CSE548 Fall 2019 Analysis of Algorithms Homework2

Tianao Wang 112819772

September 30, 2019

1. Proof by Contradiction.
   Suppose there is a edge e(a,b) which is in graph G but not in T. Due to T is a DFS tree, a must be an ancestor of b. In addition, T is a BFS tree. The distance between u-a and u-b differs at most 1.
   In conclusion, a should be ancestor to b. Moreever, a should directly connected to b. So the suppose is wrong. Therefore, G = T.

2. False.
   Suppose there is c that $diam(G)/apd(G) <= c$. Let us assume a path with k-1 nodes in order and n-k+1 nodes connecting to the last node of the straight path like a star.
   The diam(G) = k. We can enlarge the distance of nodes in the path and nodes in the star are all k. Also, there are at most k*n sets with at least one node in the path. The distance of nodes in the stars are all 2. So the $apd(G) <= (2\binom{n}{2} + k^2 n)/\binom{n}{2} <= 2 + (2k^2)/(n-1)$
   If $n - 1 > 2k^2$, then $apd(G) < 3$. If $k > 3c$, $diam(G)/apd(G) > 3c/3 = c$.
   In conclusion, the suppose is false.

3. Use directed acyclic graph(DAG) and topological sort.
   Create two nodes for each person, one for date of born and the other for date of death. According to each fact, we can connect two nodes in the directed graph. Nodes with early time point to nodes with late time. For example, A borns before A dies, a arrow is built from node(A born) to node(A die). One more example, B borns before C dies, a arrow is built from node(B born) to node(C die).
   After creating the graph, we can check whether there is a circle in the directed graph. If there is a circle in the graph, we can conclude that the facts are incorrected.
   If there is no circle in the graph, we can use topological sort to get a sorted array. Then we can follow the order in the array and give each elements a time by increaseing order.

4. Proof by Contradiction.
   Suppose G has a MST called T in a graph and there is a edge in G called $e(A, B)$ which is not belongs to T. In addtion, the cost between A and B in the MST is $weight(e')$ and $wight(e) < weight(e')$.
   Therefore, we can change the e' into e in the MST. The total cost of MST T is decreaseing. So, T is not a MST in the graph G and this is a contradiction.
   In conclusion, G has a unique minimum spanning tree.

5. First set up a array called Parent[] to store each node's parent. Then create a function called FindCircle(G,s), G is the graph and s is a start node. Using DFS search in this function. If

node s has not been marked, then mark node s and find each edge connecting to node s. For example, t is a node connected to node s. If node t also has not been marked, we can mark t and store P[t] = s and continue FindCircle(G,t). Until finding a node n has been marked, return these two nodes s and n have a circle.
Then we can write the core function. The main task is to delete 9 edges in the graph. We can use the function FindCircle() to find two nodes s,n which have circle and find the common ancestor of s,n called m. Compare e(s,m),e(n,m),e(s,n) and delete the maximum cost edge of these three edges. Continue this process for 9 times, finally we can get the MST of graph G.

6. Correct.
There is a MST T in the graph G which is also a node in graph H. For the other nodes in graph H, there must be some different from node T. For example, maybe there are different paths from node P to node Q in graph G. The path in MST must be minimum, but a path in other nodes in graph H can be different.
Therefore, all the other nodes are neighbors to the node T which is the MST in graph G. In other words, all the other nodes are connected to the node T. So the graph H is always connected.