Student Name:- Chaudhary Hamdan

Student Roll No.:- 1905387

Algorithm Lab. Class Assignment-13

CSE Group 1

Date: - 29th October 2021

1. Write a program to find Minimum Cost Spanning Tree of a given undirected graph using Kruskal's algorithm.

Program

```
// Author: Chaudhary Hamdan
// Generated at: Fri Oct 29 12:42:15 2021

#include <stdio.h>
#include <time.h>
#include <stdlib.h>

#define sf(x)          scanf("%d", &x)
#define pf             printf
#define pfs(x)         printf("%d ", x)
#define pfn(x)         printf("%d\n", x)
#define pfc(x)         printf("%d, ", x)
#define FI(i,x,y,inc)   for(int i = x; i < y; i += inc)
#define F(i,x,y)       FI(i, x, y, 1)
#define F0(i,n)        FI(i, 0, n, 1)
#define RF(i,x,y)       for(int i = x; i >= y; i--)
#define pfarr(i,a,n)    for(int i = 0; i < n-1; i++) pfs(a[i]); pfn(a[n-1]);
```

```
void i_o_from_file() {

#ifndef ONLINE_JUDGE
        freopen("C:\\Users\\KIIT\\input", "r", stdin);
        freopen("C:\\Users\\KIIT\\output", "w", stdout);
#endif
}
/* ********************************************** */

struct Edge {
        int src, dest, weight;
};

struct Graph {
        int V, E;
        struct Edge* edge;
};
struct Graph* createGraph(int V, int E)
{
        struct Graph* graph = (struct Graph*)(malloc(sizeof(struct Graph)));
        graph->V = V;
        graph->E = E;

        graph->edge = (struct Edge*)malloc(sizeof(struct Edge) * E);

        return graph;
}
```

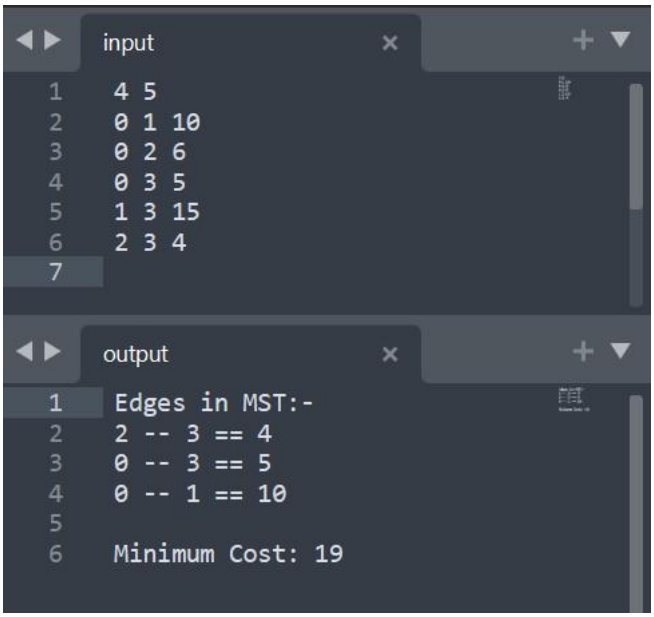
```
struct subset {
	int parent;
	int rank;
};

int find(struct subset subsets[], int i)
{
	if (subsets[i].parent != i)
		subsets[i].parent
		  = find(subsets, subsets[i].parent);

	return subsets[i].parent;
}

void Union(struct 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)
{
	struct Edge* a1 = (struct Edge*)a;
	struct Edge* b1 = (struct Edge*)b;
	return a1->weight > b1->weight;
}

void KruskalMST(struct Graph* graph)
{
	int V = graph->V;
	struct Edge
		result[V];
	int e = 0;
	int i = 0;

	qsort(graph->edge, graph->E, sizeof(graph->edge[0]), myComp);

	struct subset* subsets = (struct subset*)malloc(V * sizeof(struct subset));

	for (int v = 0; v < V; ++v) {
		subsets[v].parent = v;
		subsets[v].rank = 0;
	}

	while (e < V - 1 && i < graph->E) {
		struct 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);
            }
        }

        printf("Edges in MST:-\n");
        int minimumCost = 0;
        for (i = 0; i < e; ++i)
        {
            printf("%d -- %d == %d\n", result[i].src, result[i].dest,
result[i].weight);
            minimumCost += result[i].weight;
        }
        printf("\nMinimum Cost: %d", minimumCost);
        return;
}

int main() {

        i_o_from_file();

        /* ************************************************ */

        int V, E;

        sf(V);
```

```
    sf(E);

    struct Graph* graph = createGraph(V, E);

    F0(i, E) {

        sf(graph->edge[i].src);
        sf(graph->edge[i].dest);
        sf(graph->edge[i].weight);
    }
    KruskalMST(graph);

    return 0;
}
```

Output



2. Write a program to find Minimum Cost Spanning Tree of a given undirected graph using Prims algorithm.

Program

```
// Author: Chaudhary Hamdan
// Generated at: Fri Oct 29 12:42:30 2021

#include <stdio.h>
#include <time.h>
#include <limits.h>
#include <stdbool.h>

#define sf(x)          scanf("%d", &x)
#define pf             printf
#define pfs(x)         printf("%d ", x)
#define pfn(x)         printf("%d\n", x)
#define pfc(x)         printf("%d, ", x)
#define FI(i,x,y,inc)   for(int i = x; i < y; i += inc)
#define F(i,x,y)       FI(i, x, y, 1)
#define F0(i,n)        FI(i, 0, n, 1)
#define RF(i,x,y)       for(int i = x; i >= y; i--)
#define pfarr(i,a,n)    for(int i = 0; i < n-1; i++) pfs(a[i]); pfn(a[n-1]);

void i_o_from_file() {
#ifndef ONLINE_JUDGE
        freopen("C:\\Users\\KIIT\\input", "r", stdin);
        freopen("C:\\Users\\KIIT\\output", "w", stdout);
#endif
}
```

```
/*
**********************************************************************
*/
int V;

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;
}

int printMST(int parent[], int graph[V][V])
{
	printf("Edge \tWeight\n");
	F(i, 1, V) {
		printf("%d - %d \t%d \n", parent[i], i, graph[i][parent[i]]);
	}
}

void primsMST(int graph[V][V])
{
	int parent[V];
	int key[V];
```

```
        bool mstSet[V];

        F0(i, V) {
                key[i] = INT_MAX;
                mstSet[i] = false;
        }

        key[0] = 0;
        parent[0] = -1;

        F0(cnt, V - 1) {

                int u = minKey(key, mstSet);

                mstSet[u] = true;

                F0(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() {

        i_o_from_file();
```

```
        /* ********************************************** */

        sf(V);

        int graph[V][V];

        F0(i, V) {
                F0(j, V) {
                        sf(graph[i][j]);
                }
        }

        primsMST(graph);

        return 0;
}
```

Output