WEEK-7 ADA LAB

SHANA DIYASUJIT

1BM21CS196

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

PRIM'S:

CODE:

```
#include <stdio.h>


float cost[10][10];
int vt[10], et[10][10], vis[10], j, n;
float sum = 0.0;
int x = 1;
int e = 0;
void prims()
{
  int s, m, k, u, v;
  float min;
  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)
          {
            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;
  }
}

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("%f", &cost[i][j]);
    }
    vis[i] = 0;
  }
  prims();
  printf("The Edges of spanning tree\n");
  for (i = 1; i <= e; i++)
  {
    printf("%d --> %d\t", et[i][0],et[i][1]);
  }
  printf("weight=%f\n", sum);

}
```

OUTPUT:

```
enter the number of vertices
5
enter the cost adjacency matrix
0 1 3 2 999
1 0 999 999 999
5 999 0 3 999
2 999 3 0 1.5
999 999 999 1.5 0
The Edges of spanning tree
1 --> 2 1 --> 4 4 --> 5 4 --> 3 weight=7.500000


...Program finished with exit code 16
Press ENTER to exit console.
```

KRUSKAL'S

CODE:

```
#include<stdio.h>
float cost[10][10]; int t[10][10],parent[10],n;
void kruskal()
{
 int i,j,u,v;
 int count=0;
 int k=0;
float sum=0.0;
 for(i=0;i<n;i++)
 {
   parent[i]=i;
 }
 while(count!=n-1)
 {
  float min=999;
  for(i=0;i<n;i++)
  {
   for(j=0;j<n;j++)
   {
    if(cost[i][j]<min&&cost[i][j]!=0)
    {
      min=cost[i][j];
      u=i;
      v=j;
    }
   }
  }
  i=find(u);
  j=find(v);
  if(i!=j)
```

```
 {
  t[k][0]=u;
  t[k][1]=v;
  k++;
  count++;
  sum=sum+cost[u][v];
  union_ij(i,j);
 }
 cost[u][v]=cost[v][u]=999.0;
}
printf("Spanning Tree:\n");
for(i=0;i<n;i++)

{ if(t[i][0]!=t[i][1])
    printf("%d->%d\t",t[i][0],t[i][1]);
}
printf("\nTotal Cost=%f",sum);
getch();
}

void union_ij(int i,int j)
{
 if(i<j)
 {
 parent[j]=i;
 }
 else
 {
 parent[i]=j;
 }
}
```

```
int find(int v)
{
 while(parent[v]!=v)
 {
  v=parent[v];
 }
 return v;
}

int main()
{
 int i,j;
 printf("\nEnter the number of vertices:");
 scanf("%d",&n);
 printf("\n Enter the cost matrix:\n");
 for(i=0;i<n;i++)
 {
  for(j=0;j<n;j++)
  {
    scanf("%f",&cost[i][j]);
  }
 }
 kruskal();
 return 0;
}
```

OUTPUT:

```
Enter the number of vertices:5

 Enter the cost matrix:
0 1 5 2 999
1 0 999 999 999
5 999 0 3 999
2 3 999 0 1.5
999 999 999 1.5 0
Spanning Tree:
0->1    3->4    0->3    2->3
Total Cost=7.500000
```

From a given vertex in a weighted connected graph, find shortest paths to other vertices using **Dijkstra's algorithm.**

CODE:

```
#include<stdio.h>
#define INFINITY 9999
#define MAX 10

void dijkstra(int G[MAX][MAX],int n,int startnode);

int main()
{
   int G[MAX][MAX],i,j,n,u;
   printf("Enter no. of vertices:");
   scanf("%d",&n);
   printf("\nEnter the adjacency matrix:\n");
   for(i=0; i<n; i++)
      for(j=0; j<n; j++)
         scanf("%d",&G[i][j]);
   printf("\nEnter the starting node:");
   scanf("%d",&u);
   dijkstra(G,n,u);
   return 0;
}

void dijkstra(int G[MAX][MAX],int n,int startnode)
{

   int cost[MAX][MAX],distance[MAX],pred[MAX];
   int visited[MAX],count,mindistance,nextnode,i,j;
```

```
//pred[] stores the predecessor of each node
//count gives the number of nodes seen so far
//create the cost matrix
   for(i=0; i<n; i++)
      for(j=0; j<n; j++)
         if(G[i][j]==0)
            cost[i][j]=INFINITY;
         else
            cost[i][j]=G[i][j];
//initialize pred[],distance[] and visited[]
   for(i=0; i<n; i++)
   {
      distance[i]=cost[startnode][i];
      pred[i]=startnode;
      visited[i]=0;
   }
   distance[startnode]=0;
   visited[startnode]=1;
   count=1;
   while(count<n-1)
   {
      mindistance=INFINITY;
//nextnode gives the node at minimum distance
      for(i=0; i<n; i++)
         if(distance[i]<mindistance&&!visited[i])
         {
            mindistance=distance[i];
            nextnode=i;
         }
```

```
//check if a better path exists through nextnode
        visited[nextnode]=1;
        for(i=0; i<n; i++)
           if(!visited[i])
              if(mindistance+cost[nextnode][i]<distance[i])
              {
                 distance[i]=mindistance+cost[nextnode][i];
                 pred[i]=nextnode;
              }
        count++;
     }

//print the path and distance of each node
     for(i=0; i<n; i++)
        if(i!=startnode)
        {
           printf("\nDistance of node%d=%d",i,distance[i]);
           printf("\nPath=%d",i);
           j=i;
           do
           {
              j=pred[j];
              printf("<-%d",j);
           }
           while(j!=startnode);
        }
}
```

OUTPUT:

```
Enter no. of vertices:5

Enter the adjacency matrix:
0 3 999 7 999
3 0 4 2 999
999 4 0 5 6
7 2 5 0 4
999 999 6 4 0

Enter the starting node:0

Distance of node1=3
Path=1<-0
Distance of node2=7
Path=2<-1<-0
Distance of node3=5
Path=3<-1<-0
Distance of node4=9
Path=4<-3<-1<-0
```