//Dijkstra’s Algo

#include <iostream>

#include <vector>

using namespace std;

const int INF = INT32\_MAX; *// Set a constant to represent infinity*

*// Adds an undirected edge between nodes u and v with weight w to the adjacency list*

void addEdge(vector<vector<pair<int, int>>> &adj, int u, int v, int w)

{

    adj[u].push\_back({v, w});

    adj[v].push\_back({u, w});

}

int currmindis(int n, vector<int> &distance, vector<bool> &Visited)

{

    int mindistance = INF, currver = -1;

    for (int i = 0; i < n; i++)

    {

        if (!Visited[i] && distance[i] < mindistance)

        {

            mindistance = distance[i];

            currver = i;

        }

    }

    return currver;

}

vector<int> Distras(vector<vector<pair<int, int>>> &adj)

{

    int n = adj.size();

    vector<int> distance(n, INF), parent(n, -1);

    vector<bool> Visited(n, false);

    int src = 0;

    distance[src] = 0;

    for (int i = 0; i < n - 1; i++)

    {

        int u = currmindis(n, distance, Visited);

        Visited[u] = true;

        for (auto &edge : adj[u])

        {

            int v = edge.first;

            int w = edge.second;

            if (!Visited[v] && w + distance[u] < distance[v])

            {

                distance[v] = w + distance[u];

                parent[v] = u;

            }

        }

    }

    return parent;

}

void display(vector<int> &parent)

{

    for (int i = 1; i < parent.size(); i++)

    {

        cout << parent[i] << " - " << i << "\n";

    }

}

int main()

{

    int n, m;

    cout << "Enter No. of vertices and edges" << endl;

    cin >> n >> m;

    vector<vector<pair<int, int>>> adj(n);

    for (int i = 0; i < m; i++)

    {

        int u, v, w;

        cout << "Enter starting -ending - weight of edge " << i + 1 << endl;

        cin >> u >> v >> w;

        addEdge(adj, u, v, w);

    }

    vector<int> parent = Distras(adj);

    display(parent);

    return 0;

}

Output:

Text

Description automatically generated