zarif98sjs

About Projects Memento Blog

# LightOj 1291 (Real Life Traffic)

2020-05-11 :: ~5 min read

#lightoj #cp #problem solving #articulation #graph theory

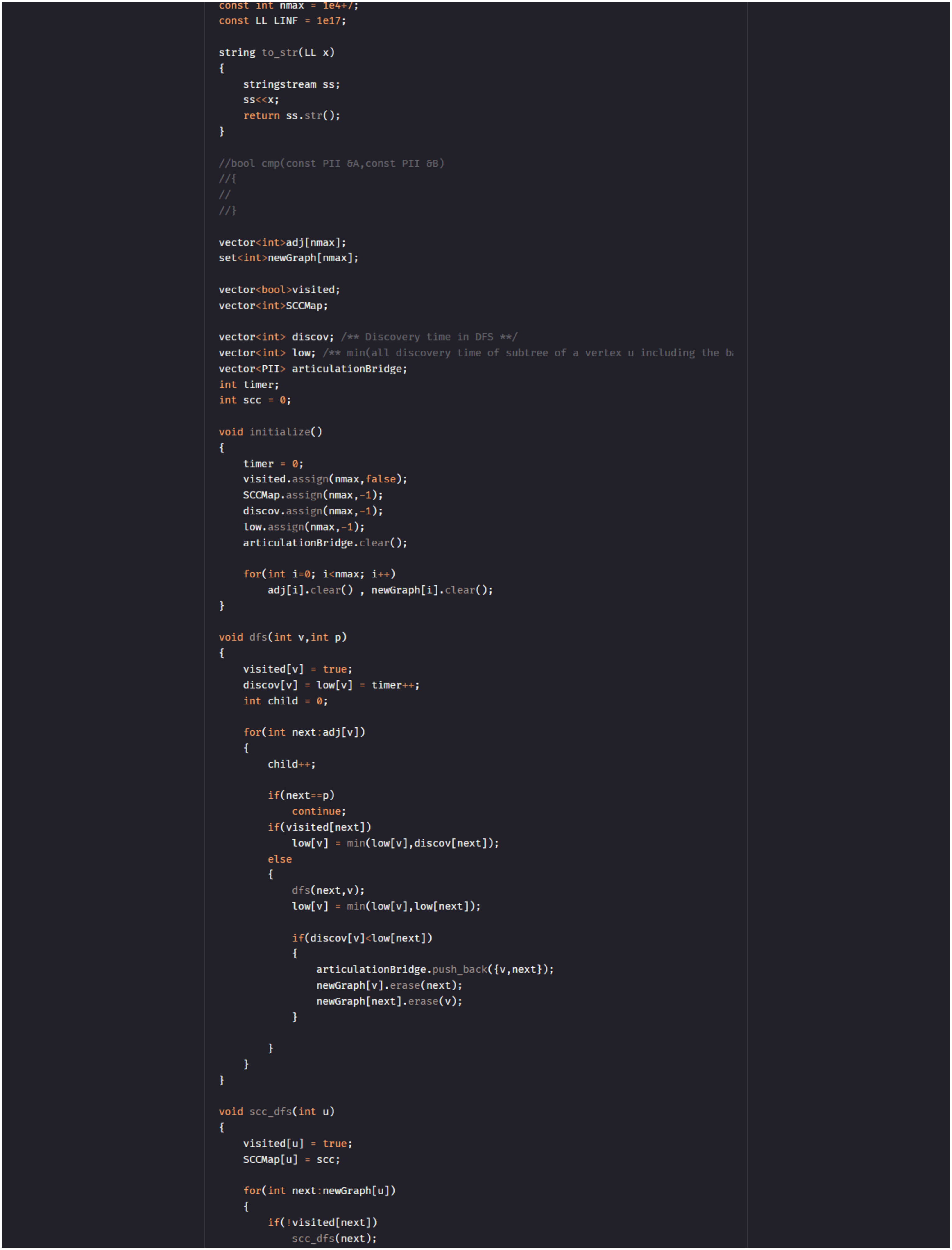
## Idea ⌗

Articulation Bridge

- Find the articulation bridges in the graph.
- If you remove the bridges, you will get several connected components.
- Think carefully, there are 2 types of components.
- The 1st one is the connected component which had more than one bridge, if you cut one bridge, it still can connect with other components through other bridge.
- The 2nd one is the component which had only one bridge, if you cut the bridge, it cannot connect with other components.
- You may think that the answer is the number of 2nd type bridge, as they need another one path to connect with other components.
- Think carefully, you may still minimize the number of edges required.
- If you connect one 2nd type component to another 2nd type component, then it requires one edge rather than two edges.

