Vashisth Zatakia
60003200163 B3 IT

Aim: Implementation of Kruskal's Algorithm for MST

Theory:

Spanning tree - A spanning tree is the subgraph of an undirected connected graph.

Minimum Spanning tree - Minimum spanning tree can be defined as the spanning tree in which the sum of the weights of the edge is minimum. The weight of the spanning tree is the sum of the weights given to the edges of the spanning tree.

Now, let's start with the main topic.

Kruskal's Algorithm is used to find the minimum spanning tree for a connected weighted graph. The main target of the algorithm is to find the subset of edges by using which we can traverse every vertex of the graph. It follows the greedy approach that finds an optimum solution at every stage instead of focusing on a global optimum.

In Kruskal's algorithm, we start from edges with the lowest weight and keep adding the edges until the goal is reached. The steps to implement Kruskal's algorithm are listed as follows -

- First, sort all the edges from low weight to high.
- Now, take the edge with the lowest weight and add it to the spanning tree. If the edge to be added creates a cycle, then reject the edge.
- Continue to add the edges until we reach all vertices, and a minimum spanning tree is created.

Complexity Analysis :

Let us assume a graph with e number of edges and n number of vertices. Kruskal's algorithm starts with sorting of edges. Time complexity of sorting algorithm= O (e log e). In Kruskal's algorithm, we have to add an edge to the spanning tree, in each iteration. This involves merging of two components. Time complexity of merging of components= O (e log n). Overall time complexity of the algorithm= O (e log e) + O (e log n)

Code:

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

Vashisth Zatakia
60003200163 B3 IT

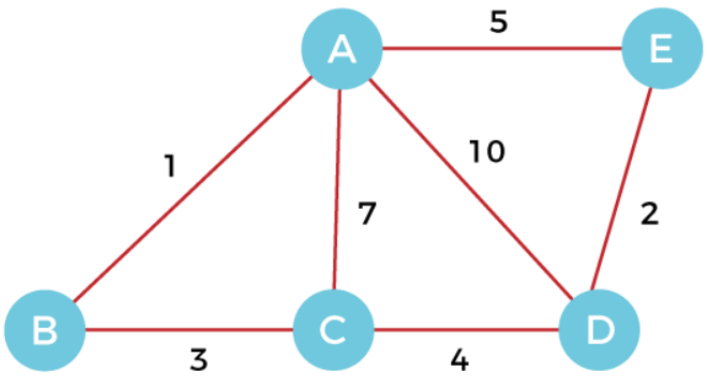
```
#include <stdlib.h>
int i, j, k, a, b, u, v, n, ne = 1;
int min, mincost = 0, cost[9][9], parent[9];
int find(int i)
{
   while (parent[i])
      i = parent[i];
   return i;
}

int uni(int i, int j)
{
   if (i != j)
   {
      parent[j] = i;
      return 1;
   }
   return 0;
}
int main()
{
   printf("Kruskal's algorithm in C\n");
   printf("========================\n");

   printf("Enter the no. of vertices:\n");
   scanf("%d", &n);
```

```
printf("\nEnter 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] = 999;
   }
}

printf("The edges of Minimum Cost Spanning Tree are\n");
while (ne < n)
{
   for (i = 1, 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) =%d\n", ne++, a, b, min);
            mincost += min;
        }
        cost[a][b] = cost[b][a] = 999;
    }
    printf("\nMinimum cost = %d\n", mincost);
    return 0;
}
```

Example of Kruskal's algorithm

Suppose a weighted graph is -



Now, let's start constructing the minimum spanning tree.

Step 1 - First, add the edge **AB** with weight **1** to the MST.

Step 2 - Add the edge **DE** with weight **2** to the MST as it is not creating the cycle.

Step 3 - Add the edge **BC** with weight **3** to the MST, as it is not creating any cycle or loop.

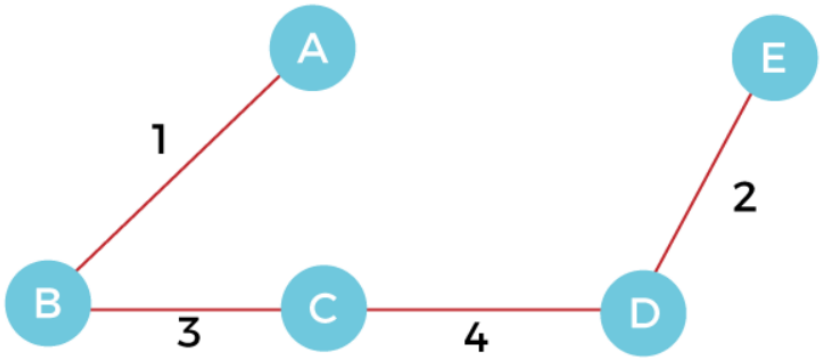
Step 4 - Now, pick the edge **CD** with weight **4** to the MST, as it is not forming the cycle.

Step 5 - After that, pick the edge **AE** with weight **5.** Including this edge will create the cycle, so discard it.

Step 6 - Pick the edge **AC** with weight **7.** Including this edge will create the cycle, so discard it.

Step 7 - Pick the edge **AD** with weight **10.** Including this edge will also create the cycle, so discard it.

So, the final minimum spanning tree obtained from the given weighted graph by using Kruskal's algorithm is -



The cost of the MST is = AB + DE + BC + CD = 1 + 2 + 3 + 4 = 10.

Output:

```
Enter the no. of vertices:
5

Enter the cost adjacency matrix:
0 1 7 10 5
1 0 3 0 0
7 3 0 4 0
10 0 4 0 2
5 0 0 2 0
The edges of Minimum Cost Spanning Tree are
1 edge (1,2) =1
2 edge (4,5) =2
3 edge (2,3) =3
4 edge (3,4) =4

Minimum cost = 10
```