NAME – ANIKET SHARMA

REG. NO – RA1911003030436

BRANCH- CSE-I

SUBJECT- Design and Analysis of Algorithms

SUBJECT CODE-18CSC204J

SUBMITTED TO- Mr. Himansu Sekhar Pattanayak

Implement Kruskal's and Prim's minimum spanning tree algorithm.

Kruskal's minimum spanning tree algorithm: -

main.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;
class Edge {
public:
    int src, dest, weight;
};
class Graph {
public:
    int V, E;
    Edge* edge;
};
Graph* createGraph(int V, int E)
{
    Graph* graph = new Graph;
    graph->V = V;
    graph->E = E;

    graph->edge = new Edge[E];

    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++;
    }
}
```

```
int myComp(const void* a, const void* b)
{
    Edge* a1 = (Edge*)a;
    Edge* b1 = (Edge*)b;
    return a1->weight > b1->weight;
}
void KruskalMST(Graph* graph)
{
    int V = graph->V;
    Edge result[V];
    int e = 0;
    int i = 0;
    qsort(graph->edge, graph->E, sizeof(graph->edge[0]),
        myComp);
    subset* subsets = new subset[(V * sizeof(subset))];
    for (int v = 0; v < V; ++v)
    {
        subsets[v].parent = v;
        subsets[v].rank = 0;
    }
    while (e < V - 1 && i < graph->E)
    {
        Edge next_edge = graph->edge[i++];
        int x = find(subsets, next_edge.src);
        int y = find(subsets, next_edge.dest);
        if (x != y) {
```

```
                result[e++] = next_edge;
                Union(subsets, x, y);
            }
    }
    cout << "Following are the edges in the constructed "
            "MST\n";
    int minimumCost = 0;
    for (i = 0; i < e; ++i)
    {
        cout << result[i].src << " -- " << result[i].dest
            << " == " << result[i].weight << endl;
        minimumCost = minimumCost + result[i].weight;
    }
    cout << "Minimum Cost Spanning Tree: " << minimumCost
        << endl;
}
int main()
{
            4
    int V = 4;
    int E = 5;
    Graph* graph = createGraph(V, E);
    graph->edge[0].src = 0;
    graph->edge[0].dest = 1;
```

```
    graph->edge[0].weight = 10;
    graph->edge[1].src = 0;
    graph->edge[1].dest = 2;
    graph->edge[1].weight = 6;
    graph->edge[2].src = 0;
    graph->edge[2].dest = 3;
    graph->edge[2].weight = 5;
    graph->edge[3].src = 1;
    graph->edge[3].dest = 3;
    graph->edge[3].weight = 15;
    graph->edge[4].src = 2;
    graph->edge[4].dest = 3;
    graph->edge[4].weight = 4;
    KruskalMST(graph);
    return 0;
}
```

OUTPUT: -

input

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


...Program finished with exit code 0
Press ENTER to exit console.
```

Prim's minimum spanning tree algorithm: -

main.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;
#define V 5
int minKey(int key[], bool mstSet[])
{
    int min = INT_MAX, min_index;
    for (int v = 0; v < V; v++)
        if (mstSet[v] == false && key[v] < min)
            min = key[v], min_index = v;
    return min_index;
}
void printMST(int parent[], int graph[V][V])
{
    cout<<"Edge \tWeight\n";
    for (int i = 1; i < V; i++)
        cout<<parent[i]<<" - "<<i<<" \t"<<graph[i][parent[i]]<<" \n";
}
void primMST(int graph[V][V])
{
    int parent[V];
    int key[V];
    bool mstSet[V];
    for (int i = 0; i < V; i++)
        key[i] = INT_MAX, mstSet[i] = false;
    key[0] = 0;
```

```
    parent[0] = -1;
    for (int count = 0; count < V - 1; count++)
    {
        int u = minKey(key, mstSet);
        mstSet[u] = true;
        for (int v = 0; v < V; v++)
            if (graph[u][v] && mstSet[v] == false && graph[u][v] < key[v])
                parent[v] = u, key[v] = graph[u][v];
    }
    printMST(parent, graph);
}
int main()
{
    int graph[V][V] = { { 0, 2, 0, 6, 0 },
                        { 2, 0, 3, 8, 5 },
                        { 0, 3, 0, 0, 7 },
                        { 6, 8, 0, 0, 9 },
                        { 0, 5, 7, 9, 0 } };

    primMST(graph);
    return 0;
}
```

OUTPUT: -