# Design & Analysis of Algorithms

# Monsoon Semester III 2020-21

**Lab - 7** Date: **20 November 2020**

**Topics: Graphs**

**AIM**

# In the lab we solved problems based on Graphs

**EXERCISE**

**Minimum spanning tree(MST):** A minimum spanning tree (MST) or minimum weight spanning tree is a subset of the edges of a [connected](https://en.wikipedia.org/wiki/Connected_graph), edge-weighted undirected graph that connects all the [vertices](https://en.wikipedia.org/wiki/Vertex_(graph_theory)) together, without any [cycles](https://en.wikipedia.org/wiki/Cycle_(graph_theory)) and with the minimum possible total edge weight. That is, it is a [spanning tree](https://en.wikipedia.org/wiki/Spanning_tree) whose sum of edge weights is as small as possible.

**Prim’s Algorithm:** Prim's Algorithm is used to find the minimum spanning tree from a graph. Prim's algorithm finds the subset of edges that includes every vertex of the graph such that the sum of the weights of the edges can be minimized. Prim's algorithm starts with the single node and explore all the adjacent nodes with all the connecting edges at every step. The edges with the minimal weights causing no cycles in the graph got selected.

***Algorithm***

***1)****Create a set mstSet that keeps track of vertices already included in MST.****2)****Assign a key value to all vertices in the input graph. Initialize all key values as INFINITE. Assign key value as 0 for the first vertex so that it is picked first.****3)****While mstSet doesn’t include all vertices   
….****a)****Pick a vertex u which is not there in mstSet and has minimum key value.   
….****b)****Include u to mstSet.   
….****c)****Update key value of all adjacent vertices of u.*

**Kruskal’s Algorithm:** Kruskal's algorithm to find the minimum cost spanning tree uses the greedy approach. This algorithm treats the graph as a forest and every node it has as an individual tree. A tree connects to another only and only if, it has the least cost among all available options and does not violate MST properties.

Below are the steps for finding MST using Kruskal’s algorithm

***1.****Sort all the edges in non-decreasing order of their weight.****2.****Pick the smallest edge. Check if it forms a cycle with the spanning tree formed so far. If cycle is not formed, include this edge. Else, discard it.****3.****Repeat step#2 until there are (V-1) edges in the spanning tree.*

**Dijkstra’s Algorithm:** Dijkstra’s algorithm is very similar to [Prim’s algorithm for minimum spanning tree](https://www.geeksforgeeks.org/prims-minimum-spanning-tree-mst-greedy-algo-5/). Like Prim’s MST, we generate a SPT (shortest path tree) with given source as root. We maintain two sets, one set contains vertices included in shortest path tree, other set includes vertices not yet included in shortest path tree. At every step of the algorithm, we find a vertex which is in the other set (set of not yet included) and has a minimum distance from the source.

**Algorithm  
*1)****Create a set*sptSet*(shortest path tree set) that keeps track of vertices included in shortest path tree, i.e., whose minimum distance from source is calculated and finalized. Initially, this set is empty.****2)****Assign a distance value to all vertices in the input graph. Initialize all distance values as INFINITE. Assign distance value as 0 for the source vertex so that it is picked first.****3)****While*sptSet*doesn’t include all vertices  
….****a)****Pick a vertex u which is not there in*sptSet*and has minimum distance value.  
….****b)****Include u to*sptSet*.  
….****c)****Update distance value of all adjacent vertices of u.*

**GRAPH**

**ANALYSIS**

The execution time of the algorithms are as follows:

Prim’s Algorithm: ***O(V2)***

Kruskal’s Algorithm: ***O(ElogE) or O(ElogV)***

Dijktra’s Algorithm: ***O(V2)***

**CONCLUSION**

Hence, different graphs problems has been implemented