Problem Set 8

Data Structures and Algorithms, Fall 2020

Due: November 19, in class.

Problem 1

One way to perform topological sorting on a directed acyclic graph $G = (V, E)$ is to repeatedly find a vertex of in-degree 0, output it, and remove it and all of its outgoing edges from the graph. Assuming we use adjacency-list representation, explain how to implement this idea so that it runs in time $O(|V|+|E|)$.

Problem 2

(a) Give an $O(|V| + |E|)$ time algorithm to compute the component graph of a directed graph $G = (V, E)$. Make sure that there is at most one edge between two vertices in the component graph your algorithm produces.

(b) Consider the two-pass SCC algorithm we introduced in class. Professor Bacon claims that the algorithm would still work correctly if it used the transpose graph in the second depth-first search and scanned the vertices in order of increasing finishing times. Do you agree with Professor Bacon? You need to justify your answer.

Problem 3

You are given a directed graph $G = (V, E)$ in which each node $u \in V$ has an associated price p_u which is a positive integer. Define the array `cost` as follows: for each $u \in V$, `cost[u]` is the price of the cheapest node reachable from u (including u itself). Your goal is to design an algorithm that fills in the entire `cost` array (i.e., for all vertices).

(a) Give an $O(|V| + |E|)$ time algorithm that works for directed acyclic graphs.

(b) Give an $O(|V| + |E|)$ time algorithm that works for all directed graphs.

Problem 4 [Bonus Problem]

A directed graph $G = (V, E)$ is "sort-of-connected" if, for every pair of vertices u and v, either u is reachable from v or v is reachable from u (or both). Give an $O(|V| + |E|)$ time algorithm to determine whether a directed graph is sort-of-connected. To get full credit, also prove your algorithm is correct.

Problem 5

(a) Let $G = (V, E)$ be a connected, undirected graph with a real-valued weight function w defined on E. Let A be a subset of E that is included in some minimum spanning tree for G, let $(S, V - S)$ be any cut of G that respects A, and let (u, v) be a safe edge for A crossing $(S, V - S)$. Professor Bacon claims (u, v) is a light edge for the cut. Do you agree with Professor Bacon? You need to justify your answer.

(b) Professor Bacon proposes 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. Do you think Professor Bacon's algorithm is correct? You need to justify your answer.

Problem 6

(a) Let $G = (V, E)$ be an undirected graph. Prove that if all its edge weights are distinct, then it has a unique minimum spanning tree.

(b) Let T be a minimum spanning tree of a graph $G = (V, E)$, and let V' be a subset of V. Let T' be the subgraph of T induced by V', and let G' be the subgraph of G induced by V'. Show that if T' is connected, then T' is a minimum spanning tree of G'.

Problem 7

Let $G = (V, E)$ be an undirected, connected graph whose weight function is $w : E \to \mathbb{R}$, and suppose that all edge weights are distinct. We define a second-best minimum spanning tree as follows. Let $\mathcal{T}$ be the set of all spanning trees of G, and let T' be a minimum spanning tree of G. Then a second-best minimum spanning tree is a spanning tree T such that $w(T) = \min_{T'' \in \mathcal{T} - \{T'\}} \{w(T'')\}$.

(a) Let T be the minimum spanning tree of G. Prove that G contains edges $(u, v) \in T$ and $(x, y) \notin T$ such that $T - \{(u, v)\} \cup \{(x, y)\}$ is a second-best minimum spanning tree of G.

(b) Let T be a spanning tree of G and, for any two vertices u and v, let `max(u,v)` denote an edge of maximum weight on the unique simple path between u and v in T. Describe an $O(|V|^2)$ time algorithm that, given T, computes `max(u,v)` for all (u, v) pairs.

(c) Give an $O(|V|^2 + |E| \lg |V|)$ time algorithm to compute a second-best minimum spanning tree of G.