CSE Ph.D. Qualifying Exam, Fall 2020
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**

   At the start of the semester you are given n homework assignments $\{a_1, ..., a_n\}$. You can do the assignments in any order, but you must turn in 1 assignment per week over the n weeks of the semester. Assignment a_i has a difficulty d_i (assume the difficulty values are distinct). If you turn in a_i on week j, you get $d_i(n - j)$ points.

   Please give a greedy algorithm which finds the order of the assignments that maximizes your points, and use the exchange argument to prove the optimality of your algorithm.

2. **Dynamic programming**

   You are given a sorted set of points $P = (P_1, P_2, ..., P_n)$ on a line. Given a constant k, show how to select a subset of $k - 1$ of these points, say (still in sorted order) $(P_{j_1}, ..., P_{j_{(k-1)}})$, so as to partition the segment from P_1 to P_n into k pieces that are as close to equal in length as possible. Specifically, writing $L = (P_n - P_1)/k$, we want to minimize the square error

   $$(P_{j_1} - P_1 - L)^2 + \sum_{i=1}^{k-2}(P_{j_{i+1}} - P_{j_i} - L)^2 + (P_n - P_{j_{k-1}} - L)^2$$

   Please design an algorithm that solves this problem optimally and runs in time polynomial in k and n. Please describe the algorithm clearly (you may give the pseudocode), give the recurrence relations, and analyze the time and space complexity of your algorithm.

3. **Dynamic Programming**

   Suppose you are taking n courses, each with a project that has to be done. Each project will be graded on the following scale: It will be assigned an integer number on a scale of 1 to $g > 1$, higher numbers being better grades. Your goal is to maximize your average grade on the n projects.

   You have a total of $H > n$ hours in which to work on the n projects cumulatively, and you want to decide how to divide up this time. For simplicity, assume H is a positive integer, and you will spend an integer number of hours on each project.

   To figure out how to best divide up your time, you have come up with a set of functions $\{f_i : i = 1, 2, ..., n\}$ to estimate the grade you will get for each project given that you spend h hours on that project. That is, if you spend $h \leq H$ hours on the project for course i, you will get a grade of $f_i(h)$. You may assume that the functions f_i are non decreasing: if $h < h'$, then $f_i(h) \leq f_i(h')$.

So the problem is: Given these functions $\{f_i\}$, decide how many hours to spend on each project (in integer values only) so that your total grade, as computed according to the f_i, is as large as possible. The running time of your algorithm should be polynomial in n and H. Please describe the algorithm clearly (you may give the pseudocode), give the recurrence relations, and analyze the time and space complexity of your algorithm.

4. **NP-completeness**

   2-Partition-n-Even Problem:
   Input: $2n$ positive integers $a_1, \ldots, a_{2n}$.
   Question: Is there a subset I of $\{1, \ldots, 2n\}$ such that $\sum_{i \in I} a_i = \sum_{i \notin I} a_i$? We let $S = \sum_{1 \leq i \leq 2n} a_i$.

   Prove that 2-Partition-n-Even is NP-complete by using the fact the 2-Partition is NP-complete. 2-Partition is the following problem: Given n positive integers $a_1, \ldots, a_n$, is there a subset I of $\{1, \ldots, n\}$ such that $\sum_{i \in I} a_i = \sum_{i \notin I} a_i$? Remember to include all the steps of the NP-completeness proof.