Name: Sarvagnya H. Purohit **Registration Number**: 201070056
Date: 19-12-2021 **Branch:** Computer Engineering
Course: DSA **Course Instructor:** Dr. Mahesh Shirole

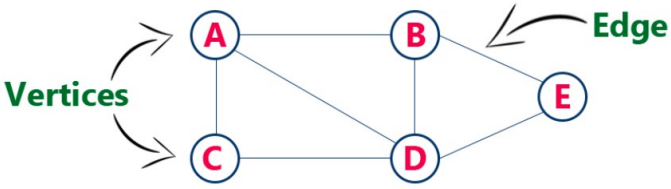
Lab Assignment 9

Aim: To implement Graph using any one representation, BFS and DFS algorithm, Prims and Kruskal's algorithm to find minimum spanning tree. Use graph of all capital cities of all states in India as nodes and distance between them as weight of the edge.

Theory:

Graph:

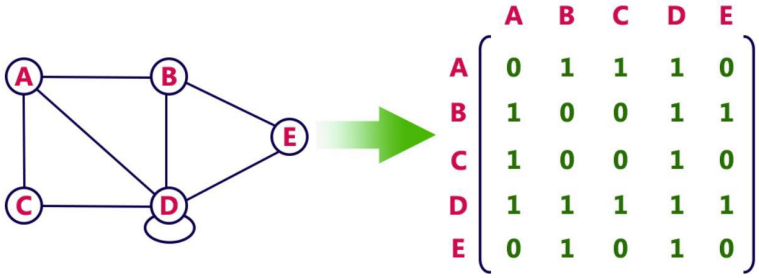
Graph is a non-linear data structure. Graph is a collection of nodes and edges in which nodes are connected with edges. Generally, a graph G is represented as G = (V, E), where V is set of vertices and E is set of edges.



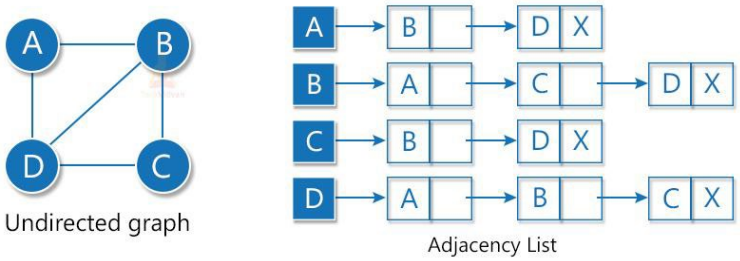
Individual data element of a graph is called as Vertex. Vertex is also known as node.
An edge is a connecting link between two vertices. Edge is also known as Arc. An edge is represented as (startingVertex, endingVertex).

Graph Representation:

1) Adjacency Matrix: An adjacency matrix is a square matrix used to represent a finite graph. The elements of the matrix indicate whether pairs of vertices are adjacent or not in the graph.



2) Adjacency List: The adjacency list is an array of linked lists where the array denotes the total vertices and each linked list denotes the vertices connected to a particular node.



Graph Traversal:

Graph traversal (also known as graph search) refers to the process of visiting each vertex in a graph. Using graph traversal, we visit all the vertices of the graph without getting into looping path. There are two graph traversal techniques – BFS and DFS.

Depth First Search:

A depth-first search (DFS) is an algorithm for traversing a finite graph. DFS visits the child vertices before visiting the sibling vertices; that is, it traverses the depth of any particular path before exploring its breadth. A stack is generally used when implementing the algorithm.

The algorithm begins with a chosen "root" vertex; it then iteratively transitions from the current vertex to an adjacent, unvisited vertex, until it can no longer find an unexplored vertex to transition to from its current location. The algorithm then backtracks along previously visited vertices, until it finds a vertex connected to yet more uncharted territory. It will then proceed down the new path as it had before, backtracking as it encounters dead-ends, and ending only when the algorithm has backtracked past the original "root" vertex from the very first step.

Depth First Search (DFS) algorithm traverses a graph in a depth-ward motion and uses a stack to remember to get the next vertex to start a search, when a dead end occurs in any iteration.

Breadth First Search:

A breadth-first search (BFS) is another technique for traversing a finite graph. BFS visits the sibling vertices before visiting the child vertices, and a queue is used in the search process. This algorithm is often used to find the shortest path from one vertex to another.

Breadth First Search (BFS) algorithm traverses a graph in a breadth-ward motion and uses a queue to remember to get the next vertex to start a search, when a dead end occurs in any iteration.

Spanning Tree, Minimum Spanning Tree:

A spanning tree is a tree that connects all the vertices of a graph with the minimum possible number of edges.

In a weighted graph, a minimum spanning tree is a spanning tree whose weight is the smallest among all possible spanning trees.

