travel 400 miles a day, but depending on the spacing of the hotels, you might want to drive more or less. If you drive d miles on a given day, then your *unhappiness* is $(400 - d)^2$. You want to break up the drive with hotel stops so that your *total unhappiness* is minimized. That is, if you stop at hotels $\{h_1 < h_2 < \cdots < h_k\}$, you drive distances $h_1 - 0$, $h_2 - h_1$, $h_3 - h_2$, $\ldots$, $M - h_k$, and have total unhappiness

$$(400 - (h_1 - 0))^2 + (400 - (h_2 - h_1))^2 + (400 - (h_3 - h_2))^2 + \cdots + (400 - M + h_k)^2$$

In this problem, you will design a dynamic programming algorithm that takes as input the end point $M > 0$ and the set of n hotel locations $0 < m_1 < \cdots < m_n < M$ and outputs a set of hotels to stop at that minimized total unhappiness.

(a) Describe the set of subproblems that your dynamic programming algorithm will consider. Your solution should look something like "For every $\ldots$, we define $\textsc{opt}(\ldots)$ to be $\ldots$"

Solution:

(b) Give a recurrence expressing the solution to each subproblem in terms of the solution to "smaller" subproblems.

Solution:

(c) Describe in pseudocode an efficient dynamic programming algorithm that finds an optimal *set* of hotels minimizing total unhappiness. Your solution may use either the "bottom-up" or "top-down" approach.

Solution:

(d) Analyze the running time and space usage of your algorithm.

Solution:

Problem 2. *Strategery*

Alice and Bob play the following game. There is a row of n tiles with values $a_1, \dots, a_n$ written on them. Starting with Alice, Alice and Bob take turns removing either the first or last tile in the row and placing it in their pile until there are no tiles remaining. For example, if Alice takes tile 1, Bob can take either tile 2 or tile n on the next turn. At the end of the game, each player receives a number of points equal to the sum of the values of their tiles minus that of the other player's tiles. Specifically, if Alice takes tiles $A \subseteq \{1, \dots, n\}$ and Bob takes tiles $B = \{1, \dots, n\} \setminus A$, then their scores are

$$\sum_{i \in A} a_i - \sum_{i \in B} a_i \quad \text{and} \quad \sum_{i \in B} a_i - \sum_{i \in A} a_i,$$

respectively. For example, if $n = 3$ and the tiles have numbers $10, 2, 8$ then taking the first tile guarantees Alice a score of at least $10 + 2 - 8 = 4$, whereas taking the last tile would only guarantee Alice a score of at least $8 + 2 - 10 = 0$.

In this question, you will design an algorithm to determine the maximum score that Alice can guarantee for herself, assuming Bob plays optimally to maximize his score.[1]

(a) Describe the set of subproblems that your dynamic programming algorithm will consider. Your solution should look something like "For every …, we define OPT(…) to be …"

Solution:

(b) Give a recurrence expressing the solution to each subproblem in terms of the solution to smaller subproblems.

Solution:

(c) Explain in English a valid order to fill your dynamic programming table in a "bottom-up" implementation of the recurrence.

Solution:

(d) Describe in pseudocode an algorithm that finds the maximum score that Alice can guarantee for herself. Your implementation may be either "bottom-up" or "top-down."

Solution:

(e) Analyze the running time and space usage of your algorithm.

Solution:

[1] **Hint:** Note that the sum of their scores is always 0, so if Bob is playing optimally to maximize his own score, then he is also playing optimally to minimize Alice's score.