Name: Tirth Hihoriya

Roll no.: 18bce244

Prac-4 : Find minimun cut-edges and cut-vertices of given Graph

```
#include<bits/stdc++.h>
using namespace std;

vector<vector<int>>a;
vector<vector<int>>b;
const int MAX = 1000;
int n;
int c[MAX],d[MAX],l[MAX],pred[MAX];
int t = 0;
bool art[MAX];
vector<bool> visited;
vector<int> tin, low;
int timer;

int cut_vertex(int src)
{
    c[src] = 1;
    l[src] = d[src] = ++t;

    for (int i = 0; i < a[src].size(); ++i){
        int w = a[src][i];
        if(!c[w]){
            pred[w] = src;
            cut_vertex(w);

            if(pred[src] == -1 && src!=0)
            {
                if(i >= 1)
                {
                    art[src] = true;
                }
            }
            else if(l[w] >= d[src] && src!=0)
            {
                art[src] = true;
            }

            l[src]=min(l[src],l[w]);
        }
        else if(w != pred[src])
        {
            l[src]=min(l[src],d[w]);
        }
    }
    return 0;
}

void cut_edge(int v, int p = -1)
 {
    visited[v] = true;
    tin[v] = low[v] = timer++;
    for (int to : a[v])
        {
        if (to == p)
        {
            continue;
        }
        if (visited[to])
            {
            low[v] = min(low[v], tin[to]);
        }
        else
        {
            cut_edge(to, v);
            low[v] = min(low[v], low[to]);
            if (low[to] > tin[v])
        {
            b[v].push_back(to);
        }
        }
    }
}

void cut_edge_func(int n)
 {
    timer = 0;
    visited.assign(n, false);
    tin.assign(n, -1);
    low.assign(n, -1);
    for (int i = 0; i < n; ++i) {
        if (!visited[i])
            cut_edge(i);
    }
}


int main()
{
    int e, x, y;
    cout<<"Enter no of nodes and no of edges"<<endl;
    cin>>n;
    cin>>e;
    a = vector<vector<int>>(n);
    b = vector<vector<int>>(n);
    for (int i = 0; i < e; ++i)
    {
        cout<<"Enter edge:"<<endl;
        cin>>x>>y;
        a[x].push_back(y);
        a[y].push_back(x);
    }
    cout<<endl;

    cout<<"Graph:"<<endl;
     for(int i=0;i<a.size();i++)
        {
        cout<<"Node "<<i<<" : ";
        for(int x : a[i])
        {
            cout<<i<<" -> "<<x<<"  ";
        }
        cout<<endl;
    }
    cout<<endl;


    cut_vertex(0);
    cout<<"Cut vertices:"<<endl;
    for(int i = 0; i < n;i++)
    {
        if(art[i]==true)
        {
        cout<<i<<endl;
        }
    }
    cout<<endl;


    cut_edge_func(n);
     cout<<"Cut edges:"<<endl;
   for(int i=0;i<b.size();i++)
        {
        for(int x : b[i])
        {
            cout<<i<<" -> "<<x<<"  "<<endl;
        }
    }

    return 0;
}
```

OUTPUT :

```
Enter no of nodes and no of edges
7
8
Enter edge:
0 1
Enter edge:
0 2
Enter edge:
1 2
Enter edge:
1 6
Enter edge:
1 3
Enter edge:
1 4
Enter edge:
3 5
Enter edge:
4 5

Graph:
Node 0 : 0 -> 1  0 -> 2
Node 1 : 1 -> 0  1 -> 2  1 -> 6  1 -> 3  1 -> 4
Node 2 : 2 -> 0  2 -> 1
Node 3 : 3 -> 1  3 -> 5
Node 4 : 4 -> 1  4 -> 5
Node 5 : 5 -> 3  5 -> 4
Node 6 : 6 -> 1

Cut vertices:
1

Cut edges:
1 -> 6



Enter no of nodes and no of edges
4 3
Enter edge:
0 1
Enter edge:
1 2
Enter edge:
2 3

Graph:
Node 0 : 0 -> 1
Node 1 : 1 -> 0  1 -> 2
Node 2 : 2 -> 1  2 -> 3
Node 3 : 3 -> 2

Cut vertices:
1
2

Cut edges:
0 -> 1
1 -> 2
2 -> 3
```