Prim's Algorithm:

Prim's algorithm is a greedy algorithm that finds a minimum spanning tree for a weighted undirected graph.

Following are the steps to find minimum spanning tree using Prim's algorithm:

1. Initialize a tree with a single vertex, chosen arbitrarily from the graph.
2. Grow the tree by one edge: of the edges that connect the tree to vertices not yet in the tree, find the minimum-weight edge, and transfer it to the tree.
3. Repeat step 2 (until all vertices are in the tree).

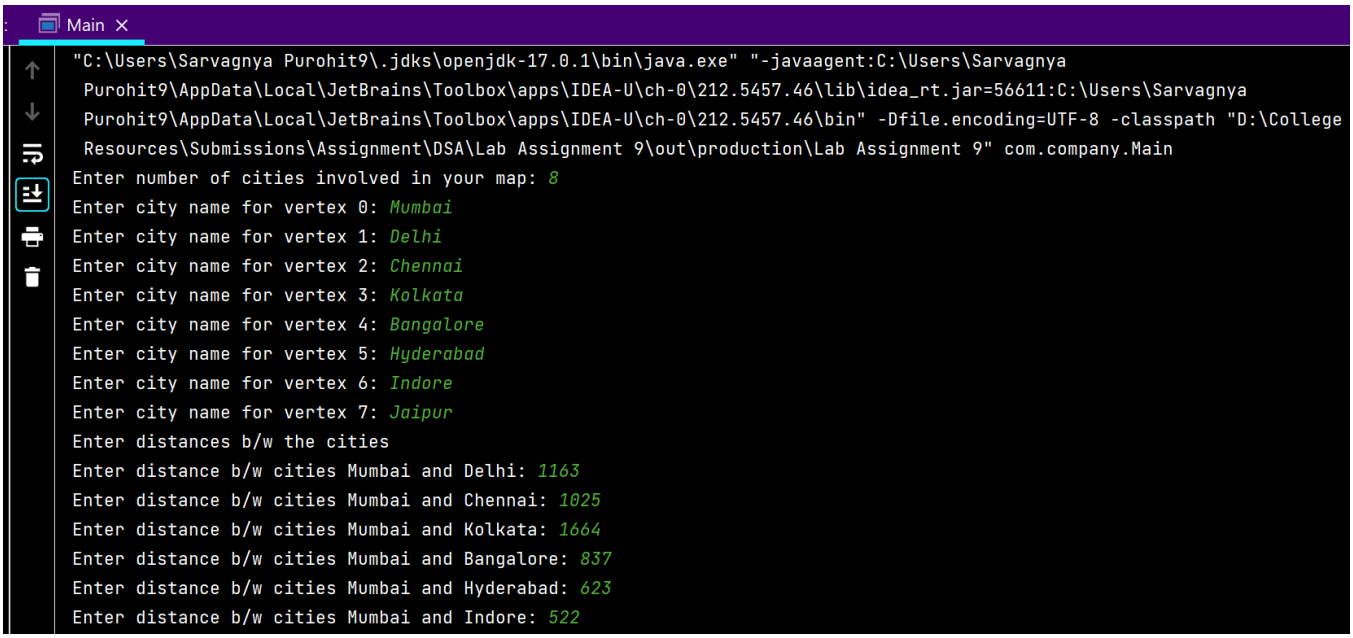
Kruskal's Algorithm:

