LAB TASK – 9

Q1)MST implementation using Kruskal’s Algorithm

**Code :**

#include <stdio.h>

#define V 5

#define E 7

typedef struct {

    int start, end, weight;

} Edge;

typedef struct {

    int parent;

    int rank;

} Subset;

int findSet(Subset subsets[], int i) {

    if (subsets[i].parent != i)

        subsets[i].parent = findSet(subsets, subsets[i].parent);

    return subsets[i].parent;

}

void unionSets(Subset subsets[], int x, int y) {

    int xroot = findSet(subsets, x);

    int yroot = findSet(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++;

    }

}

void kruskalAlgorithm(Edge edges[]) {

    Edge mst[V];

    int e = 0, i = 0;

    // bubble sort edges by weight

    for (int a = 0; a < E - 1; a++)

        for (int b = 0; b < E - a - 1; b++)

            if (edges[b].weight > edges[b + 1].weight) {

                Edge temp = edges[b];

                edges[b] = edges[b + 1];

                edges[b + 1] = temp;

            }

    Subset subsets[V];

    for (int v = 0; v < V; v++) {

        subsets[v].parent = v;

        subsets[v].rank = 0;

    }

    while (e < V - 1 && i < E) {

        Edge nextEdge = edges[i++];

        int x = findSet(subsets, nextEdge.start);

        int y = findSet(subsets, nextEdge.end);

        if (x != y) {

            mst[e++] = nextEdge;

            unionSets(subsets, x, y);

        }

    }

    printf("Edge \tWeight\n");

    for (i = 0; i < e; i++)

        printf("%d - %d \t%d\n", mst[i].start, mst[i].end, mst[i].weight);

}

int main() {

    Edge edges[E] = {

        {0, 1, 4},

        {0, 3, 8},

        {1, 2, 2},

        {1, 3, 6},

        {1, 4, 5},

        {2, 4, 7},

        {3, 4, 9}

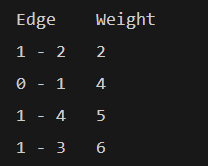
    };

    kruskalAlgorithm(edges);

    return 0;

}

**Output :**



Q2) MST implementation using Prim’s Algorithm

**Code :**

#include <stdio.h>

#include <limits.h>

#define V 5

int findMinKey(int cost[], int visited[]) {

    int min = INT\_MAX, minIndex;

    for (int i = 0; i < V; i++)

        if (visited[i] == 0 && cost[i] < min)

            min = cost[i], minIndex = i;

    return minIndex;

}

void primAlgorithm(int graph[V][V]) {

    int parent[V];

    int cost[V];

    int visited[V];

    for (int i = 0; i < V; i++)

        cost[i] = INT\_MAX, visited[i] = 0;

    cost[0] = 0;

    parent[0] = -1;

    for (int i = 0; i < V - 1; i++) {

        int u = findMinKey(cost, visited);

        visited[u] = 1;

        for (int v = 0; v < V; v++)

            if (graph[u][v] && visited[v] == 0 && graph[u][v] < cost[v])

                parent[v] = u, cost[v] = graph[u][v];

    }

    printf("Edge \tWeight\n");

    for (int i = 1; i < V; i++)

        printf("%d - %d \t%d\n", parent[i], i, graph[i][parent[i]]);

}

int main() {

    int graph[V][V] = {

        {0, 4, 0, 8, 0},

        {4, 0, 2, 6, 5},

        {0, 2, 0, 0, 7},

        {8, 6, 0, 0, 9},

        {0, 5, 7, 9, 0}

    };

    primAlgorithm(graph);

    return 0;

}

**Output :**

