22AIE203 – Data Structures and Algorithm - 2

LAB EXP 3

Kruskal's algorithm

NAME :GURUPRASATH M R

ROLLNO:CH.EN.U4.AIE22015

CLASS:2 YEAR 3 SEM

Code

#include <stdio.h>

#define I 32767  // Infinity

#define V 7  // # of vertices in Graph

#define E 9  // # of edges in Graph

void PrintMCST(int T[][V-1], int A[][E]){

    printf("\nMinimum Cost Spanning Tree Edges\n\n");

    for (int i = 0; i < V-1; i++){

        printf("[%d]-----[%d]\n", T[0][i], T[1][i]);

    }

    printf("\n");

}

// Set operations: Union and Find

void Union(int u, int v, int s[]){

    if (s[u] < s[v]){

        s[u] += s[v];

        s[v] = u;

    } else {

        s[v] += s[u];

        s[u] = v;

    }

}

int Find(int u, int s[]){

    int x = u;

    int v = 0;

    while (s[x] > 0){

        x = s[x];

    }

    while (u != x){

        v = s[u];

        s[u] = x;

        u = v;

    }

    return x;

}

void KruskalsMCST(int A[3][9]){

    int T[2][V-1];  // Solution array

    int track[E] = {0};  // Track edges that are included in solution

    int set[V+1] = {-1, -1, -1, -1, -1, -1, -1, -1};  // Array for finding cycle

    int i = 0;

    while (i < V-1){

        int min = I;

        int u = 0;

        int v = 0;

        int k = 0;

        // Find a minimum cost edge

        for (int j = 0; j < E; j++){

            if (track[j] == 0 && A[2][j] < min){

                min = A[2][j];

                u = A[0][j];

                v = A[1][j];

                k = j;

            }

        }

        // Check if the selected min cost edge (u, v) forms a cycle or not

        if (Find(u, set) != Find(v, set)){

            T[0][i] = u;

            T[1][i] = v;

            // Perform union

            Union(Find(u, set), Find(v, set), set);

            i++;

        }

        track[k] = 1;

    }

    PrintMCST(T, A);

}

int main() {

    int edges[3][9] = {{ 1, 1,  2,  2, 3,  4,  4,  5,  5},

                       { 2, 6,  3,  7, 4,  5,  7,  6,  7},

                       {25, 5, 12, 10, 8, 16, 14, 20, 18}};

    KruskalsMCST(edges);

    return 0;

}

Output



