**Kruskal‟s Algorithm and prim’s algorithm**

**Aim :**Write a program to implement Kruskal‟s Algorithm and prim’s algorithm.

**Theory:**

**Graphs :**A graph is a set of *vertices* and *edges* which connect them. We write:

**G = (V,E)**

where**V** is the set of vertices and the set of edges,

**E = { (vi,vj) }**

where**vi** and **vj**are in **V**.

**Paths :**A *path*, p, of length, k, through a graph is a sequence of connected vertices:

**p = <v0,v1,...,vk>**

where, for all **i** in (0,**k**-1):

**(vi,vi+1)** is in **E**.

**Cycles :**A graph contains no *cycles* if there is no path of non-zero length through the graph, p =

<v0,v1,...,vk> such that v0 = vk.

**Spanning Trees :**A *spanning tree* of a graph, G, is a set of |V|-1 edges that connect all vertices

of the graph.

**Minimum Spanning Tree :**

Given a connected, undirected graph, a spanning tree of that graph is a subgraph which is a treeand connects all the vertices together. A single graph can have many different spanning trees.We can also assign a *weight* to each edge, which is a number representing how unfavorable it is,and use this to assign a weight to a spanning tree by computing the sum of the weights of theedges in that spanning tree. A minimum spanning tree (MST) or minimum weight spanning treeis then a spanning tree with weight less than or equal to the weight of every other spanning tree.More generally, any undirected graph (not necessarily connected) has a minimum spanningforest, which is a union of minimum spanning trees for its connected components.In general, it is possible to construct multiple spanning trees for a graph, **G**. If a cost, **cij**, isassociated with each edge, **eij= (vi,vj)**, then the minimum spanning tree is the set of edges, **Espan**,forming a spanning tree, such that: **C = sum( cij**| all **eij**in **Espan**) is a minimum.

.

**Kruskal'sAlgorithm :**

**Kruskal's algorithm** is an algorithm in graph theory that finds a minimum spanning tree for aconnected weighted graph. This means it finds a subset of the edges that forms a tree that

includes every vertex, where the total weight of all the edges in the tree is minimized. If the

graph is not connected, then it finds a minimum spanning forest (a minimum spanning tree foreach connected component). Kruskal's algorithm is an example of a greedy algorithm.

This algorithm is continually to select the edges in the order of smallest weights and accepts anedge if it does not form a cycle. Initially the forest consists of **n** single node trees (and no edges).

At each step, we add one (the cheapest one) edge so that it joins two trees together. If it were toform a cycle, it would simply link two nodes that were already part of a single connected tree, sothat this edge would not be needed.

**Algorithm :**

1. Let „G‟ be a connected weighted graph having „v‟ vertices and „e‟ edges.

2. Let „x‟ be a set of all edges of „G‟ arranged in increasing order of weights.

3. Let „T‟ be the minimum spanning tree which is initially empty.

4. while( T contains lesser than v-1 edges AND x is not empty)

{

a. Let „w‟ be the next edge of set „x‟. i.e. an edge of lowest cost in „x‟.

b. Remove „w‟ from „x‟.

c. if ( „w‟ does not create a cycle in „T‟)

Add „w‟ to „T‟

else

Discard „w‟

}

5. Stop

**Prim's algorithm** is an algorithm in graph theory that finds a minimum spanning tree for a

connected weighted graph. This means it finds a subset of the edges that forms a tree that

includes every vertex, where the total weight of all the edges in the tree is minimized.The algorithm continuously increases the size of a tree starting with a single vertex until it spansall the vertices.

Input: A connected weighted graph with vertices V and edges E.

Initialize: **Vnew= {x},** where x is an arbitrary node (starting point) from V, **Enew= {}**

Repeat until **Vnew= V:**

oChoose edge **(u,v)** from E with minimal weight such that u is in **Vnew**and v is not

(if there are multiple edges with the same weight, choose arbitrarily)

oAdd v to **Vnew**, add (u, v) to **Enew**

Output: **Vnew**and **Enew**describe a minimal spanning tree

**Algorithm :**

1. Let „adj‟ be the adjacency matrix of graph „G‟ having „v‟ vertices numbered from 1 to v

and having „e‟ edges.

2. Let distance, path and visited be arrays of „v‟ elements each.

3. Array „distance‟ is initialized to ∞, while „path‟ array and „visited‟ array is initialized to

0.

4. Let current = 1.

5. Let number of vertices already added to the trace be given as nv and let nv=1.

6. Repeat steps 7,8 and 9 while nv ≠ v.

7. for i= 1 to v

if ( adj[current][i] ≠ 0)

if ( visited[i] ≠ 1)

if ( distance[i] >adj[current][i])

{

distance[i] = adj[current][i]

path[i]=current

}

8. min = ∞ (in program min=32767)

for i= 1 to v

if ( visited[i] ≠ 1)

if ( distance[i] < min)

{

min = distance[i]

current = i

}

9. visited[current]=1

10. nv = nv+1

11. Let c be the minimum cost, initially c=0

12. for i=2 to v

c = c+distance[i]

13. for i = 2 to v

Display vertex i is connected to vertex path[i].

14. Stop

**Conclusion:** The time required by Prim's algorithm is O(|V|2). It will be reduced to (|E|log|V|)

if heap is used to keep {v: L(v) < infinity}.A simple implementation using an adjacency matrix graph representation and searching an array of weights to find the minimum weight edge to add requires O(*V2*) running time. Using a simplebinary heap data structure and an adjacency list representation, Prim's algorithm can be shown torun in time O(*E* log *V*) where E is the number of edges and V is the number.The time required by Kruskal's algorithm is O(|E|log|EV|). Using a simple binaryheap data structure and an adjacency list representation, Kruskal‟s algorithm can be shown to runin time O(*E* log *V*) where E is the number of edges and V is the number of vertices.