CIS 675 Midterm 2

Instructions:

1. Your exam grade will be based on your performance in the oral exam. The solution you present must match your written solution.

2. You are permitted to use your textbook, lecture materials, and existing online sources.

3. You are permitted to collaborate with other currently enrolled CIS 675 students. However, you are *strongly discouraged* from doing so. If you do not create your own solutions, you will do poorly on the oral exam.

4. You are *not* permitted to post exam questions online for others to answer (e.g., on StackOverflow or Chegg).

5. If you are confused about what any of the problems is asking, reach out to the instructor or TA immediately. It is better to clarify now than during the exam!

Problem 1: Suppose you are shoveling snow from driveways to earn money. There is a very large set of n driveways that need to be shoveled. Assume that shoveling a driveway of length d_i will pay you $\$p_i$. All driveways of the same length give the same payout; but it isn't necessarily the case that larger driveways pay more money. If you choose to shovel a driveway, you must perform the required t_i units of time consecutively (in other words, you cannot stop shoveling a driveway and come back to it later). You must finish shoveling by time T, which is when you go to bed. Assume there are infinitely many driveways of each length, and all lengths and payments are integers. You only get paid for a driveway if you finish shoveling the entire thing. Design an efficient algorithm to determine which driveway lengths (and how many of each) to shovel so that you maximize your payout.

Problem 2: Suppose that you are running a bakery, and have a set of n potential customers who would like you to bake a cake for them. Each potential customer i will pay you c_i if you complete their order. However, because some of these cakes require special ingredients that need to be ordered, there is a constraint that you cannot start customer i's order until time t_i. In other words, you can start that order at anytime t_i or later, but you cannot start before that time. Your bakery closes at time T, so you cannot bake any cakes after that. Each cake takes 5 units of time to complete, and these units of time *must* be consecutive. Design an efficient algorithm to determine which cakes to bake, and when to bake them, so that you receive the maximum possible amount of payment.

Problem 3: Suppose you have a line of tokens in positions $1, ..., N$. Each token i has a value v_i written on its top side and $-v_i$ written on the bottom side. (v_i may itself be positive or negative). You want to specify a starting position s and an ending position t, and take *all* the tokens in between i and j. For the first token you choose, you can keep it in its original position, or flip it. However, after that point, you must flip tokens in *alternating* order. For example, if you choose not to flip the first selected token, you must flip the second one, and then you cannot flip the third, but you must flip the fourth, etc. The total score you get is equal to the sum of the values that are face-up for the selected tokens (v_i for the tokens that are not flipped, $-v_i$ for the tokens that are flipped). Design an efficient algorithm to to determine s and t so that the score is maximized.

Problem 4: Suppose you are trying to build a human ladder (a sequence of people, each standing on the previous person's shoulders, in order to, for example, get over a tall wall). Assume that there is no limit on how many people can be in this ladder. A person can only stand on another person's shoulders if the person on top has a lighter weight than the person below. You have a set of N people, each with a specified height and weight (it is possible that two people have the same height, weight, or both). However, some of these people hate each other, and a person cannot stand on another person's shoulders if either of them hates the other. You are given a list of pairs of people who hate each other. Design an efficient algorithm to determine which people to include in the ladder, and in which order, so that the height of the ladder is maximized.

Problem 5: Suppose you have a set of N robots which can perform tasks for you. You want to use as many robots as possible so that you finish the tasks quickly. Each robot is controlled by radio signals sent from a controller device. Each robot i can send/receive communications in the frequency range $[b_i, t_i]$, where $b_i < t_i$. But if two robots are on at the same time, and those two robots have overlapping communication frequency ranges, then they may interfere with one another. To avoid this, you want to ensure that none of the robots that are on at the same time have overlapping frequency ranges. Design an efficient algorithm to determine which robots you can have on at the same time so that the total number of robots that are turned on is maximized, while ensuring that none of them have overlapping frequency ranges.

Problem 6: Suppose you have a set of N robots which can perform tasks for you. You want to turn on as many robots as possible so that you finish the tasks quickly. Each robot is controlled by radio signals sent from a controller device. Each robot i can receive communications in the frequency range $[b_i, t_i]$, where $b_i < t_i$. To turn on a robot, you broadcast an ON signal at any frequency within its range. One signal can turn on multiple robots, as long as the signal frequency is within the frequency ranges of all of those robots. Sending such signals is time-consuming, so you want to turn on all of the robots with as few signals as possible. Design an efficient algorithm to determine which frequencies you should broadcast your ON signal on so that the total number of such broadcasts is minimized, and all robots are turned on.