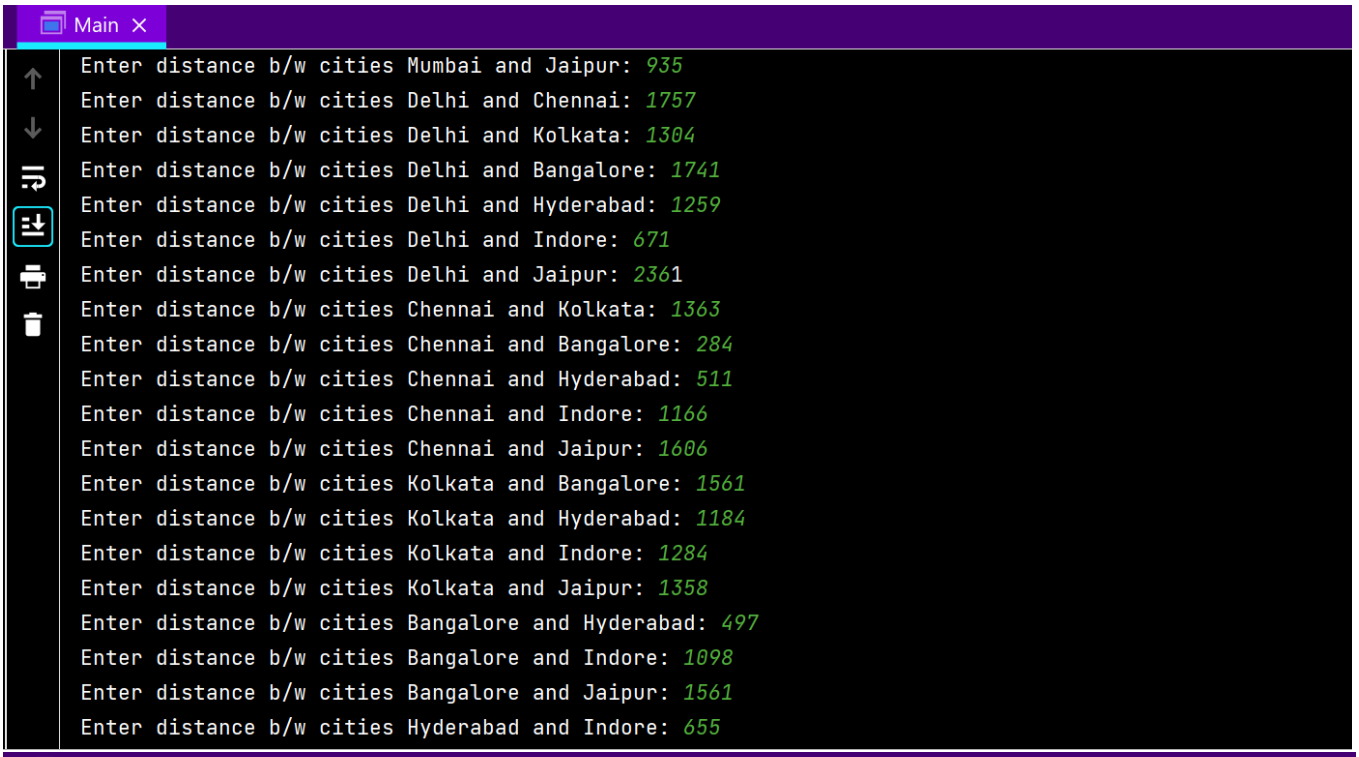
Kruskal's algorithm to find the minimum cost spanning tree uses the greedy approach. This algorithm treats the graph as a forest and every node it has as an individual tree. A tree connects to another only and only if, it has the least cost among all available options and does not violate MST properties.

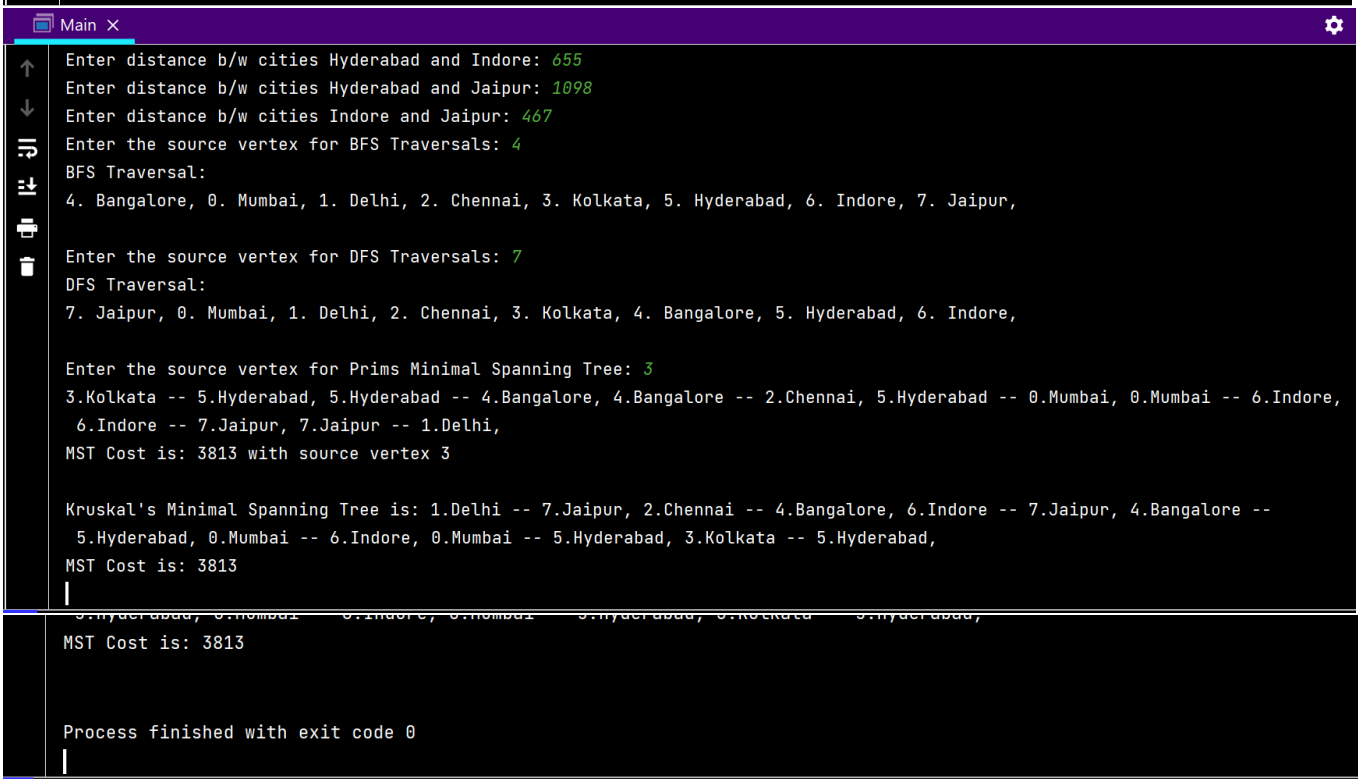
Below are the steps for finding MST using Kruskal's algorithm

