Date 01/02/24.   Expt. No. 04

Expt. Name   Page No. 9

AIM: Find out the minimum cost spanning tree using Kruskal's Algorithm with help of the Greedy Approach.

Algorithm:

```
01. E is the set at Edge in G.G has n vertices.
    Cost [u,v] is the.
02. cost at Edge (u,v). t is the set af edges in the
    minimum-cost.
03. Spanning tree the find cost is returned.
04. Construct a Heap out af the edge costs using
    Heapify :-
05. for i=1 to n do parent[i] = 1,
06. Each vertex in a different set
            i = 0; min-cost = 0.0;
07. i = 0; min-cost = 0.0;
08. while (i<n-1) and (heap not empty) do {
09. Delete minimum cost edge (u,v) from the heap
            j = find (u); k = Find (w)
        if (j != k) then { i = i+1
            t [i,1] = 0;
            t [i,2] = v;
10. Mincost = mincost +cost [0,v];
```

Teacher's Signature:

Date

Expt. No.

Expt. Name

Page No. 10

```
11.  Union (j, k)
     }
12.  if (i != n-1) then write ("No spanning tree")
      else  return
        mincost
     }
13.  END.
```

Code:

```
# include <stdio.h>
# include <conio.h>
# include <stdlib.h>

int i, j, a, b, u, v, n, ne = 1;
 int min, mincost = 0, cost [9][9], parent [9];
 int Find (int);
  int union (int, int);

void main() {
 printf ("Enter the no. of vertices : \n");
  scanf ("%d", &n);
 printf ("\n Enter the cost adjacency Matrix:");
  for (i=n; i<=n; i++)
   {
     for (j=1; j<=n; j++)
       {
         scanf ("%d", & cost [i][j]);
         if (cost [i][j] == 0)
             cost [i][j] = 999;
       }
   }
```

Teacher's Signature:

Output:

```
Enter the No. of Vertices : 3
Enter the cost of adjacency Matrix : 1 2 3 4 5 6 7 8 9

the Edges of minimum cost
            Spanning tree are :-

   1 Edge (1,2) = 2
    2 Edge (1,3) = 3

          Minimum Cost = 5.
```

Date

Expt. No.

Expt. Name

Page No. 11

```
printf("the Edges of minimum cost Spanning
        tree are : ");
 while (ne < n)
  {
  for (i=1 ; min = 999 ; i<=n ; i++) {
      for (j=1 ; j<=n ; j++)
      {
         min = cost [i] [j]
           a = u = i;
           b = r = j;
         }
      }
  }
 u = find (u);
 r = find (v,r);

if (uni (v,r))
 {
  printf ("%d edge (%d, %d) = %d\n", ne++, a, b,
          min);
  min cost += min ;
 }
 cost [a][b] = cost [b][a] = 999;
}
 printf ("Minimum costs : %d ), mincost);
 getch ();
}
int find (int i)
{
  while ( parent [i])
     i = parent [i];
```

Teacher's Signature:

Date

Expt. No.

Expt. Name

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

[illegible]