CSE Ph.D. Qualifying Exam, Fall 2022
Algorithms

Instructions:

Please answer three of the following four questions. All questions are graded on a scale of 10. If you answer all four, all answers will be graded and the three lowest scores will be used in computing your total.

Questions:

1. **Greedy** Consider the following job scheduling problem:

   You have n jobs $J_1, J_2, ..., J_n$, one supercomputer and n identical PCs. Each jobs needs to be first *preprocessed* on the supercomputer, and then it needs to be *finished* on one of the PCs. Job J_i needs p_i seconds of time on the supercomputer, followed by f_i seconds of time on a PC.

   Since there are n PCs, the finishing of the jobs can be performed fully in parallel, where each PC can process one job. However, the supercomputer can only work on a single job at a time, so you need to work out an order in which to schedule the jobs to the supercomputer. As soon as a job is done on the supercomputer, it can be handed off to a PC for finishing, and the supercomputer can start to process the next job.

   The *completion time* of all jobs is the earliest time at which all jobs will have finished processing on the PCs. Describe a greedy algorithm which produces a schedule that minimizes the completion time, and prove the correctness of your algorithm. Your proof of correctness can use an exchange argument.

2. **Dynamic Programming: longest subsequence of increasing numbers**

   You are given a list of distinct numbers $a_1, a_2, ..., a_n$. Please design a dynamic programming algorithm which finds the longest subsequence of numbers where the numbers are strictly increasing from smaller indices to larger indices. A subsequence means a subset of numbers from the original list where the relative positions of numbers should be maintained but the indices need not be consecutive. That is, in subsequence $a_{i_1}, a_{i_2}, ..., a_{i_m}$, we have $i_1 < i_2 < ... < i_m$.

   For example, given a list of numbers $\{82, 77, 65, 89, 83, 68, 88, 71, 91\}$, one optimal solution is $\{77, 83, 88, 91\}$.

   Please describe the algorithm clearly (you may give the pseudocode), give the recurrence relations, and analyze the time and space complexity of your algorithm. Remember to include steps to output the optimal subsequence (you do not need to output all the optimal solutions if there are more than one.)

3. **Dynamic Programming: meal delivery**

   As the new semester starts, George is making plan for ordering meal deliveries from a restaurant for the entire semester. For each week, George has two choices: either skip the week or order meal for the entire week. Since the restaurant shares the menu of each week, George can give a tasty score to the menu of each week depending on how much he likes the food. Suppose there are n weeks, the tasty scores are $x_1, x_2, ..., x_n$, where each score is an integer, and can be positive, 0, or negative.

Since the restaurant encourages ordering of consecutive weeks, there can be penalties for skipping weeks:

- If George skips one week, there is no penalty.
- If George skips two or three consecutive weeks, there will be a penalty of 20 points.
- Skipping for four weeks or more than four weeks is not allowed. This can be considered as a penalty of ∞ points.

The overall happiness score of the semester is the sum of all the tasty scores of the weeks George decides to order delivery, minus the corresponding penalty for the weeks that are skipped. For example, if there are 5 weeks, and the tasty scores are $\{10, -10, -5, 15, 6\}$, and the plan for the 5 weeks is $\{order, skip, skip, order, order\}$, the overall happiness score is $10 - 20 + 15 + 6 = 11$.

Please design a dynamic programming algorithm to find the weekly plan for George that maximizes the overall happiness score of the semester. Please describe the algorithm clearly (you may give the pseudocode), give the recurrence relations, and analyze the time and space complexity of your algorithm. Remember to include steps to output the optimal weekly plan.

4. **NP-complete : DOUBLE-SAT**

Recall that the SATISFIABILITY (SAT) problem is: given a Boolean formula ϕ of n variables $x_1, x_2, ..., x_n$, determine whether there exist at least one assignment of the n variables that evaluate ϕ to be TRUE. SAT is known to be a NP-complete problem.

The DOUBLE-SAT problem is: given a Boolean formula ϕ of n variables $x_1, x_2, ..., x_n$, determine whether there exist at least *two* assignments of the n variables that evaluate ϕ to be TRUE. Prove that DOUBLE-SAT is NP-complete using the fact that SAT is NP-complete.

For this problem, you can assume that ϕ is in the Conjunctive Normal Form (CNF) for both SAT and DOUBLE-SAT (no points will be taken off if you make this assumption). But it is possible to write the proof without this assumption.

Please remember to include all steps of the NP-completeness proof.