Problem Set 12

Data Structures and Algorithms, Fall 2022

Due: December 15, in class.

Problem 1

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 on some critical path.

Problem 2

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).

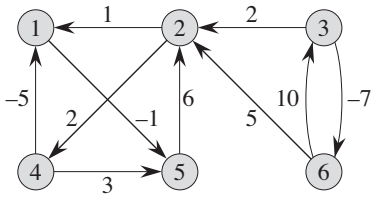
(a) Describe and analyze an efficient algorithm for the following problem: Given a set of exchange rates $r_{i,j}$, and two currencies s and t, find the most advantageous sequence of currency exchanges for converting currency s into currency t.

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 $r_{i_1,i_2} \cdot r_{i_2,i_3} \cdot \cdots \cdot r_{i_{k-1},i_k} \cdot r_{i_k,i_1} > 1$. 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.

Problem 3

(a) Run the Floyd-Warshall algorithm on the following weighted, directed graph. Show the $dist$ values for all pairs of nodes after each iteration of the outer-most loop.



(b) How to use the Floyd-Warshall algorithm's output to detect the presence of a negative-weight cycle?

Problem 4

(a) Use Johnson's algorithm to find the shortest paths between all pairs of vertices in the graph used in Problem 3.(a), visualize algorithm's execution via a figure similar to Figure 25.6 in the CLRS textbook.

(b) Suppose that we run Johnson's algorithm on a directed graph G with weight function w. Assume G does not contain negative-weight cycles. Show that if G contains a 0-weight cycle c, then $\hat{w}(u, v) = 0$ for every edge (u, v) in c.

Problem 5

Suppose that we wish to maintain the transitive closure of a directed graph $G = (V, E)$ as we insert edges into E. That is, after each edge has been inserted, we want to update the transitive closure of the edges inserted so far. Assume that the graph G has no edges initially and that we represent the transitive closure as a boolean matrix.

(a) Show how to update the transitive closure of a graph $G = (V, E)$ in $O(|V|^2)$ time when a new edge is added to G.

(b) Give an example of a graph G and an edge e such that $\Omega(|V|^2)$ time is required to update the transitive closure after the insertion of e into G, no matter what algorithm is used.

(c) Describe an algorithm for updating the transitive closure as edges are inserted into the graph. For any sequence of x insertions, your algorithm should run in total time $\sum_{i=1}^{x} t_i = O(|V|^3)$, where t_i is the time to update the transitive closure upon inserting the i^{th} edge. Prove that your algorithm attains this time bound.

Problem 6

Consider the 0-1 knapsack problem. Given array $v[1, \cdots, n]$ denoting the values of the n items, array $w[1, \cdots, n]$ denoting the weights of the n items, and W denoting the capacity of the knapsack. Devise an algorithm as efficient as possible to correctly solve the problem. You need to analyze the runtime of your algorithm.