LAB. TEST-2

NAME -: TUSHAR . A. PAI

USN -: 1BM19CS174

SEC | SEM -: D | 4th

DATE -: 5th July 2021

Sign -: Tushar

Program - 6 Kruskals algorithm

```
# include <stdio.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 i)
{ while (parent [i])
      i=parent [i];
  return i;
}
```

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

 void main() {
  printf ("Enter the no. 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]);
             if (cost[i].[j]==0)
                 cost[i][j]=99;
        }
  }
  printf ("The edges of Minimum cost Spanning tree are \n");
  while (ne<n)
  {     for (i=0, 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) \n", ne++, a,b,min);
       mincost += min;
     }.
    cost[a][b] = cost[b][a] = 999.
  }
  printf("\n \t Minimum cost = %d \n", mincost);
}
```

Modification.

To detect cycles in the graph we use the Union-Find algorithm as it uses an incremental edge adding approach to detect cycles

```
void kruskals ()
{ count=0, k=0, sum=0; {
   for(i=0; i<n; i++)
     parent[i]=i;
  while (count != num_edge)
   { 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)
   { printf("Vertex forming cycle %d & %d \n", u, v);
       count++;}
   }.
   :
   :
```

everything else same as original program