**Design and implement C/C++ Program to find Minimum Cost Spanning Tree of a given connected undirected graph using Kruskal's algorithm.**

ALGORITHM: Kruskal ( G )

//Input: A weighted connected graph G = < V, E >.

//Output: The set of edges composing a minimum spanning tree of G.

Step 1: Sort E in increasing order of the edge weights w(ei1) <=…..<= w(ei|E| )

Step 2: Et = NULL; ecounter = 0 //initialize the set of tree edges and its size

Step 3: k = 0 //initialize the number of processed edges

Step 4: while ecounter < | V | - 1 do {

Step 5: k = k + 1

Step 6: if Et U { eik } is acyclic then {

Step 7: Et = Et U { eik }

Step 8: ecounter = ecounter + 1 } // end if } // end while

Step 9: return Et

**Program**

#include<stdio.h>

int cost[10][10];

**void krush(int n)**

**{**

int parent[10],i,j,ne=0,min,mincost=0,a,b,u,v;

for(i=1;i<=n;i++)

{

parent[i]=0;

}

while(ne!=n-1)

{

min=999;

for(i=1;i<=n;i++)

{

for(j=1;j<=n;j++)

{

**if(cost[i][j]<min)**

{

min=cost[i][j];

a=i;

u=i;

b=j;

v=j;

}

}

}

while(parent[u]!=0)

u=parent[u];

while(parent[v]!=0)

v=parent[v];

**if(u!=v)**

{

printf("\nVertex(%d-->%d)=%d",a,b,min);

ne++;

parent[v]=u;

mincost=mincost+min;

}

cost[a][b]=cost[b][a]=999;

}

printf("\nCost of Spanning Tree is = %d\n",mincost);

**}**

**int main()**

**{**

int i,j,n;

printf("Enter the no of nodes\n");

scanf("%d",&n);

printf("Enter the matrix\n");

for(i=1;i<=n;i++)

{

for(j=1;j<=n;j++)

{

scanf("%d",&cost[i][j]);

}

}

**krush(n);**

return 0;

**}**

**Output:**

Enter the no of nodes

3

Enter the matrix

999 10 15

10 999 20

15 20 999

Vertex(1-->2)=10

Vertex(1-->3)=15

Cost of Spanning Tree is = 25

Design and implement C/C++ Program to find Minimum Cost Spanning Tree of a given connected undirected graph using Prim's algorithm.

Prim ( G )

//Input: A weighted connected graph G = < V, E >

//Output: Et, the set of edges composing a minimum spanning tree of G

Step 1: Vt= { v0 } //the set of tree vertices can be initialized with any vertex

Step 2: Et = NULL Step 3: for i = 1 to | V | - 1 do {

Step 4: find a minimum weight edge e\*= (v\*, u\*) among all the edges (v, u)

Step 5: such that v is in Vt and u is in V – Vt

Step 6: Vt = Vt U {u \*}

Step 7: Et = Et U {e\*} } //end for

Step 8: return Et

Program

#include<stdio.h>

int a[10][10];

**void prims(int n)**

**{**

int i,j;

int mincost=0,min,u,v,visited[10],ne=0;

for(i=1;i<=n;i++)

{

visited[i]=0;

}

visited[1]=1;

while(ne<n-1)

{

min=999;

for(i=1;i<=n;i++)

{

if(visited[i]==1)

{

for(j=1;j<=n;j++)

{

**if(a[i][j]<min)**

{

min=a[i][j];

u=i;

v=j;

}

}

}

}

**if(visited[u]==0||visited[v]==0)**

**{**

printf("From vertex %d to %d cost=%d\n",u,v,min);

ne++;

visited[v]=1;

mincost=mincost+min;

**}**

a[u][v]=a[v][u]=9999;

}

printf("The Mincost is %d\n",mincost);

**}**

**int main()**

**{**

int i,j,n;

printf("Enter the no of nodes\n");

scanf("%d",&n);

printf("Enter matrix\n");

for(i=1;i<=n;i++)

{

for(j=1;j<=n;j++)

{

scanf("%d",&a[i][j]);

}

}

prims(n);

return 0;

**}**

**Output:**

Enter the no of nodes

5

Enter matrix

999 2 999 6 999

2 999 3 8 5

999 3 999 999 7

6 8 999 999 9

999 5 7 9 999

From vertex 1 to 2 cost=2

From vertex 2 to 3 cost=3

From vertex 2 to 5 cost=5

From vertex 1 to 4 cost=6

The Mincost is 16