ASSINGMENT NO.	**7**
TITLE	To write a program for Graph creation and find its minimum cost using Prim's or Kruskal's algorithm.
PROBLEM STATEMENT /DEFINITION	You have a business with several offices; you want to lease phone lines to connect them up 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. **OR** Tour operator organizes guided bus trips across the Maharashtra. Tourists may have different preferences. Tour operator offers a choice from many different routes. Every day the bus moves from starting city S to another city F as chosen by client. On this way, the tourists can see the sights alongside the route travelled from S to F. Client may have preference to choose route. There is a restriction on the routes that the tourists may choose from, the bus has to take a short route from S to F or a route having one distance unit longer than the minimal distance. Two routes from S to F are considered different if there is at least one road from a city A to a city B which is part of one route, but not of the other route
OBJECTIVE	• To understand concept of graph & minimum cost spanning tree. • To understand different minimum cost spanning tree algorithms. • To implement minimum spanning tree algorithms.
OUTCOME	• Graph implementation using Adjacency matrix • Apply and implement MST Algorithms to find total minimum cost
S/W PACKAGES AND HARDWARE APPARATUS USED	• (64-bit)64-BIT Fedora 17 or latest 64-bit update of equivalent Open source OS • Programming Tools (64-bit) Latest Open source update of Eclipse Programming frame work
REFERENCES	1. E. Horowitz S. Sahani, D. Mehata, "Fundamentals of data structures in C++", Galgotia Book Source, New Delhi, 1995, ISBN: 1678298 2. Sartaj Sahani, ―Data Structures, Algorithms andApplications in C++‖, Second Edition, University Press, ISBN:81-7371522 X.

INSTRUCTIONS FOR WRITING JOURNAL	1. Date 2. Assignment no. and title 3. Problem definition 4. Learning Objective 5. Learning Outcome 6. Software / Hardware requirement 7. Concepts related Theory 8. Algorithms/ Pseudo Code 9. Class ADT 10. Test cases 11. Conclusion/Analysis

Prerequisites:

- Basic knowledge of graph
- Graph representation method (Adjacency matrix)
- Object oriented programming features

Learning Objectives

- To understand concept of graph and minimum cost spanning tree.
- To understand minimum cost spanning tree algorithms.

Learning Outcomes

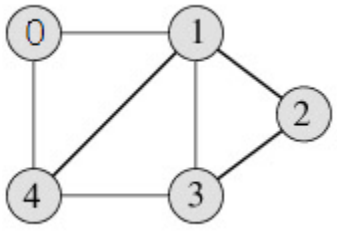
After successful completion of this assignment, students will be able to

- Implement graph using adjacency matrix or adjacency list.
- Apply and implement MST Algorithms to find total minimum cost

Concepts related Theory:

Representation of Graph

Following is an example undirected graph with 5 vertices.



Using 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 undirected graph is always symmetric.

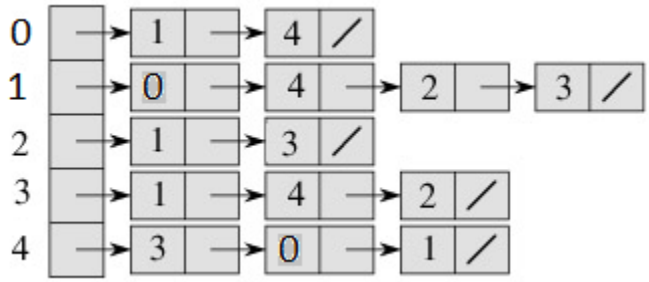
Adjacency Matrix is also used to represent weighted graphs. If adj[i][j] = w, then there is an edge from vertex i to vertex j with weight w.

The adjacency matrix for the above example graph is:

	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

- **Using Adjacency list**

An array of linked lists is used. Size of the array is equal to number of vertices. Let the array be array[]. An entry array[i] represents the linked list of vertices adjacent to the ***i***th vertex. This representation can also be used to represent a weighted graph. The weights of edges can be stored in nodes of linked lists. Following is adjacency list representation of the above graph.



- **Minimum Spanning Tree:**

Give a graph G = (V, E), the minimum spanning tree (MST) is a weighted graph G' = (V, E') such that:

- E' Í E
- G' is connected
- 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

Kruskal's Algorithm:

KRUSKAL(G):
A = ∅
For each vertex v ∈ G.V:
  MAKE-SET(v)
For each edge (u, v) ∈ G.E ordered by increasing order by weight(u, v):
  if FIND-SET(u) ≠ FIND-SET(v):
  A = A ∪ {(u, v)}
  UNION(u, v)
return A

Conclusion: We have successfully calculated total minimum cost of graph using minimum spanning tree algorithm.

Review Questions:
1) What is minimum spanning tree?
2) What are the algorithms to find minimum spanning tree?
3) What is Time and space complexity of the algorithm used?
4) What is adjacency list and adjacency matrix?
5) Difference between adjacency list and adjacency matrix.?
6) Draw and compare graph using Prim's and Kruskal's algorithm?
7) Explain steps in kruskal's algorithm ?
8) Explain steps in prim's algorithm?
9) What are Applications of minimum spanning tree?
10) Explain number of edges in any Minimum Spanning Tree?