Kruskal Algorithm Coding with c++

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

// Structure to represent an edge
struct Edge {
    int u, v, weight;
};

// Comparator to sort edges by weight
bool compare(Edge a, Edge b) {
    return a.weight < b.weight;
}

// Disjoint Set (Union-Find) with path compression and union by rank
class DisjointSet {
    vector<int> parent, rank;

public:
    // Initialize the DSU
    DisjointSet(int n) {
        parent.resize(n);
        rank.resize(n, 0);
        for (int i = 0; i < n; i++)
            parent[i] = i;
    }

    // Find the representative of a set with path compression
    int find(int u) {
        if (parent[u] != u)
            parent[u] = find(parent[u]);
        return parent[u];
    }

    // Union two sets by rank
```

```
    void unionByRank(int u, int v) {
        int pu = find(u);
        int pv = find(v);
        if (pu == pv) return;  // Already in the same set

        if (rank[pu] < rank[pv]) {
            parent[pu] = pv;
        } else if (rank[pu] > rank[pv]) {
            parent[pv] = pu;
        } else {
            parent[pu] = pv;
            rank[pv]++;
        }
    }
};

// Function to find MST using Kruskal's Algorithm
int kruskalMST(int V, vector<Edge>& edges) {
    // Step 1: Sort edges by weight
    sort(edges.begin(), edges.end(), compare);

    // Step 2: Initialize Disjoint Set
    DisjointSet ds(V);
    int mstWeight = 0;

    // Step 3: Pick edges one by one
    for (auto edge : edges) {
        if (ds.find(edge.u) != ds.find(edge.v)) {
            cout << "Selected Edge: " << edge.u << " - " << edge.v << " (" << edge.weight << ")\n";
            mstWeight += edge.weight;
            ds.unionByRank(edge.u, edge.v);
        }
    }

    return mstWeight;
}

// Driver code
int main() {
    int V = 4;  // Number of vertices
```

```
    vector<Edge> edges = {
        {0, 1, 10},
        {0, 2, 6},
        {0, 3, 5},
        {2, 3, 4}
    };

    int totalWeight = kruskalMST(V, edges);
    cout << "Total weight of MST: " << totalWeight << endl;

    return 0;
}
```

Output

```
Selected Edge: 2 - 3 (4)
Selected Edge: 0 - 3 (5)
Selected Edge: 0 - 1 (10)
Total weight of MST: 19
```