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. Union-find algorithm can be used to detect any cycle formed.
3. Repeat step 2 until there are (V-1) edges in the spanning tree.

Test Data and Output of the program:




Main
Enter distance b/w cities Mumbai and Jaipur: 935
Enter distance b/w cities Delhi and Chennai: 1757
Enter distance b/w cities Delhi and Kolkata: 1304
Enter distance b/w cities Delhi and Bangalore: 1741
Enter distance b/w cities Delhi and Hyderabad: 1259
Enter distance b/w cities Delhi and Indore: 671
Enter distance b/w cities Delhi and Jaipur: 2361
Enter distance b/w cities Chennai and Kolkata: 1363
Enter distance b/w cities Chennai and Bangalore: 284
Enter distance b/w cities Chennai and Hyderabad: 511
Enter distance b/w cities Chennai and Indore: 1166
Enter distance b/w cities Chennai and Jaipur: 1606
Enter distance b/w cities Kolkata and Bangalore: 1561
Enter distance b/w cities Kolkata and Hyderabad: 1184
Enter distance b/w cities Kolkata and Indore: 1284
Enter distance b/w cities Kolkata and Jaipur: 1358
Enter distance b/w cities Bangalore and Hyderabad: 497
Enter distance b/w cities Bangalore and Indore: 1098
Enter distance b/w cities Bangalore and Jaipur: 1561
Enter distance b/w cities Hyderabad and Indore: 655


Main
Enter distance b/w cities Hyderabad and Indore: 655
Enter distance b/w cities Hyderabad and Jaipur: 1098
Enter distance b/w cities Indore and Jaipur: 467
Enter the source vertex for BFS Traversals: 4
BFS Traversal:
4. Bangalore, 0. Mumbai, 1. Delhi, 2. Chennai, 3. Kolkata, 5. Hyderabad, 6. Indore, 7. Jaipur,
Enter the source vertex for DFS Traversals: 7
DFS Traversal:
7. Jaipur, 0. Mumbai, 1. Delhi, 2. Chennai, 3. Kolkata, 4. Bangalore, 5. Hyderabad, 6. Indore,
Enter the source vertex for Prims Minimal Spanning Tree: 3
3.Kolkata -- 5.Hyderabad, 5.Hyderabad -- 4.Bangalore, 4.Bangalore -- 2.Chennai, 5.Hyderabad -- 0.Mumbai, 0.Mumbai -- 6.Indore,
6.Indore -- 7.Jaipur, 7.Jaipur -- 1.Delhi,
MST Cost is: 3813 with source vertex 3
Kruskal's Minimal Spanning Tree is: 1.Delhi -- 7.Jaipur, 2.Chennai -- 4.Bangalore, 6.Indore -- 7.Jaipur, 4.Bangalore --
5.Hyderabad, 0.Mumbai -- 6.Indore, 0.Mumbai -- 5.Hyderabad, 3.Kolkata -- 5.Hyderabad,
MST Cost is: 3813

MST Cost is: 3813
Process finished with exit code 0

Conclusion:

In this lab assignment, we learnt about the real-life applications of graph data structure and how we can represent them in a program. A graph is a very powerful data structure which can allow us to analyze complex, multi-connected networks. We learnt how to represent the maps of cities as a graph, find the order of cities closest to a provided city (BFS traversal), find the minimum path (MST) to cover all the cities using Prim's and Kruskal's algorithms. We also learnt how a stack is used to resemble the recursive nature of the DFS traversals, and how a queue is used to carry out BFS traversals.