MATH 416, PROBLEM SET 6

Comments about homework.

- Solutions to homework should be written clearly, with justification, in complete sentences. Your solution should resemble something you'd write to teach another student in the class how to solve the problem.
- You are encouraged to work with other 416 students on the homework, but solutions must be written independently. Include a list of your collaborators at the top of your homework.
- You should submit your homework on Gradescope, indicating to Gradescope where the various pieces of your solutions are. The easiest (and recommended) way to do this is to start a new page for each problem.
- Attempting and struggling with problems is **critical** to learning mathematics. Do not search for published solutions to problems. I don't have to tell you that doing so constitutes academic dishonesty; it's also a terrible way to get better at math.

  If you get stuck, ask someone else for a hint. Better yet, go for a walk.

Problem 1. A small business (say, a small photocopying service with a single large machine) faces the following scheduling problem. Each morning they receive a set of jobs from customers. They want to do the jobs on their single machine in an order that keeps their customers happiest. Customer k's job will take time t_k to complete. Given a schedule (i.e., an ordering of the jobs), let C_k denote the finish time of job k. For example, if job j is the first to be done, we would have $C_j = t_j$. Each job k also has a given weight w_k that represents the job's importance to the business. So the company decides to order the jobs in order to minimize the weighted sum of the completion times, $\sum_{k=1}^{n} w_k C_k$.

Design an efficient algorithm to solve this problem. That is, you are given a list of n jobs, the k^{th} with processing time t_k and a weight w_k. Order the jobs so as to minimize the weighted sum of the completion times, $\sum_{k=1}^{n} w_k C_k$.

Example. Suppose there are two jobs: the first takes time $t_1 = 1$ and has weight $w_1 = 10$; the second takes time $t_2 = 3$ and has weight $w_2 = 2$. Then doing job 1 first would yield a weighted completion time of $10 \cdot 1 + 2 \cdot 4 = 18$, while doing the second job first would yield the larger weighted completion time of $10 \cdot 4 + 2 \cdot 3 = 46$.

Problem 2. A sequence is *palindromic* if it is the same whether read left-to-right or right-to-left. For instance, the sequence

`A C G T G T C A A A A T C G`

has many palindromic subsequences, including `ACGCA` and `AAAA` (but not `ACT`). (Notice that subsequences are not required to be contiguous.) Devise an algorithm that takes as input a sequence $x[1 \ldots n]$ and returns the (length of the) longest palindromic subsequence. Its running time should be $O(n^2)$.

Problem 3. A shuffle of two strings X and Y is formed by interspersing the characters into a new string, keeping the characters of X and Y in the same order. For example, the string BANANAANANAS is a shuffle of the strings BANANA and ANANAS in several ways:

BANANAANANAS BANANAANANAS BANANAANANAS.

(a) Given three strings $A[1 \dots m]$, $B[1 \dots n]$, $C[1 \dots m+n]$, describe an efficient algorithm to determine whether C is a shuffle of A and B. How fast does it run?

(b) A **smooth** shuffle of X and Y is a shuffle of X and Y that never uses more than two consecutive symbols of either string. For example, the shuffling BANANAANANAS. Describe an efficient algorithm to decide, given three strings X, Y, and Z, whether Z is a smooth shuffle of X and Y. How fast does it run?

Problem 4. Recall the interval-scheduling problem in which we sought to minimize the maximum lateness. There are n jobs, the i^{th} with a deadline d_i and a required processing time t_i, and all jobs are available to start at time $t = 0$. Each job needs to be *scheduled*, i.e., assigned a start time s_i and finish time $f_i = s_i + t_i$, and different jobs should be assigned non-overlapping intervals. As usual, such an assignment of times will be called a *schedule*.

In this problem, we consider a problem with the same setup but a different objective. We assume that each job must be completed by its deadline or not at all. We'll say that a subset J of the jobs is *schedulable* if there is a schedule for the jobs in J so that each of them finishes by its deadline. Your task is to select a schedulable subset of maximum possible size and give a schedule your subset that allows each job (in your subset) to finish on time.

(a) Prove that there is an optimal solution J (i.e., a schedulable set of maximal size) in which the jobs in J are scheduled in increasing order of their deadlines.

(b) Assume that all deadlines d_i and time durations t_i are integers. Give an algorithm to find an optimal solution. Your algorithm should run in time polynomial in the number n of jobs and the maximum deadline $D = \max_i d_i$.

Problem 5. You are going on a long trip. You start on the road at mile post 0. Along the way there are n hotels, at mile posts $a_1 < a_2 < \cdots < a_n$, where each a_i is measured from the starting point. The only places you are allowed to stop are at these hotels, but you can choose which of the hotels you stop at. You must stop at the final hotel (located at distance a_n), which is your destination.

You'd ideally like to travel 200 miles per day, but this may not be possible, depending on the spacing of the hotels. If you travel x miles during a day, the *penalty* for that day is $(200 - x)^2$. You want to plan your trip so as to minimize the total penalty, that is, the sum, over all travel days, of the daily penalties. Give an efficient algorithm that determines the optimal sequence of hotels at which to stop.