Copy

```
/** Which of the favors of your Lord will you deny ? **/

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

#define LL long long
#define PII pair<int,int>
#define PLL pair<LL,LL>
#define MP make_pair
#define F first
#define S second
#define INF INT_MAX

#define ALL(x) (x).begin(), (x).end()
#define DBG(x) cerr << __LINE__ << " says: " << #x << " = " << (x) << endl

#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;

template<class TIn>
using indexed_set = tree<
                    TIn, null_type, less<TIn>,
                    rb_tree_tag, tree_order_statistics_node_update>;

/*
PBDS
-------------------------------------------------
1) insert(value)
2) erase(value)
3) order_of_key(value) // 0 based indexing
4) *find_by_order(position) // 0 based indexing
*/

inline void optimizeIO()
{
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
}
```



```
const int nmax = 1e4+7;
const LL LINF = 1e17;

string to_str(LL x)
{
    stringstream ss;
    ss<<x;
    return ss.str();
}

//bool cmp(const PII &A,const PII &B)
//{
//
//}

vector<int>adj[nmax];
set<int>newGraph[nmax];

vector<bool>visited;
vector<int>SCCMap;

vector<int> discov; /** Discovery time in DFS **/
vector<int> low; /** min(all discovery time of subtree of a vertex u including the b
vector<PII> articulationBridge;
int timer;
int scc = 0;

void initialize()
{
    timer = 0;
    visited.assign(nmax,false);
    SCCMap.assign(nmax,-1);
    discov.assign(nmax,-1);
    low.assign(nmax,-1);
    articulationBridge.clear();

    for(int i=0; i<nmax; i++)
        adj[i].clear() , newGraph[i].clear();
}

void dfs(int v,int p)
{
    visited[v] = true;
    discov[v] = low[v] = timer++;
    int child = 0;

    for(int next:adj[v])
    {
        child++;

        if(next==p)
            continue;
        if(visited[next])
            low[v] = min(low[v],discov[next]);
        else
        {
            dfs(next,v);
            low[v] = min(low[v],low[next]);

            if(discov[v]<low[next])
            {
                articulationBridge.push_back({v,next});
                newGraph[v].erase(next);
                newGraph[next].erase(v);
            }

        }
    }
}

void scc_dfs(int u)
{
    visited[u] = true;
    SCCMap[u] = scc;

    for(int next:newGraph[u])
    {
        if(!visited[next])
            scc_dfs(next);
    }
}

int main()
{
    //freopen("out.txt","w",stdout);

    optimizeIO();

    int tc;
    cin>>tc;

    for(int q=1;q<=tc;q++)
    {
        initialize();

        int n,m;
        cin>>n>>m;

        for(int i=1; i<=m; i++)
        {
            int a,b;
            cin>>a>>b;
            adj[a].push_back(b);
            adj[b].push_back(a);

            newGraph[a].insert(b);
            newGraph[b].insert(a);
        }

        for(int i=0; i<n; i++)
        {
            if(!visited[i])
                dfs(i,-1);
        }

        visited.assign(nmax,false);
        scc = 0;

        for(int i=0; i<n; i++)
        {
            if(!visited[i])
            {
                scc_dfs(i);
                scc++;
            }
        }

        for(auto bridge:articulationBridge)
        {
            numBridgesConToComp[SCCMap[bridge.F]]++;
            numBridgesConToComp[SCCMap[bridge.S]]++;
        }

        int cc = 0;

        for(int i=0;i<scc;i++)
        {
            if(numBridgesConToComp[i]==1)
                cc++;
        }

        int ans = (cc+1)/2;

        cout<<"Case "<<q<<": ";
        cout<<ans<<endl;
    }

    return 0;
}
```

READ OTHER POSTS

← LightOj 1308 (Ant Network)

LightOj 1300 (Odd Personality) →

**0 Comments** - *powered by utteranc.es*

Write Preview

Sign in to comment

Styling with Markdown is supported

Sign in with GitHub