CS535 Design and Analysis of Algorithms - Assignment 1

Batkhishig Dulamsurankhor - A20543498

September 5, 2024

Problem 1

Solution:

Let's prove that these two statements are equivalent by deriving one statement from another.

1. If a connected subgraph is on n vertices has $n-1$ edges, then the connected subgraph must have exactly one path between every pair of vertices.

   Let's say we have a connected subgraph $G = (V, E)$ that has n vertices and $n-1$ edges, but G also has a pair of vertices that have more than one distinct path between them. This implies that there is a cycle/cycles in the graph which makes it possible to connect the vertices by more than one path. However, a graph with cycle but has the property a cannot exist since having a cycle will result in a loose vertex and a disconnected subgraph. Therefore, property a and b must be both true.

2. If a connected subgraph have exactly one path between every pair of vertices, then the connected subgraph is on n vertices has $n-1$ edges.

   If there exists only one path between every pair of vertices, the graph must be acyclic as we have proven above. For the subgraph to be connected, it must follow the basic property of a tree, which is having $n-1$ edges for n number of vertices.

Problem 2

Solution:

1. The correctness of Prim's algorithm

   Prim's algorithm is a greedy algorithm that after each iteration we choose the vertex that has smallest weight edge to the result set from the unvisited set of edges and remove edges from cutset if they are connected to the vertex, and add the the rest of the connected edge to the cutset. In class, we kept the cutset using heap data structure.

   Firstly, the resulting subgraph will always be a connected tree, since each vertex will be added only once to the visited set (the result set) and it is impossible for a back edge to be added to the result for the reason that a vertex can be added to the result only once.

   Also, on each iteration, the result set will always be a MST for the current result set. We can argue that it is not a MST and some edge e can be replaced with an edge $e\prime$ where $weight(e) > weight(e\prime)$ and both connected to vertex v. However, when we added the vertex v to the result set, we added the minimum weight edge in the cut set and minimum weight edge e is more than $e\prime$ which results in contradiction and $e\prime$ must be chosen. This is true for every iteration, including the last one that gives the final answer.

2. The correctness of Boruvka/Sollin's algorithm

   Boruvka/Sollin's algorithm is also a greedy algorithm. We start off by choosing the smallest weight edge for all vertices, and merge the connected edges to treat them as single edges. Then, run the same process on the new graph and keep repeating it until we get a MST.

   Similar to above proof, at any given moment, resulting tree/trees will always be MSTs. Because, we are choosing the minimum edge for vertices/merged vertices. By combining trees/vertices together with the smallest weighted edge between them, we are building bigger and fewer trees as long as the given graph is connected. Since the number of edges will always be less than the number of vertices and we prevent cycles by treating the merged edges as one to not choose an edge connects to itself again, we can conclude that the result after each step is a tree. This is valid for every iteration, so by conduction the algorithm is will give MST.

Problem 3

Solution:

MST in the original graph remains the same after we add $-min_e w(e)$ to all the edges. This is because to calculate MST, the relative differences between the edge weights are used and not their individual values.

Let's say vertex v is connected to vertex u by edges e_1, e_2, e_3 with the corresponding weights $-4, -2, 3$. The minimum edge between the two vertex would be e_1, because it has the least weight. Now let's add $-min_e w(e) = 4$ to each of the vertex, and the modifed edges are $e_1\prime, e_2\prime, e_3\prime$ will have $0, 2, 7$ respectively. Here, the minimum edge is $e_1\prime$ and even after modification, we still have the same edge.

$$
\begin{aligned}
&(w(e_1) + (-min_e w(e))) - (w(e_2) + (-min_e w(e))) \\
&\quad = (w(e_1) - min_e w(e)) - (w(e_2) - min_e w(e)) \\
&\quad = w(e_1) - min_e w(e) - w(e_2) + min_e w(e) \\
&\quad = w(e_1) - w(e_2)
\end{aligned}
$$

Since all algorithms to calculate MST, like Kruskal's, Prim's and Boruvka/Sollin's etc, involves relative difference between edge weights, MST in the original graph will stay the same after the modification.