**Assignment 6**

**Name :-** Dhruv Thakkar

Branch :- CS

Division :- A

Roll no. :- 69

GR Number :- 12111512

**Problem Statement**

Write a Program to implement Prims algorithm to find

minimum spanning tree of a user defined graph. Use Adjacency

List to represent a graph.

**Solution :**

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <limits.h>

struct ListNode{

    int data;

    struct ListNode \*next;

    int weight;

};

void insertEdge(int a, int b,int weight, struct ListNode \*adjList[]){

    if(adjList[a]==NULL){

        adjList[a] = (struct ListNode\*)malloc(sizeof(struct ListNode));

        adjList[a]->data = b;

        adjList[a]->weight = weight;

        adjList[a]->next=NULL;

    }

    else if(adjList[a] != NULL){

        struct ListNode\* temp = (struct ListNode\*)malloc(sizeof(struct ListNode));

        temp->data = b;

        temp->weight = weight;

        temp->next = adjList[a];

        adjList[a]=temp;

    }

}

void primMST(struct ListNode \*adjList[], int vertex, int start){

    int parent[vertex];

    int key[vertex];

    bool mstSet[vertex];

    int totalWeight = 0;

    for(int i = 0; i < vertex; i++){

        key[i] = INT\_MAX;

        mstSet[i] = false;

    }

    key[0] = start;

    parent[0] = -1;

    for(int i = 0; i < vertex-1; i++){

        int u, minKey = INT\_MAX;

        for(int j = 0; j < vertex; j++){

            if(!mstSet[j] && key[j] < minKey){

                minKey = key[j];

                u = j;

            }

        }

        mstSet[u] = true;

        struct ListNode\* temp = adjList[u];

        while(temp != NULL){

            int v = temp->data;

            int weight = temp->weight;

            if(!mstSet[v] && weight < key[v]){

                parent[v] = u;

                key[v] = weight;

            }

            temp = temp->next;

        }

        totalWeight += key[u];

    }

    printf("Minimum Cost Spanning Tree:\n");

    for(int i = 1; i < vertex; i++){

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

    }

    printf("Total cost of MST: %d\n", totalWeight);

}

void printAdjList(struct ListNode \*adjList[],int vertex){

    printf("The Adjacency List You Gave ! \n");

    for(int i =0 ; i<vertex; i++){

        if(adjList[i]==NULL){

            printf("%d -> NULL\n",i);

        }

        else{

            struct ListNode\* temp = adjList[i];

            printf("%d : --(w:%d)-%d",i,adjList[i]->weight,adjList[i]->data);

            while(temp->next!=NULL){

                temp = temp->next;

                printf("--(w:%d)->%d ",temp->weight,temp->data);

            }

            printf("\n");

        }

    }

}

void getAdjacencyListInput(struct ListNode \*adjList[],int vertex){

    for(int i=0; i<vertex; i++){

        adjList[i] = NULL;

    }

    printf("Enter the number edges\n");

    int n;

    scanf("%d",&n);

    for(int i=0 ; i<n ; i++){

        int a,b,w;

        printf("Enter the edge vertices and its weight\n");

        scanf("%d%d%d",&a,&b,&w);

        insertEdge(a,b,w,adjList);

        insertEdge(b,a,w,adjList);

    }

    // for(int i=0 ; i<vertex; i++){

    //     printf("Enter the number of vertices attached with vertex %d\n",i);

    //     int vertexNo;

    //     scanf("%d",&vertexNo);

    //     for(int j =0 ; j<vertexNo; j++){

    //         printf("Enter the vertex attached with vertex %d and its weight\n",i);

    //         int tempVertex,weight;

    //         scanf("%d",&tempVertex);

    //         scanf("%d",&weight);

    //         insertEdge(i,tempVertex,weight,adjList);

    //     }

    //     printf("All vertices for vertex %d are inserted in AdjList\n\n",i);

    // }

    printf("\nAdjacency List created successfully !!!\n \n");

}

void main(){

    int vertex;

    printf("Enter number of vertex\n");

    scanf("%d",&vertex);

    struct ListNode \*adjList[vertex];

    getAdjacencyListInput(adjList,vertex);

    printAdjList(adjList,vertex);

    int start;

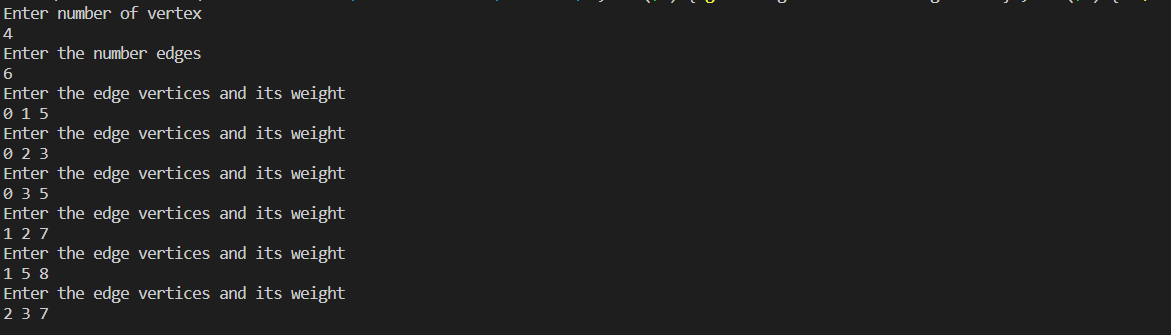
    printf("Enter the start vertex ");

    scanf("%d",&start);

    primMST(adjList,vertex,start);

}

**OUTPUT:**

****

****