CS 577: Introduction to Algorithms — Homework 6

Out: 03/16/21 **Due: 03/23/21**

Name: ______________________ Wisc ID: ______________________

Ground Rules

- Answer the questions in the boxes provided on the question sheets. If you run out of room for an answer, add a page to the end of the document. Do **not** feel obligated to fill the entire solution box. The size of the box does **not** correspond to the intended solution length.
- The homework is to be done and submitted individually. You may discuss the homework with others in either section but you must write up the solution *on your own*.
- You are not allowed to consult any material outside of assigned textbooks and material the instructors post on the course websites. In particular, consulting the internet will be considered plagiarism and penalized appropriately.
- The homework is due at 11:59 PM CST on the due date. No extensions to the due date will be given under any circumstances.
- Homework must be submitted electronically on Gradescope.

Problem 1

After running the teleportation delivery company *Algo Express* for many years, you discover the power of dynamic programming. You leave the company to start a new venture (*DPAlgo Express*) that can process very big delivery orders. In particular, each order now takes several days for the teleportation machine to complete.

Suppose on a certain day, n customers give you packages to deliver. Each delivery i should be made within d_i days, takes t_i days to deliver, and the customer pays you p_i dollars for doing it on time (if you don't do it on time, you get paid 0 dollars). On-time delivery means that if package i is due within $d_i = k$ days, the delivery should be completed on or before day k to be on time (that is, it should start on or before day $k - t_i$). As before, your teleportation machine can only make one delivery at a time.

Input: A set of n deliveries with due dates $d_i \in \mathbb{N}, d_i \geq 1$, number of days needed for delivery $t_i \in \mathbb{N}, t_i \geq 1$ and payments $p_i > 0$ for each delivery $i \in \{1, \cdots, n\}$.

(a) Describe an efficient algorithm to determine which deliveries to make and in what order so as to maximize your profit. (Note: unlike the previous version of the problem, deliveries may now take more than one day). Your algorithm should have a pseudo-polynomial running time – running time polynomial in n and T, where T is the latest deadline among all deliveries.

(b) Provide a clear explanation for your recurrence relation for part (a) and analyze its running time.

Problem 2

Two players are arguing over which of their characters is the best at 1v1 combat in a tabletop RPG game, and they have asked you to help solve this dispute. In this game, combat works as follows: each character starts with a certain number of hitpoints, and players take turns selecting actions that could either harm their opponent or benefit themselves. The first character to reach 0 or fewer hitpoints loses.

Player 1's character starts with H_1 hitpoints and has two actions to choose from on each turn: attack and flex. Their character starts with an attack power of 1, and each time they choose the flex option their attack power increases by 1, up to a maximum of 5. If they choose to attack, then their character deals their attack power times A_1 damage to their opponent, removing that same amount of hitpoints. Player two's character starts with H_2 hitpoints and also has two actions to choose from: attack and heal. If they choose to heal, then their character recovers 20 of their missing hitpoints (up to a maximum of H_2). Because heal is a spell, it can only be used a total of two times in combat. If player 2 chooses to attack, then they deal A_2 points of damage to their opponent.

Because player 1's character initiative attribute is greater than player 2's character, player 1 is always the first one to choose an action. Notice that there are no draws, and if both players play optimally then one of them is always guaranteed to win. The problem you are asked to solve is the following: given as input integer values H_1, A_1, H_2 and A_2, decide which player is guaranteed to win when both play optimally.

(a) Give a dynamic programming algorithm to solve this problem. That is, describe your algorithm by including a clear statement of your recurrence, any necessary base case(s), and your final output.

(b) Provide a clear explanation for your recurrence relation for part (a) and analyze its running time.