Pune Institute of Computer Technology, Pune
Department of Computer Engineering

Sub - DSAL Date:- 29/05/2020

Name : - Shrikrushna Santosh Zirape
Roll No : - 21286

Assignment - 07

Problem Statement :-

You have a business with several offices; you want to lease phone lines to connect them 4up with each other; and the phone company charges different amounts of money to connect different pairs of cities. You want a set of lines that connects all your offices with a minimum total cost. Solve the problem by suggesting appropriate data structures.

Software / Hardware Requirements :

1. 64-bit OS Linux or its derivative.
2. Open source c++ programming tools like G++ / GCC

Learning Objective :-

1. To understand concept of graph and minimum cost spanning tree
2. To understand different minimum cost spanning tree algorithms.
3. To implement a minimum spanning tree algorithm.

Learning Outcome :-

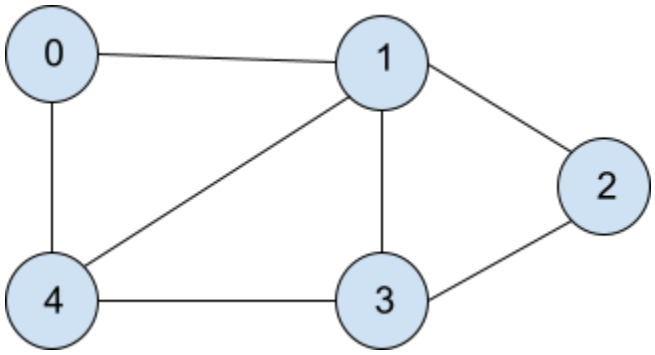
1. Graph implementation using Adjacency matrix or Adjacency list
2. Find total minimum cost using MST Algorithm

Theory:

Graph is a data structure that consists of following two components:

1. A finite set of vertices also called as nodes.
2. A finite set of ordered pairs of the form (u, v) called edge. The pair is ordered because (u, v) is not the same as (v, u) in case of directed graphs(di-graph). The pair of forms (u, v) indicates that there is an edge from vertex u to vertex v. The edges may contain weight/value/cost.

Graphs are used to represent many real life applications. Graphs are used to represent networks. The networks may include paths in a city or telephone network or circuit network. Graphs are also used in social networks like linkedIn, facebook. For example, in facebook, each person is represented with a vertex(or node). Each node is a structure and contains information like person id, name, gender and locale. See this for more applications of graph. Following is an example undirected graph with 5 vertices.



Adjacency Matrix: Adjacency Matrix is a 2D array of size V x V where V is the number of vertices in a graph. Let the 2D array be adj[][], a slot adj[i][j] = 1 indicates that there is an edge from vertex i to vertex j. Adjacency matrix

for an undirected graph is always symmetric. Adjacency Matrix is also used to represent weighted graphs. If adj[i][j] = w, then there is an edge from vertex i to vertex j with weight w.

The adjacency matrix for the above example graph is:

Adjacency Matrix Representation

	0	1	2	3	4
0	0	1	0	0	1
1	1	0	1	1	1
2	0	1	0	1	0
3	0	1	1	0	1
4	1	1	0	1	0

Minimum Spanning Tree :-

Give a graph G = (V, E), the minimum spanning tree (MST) is a weighted graph G’ = (V, E’)

such that:

1. E’ Í E
2. G’ is connected
3. G has the minimum cost

A minimum spanning tree (MST) or minimum weight spanning tree is a subset of the edges of a connected, edge-weighted undirected graph that connects all the vertices together, without any cycles and with the minimum possible total edge weight. That is, it is a spanning tree whose sum of edge

weights is as small as possible. More generally, any undirected graph (not necessarily connected) has a minimum spanning forest, which is a union of the minimum spanning trees for its connected components. There are quite a few use cases for minimum spanning trees. One example would be a telecommunications company which is trying to lay out cables in new neighborhood.

➢ **Prim's Algorithm**

Step 1: Select any vertex

Step 2: Select the shortest edge connected to that vertex.

Step 3: Select the shortest edge connected to any vertex already connected

Step 4: Repeat step 3 until all vertices have been connected

➢ **Kruskal's Algorithm:**

Step 1: Enter number of cities (vertices in graph).

Step 2: Enter the cost of connectivity between each pair of cities (edges in graph).

Step 3: Initialize cost_of_connectivity to 0.

Step 4: Sort all the edges in non-decreasing order of their cost.

Step 5: Pick the smallest cost edge

Step 6: Check if it forms a cycle with the already include edges in the minimum spanning tree

Step 7: If cycle is not formed, include this edge in MST, else discard it

Step 8: Add weight of the selected edge to the cost_of_connectivity.

Step 9: Repeat step 5 ,6, 7 until there are (v-1) edges in the graph.

Step 10: cost_of_connectivity will have minimum cost in the end

Algorithm:

• Prim's Algorithm // input: a graph G

// output: E: a MST for G

1. Select a starting node, v

2. T ß {v} //the nodes in the MST

3. E ß {} //the edges in the MST

4. While not all nodes in G are in the T do

    Choose the edge v' in G − T such that there is a v in T:

    weight (v,v') is the minimum in

    {weight(u,w) : w in G − T and u in T}

    T ß T È {v'}

    E ß E È {(v,v')}

5. return E

Complexity:-

Prims - O(V^2)
Kruskal- O(V^2)

Test Cases

SR No	Test Input	Exp Output	Act Output	Result
1	V = 4 e =4 1 2 =>3 2 3 =>4 3 4 =>5 1 4 =>6	**Adjmax :** 0 3 0 6 3 0 4 0 0 4 0 5 6 0 5 0 **prims(0)** :- 0 3 0 0 3 0 4 0 0 4 0 5 0 0 5 0 ->12 **Kruskal**: 0 3 0 0 3 0 4 0 0 4 0 5 0 0 5 0 ->12	**Adjmax :** 0 3 0 6 3 0 4 0 0 4 0 5 6 0 5 0 **prims(0)** :- 0 3 0 0 3 0 4 0 0 4 0 5 0 0 5 0 ->12 **Kruskal**: 0 3 0 0 3 0 4 0 0 4 0 5 0 0 5 0 ->12	Pass

Conclusion :

We implemented a program for Minimum Spanning Tree.