CSCI 570 - Spring 2021 - HW 2

Due Sunday Feb. 22 (by 4:00 AM)

Problem 1 (20 points)

Suppose you are given two sets A and B, each containing n positive integers. You can choose to reorder each set however you like. After reordering, let a_i be the i-th element of set A, and let b_i be the i-th element of set B. You then receive a payoff on $\prod_{i=1}^{n} a_i^{b_i}$. Give an algorithm that will maximize your payoff (6 points). Prove that your algorithm maximizes the payoff (10 points) and state its running time (4 points).

Problem 2 (20 points)

Suppose you are to drive from USC to Santa Monica along I-10. Your gas tank, when full, holds enough gas to go p miles, and you have a map that contains the information on the distances between gas stations along the route. Let $d_1 < d_2 < ... < d_n$ be the locations of all the gas stations along the route where d_i is the distance from USC to the i-th gas station. We assume that the distance between neighboring gas stations is at most p miles. Your goal is to make as few gas stops as possible along the way. Give the most efficient algorithm to determine at which gas stations you should stop (6 points) and prove that your strategy yields an optimal solution (10 points). Give the time complexity of your algorithm as a function of n (4 points). Suppose the gas stations have already been sorted by their distances to USC.

Problem 3 (20 points)

Some of your friends have gotten into the burgeoning field of time-series data mining, in which one looks for patterns in sequences of events that occur over time. Purchases at stock exchanges-what's being bought-are one source of data with a natural ordering in time. Given a long sequence S of such events, your friends want an efficient way to detect certain "patterns" in them—for example, they may want to know if the four events

buy Yahoo, buy eBay, buy Yahoo, buy Oracle

occur in this sequence S, in order but not necessarily consecutively.

They begin with a collection of possible events (e.g., the possible transactions) and a sequence S of n of these events. A given event may occur multiple times in S (e.g., Yahoo stock may be bought many times in a single sequence S). We will say that a sequence S' is a subsequence of S if there is a way to delete certain of the events from S so that the remaining events, in order, are equal to the sequence S'. So, for example, the sequence of four events above is a subsequence of the sequence

buy Amazon, buy Yahoo, buy eBay, buy Yahoo, buy Yahoo, buy Oracle

Their goal is to be able to dream up short sequences and quickly detect whether they are subsequences of S. So this is the problem they pose to you: Give an algorithm that takes two sequences of events-S' of length m and S of length n, each possibly containing an event more than once-and decides in time $O(m+n)$ whether S' is a subsequence of S. Prove that your algorithm outputs "yes" if S' is indeed a substring of S (hint: induction).

Problem 4 (20 points)

You are consulting for a trucking company that does a large amount of business shipping packages between New York and Boston. The volume is high enough that they have to send a number of trucks each day between the two locations. Trucks have a fixed limit W on the maximum amount of weight they are allowed to carry. Boxes arrive at the New York station one by one, and each package i has a weight w_i. The trucking station is quite small, so at most one truck can be at the station at any time. Company policy requires that boxes are shipped in the order they arrive; otherwise, a customer might get upset upon seeing a box that arrived after his make it to Boston faster. At the moment, the company is using a simple greedy algorithm for packing: they pack boxes in the order they arrive, and whenever the next box does not fit, they send the truck on its way.

But they wonder if they might be using too many trucks, and they want your opinion on whether the situation can be improved. Here is how they are thinking. Maybe one could decrease the number of trucks needed by sometimes sending off a truck that was less full, and in this way allow the next few trucks to be better packed.

Prove that, for a given set of boxes with specified weights, the greedy algorithm currently in use actually minimizes the number of trucks that are needed. Your proof should follow the type of analysis we used for the Interval Scheduling Problem: it should establish the optimality of this greedy packing algorithm by identifying a measure under which it "stays ahead" of all other solutions.

Problem 5 (20 points)

Solve the following recurrences by giving tight Θ-notation bounds in terms of n for sufficiently large n. Assume that $T(\cdot)$ represents the running time of an

algorithm, i.e. $T(n)$ is positive and non-decreasing function of n and for small constants c independent of n, $T(c)$ is also a constant independent of n. Note that some of these recurrences might be a little challenging to think about at first. Each question has 4 points. For each question, you need to explain how the Master Theorem is applied (2 points) and state your answer (2 points).

- (a) $T(n) = 4T(n/2) + n^2 \log n$.
- (b) $T(n) = 8T(n/6) + n \log n$.
- (c) $T(n) = \sqrt{6006}T(n/2) + n^{\sqrt{6006}}$.
- (d) $T(n) = 10T(n/2) + 2^n$.
- (e) $T(n) = 2T(\sqrt{n}) + log_2 n$.