Solutions to Homework Assignment 7

CS 430 Introduction to Algorithms
Spring 2018

Solution:

1. No it doesn't. Here is a simple counter-example. Consider the directed graph $G(V, \vec{E})$, where

$$V = \{v_1, v_2, v_3\}$$
$$\vec{E} = \{\overrightarrow{(v_1, v_2)}, \overrightarrow{(v_1, v_3)}, \overrightarrow{(v_3, v_1)}\}$$

   A first DFS may result in the finishing time: v_3, v_2, v_1. If we run the second DFS in this order, it would suggest that all vertices are within one strongly connected component, while v_2 is actually not strongly connected with the rest.

2. For a vertex v, $v.d$ in BFS basically means the shortest distance from root to v. With this in mind it is easy to prove the statements.

   1. Suppose there exists a back edge (u, v), we have $u.d - v.d \geq 2$. Then by taking the path from root to v then the edge (u, v) we obtain a path to u with distance smaller than $u.d$, which is a contradiction.

      Suppose there exists a forward edge (u, v), we have $v.d - u.d \geq 2$. Then by taking the path from root to u then the edge (u, v) we obtain a path to v with distance smaller than $v.d$, which is a contradiction.

      Therefore neither back edges nor forward edges exist in a BFS tree.
   2. According to the algorithm on Page 595, $v.d$ is assigned to be $u.d + 1$ only when the tree edge (u, v) is discovered and . Since it is never modified afterwards we have $v.d = u.d + 1$.
   3. Since in 1. we proved that we cannot have any edge (u, v) with $u.d - v.d \geq 2$ or $v.d - u.d \geq 2$, all we are left with are edges with $|u.d - v.d| \leq 1$. An edge (u, v) won't have $u.d = v.d + 1$ since otherwise it would have been discovered earlier as (v, u). Therefore the only possible cases are $u.d = v.d$ and $u.d + 1 = v.d$.

3. Suppose we have edge $e(u, v)$ in G whose weight is decreased. There are two cases.

Case 1. If $e \in T$, simply updated the weight of e in T and that is your new minimum spanning tree.

Case 2. If $e \notin T$, use DFS to find the path P from vertex u to v in T. Find the largest weighted edge e_{max} on P. Compare the weights of e and e_{max} and keep the lower weighted edge in T. Now T is the new minimum spanning tree.

   Both cases can be done in $O(V + E)$.

4. Simply run Dijkstra's Shortest Path from vertex s. After it finishes, the shortest $s - t$ path found by the algorithm will be the shortest $s - t$ path with minimum number of edges. The time complexity is $O(V \lg V)$.

   This is because Dijkstra's algorithm finds the paths based on the number of edges on them. We will prove that any shortest path found within the first n iterations contains at most n edges.

   Proof by induction: Induct on the number of iterations.

   Base case: In the 1^{st} iteration any path found is a direct edge to s.

   Inductive step: In the i^{th} iteration, if a path is updated, it must contain some path from the previous iteration, which has at most $i - 1$ edges, and a new edge. Combining them we have that this path contains at most i edges.

   Now it is easy to see that any shortest $s - t$ path discovered by the algorithm contains the least number of edges.