Prims Algorithm

Deva Shriyansh A

19070122049

#include <iostream>

#define infinity 1000

#define MAX 10

int adj[MAX][MAX], n, spanning[MAX][MAX];

int primsAlgo()

{

    int cost[MAX][MAX];

    int u, v, minimumDistance, distance[MAX], from[MAX];

    int visited[MAX], numEdges, minimumCost;

    // cost[][] matrix and spanning[][] creation

    for (int i = 0; i < n; i++)

        for (int j = 0; j < n; j++)

        {

            if (adj[i][j] == 0)

                cost[i][j] = infinity;

            else

                cost[i][j] = adj[i][j];

            spanning[i][j] = 0;

        }

    // visited[],distance[] and from[] initialised here

    distance[0] = 0;

    visited[0] = 1;

    for (int i = 1; i < n; i++)

    {

        distance[i] = cost[0][i];

        from[i] = 0;

        visited[i] = 0;

    }

    minimumCost = 0;  //cost of spanning tree

    numEdges = n - 1; //no. of edges to be added

    while (numEdges > 0)

    {

        //locating the vertex at minimum distance from the tree

        minimumDistance = infinity;

        for (int i = 1; i < n; i++)

            if (visited[i] == 0 && distance[i] < minimumDistance)

            {

                v = i;

                minimumDistance = distance[i];

            }

        u = from[v];

        //edge insertion in spanning tree

        spanning[u][v] = distance[v];

        spanning[v][u] = distance[v];

        numEdges--;

        visited[v] = 1;

        // distance[] array updation

        for (int i = 1; i < n; i++)

            if (visited[i] == 0 && cost[i][v] < distance[i])

            {

                distance[i] = cost[i][v];

                from[i] = v;

            }

        minimumCost = minimumCost + cost[u][v];

    }

    return (minimumCost);

}

int main()

{

    int totalCost;

    std::cout << "Enter no. of vertices:";

    std::cin >> n;

    std::cout << "\nEnter the adjacency matrix:\n";

    for (int i = 0; i < n; i++)

        for (int j = 0; j < n; j++)

            std::cin >> adj[i][j];

    totalCost = primsAlgo();

    std::cout << "\nspanning tree matrix:\n";

    for (int i = 0; i < n; i++)

    {

        std::cout << "\n";

        for (int j = 0; j < n; j++)

            std::cout << spanning[i][j] << "\t";

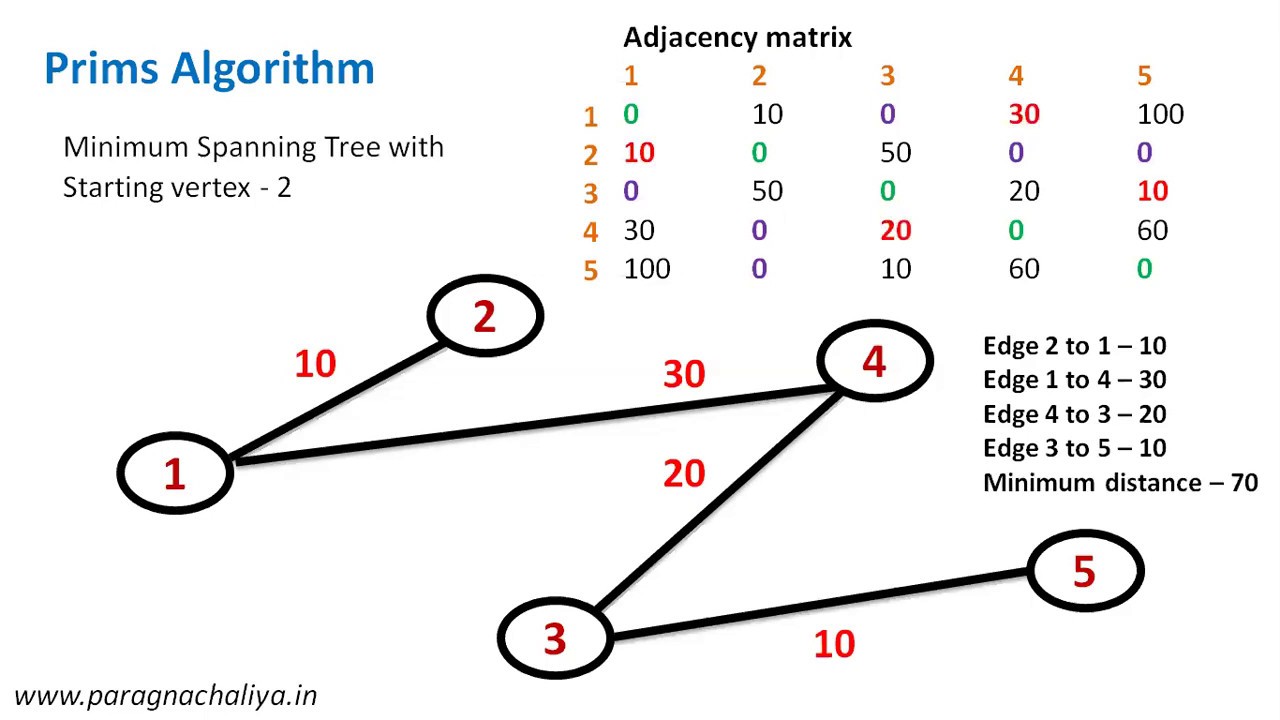
    }

    std::cout << "\n\nTotal cost of spanning tree = " << totalCost;

    return 0;

}

**Output**

The reference image is : 

**Running the code :**

