# Experiment No. 2.3

**Student Name: Jesu Yuvraj** **UID: 22MCI10074**
**Branch: MCA** **Section/Group: MAM-1/ Grp B**
**Semester: I** **Date of Performance: 17th Nov 22**
**Subject Name: DAA LAB** **Subject Code: 22CAP-636**

## 1. Aim/Overview of the practical:

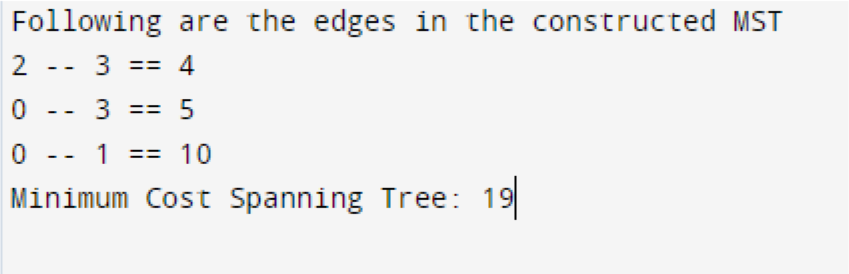
Find Minimum Cost Spanning Tree of a given undirected graph using Kruskal's algorithm.

## 2. Code for experiment/practical:

3.
```
# include <bits/stdc++.h>
using
namespace
std;


class DSU {
int * parent;
int * rank;

public:
    DSU(int


n)
{
    parent = new
int[n];
rank = new
int[n];

for (int i = 0; i < n; i++)
{
    parent[i] = -1;
rank[i] = 1;
}
}
int
find(int
i)
{
if (parent[i] == -1)
return i;

return parent[i] = find(parent[i]);
}
void
```

```
unite(int
x, int
y)
{
int
s1 = find(x);
int
s2 = find(y);

if (s1 != s2) {
if (rank[s1] < rank[s2]) {
parent[s1] = s2;
rank[s2] += rank[s1];
}
else {
parent[s2] = s1;
rank[s1] += rank[s2];
}
}
}
};

class Graph {
vector < vector < int > > edgelist;
int V;

public:
    Graph(int


V) {this->V = V;}

void
addEdge(int
x, int
y, int
w)
{
    edgelist.push_back({w, x, y});
}

void
kruskals_mst()
{
    sort(edgelist.begin(), edgelist.end());
DSU
s(V);
int
ans = 0;
cout << "Following are the edges in the " \
        "constructed MST" \
<< endl;
for (auto edge: edgelist)
{
    int
w = edge[0];
int
x = edge[1];
int
y = edge[2];
if (s.find(x) != s.find(y))
{
    s.unite(x, y);
ans += w;
cout << x << " -- " << y << " == " << w \
<< endl;
}
}

cout << "Minimum Cost Spanning Tree: " << ans;
}
};

int
main()
{
    Graph
g(4);
g.addEdge(0, 1, 10);
g.addEdge(1, 3, 15);
g.addEdge(2, 3, 4);
g.addEdge(2, 0, 6);
g.addEdge(0, 3, 5);
g.kruskals_mst();
return 0;
}
```

## 4. Output:



```
Following are the edges in the constructed MST
2 -- 3 == 4
0 -- 3 == 5
0 -- 1 == 10
Minimum Cost Spanning Tree: 19
```

**************************************** THE END *************************************************