CS41 Homework 7

This homework is due at 10PM on Sunday, November 3. Write your solution using \LaTeX. Submit this homework using github as a .tex file. This is a partnered homework. You should primarily be discussing problems with your homework partner. It’s ok to discuss approaches at a high level with others. However, you should not reveal specific details of a solution, nor should you show your written solution to anyone else. The only exception to this rule is work you’ve done with a lab partner while in lab. In this case, note (in your homework submission poll) who you’ve worked with and what parts were solved during lab.

The focus of this lab is on designing and implementing dynamic programs. Your algorithms should be dynamic programs. They should be clearly correct and clearly run in polynomial time. However, proving correctness or analyzing runtime will not be a major component of your grade this week.

1. Implementing RNA Substructure. For this problem, your task is to write a program that takes as input the name of a file containing a single string representing an RNA molecule and outputs the size of the largest matching according to the rules for RNA Substructure discussed in class.

   (a) You may write your program in C++ or Python. We’ve provided some limited starter code for the C++ version.

   (b) You’re welcome to test your program using test files in /home/brody/public/cs41/rna_test_data.

   (c) We wrote two implementations of the RNA Substructure problem, which we call rna-A and rna-B, located in /home/brody/public/cs41. You’re welcome to run one or both of these programs. Your answers should match the output of either of these programs. One of the programs we wrote is much faster than the other.

   (d) The runtime of your algorithm should be comparable to the more efficient of rna-A and rna-B.

You are welcome to write your own test inputs and use them to debug and test runtime; you are not limited to the provided test files.

2. Harry’s Hoagie House. Harry runs a Hoagie House. His customers are regular and pushy – each day, they order hoagies at the same time and expect them to be made immediately. (If their hoagie is not started as soon as they order it, the customers will walk out in a huff.) Harry used to have several employees to help him make hoagies, but they recently all quit to prepare for a pterodactyl hunt. Now, Harry is on his own. Harry actually doesn’t care much about his customers or how many hoagies he can make; he only cares about how much money he makes.

Design and analyze a polynomial-time dynamic programming algorithm to help Harry maximize his profits. The input to your algorithm is a list of \( n \) hoagie orders. Each hoagie order has:

- a start time \( s_i \).
- a finish time \( f_i \).
- a profit \( p_i \).

Assume the hoagies are sorted by finish time. Your algorithm should return the maximum amount of profit Harry can earn by making hoagies one at a time.

Hints:

- Given hoagie \( k \), it might help to know the greatest \( i \) such that \( f_i \leq s_k \). How would you compute this?
- It might help to create a “virtual hoagie” \( 0 \) such that \( s_0 = f_0 = 0 \) and \( p_0 = 0 \).
3. **Moving on a Checkerboard.** Suppose you are given an $n \times n$ checkerboard and a single checker. You must move the checker from the bottom edge of the board to the top edge of the board according to the following rule. At each step, you may move the checker to one of three squares:

(a) the square immediately above,
(b) the square that is one up and one to the left (but only if the checker is not already in the leftmost column), or
(c) the square that is one up and one to the right (but only if the checker is not already in the rightmost column).

Each time you move from square $x$ to square $y$ you receive $P(x, y)$ dollars. You are given the values $P(x, y)$ for all pairs $(x, y)$ for which a move from $x$ to $y$ is legal. $P(x, y)$ may be negative.

Give a polynomial-time algorithm that figures out the set of moves that will move the checker from somewhere along the bottom edge to somewhere along the top edge while gathering as many dollars as possible. Your algorithm is free to pick any square along the bottom edge as a starting point, and any square along the top edge as a destination, in order to maximize the number of dollars gathered along the way. What is the runtime of your algorithm?