Name: Shah Siddh Tejaskumar

Reg.no.: 20BCE1937

---------------------------------------------------------------------------------------------

Graph assignment 2

------------------------------------------------------------------------------------------------

// code for prims algo

#include <iostream>

#include <bits/stdc++.h>

using namespace std;

# define n 5

int selectMinVertex(vector<int>& value , vector<bool>& setMST)

{

    int min = INT\_MIN;

    int vertex;

    for (int i=0; i<n ; i++)

    {

        if (setMST[i] == false && value[i] < min)

        {

            vertex = i;

            min = value[i];

        }

    }

    return vertex;

}

void prims (int graph[n][n], int value[])   // prims algorithm

{

    int parent[n];  // maintaing the parent of nodes

    vector<int> weights (n , INT\_MAX);   // creating weight vector with initial value INFINITY

    vector<bool> setMST (n, false);    // created to keep track whether nodes has been visited or not

    parent[0] = -1; // so the starting node has no parent

    weights[0] = 0; // so as it is 1st node

    for (int i=0; i<n-1; i++)

    {

        int u = selectMinVertex (weights, setMST);  // selecting the next minimum node to explore

        setMST[u] = true;  // marking that node has been visited

        for (int j=0; j<n ; j++)

        {

            if (graph[u][j] != 0 && setMST[j]==false && graph[u][j] < weights[j])

            {

                weights[j] = graph[u][j];

                parent[j] = u;

            }

        }

    }

    for (int i=1; i<n; i++)

    {

        cout << parent[i] << " <-> " << value[i] << ": " << graph[parent[i]][value[i]] << endl;

    }

}

int main()

{

    int value[n];

    for (int i = 0; i < n; i++)  // vertex input

    {

        cin >> value[i];

    }

    cout << "Enter the distances (0 if self loop exist , -1 if not directly reachable)" << endl;

    int graph[n][n];

    for (int i = 0; i < n; i++)  // weight input if edge exist

    {

        for (int j = i; j < n; j++)

        {

            cout << "Distance between the " << value[i] << " and " << value[j] << ": ";

            cin >> graph[i][j];

        }

    }

    prims(graph, value);

    return 0;

}

------------------------------------------------------------------------------

// code for kruskal

#include <iostream>

#include <bits/stdc++.h>

using namespace std;

class edge

{

public:

    int source;

    int dest;

    int weight;

};

bool comparator(edge e1, edge e2) // sorting helper

{

    return e1.weight < e2.weight;

}

int findParent (int v, int \* parent)

{

    if (parent[v] == v)

        return v;

    return findParent (parent[v], parent);

}

void kruskal(edge \*input, int n, int e) // kruskal algo

{

    sort(input, input + e, comparator); // sorting the the edges in ascending order

    edge \*output = new edge[n - 1]; // creating the output array of size (n-1)

    int \* parent  = new int[n];

    for (int i=0; i<n ;i++)

        parent[i] = i;

    int i = 0;

    int count = 0;  // maintaining no of elements in output array

    while (count != n-1) // iterating till we get n-1 edges

    {

        edge currentEdge = input[i];

        // checking if the currentEdge is valid to enter in the output array

        // basically checking if the newly entered edge doesnt form a cycle in tree

        int sourceParent = findParent (currentEdge.source , parent);

        int destParent = findParent (currentEdge.dest , parent);

        if (sourceParent != destParent)

        {

            output[count] = currentEdge;

            count ++ ;

        }

        i++;

    }

    for (int i=0; i<n-1; i++)  // displaying final egdes of output array

        cout << output[i].source << " <-> " << output[i].dest << ": " << output[i].weight << endl;

}

int main()

{

    int n, e; // n = no\_of\_vertices, e = no\_of\_edges

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

    cin >> n;

    cout << "Enter the no. of edges: ";

    cin >> e;

    edge \*input = new edge[n]; // creating the edge type of input array

    for (int i = 0; i < e; i++) // taking the no. of edges as input

    {

        int s, d, w;

        cout << "Enter the source\_vertex: ";

        cin >> s;

        cout << "Enter the dest\_vertex: ";

        cin >> d;

        cout << "Enter the weight: ";

        cin >> w;

        input[i].source = s;

        input[i].dest = d;

        input[i].weight = w;

    }

    kruskal(input, n, e);

    return 0;

}