|  |  |
| --- | --- |
| **Title:** | **Prim’s Algorithm** |
| **Problem Statement** | **Write a program in C to represent graph using adjacency list or matrix and generate minimum spanning tree using Prim’s algorithm.** |

**Programmer Name: Ansh Shah**

**Batch: H5**

***1.Program:***

#include <stdio.h>

#include <limits.h> // For INT\_MAX

#define MAX 100 // Maximum number of vertices

int adjMatrix[MAX][MAX]; // Adjacency matrix representation of the graph

int vertices; // Number of vertices in the graph

// Function to find the vertex with the minimum key value from the set of vertices not yet included in the MST

int minKey(int key[], int mstSet[]) {

int min = INT\_MAX;

int minIndex = -1;

int v = 0;

while (v < vertices) {

if (mstSet[v] == 0 && key[v] < min) {

min = key[v];

minIndex = v;

}

v++;

}

return minIndex;

}

// Function to implement Prim's algorithm to find the MST

void primMST() {

int parent[MAX]; // Array to store constructed MST

int key[MAX]; // Key values used to pick minimum weight edge

int mstSet[MAX]; // To represent the set of vertices included in the MST

// Initialize all keys as INFINITE

int i = 0;

while (i < vertices) {

key[i] = INT\_MAX;

mstSet[i] = 0;

parent[i] = -1;

i++;

}

// First vertex is picked first

key[0] = 0; // Make key 0 so that this vertex is picked first

parent[0] = -1; // First node is always root of MST

int count = 0; // Count of vertices in the MST

while (count < vertices - 1) {

int u = minKey(key, mstSet); // Get the minimum key vertex

mstSet[u] = 1; // Add the picked vertex to the MST

count++;

// Update key value and parent index of the adjacent vertices of the picked vertex

int v = 0;

while (v < vertices) {

if (adjMatrix[u][v] && mstSet[v] == 0 && adjMatrix[u][v] < key[v]) {

parent[v] = u;

key[v] = adjMatrix[u][v];

}

v++;

}

}

// Print the constructed MST

printf("Edge \tWeight\n");

i = 1;

while (i < vertices) {

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

i++;

}

}

// Function to add an edge to the adjacency matrix

void addEdge(int u, int v, int weight) {

adjMatrix[u][v] = weight;

adjMatrix[v][u] = weight; // Since this is an undirected graph

}

// Main function

int main() {

int edges, u, v, weight;

// Get the number of vertices from the user

printf("Enter the number of vertices: ");

scanf("%d", &vertices);

// Initialize adjacency matrix to 0

int i = 0;

while (i < vertices) {

int j = 0;

while (j < vertices) {

adjMatrix[i][j] = 0;

j++;

}

i++;

}

// Get the number of edges from the user

printf("Enter the number of edges: ");

scanf("%d", &edges);

// Input edges for the graph

i = 0;

while (i < edges) {

printf("Enter edge (u v weight): ");

scanf("%d %d %d", &u, &v, &weight);

addEdge(u, v, weight);

i++;

}

// Perform Prim's algorithm to find the MST

primMST();

return 0;

}

**Output:**

Enter the number of vertices: 3

Enter the number of edges: 3

Enter edge (u v weight): 1

2

3

Enter edge (u v weight): 4

5

6

Enter edge (u v weight): 7

8

9

Edge Weight

-1 - 1 0

-1 - 2 0