Problem Set 11

Data Structures and Algorithms, Fall 2019

Due: December 5, in class.

From CLRS

Exercise 16.3-6, 16.3-8, 24.1-1, 24.1-3, 24.1-5 (also argue correctness), 24.2-4, 24.3-5, 24.3-8, 24.5-7.

Additional Problem One

A server has n customers waiting to be served. The service time required by each customer is known in advance: it is t_i minutes for customer i. So if, for example, the customers are served in order of increasing i, then the i^{th} customer has to wait $\sum_{j=1}^{i} t_j$ minutes.

We wish to minimize the total waiting time

$$T = \sum_{i=1}^{n} (\text{time spent waiting by customer } i)$$

Describe and analyze an efficient algorithm for computing the optimal order in which to process the customers. Remember to prove your algorithm is correct.

Additional Problem Two

Consider the following process. At all times you have a single positive integer x, which is initially equal to 1. In each step, you can either *increment* x or *double* x. Your goal is to produce a target value n. For example, you can produce the integer 10 in four steps as follows:

$$1 \xrightarrow{+1} 2 \xrightarrow{\times 2} 4 \xrightarrow{+1} 5 \xrightarrow{\times 2} 10$$

Obviously you can produce any integer n using exactly $n-1$ increments, but for almost all values of n, this is horribly inefficient. Describe and analyze an algorithm to compute the minimum number of steps required to produce any given integer n. Remember to prove your algorithm is correct.

Additional Problem Three

There is an (undirected) network of roads $G = (V, E)$ connecting a set of cities V. Each road in E has an associated length l_e. There is a proposal to add one new road to this network, and there is a list E' of pairs of cities between which the new road can be built. Each such potential road $e' \in E'$ has an associated length. As a designer for the public works department you are asked to determine the road $e' \in E'$ whose addition to the existing network G would result in the maximum decrease in the driving distance between two fixed cities s and t in the network. Describe and analyze an efficient algorithm for solving this problem. Remember to argue your algorithm is correct.

Additional Problem Four

Let $c_1, c_2, \cdots, c_n$ be various currencies; for instance, c_1 might be dollars, c_2 pounds, and c_3 lire. For any two currencies c_i and c_j, there is an exchange rate $r_{i,j}$; this means that you can purchase $r_{i,j}$ units of currency c_j in exchange for one unit of c_i. These exchange rates satisfy the condition that $r_{i,j} \cdot r_{j,i} < 1$, so that if you start with a unit of currency c_i, change it into currency c_j and then convert back to currency c_i, you end up with less than one unit of currency c_i (the difference is the cost of the transaction).

(a) Describe and analyze an efficient algorithm for the following problem: Given a set of exchange rates $r_{i,j}$, and two currencies s and t, find the most advantageous sequence of currency exchanges for converting currency s into currency t.

The exchange rates are updated frequently, reflecting the demand and supply of the various currencies. Occasionally the exchange rates satisfy the following property: there is a sequence of currencies $c_{i_1}, c_{i_2}, \cdots, c_{i_k}$ such that $r_{i_1,i_2} \cdot r_{i_2,i_3}, \cdots \cdot r_{i_{k-1},i_k} \cdot r_{i_k,i_1} > 1$. This means that by starting with a unit of currency c_{i_1} and then successively converting it to currencies $c_{i_2}, c_{i_3}, \cdots, c_{i_k}$ and finally back to c_{i_1}, you would end up with more than one unit of currency c_{i_1}. Such anomalies last only a fraction of a minute on the currency exchange, but they provide an opportunity for risk-free profits.

(b) Describe and analyze an efficient algorithm for detecting the presence of such an anomaly.

Bonus Problem

Consider the following generalized set cover problem: Given a universe U of n elements, a collection of subsets of U, $\mathcal{S} = \{S_1, \cdots, S_k\}$, and a cost function $c : \mathcal{S} \to \mathbb{Q}^+$, find a minimum *cost* sub-collection of $\mathcal{S}$ that covers all elements of U.[1]

This problem is suspected to be hard, in the sense that there might not exist a polynomial (with respect to the length of the input) time algorithm that can solve the problem exactly. However, good approximation algorithms do exist for this problem. In particular, if the optimal solution incurs a cost of OPT, we can efficiently find a solution that costs at most $O(OPT \cdot \ln n)$. Can you devise one such algorithm? Prove your algorithm indeed gives a solution that costs at most $O(OPT \cdot \ln n)$, and analyze the runtime of your algorithm. (*Hint: simply be greedy!*)

[1]Recall in class we have discussed another version of the problem, in which the goal is to find the minimum *size* sub-collection of $\mathcal{S}$ that covers all elements of U.