G. H. RAISONI COLLEGE OF ENGG., NAGPUR
(An Autonomous Institute under UGC Act 1956)
Department of Computer Science & Engg.

Date: 08/08/2020

Practical Subject: Design and Analysis of Algorithms
Session: 2020-21

Student Details:

Roll Number	58
Name	Shivam Tawari
Semester	3
Section	A
Branch	Artificial Intelligence

Practical Details: Practical Number- 6

Practical Aim	To implement and analyze time complexity of Minimum Cost spanning Tree Algorithm.
Theory & Algorithm	**Theory:** ***Minimum Cost Spanning Tree:*** The cost of the spanning tree is the sum of the weights of all the edges in the tree. There can be many spanning trees. Minimum spanning tree is the spanning tree where the cost is minimum among all the spanning trees. There also can be many minimum spanning trees. Minimum spanning tree has direct application in the design of networks. It is used in algorithms approximating the travelling salesman problem, multi-terminal minimum cut problem and minimum-cost weighted perfect matching. ***Kruskal's Algorithm:*** Kruskal's algorithm to find the minimum cost spanning tree uses the greedy approach. This algorithm treats the graph as a forest and every node it has as an individual tree. A tree connects to

	another only and only if, it has the least cost among all available options and does not violate MST properties. **Algorithm:** Step 1: START Step 2: Input Graph Step 3: Sort edge list in ascending order Step 4: Select edge with least weight i.e., first in edge list Step 5: Check if it forms a cycle with spanning tree formed so far. If cycle is not formed, include this edge else discard. Step 6: Remove edge from edge list Step 7: Go to Step 4 until there are V-1 edges Step 8: STOP
Complexity	O (E log E) **or** O (E log V)

Program	main.cpp Run
	1 #include "iostream"
	2 using namespace std;
	3
	4 class Edge
	5 {
	6 public:
	7 int src, dest, weight;
	8 };
	9 class Graph
	10 {
	11 public:
	12 int V, E;
	13 Edge* edge;
	14 };
	15 Graph* createGraph(int V, int E)
	16 {
	17 Graph* graph = new Graph;
	18 graph->V = V;
	19 graph->E = E;
	20
	21 graph->edge = new Edge[E];
	22
	23 return graph;

```
}
class subset
{
    public:
    int parent;
    int rank;
};
int find(subset subsets[], int i)
{
    if (subsets[i].parent != i)
        subsets[i].parent = find(subsets, subsets[i].parent);

    return subsets[i].parent;
}
void Union(subset subsets[], int x, int y)
{
    int xroot = find(subsets, x);
    int yroot = find(subsets, y);
    if (subsets[xroot].rank < subsets[yroot].rank)
        subsets[xroot].parent = yroot;
    else if (subsets[xroot].rank > subsets[yroot].rank)
        subsets[yroot].parent = xroot;
    else
    {
        subsets[yroot].parent = xroot;
        subsets[xroot].rank++;
    }
}
```

```
52  int myComp(const void* a, const void* b)
53  {
54      Edge* a1 = (Edge*)a;
55      Edge* b1 = (Edge*)b;
56      return a1->weight > b1->weight;
57  }
58  void KruskalMST(Graph* graph)
59  {
60      int V = graph->V;
61      Edge result[V];
62      int e = 0;
63      int i = 0;
64      qsort(graph->edge, graph->E, sizeof(graph->edge[0]), myComp);
65      subset *subsets = new subset[( V * sizeof(subset) )];
66      for (int v = 0; v < V; ++v)
67      {
68          subsets[v].parent = v;
69          subsets[v].rank = 0;
70      }
71      while (e < V - 1 && i < graph->E)
72      {
73          Edge next_edge = graph->edge[i++];
74
75          int x = find(subsets, next_edge.src);
76          int y = find(subsets, next_edge.dest);
77          if (x != y)
78          {
79              result[e++] = next_edge;
```

```
            Union(subsets, x, y);
        }
    }
    cout<<"\n Following are the edges in the constructed MST\n";
    for (i = 0; i < e; ++i)
        cout<<result[i].src<<" -- "<<result[i].dest
            <<" == "<<result[i].weight<<endl;
    return;
}

int main()
{
    int V, E;
    int src, dest, weight;

    cout << "\n Name: Shivam Tawari";
    cout << "\nSection: A";
    cout << "\nRoll Number: 58";

    cout << "\nEnter Vertices V and Edges E: ";
    cin >> V >> E;

    Graph* graph = createGraph(V, E);

    for(int i=0; i<E; i++)  {
        cout << " Enter source, destination and weight for edge "
             << i+1 << ": ";
        cin >> src >> dest >> weight;
        graph->edge[i].src = src;
        graph->edge[i].dest = dest;
        graph->edge[i].weight = weight;
    }

    KruskalMST(graph);

    return 0;
}
```

Output	

Output Clear

```
g++ -o /tmp/6m8VY0duxJ.o /tmp/6m8VY0duxJ.cpp
/tmp/6m8VY0duxJ.o
Name: Shivam Tawari
Section: A
Roll Number: 58
Enter Vertices V and Edges E: 5
6
Enter source, destination and weight for edge 1: 0
1
10
Enter source, destination and weight for edge 2: 2
2
3
Enter source, destination and weight for edge 3: 5
2
0
Enter source, destination and weight for edge 4: 2
6
1
Enter source, destination and weight for edge 5: 0
2
3
Enter source, destination and weight for edge 6: 5
3
1
Following are the edges in the constructed MST
5 -- 2 == 0
5 -- 3 == 1
0 -- 1 == 10
```