Experiment No-7

Vilakshan

Batch – B

Problem statement: Represent a graph using an adjacency list or array

and generate a minimum spanning tree using Kruskal’s

algorithm.

Code:

#include <bits/stdc++.h>

using *namespace* std;

typedef pair<*int*, *int*> iPair;

*struct* Graph

{

*int* V, E;

    vector<pair<*int*, iPair>> edges;

    Graph(*int* *V*, *int* *E*)

    {

        this->V = *V*;

        this->E = *E*;

    }

*void* addEdge(*int* *u*, *int* *v*, *int* *w*)

    {

        edges.push\_back({w, {u, v}});

    }

*int* kruskalMST();

};

*struct* DisjointSets

{

*int* \*parent, \*rnk;

*int* n;

    DisjointSets(*int* *n*)

    {

        this->n = *n*;

        parent = new *int*[*n* + 1];

        rnk = new *int*[*n* + 1];

        for (*int* i = 0; i <= *n*; i++)

        {

            rnk[i] = 0;

            parent[i] = i;

        }

    }

*int* find(*int* *u*)

    {

        if (*u* != parent[*u*])

            parent[*u*] = find(parent[*u*]);

        return parent[*u*];

    }

*void* merge(*int* *x*, *int* *y*)

    {

*x* = find(*x*), *y* = find(*y*);

        if (rnk[*x*] > rnk[*y*])

            parent[*y*] = *x*;

        else // If rnk[x] <= rnk[y]

            parent[*x*] = *y*;

        if (rnk[*x*] == rnk[*y*])

            rnk[*y*]++;

    }

};

*int* Graph::kruskalMST()

{

*int* mst\_wt = 0; // Initialize result

    // Sort edges in increasing order on basis of cost

    sort(edges.begin(), edges.end());

    // Create disjoint sets

    DisjointSets ds(V);

    // Iterate through all sorted edges

    vector<pair<*int*, iPair>>::iterator it;

    for (it = edges.begin(); it != edges.end(); it++)

    {

*int* u = it->second.first;

*int* v = it->second.second;

*int* set\_u = ds.find(u);

*int* set\_v = ds.find(v);

        if (set\_u != set\_v)

        {

            // Current edge will be in the MST

            // so print it

            cout << u << " - " << v << endl;

            // Update MST weight

            mst\_wt += it->first;

            // Merge two sets

            ds.merge(set\_u, set\_v);

        }

    }

    return mst\_wt;

}

*int* main()

{

*int* V = 9, E = 14;

    Graph g(V, E);

    // making above shown graph

    g.addEdge(0, 1, 4);

    g.addEdge(0, 7, 8);

    g.addEdge(1, 2, 8);

    g.addEdge(1, 7, 11);

    g.addEdge(2, 3, 7);

    g.addEdge(2, 8, 2);

    g.addEdge(2, 5, 4);

    g.addEdge(3, 4, 9);

    g.addEdge(3, 5, 14);

    g.addEdge(4, 5, 10);

    g.addEdge(5, 6, 2);

    g.addEdge(6, 7, 1);

    g.addEdge(6, 8, 6);

    g.addEdge(7, 8, 7);

    cout << "Edges of MST are \n";

*int* mst\_wt = g.kruskalMST();

    cout << "\nWeight of MST is " << mst\_wt;

    return 0;

}

OUTPUT:

