Homework #7
Introduction to Algorithms/Algorithms 1
600.363/463
Spring 2014
Solutions

March 26, 2014

Problem 1 (20 points)

Suppose that we have an undirected graph $G = (V, E)$ with all edge weights distinct. Prove that G has a unique minimum spanning tree.

0.1 Solution

Suppose by way of contradiction that G has two minimum spanning trees $T^{(1)} = \{e_1^{(1)}, e_2^{(1)}, \ldots, e_{n-1}^{(1)}\}$ and $T^{(2)} = \{e_1^{(2)}, e_2^{(2)}, \ldots, e_{n-1}^{(2)}\}$, so that $w(T^{(1)}) = w(T^{(2)})$. If G has fewer than n edges, than there can be at most one minimum spanning tree, so assume that G has at least n edges. Assume without loss of generality that the edges in $T^{(1)}$ and $T^{(2)}$ are labeled in order by weight so that $w_{e_1^{(1)}} < w_{e_2^{(1)}} < \cdots < w_{e_{n-1}^{(1)}}$ and $w_{e_1^{(2)}} < w_{e_2^{(2)}} < \cdots < w_{e_{n-1}^{(2)}}$. Now, consider the smallest index i for which $e_i^{(1)} \neq e_i^{(2)}$. Such an index must exist by our assumption that $T^{(1)} \neq T^{(2)}$. Suppose without loss of generality that $w_{e_i^{(1)}} < w_{e_i^{(2)}}$, and consider what happens when we add edge $e_i^{(1)}$ to $T^{(2)}$. Since $T^{(2)}$ is a tree and $e_i^{(1)} \notin T^{(2)}$, $T^{(2)} \cup \{e_i^{(1)}\}$ must contain a cycle. Further, since $\cup_{j=1}^{i}\{e_i^{(1)}\} \subseteq T^{(1)}$, this cycle must include at least one edge $e_j^{(2)}$ for some $j \geq i$. By assumption, $w_{e_i^{(1)}} < w_{e_i^{(2)}} < w_{e_j^{(2)}}$. Thus, $T^{(2)} \cup \{e_i^{(1)}\} - \{e_j^{(2)}\}$ is a tree, and has strictly smaller weight than $T^{(2)}$, contradicting our assumption that $T^{(2)}$ is a minimum spanning tree.

Problem 2 (20 points)

Let $G = (V, E)$ be a connected, unweighted, undirected graph and let $u, v \in V$ be two vertices in graph G. Since G is connected, there exists a shortest path between nodes u and v, with some length $\delta(u, v)$. Of course, it is possible that there are many paths from u to v that all have this length. Call the number of $u - v$ paths of this length the *connection strength* between vertices u and v. That is, the connection strength between vertices u and v is the number of paths from u to v of length $\delta(u, v)$. Since G is connected, the connection strength between two vertices must be at least 1. Give an algorithm that takes a connected, unweighted, undirected graph G along with two vertices u and v in G, and returns the connection strength between vertices u and v. Your algorithm should run in $O(|V| + |E|)$ time. Prove the correctness of your algorithm and prove its runtime.

0.2 Solution

It will suffice to make a minor addition to BFS. To see the intuition, suppose that $\delta(u, v) = d$. Consider the set B of all nodes at distance $d - 1$ from u. Let $c(u, v)$ denote the connection strength between nodes u and v. Then $B = \{t : \delta(u, t) = d - 1, (t, v) \in E\}$. Thus, to calculate the connection strength between u and v, it suffices to calculate the quantity $\sum_{t \in B : \delta(u,t) = d-1} c(u, t)$, since a shortest path from u to any $t \in B$ can be extended to be a shortest path to node v by definition of B. Thus, we see the intuition behind our algorithm: we will perform BFS starting at u, calculating $c(t)$ for all nodes at depth $d = 1, 2, \ldots$ until we reach depth $d = \delta(u, v)$ (we know that we will reach this depth, since the graph is connected). Thus, we simply need to add an extra step to BFS whereby for each node t at depth d_t, we update the connection strength for each of its neighbors that are at depth $d_t + 1$ by adding $c(u, t)$ to each neighbor's connection strength.