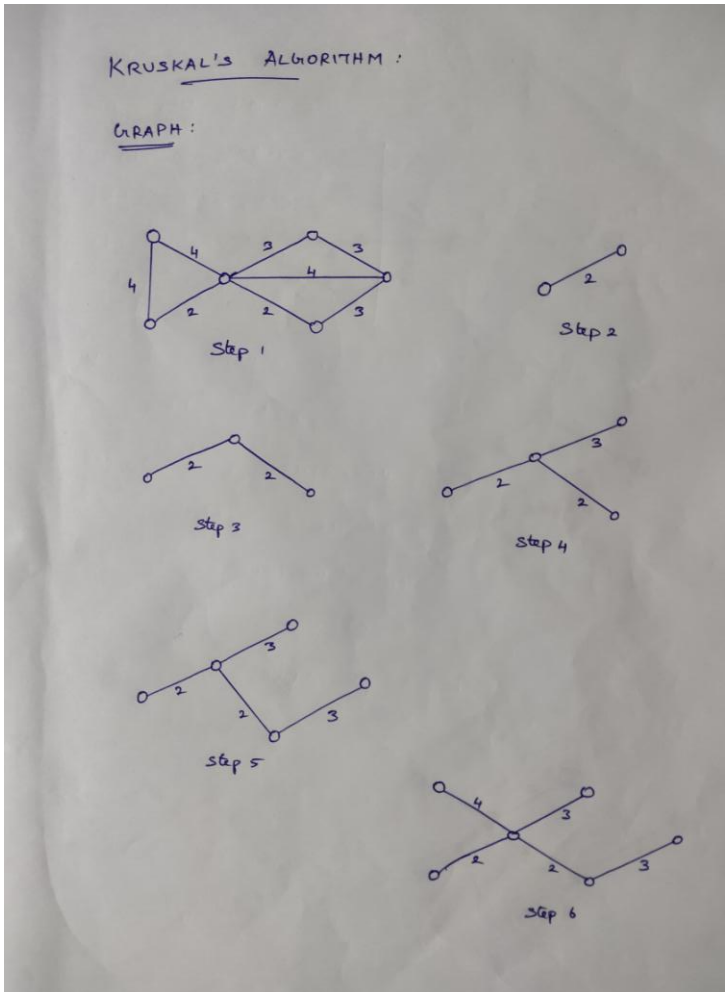
NAME : JAGADEESH R
REGNO : 2021506314
ADS LAB 11a : Kruskal's Algorithm Graph

SOURCE CODE :

```
#include <bits/stdc++.h>
#include <iostream>
using namespace std;
#define edge pair<int, int>
class Graph {
 private:
 vector<pair<int, edge> > G;
 vector<pair<int, edge> > T;
 int *parent;
 int V;
 public:
 Graph(int V);
 void AddWeightedEdge(int u, int v, int w);
 int find_set(int i);
 void union_set(int u, int v);
 void kruskal();
 void print();
};
Graph::Graph(int V) {
 parent = new int[V];
 for (int i = 0; i < V; i++)
   parent[i] = i;
 G.clear();
 T.clear();
}
void Graph::AddWeightedEdge(int u, int v, int w) {
 G.push_back(make_pair(w, edge(u, v)));
}
int Graph::find_set(int i) {
 if (i == parent[i])
   return i;
 else
   return find_set(parent[i]);
}
void Graph::union_set(int u, int v) {
 parent[u] = parent[v];
}
void Graph::kruskal() {
 int i, uRep, vRep;
 sort(G.begin(), G.end());
 for (i = 0; i < G.size(); i++) {
```

```
    uRep = find_set(G[i].second.first);
    vRep = find_set(G[i].second.second);
    if (uRep != vRep) {
      T.push_back(G[i]);
      union_set(uRep, vRep);
    }
  }
}
void Graph::print() {
    int weight = 0;
  cout << "Kruskal's Algorithm " << endl;
  for (int i = 0; i < T.size(); i++) {
    cout << "Edge " << T[i].second.first << " - " << T[i].second.second << " and its Weight " <<
T[i].first << endl;
    weight = weight + T[i].first;
  }
  cout << "Total weight of the graph is : " << weight << endl;
}
int main() {
  Graph g(6);
  g.AddWeightedEdge(0, 1, 4);
  g.AddWeightedEdge(0, 2, 4);
  g.AddWeightedEdge(1, 2, 2);
  g.AddWeightedEdge(1, 0, 4);
  g.AddWeightedEdge(2, 0, 4);
  g.AddWeightedEdge(2, 1, 2);
  g.AddWeightedEdge(2, 3, 3);
  g.AddWeightedEdge(2, 5, 2);
  g.AddWeightedEdge(2, 4, 4);
  g.AddWeightedEdge(3, 2, 3);
  g.AddWeightedEdge(3, 4, 3);
  g.AddWeightedEdge(4, 2, 4);
  g.AddWeightedEdge(4, 3, 3);
  g.AddWeightedEdge(5, 2, 2);
  g.AddWeightedEdge(5, 4, 3);
  g.kruskal();
  g.print();
  return 0;
}
```

GRAPH :



OUTPUT :

```
Output

/tmp/p4AszD9rJx.o
Kruskal's Algorithm
Edge 1 - 2 and its Weight 2
Edge 2 - 5 and its Weight 2
Edge 2 - 3 and its Weight 3
Edge 3 - 4 and its Weight 3
Edge 0 - 1 and its Weight 4
Total weight of the graph is : 14
```