Program 1: Kruskal's Algorithm

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

DESCRIPTION: Kruskal's is a greedy algorithm for finding minimum cost spanning tree that provides an optimal solution. The algorithm begins by sorting the graph's edges in nondecreasing order of their weights. Then, starting with the empty subgraph, it scans this sorted list, adding the next edge on the list to the current subgraph if such an inclusion does not create a cycle and simply skipping the edge otherwise. It is implemented with help of Union-Find algorithm which is given as below:

- **union (x, y)** constructs the union of the disjoint subsets Sx and Sy containing x and y, respectively, and adds it to the collection to replace Sx and Sy, which are deleted from it.
- **Find(x)** returns a subset containing x.

```
ALGORITHM  Kruskal(G)
    //Kruskal's algorithm for constructing a minimum spanning tree
    //Input: A weighted connected graph $G = \langle V, E \rangle$
    //Output: $E_T$, the set of edges composing a minimum spanning tree of $G$
    sort $E$ in nondecreasing order of the edge weights $w(e_{i_1}) \le \cdots \le w(e_{i_{|E|}})$
    $E_T \leftarrow \varnothing$;   $ecounter \leftarrow 0$      //initialize the set of tree edges and its size
    $k \leftarrow 0$                            //initialize the number of processed edges
    while $ecounter < |V| - 1$ do
        $k \leftarrow k + 1$
        if $E_T \cup \{e_{i_k}\}$ is acyclic
            $E_T \leftarrow E_T \cup \{e_{i_k}\}$;   $ecounter \leftarrow ecounter + 1$
    return $E_T$
```

Program:

```
#include<stdio.h>

int ne=1,min_cost=0;


void main()

{

int n,i,j,min,a,u,b,v,cost[20][20],parent[20];

printf("Enter the no. of vertices:");

 scanf("%d",&n);

printf("\nEnter the cost matrix:\n");

for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
scanf("%d",&cost[i][j]);
for(i=1;i<=n;i++)
parent[i]=0;
printf("\nThe edges of spanning tree are\n");
 while(ne<n)
{
min=999;
 for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
if(cost[i][j]<min)
{
min=cost[i][j];
a=u=i;
b=v=j;
}
}
}
while(parent[u]){

u=parent[u];
printf("the value of u is %d",u);
}
while(parent[v]){
v=parent[v];
printf("the value of v is %d",v);
}
if(u!=v)
{
printf("Edge %d\t(%d->%d)=%d\n",ne++,a,b,min); min_cost=min_cost+min;
parent[v]=u;
}
cost[a][b]=cost[b][a]=999;
}
printf("\nMinimum cost=%d\n",min_cost);
}
```

Output: // Include your Output here

Learning Outcomes: // Include your Learning outcomes here