# Experiment 4

***Aim*:**Find out the Minimum Cost Spanning Tree using Kruskal's Algorithm with the help of the Greedy Approach.

## *4.1CO Attained:* CO2 and CO4

## 4.2 *Objective:*

Kruskal's algorithm: Kruskal's algorithm finds the minimum spanning tree for a weighted connected graph G=(V,E) to get an acyclic sub graph with |V|-1 edges for which the sum of edge weights is the smallest.

## 4.3 *Resources:* Turbo c/Dev C++

## 4.4 *Program Logic:*

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.

```
1 Algorithm Kruskal(E , cost, n ,t)
2 // E is the set of edges in G.G has n vertices. cost [u, v] is the
3 // cost of edge (u,v). t is the set of edges in the minimum-cost
4 // spanning tree. The final cost is returned.
5 {
6 Construct a heap out of the edge costs using Heapify;
7 for i :=1 to n do parent[i]:=-1;
8 // Each vertex is in a different set.
9 i :=0; mincost:=0.0;
10 while ((i< n-1) and (heap not empty)) do
11 {
12 Delete a minimum cost edge(u,v) from the heap
13 and reheapify using Adjust;
14 j :=Find(u); k :=Find(w);
15 if (j != k) then
16 {
17 i: =i+ l;
18 t [i, 1]:=u; t [i, 2]:=v;
19 mincost:=mincost+ cost[u, v];
20 Union(j,k);
21 }
22 }
23 if (i!=n -1) then write ("No spanning tree");
24 else return mincost;
```

```
25}
```

## 4.5 ***Procedure:***

1. Create: Open Dev C++/C and write a program after that save the program with the .c extension.
2. Compile: Alt + F9
3. Execute: Ctrl + F10

## 4.6 ***Program Code:***

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
int i, j, k, a, b, u, v, n, ne = 1;
int min, mincost = 0, cost[9][9], parent[9];
int find(int);
int uni(int, int);
void main()
{
printf("Enter the no. of vertices:\n");
scanf("%d", &n);
printf("\nEnter the cost adjacency matrix:\n");
for (i = 1; i <= n; i++)
  {
  for (j = 1; j <= n; j++)
{
scanf("%d", &cost[i][j]);
      if (cost[i][j] == 0)
cost[i][j] = 999;
    }
  }
printf("The edges of Minimum Cost Spanning Tree are\n");
```

```
  while (ne < n)
  {
    for (i = 1, min = 999; i<= n; i++)
    {
      for (j = 1; j <= n; j++)
      {
        if (cost[i][j] < min)
        {
          min = cost[i][j];
          a = u = i;
          b = v = j;
        }}
    }
    u = find(u);
    v = find(v);
    if (uni(u, v))
    {
printf("%d edge (%d,%d) =%d\n", ne++, a, b, min);
mincost += min;
    }
    cost[a][b] = cost[b][a] = 999;
  }
printf("\nMinimum cost = %d\n", mincost);
getch();
}
int find(int i)
{
  while (parent[i])
i = parent[i];
  return i;
```

```
}
int uni(int i, int j)
{
  if (i != j)
  {
    parent[j] = i;
    return 1;
  }
  return 0;
}
```

### 4.7 ***Conclusion:***

```
          Implementation of Kruskal's algorithm

  Enter the no. of vertices:7

  Enter the cost adjacency matrix:
  0 28 0 0 0 10 0
  28 0 16 0 0 0 14
  0 16 0 12 0 0 0
  0 0 12 0 22 0 18
  0 0 0 22 0 25 24
  10 0 0 0 25 0 0
  0 14 0 0 24 0 0
  The edges of Minimum Cost Spanning Tree are
  1 edge (1,6) =10
  2 edge (3,4) =12
  3 edge (2,7) =14
  4 edge (2,3) =16
  5 edge (4,5) =22
  6 edge (5,6) =25

          Minimum cost = 99
```

### 4.8 ***Analysis:***

The time complexity of Kruskal's Algorithm = O(ElogV) or O(ElogE). where e is the number of edges and v is the number of vertices.

### 4.9 ***Lab Viva Questions:***

1. What is the time complexity of Kruskal's algorithm.
2. Define spanning tree.
3. Define minimum cost spanning tree.