CSOR 4231: HW 5

Problem 1 Give an $O(V+E)$-time algorithm that, given a directed graph $G = (V, E)$, constructs another graph $G' = (V, E')$ such that G and G' have the same strongly connected components, G' has the same component graph as G, and $|E'|$ is as small as possible.

Problem 2 Consider a new divide-and-conquer algorithm for computing minimum spanning trees, which goes as follows. Given a graph $G = (V, E)$, partition the set V of vertices into two sets V_1 and V_2 such that $|V_1|$ and $|V_2|$ differ by at most 1. Let E_1 be the set of edges that are incident only on vertices in V_1, and let E_2 be the set of edges that are incident only on vertices in V_2. Recursively solve a minimum-spanning-tree problem on each of the two subgraphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. Finally, select the minimum-weight edge in E that crosses the cut (V_1, V_2), and use this edge to unite the resulting two minimum spanning trees into a single spanning tree.

Either argue that the algorithm correctly computes a minimum spanning tree of G, or provide an example for which the algorithm fails.

Problem 3

a) Give a simple example of a connected graph such that the set of edges $\{(u, v)$ such that there exists a cut $(S, V - S)$ such that (u, v) is a light edge crossing $(S, V - S)\}$ does not form a minimum spanning tree.

b) Show that a graph has a unique minimum spanning tree if, for every cut of the graph, there is a unique light edge crossing the cut. Show that the converse is not true by giving a counterexample.

Problem 4

Arbitrage is the use of discrepancies in currency exchange rates to transform one unit of a currency into more than one unit of the same currency. For example, suppose that 1 U.S. dollar buys 64 Indian rupees, 1 Indian rupee buys 1.8 Japanese yen, and 1 Japanese yen buys 0.009 U.S. dollars. Then, by converting currencies, a trader can start with 1 U.S. dollar and buy $64 \times 1.8 \times 0.009 = 1.0368$ U.S. dollars, thus turning a profit of 3.68 percent.

Suppose that you are given n currencies $c_1, c_2, \ldots, c_n$ and an $n \times n$ table R of exchange rates, such that one unit of currency c_i buys $R[i, j]$ units of currency c_j.

1. Give an efficient algorithm to determine whether or not there exists a

sequence of currencies $< c_{i_1}, c_{i_2}, \ldots, c_{i_k} >$ such that

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

Analyze the running time of your algorithm.

2. Give an efficient algorithm to print out such a sequence if one exists. Analyze the running time of your algorithm.

Problem 5.

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. (Here, the shortest path is by weight, not the number of edges.) 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.

Problem 6

Let $G = (V, E)$ be a directed graph with weight function w, and let $n = |V|$. We define the mean weight of a cycle $c =< e_1, e_2, \ldots, e_k >$ of edges in E to be

$$\mu(c) = \frac{1}{k} \sum_{i=1}^{k} w(e_i) \ .$$

Let $\mu^* = \min\{\mu(c)$ such that c is a directed cycle in $G\}$. We call a cycle c for which $\mu(c) = \mu^*$ a minimum mean-weight cycle. This problem investigates an efficient algorithm for computing μ^*.

Assume without loss of generality that every vertex $v \in V$ is reachable from a source vertex $s \in V$. Let $\delta(s, v)$ be the weight of a shortest path from s to v, and let $\delta_k(s, v)$ be the weight of a shortest path from s to v consisting of *exactly* k edges. If there is no path from s to v with exactly k edges, then $\delta_k(s, v) = \infty$.

1. Show that if $\mu^* = 0$, then G contains no negative-weight cycles and $\delta(s, v) = \min\{\delta_k(s, v)$ such that $0 \leq k \leq n - 1\}$ for all vertices $v \in V$.

2. Show that if $\mu^* = 0$, then

$$\max\{\frac{\delta_n(s, v) - \delta_k(s, v)}{n - k} \text{ such that } 0 \leq k \leq n - 1\} \geq 0$$

for all vertices $v \in V$. (Hint: Use both properties from part (a).)

3. Let c be a 0-weight cycle, and let u and v be any two vertices on c. Suppose that $\mu^* = 0$ and that the weight of the simple path from u to v along the cycle is x. Prove that $\delta(s, v) = \delta(s, u) + x$. (Hint: The weight of the simple path from v to u along the cycle is $-x$.)

4. Show that if $\mu^* = 0$, then on each minimum mean-weight cycle there exists a vertex v such that

$$\max\{\frac{\delta_n(s,v) - \delta_k(s,v)}{n-k} \text{ such that } 0 \le k \le n-1\} = 0 \ .$$

(Hint: Show how to extend a shortest path to any vertex on a minimum mean-weight cycle along the cycle to make a shortest path to the next vertex on the cycle.)

5. Show that if $\mu^* = 0$, then the minimum value of

$$\max\{\frac{\delta_n(s,v) - \delta_k(s,v)}{n-k} \text{ such that } 0 \le k \le n-1\} \ ,$$

taken over all vertices $v \in V$, equals 0.

6. Show that if you add a constant t to the weight of each edge of G, then μ^* increases by t. Use this fact to show that μ^* equals the minimum value of

$$\max\{\frac{\delta_n(s,v) - \delta_k(s,v)}{n-k} \text{ such that } 0 \le k \le n-1\} \ ,$$

taken over all vertices $v \in V$.

7. Give an $O(VE)$-time algorithm to compute μ^*.