CS41 Homework 8

This homework is due at 11:59PM on Sunday, November 1. Write your solution using LaTeX. Submit this homework using **github** as a file called **hw8.tex**. This is a **partnered homework**. It's ok to discuss approaches at a high level with others; however, you should **primarily discuss approaches with your homework partner.**

If you do discuss problems with others, 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.

1. **Divide and conquer minimum spanning trees?**

   Joshua has a really cool idea for a divide and conquer algorithm which will find a MST. Given a connected, undirected graph $G = (V, E)$ with weighted edges, Joshua's algorithm does the following:

   - Divides the graph into two pieces, $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. ($V_1 \cup V_2 = V$ and V_1 and V_2 are disjoint. E_1 is the edges in E with both endpoints in V_1, and E_2 is the edges in E with both endpoints in V_2.)
   - Recursively finds the MSTs T_1 for G_1 and T_2 for G_2.
   - Finds the lowest-weight edge $e = (u, v)$ with $u \in V_1$ and $v \in V_2$.
   - Returns the minimum spanning tree $T_1 \cup T_2 \cup \{e\}$.

   Unfortunately, this algorithm does not work. Give an example input graph G with weights and describe a run of this algorithm where the algorithm does not return a minimum spanning tree on G.

2. **Evaluating investment strategies.**

   An investment company has designed several new trading strategies and wants to compare them. Unfortunately, the strategies were tested on different commodities over different time periods, so the total profit earned is not a fair comparison. Instead, the company wants to compare each strategy's profit to the maximum possible profit that could have been made (with perfect hindsight) during the same time span. Each strategy was allowed to buy a fixed amount of the commodity once and then sell what it bought at some later date.

   During each testing period, the investment company recorded the minimum and maximum prices that were reached each day. Your algorithm receives a list of n pairs, where the i^{th} pair has the minimum price of the commodity and the maximum price of the commodity on day i. Your algorithm should output the largest price difference that could have been achieved by buying at the minimum price on day i and selling at the maximum price on day $j > i$.

   For example, suppose $n = 3$ and the (min, max) prices were $[(8, 14), (1, 2), (4, 6)]$. Then you should return a per-unit profit of 5, corresponding to buying for 1 on day 2 and selling for 6 on day three (recall that the trading strategies must buy first, and can't sell on the same day).

   Clearly, there is a simple algorithm that takes time $O(n^2)$: try all possible pairs of buy/sell days and see which makes the most money. Design a divide and conquer algorithm to determine the best profit in hindsight more efficiently. Set up a recurrence that describes the running time of your algorithm, and solve it to show that your algorithm is faster than $O(n^2)$.

3. **Harry's Hoagie House** Harry runs a takeout hoagie shop. His customers are regular and pushy – they order hoagies at the same time and expect them to be made immediately. Harry used to have several employees to help him make hoagies, but they recently all quit to work for a competitor. 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 make by making hoagies one at a time.

Hints:

- Focus on the first two steps of the dynamic programming process; don't stress about pseudocode.
- Focus on the **choice** you might make to construct an optimal solution. For example, with the Steel Rod Problem, our choice was where to make the left-most cut.
- 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 make a "virtual hoagie" 0 such that $s_0 = f_0 = 0$ and $p_0 = 0$.

4. **(extra challenge) Divide and conquer for minimum spanning trees (V2.0)**

   In problem 1, we considered a divide-and-conquer approach to the minimum spanning tree problem. You constructed a weighted, connected graph G and a particular execution so that the algorithm did not return a minimum spanning tree of G.

   Is it possible to "patch" this algorithm to work, if the vertex partition is chosen cleverly? That is, can we do a little bit of conquering *before* the divide step(s), which will make this divide-and-conquer MST algorithm work?

   If YES, then describe how to fix this divide and conquer algorithm to be correct. If NO, then argue why no rule for dividing G can make the algorithm correct.