Problem Set 10

Data Structures and Algorithms, Fall 2020

Due: Dec 3, in class.

Problem 1

Suppose there are n people with the sizes of their feet given in an array $P[1 \cdots n]$, and n pairs of shoes with sizes given in an array $S[1 \cdots n]$. Design an algorithm to assign a pair of shoes to each person, so that the average difference between the size of a person's feet and the size of his/her assigned pair of shoes is as small as possible. Specifically, the algorithm should compute a permutation $\pi : [1 \cdots n] \to [1 \cdots n]$ such that $(1/n) \sum_{i=1}^{n} |P[i] - S[\pi(i)]|$ is minimized. To get full credit, your algorithm should have time complexity $O(n \log n)$, and you need to prove the correctness of your algorithm.

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: Be Greedy!)*

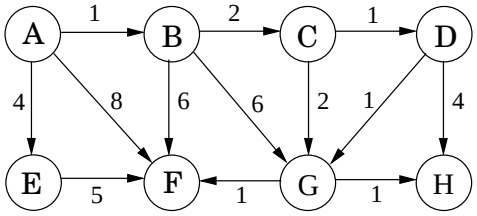
Problem 2

(a) Give a weighted, directed graph $G = (V, E)$ with no negative-weight cycles. Use this graph to demonstrate minimum spanning trees and shortest path trees are not necessarily identical.

(b) Given a weighted, directed graph $G = (V, E)$ with no negative-weight cycles, let m be the maximum over all vertices $v \in V$ of the minimum number of edges in a shortest path from the source s to v. Suggest a simple change to the Bellman-Ford algorithm that allows it to terminate in $m + 1$ passes, even if m is not known in advance.

(c) Suppose Dijkstra's algorithm is run on the following graph, starting at node A. (I) Draw a table showing the intermediate distance values of all the nodes at each iteration of the algorithm. (II) Show the final shortest-path tree.

[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.



Problem 3

You are given a set of cities, along with the pattern of highways between them, in the form of an undirected graph $G = (V, E)$. Each stretch of highway $e \in E$ connects two of the cities, and you know its length in miles, l_e. You want to get from city s to city t. There's one problem: your car can only hold enough gas to cover L miles. There are gas stations in each city, but not between cities. Therefore, you can only take a route if every one of its edges has length $l_e \leq L$.

(a) Given the limitation on your car's fuel tank capacity, show how to determine in $O(|V| + |E|)$ time whether there is a feasible route from s to t. You need to briefly argue the correctness of your algorithm.

(b) You are now planning to buy a new car, and you want to know the minimum fuel tank capacity that is needed to travel from s to t. Give an $O((|V| + |E|) \log |V|)$ time algorithm to determine this. You need to briefly argue the correctness of your algorithm.

Problem 4

Let $G = (V, E)$ be a weighted, directed graph with weight function w. Assume G has no negative-weight cycles. Give an $O(|V| \cdot |E|)$-time algorithm to find, for each $v \in V$, the value $\min_{u \in V}\{dist(u, v)\}$. Also briefly argue the correctness of your algorithm. *(Hint: Modify the Bellman-Ford algorithm; you can solve this problem without using an $O(|V| \cdot |E|)$-time all-pairs-shortest-paths algorithm.)*

Problem 5

The PERT chart formulation discussed in class is somewhat unnatural. In a more natural structure, vertices would represent jobs and edges would represent sequencing constraints; that is, edge (u, v) would indicate that job u must be performed before job v. We would then assign weights to vertices instead of edges. In this problem, you are given a DAG graph $G = (V, E)$ with a weight function $w : V \to \mathbb{R}^+$. Assume G has a unique source s and a unique sink t. Design an $O(|V| + |E|)$ time algorithm that computes, for each $v \in V$:

- The earliest start time of the job represented by v, assuming the job represented by s starts at 0.
- The latest start time of the job represented by v, without affecting the project's duration.
- Whether v is in some critical path.

Problem 6

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.