#### **LAB PROGRAM 9**

# AIM: Find Minimum Cost Spanning Tree of a given undirected graph using Prim/Kruskal's algorithm

# SOURCE CODE PRIMS ALGORITHM

```
#include<stdio.h>
#include<conio.h>
int cost[10][10],vt[10],et[10][10],vis[10],j,n;
int sum=0;
int x=1;
int e=0;
void prims();
void main()
{
 int i;
 printf("enter the number of vertices\n");
 scanf("%d",&n);
 printf("enter the cost adjacency matrix\n");
 for(i=1;i<=n;i++)
 {
   for(j=1;j<=n;j++)
    {
         scanf("%d",&cost[i][j]);
   }
   vis[i]=0;
  }
```

```
prims();
  printf("edges of spanning tree\n");
  for(i=1;i<=e;i++)
  {
        printf("%d,%d\t",et[i][0],et[i][1]);
  }
  printf("weight=%d\n",sum);
}
void prims()
{
 int s,min,m,k,u,v;
 vt[x]=1;
 vis[x]=1;
 for(s=1;s<n;s++)
 {
   j=x;
    min=999;
   while(j>0)
   {
          k=vt[j];
         for(m=2;m<=n;m++)
         {
          if(vis[m]==0)
          {
                if(cost[k][m]<min)</pre>
                {
                  min=cost[k][m];
                  u=k;
                  v=m;
                }
```

```
}

j--;

}

vt[++x]=v;

et[s][0]=u;

et[s][1]=v;

e++;

vis[v]=1;

sum=sum+min;
}
```

#### **OUTPUT SCREENSHOT**

```
Inter the number of vertices
4
enter the cost adjacency matrix
0 3 1 6
3 0 5 0
1 5 0 4
6 0 4 0
edges of spanning tree
1,3 1,2 2,4 weight=4

Process returned 9 (0x9) execution time: 54.302 s
Press any key to continue.
- X

Enter the number of vertices
4
enter the cost adjacency matrix
3 0 5 0
1 5 0 4
6 0 4 0
edges of spanning tree
1,3 1,2 2,4 weight=4

Process returned 9 (0x9) execution time: 54.302 s
Press any key to continue.
```

## **KRUSHKALS ALGORITHM**

#include<stdio.h>

#include<conio.h>

```
int find(int v,int parent[10])
{
 while(parent[v]!=v)
   v=parent[v];
 }
 return v;
}
void union1(int i,int j,int parent[10])
{
 if(i<j)
   parent[j]=i;
 else
   parent[i]=j;
}
void kruskal(int n,int a[10][10])
{
 int count,k,min,sum,i,j,t[10][10],u,v,parent[10];
 count=0;
 k=0;
 sum=0;
 for(i=0;i<n;i++)
   parent[i]=i;
 while(count!=n-1)
 {
   min=999;
   for(i=0;i<n;i++)
```

```
{
      for(j=0;j<n;j++)
      {
       if(a[i][j]<min && a[i][j]!=0)
        {
             min=a[i][j];
             u=i;
             v=j;
        }
      }
 }
 i=find(u,parent);
 j=find(v,parent);
 if(i!=j)
 {
      union1(i,j,parent);
      t[k][0]=u;
      t[k][1]=v;
      k++;
      count++;
      sum=sum+a[u][v];
 }
 a[u][v]=a[v][u]=999;
}
if(count==n-1)
{
 printf("spanning tree\n");
 for(i=0;i<n-1;i++)
```

```
{
        printf("%d %d\n",t[i][0],t[i][1]);
   }
    printf("cost of spanning tree=%d\n",sum);
  }
  else
   printf("spanning tree does not exist\n");
 }
void main()
{
 int n,i,j,a[10][10];
 printf("enter the number of nodes\n");
 scanf("%d",&n);
 printf("enter the adjacency matrix\n");
 for(i=0;i<n;i++)
  for(j=0;j<n;j++)
       scanf("%d",&a[i][j]);
 kruskal(n,a);
}
```

## **OUTPUT SCREENSHOT**

```
enter the number of nodes

senter the adjacency matrix
0 2 0 6 0
2 0 3 8 5
0 3 0 0 7
6 8 0 0 9
9 5 7 9 0
spanning tree
0 1
1 2
1 4
0 3
cost of spanning tree=16

Process returned 25 (0x19) execution time: 85.791 s

Press any key to continue.
```