Analysis of Algorithms

Homework 3

General Instructions.

For a dynamic programming solution, be sure to

- Prove that an optimal substructure exists
- Say in words what your variables are computing
- Give a recurrence and a non-recursive pseudocode
- Analyze the running time

PROBLEM 1.

Consider a sorting problem in which we do not know the exact numbers. Instead, for each number, we know an interval on the real line to which it belongs. We are given n closed intervals of the form $[a_i, b_i]$ where $a_i \leq b_i$. We wish to produce a permutation $\langle i_1, \dots i_n \rangle$ of the intervals such that for $j = 1,2, \dots, n$ there exist $c_j \in [a_{i_j}, b_{i_j}]$ satisfying $c_1 \leq c_2 \leq \cdots \leq c_n$. Design a randomized algorithm that does so in expected $\Theta(n \lg n)$ time, but runs in $\Theta(n)$ when all of the intervals overlap. *Hint:* Start by quicksorting the left endpoints, but take advantage of overlapping intervals to make an improvement. Your algorithm should not be explicitly checking if all the intervals overlap.

PROBLEM 2.

Consider modifying the PARTITION procedure by randomly picking three elements from the array and partitioning about their median. As a function of α such that $0 < \alpha < 1$, approximate the probability of getting at worst an α-to-$(1 - \alpha)$ split. Does this change improve the asymptotic runtime of quicksort?

PROBLEM 3.

Give an efficient algorithm to find the longest palindrome that is a subsequence of a given input string (e.g. given *character* you should return *carac*). Analyze the runtime of your algorithm.
(A palindrome is a string that reads the same forwards and backwards, such as *racecar* or *tenet*).

PROBLEM 4.

A certain string-processing language allows a programmer to break a string into two pieces. Because this operation copies the string, it costs n time units to break a string of n characters into two pieces. Suppose a programmer wants to break a string into many pieces. The order in which the breaks occur can affect the total amount of time used.

For example: suppose that I want to break the string *cliffsteinalgorithms* to *cliff, stein, algorithms*. If I break from left-to-right, the first break takes 20 time units and the second break takes 15, so it takes 35 time units. If I break from right-to-left, similarly, it takes 30 time units.

Design an algorithm that, given a string and the indices of characters after which to break (in the example, ⟨*cliffsteinalgorithms*, 5, 10⟩) determines the least-cost sequence of breaks.

PROBLEM 5.

The input to this problem is a sequence of n points $p_1, \ldots, p_n$ in the Euclidean plane. You are to find the shortest routes for the two taxis to service these requests in order. Let us be more specific:

- The two taxis start at the origin. If a taxi visits a point p_i before p_j then it must be the case that $i < j$ (stop and think about what this sentence means).
- Each point must be visited by at least one of the two taxis.
- The cost of a routing is just the total distance traveled by the first taxi plus the total distance traveled by the second taxi.

Construct an efficient algorithm to compute the optimal service sequence for each taxi.

PROBLEM 6.

You are starting an electric car company. You build cars that can travel m miles in between charges. Your main objective is to make sure that people can drive from New York to San Francisco along Route 80. You need to decide where to build charging stations. You have a set of potential locations $x_1, \ldots, x_n$ and for each location i, you are given a distance d_i and a price p_i. d_i specifies how many miles along Route 80 (from New York) each location is, and p_i is the cost of building a charging station at that location. You may assume that for every i, $d_{i+1} > d_i$, and $d_{i+1} - d_i \leq m$ so that if you build every station, a drive is guaranteed to be able to successfully travel from New York to San Francisco.

Find an algorithm which finds a set of locations to open that guarantee that a driver will not run out of electricity and which has minimum total cost.