Bridges

Given an undirected graph containing N vertices and M edges, find all the articulation points and the bridges in the graph.

Input

- The first line consists of two space-separated integers denoting N and M,
- M lines follow, each containing two space-separated integers X and Y denoting there is an edge between X and Y.

Output

- One line consists of two integers denoting the number of articulation points and the number of bridges.

Bridges

Input

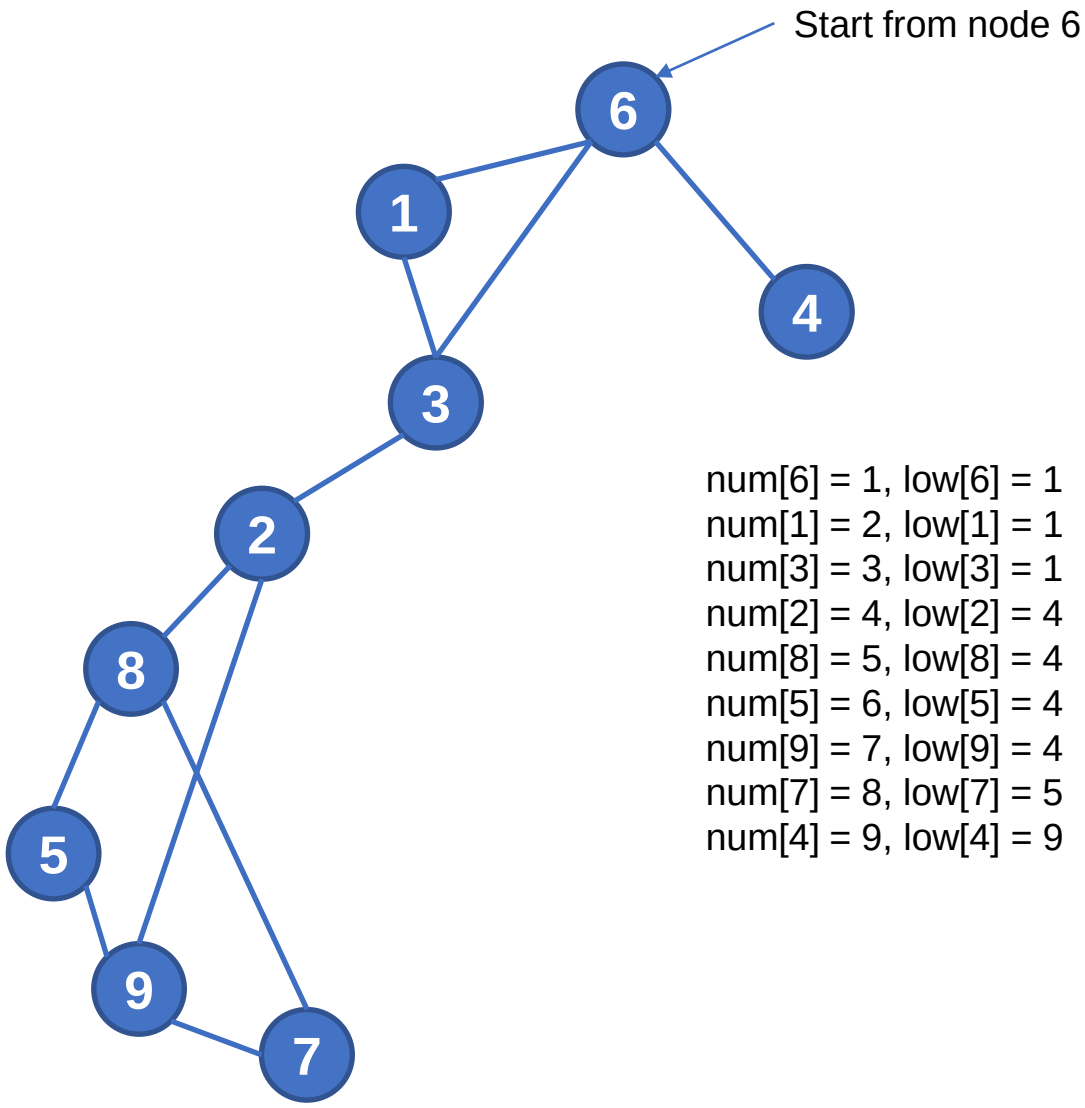
10 12
1 10
10 2
10 3
2 4
4 5
5 2
3 6
6 7
7 3
7 8
8 9
9 7

Output

4 3

Bridges

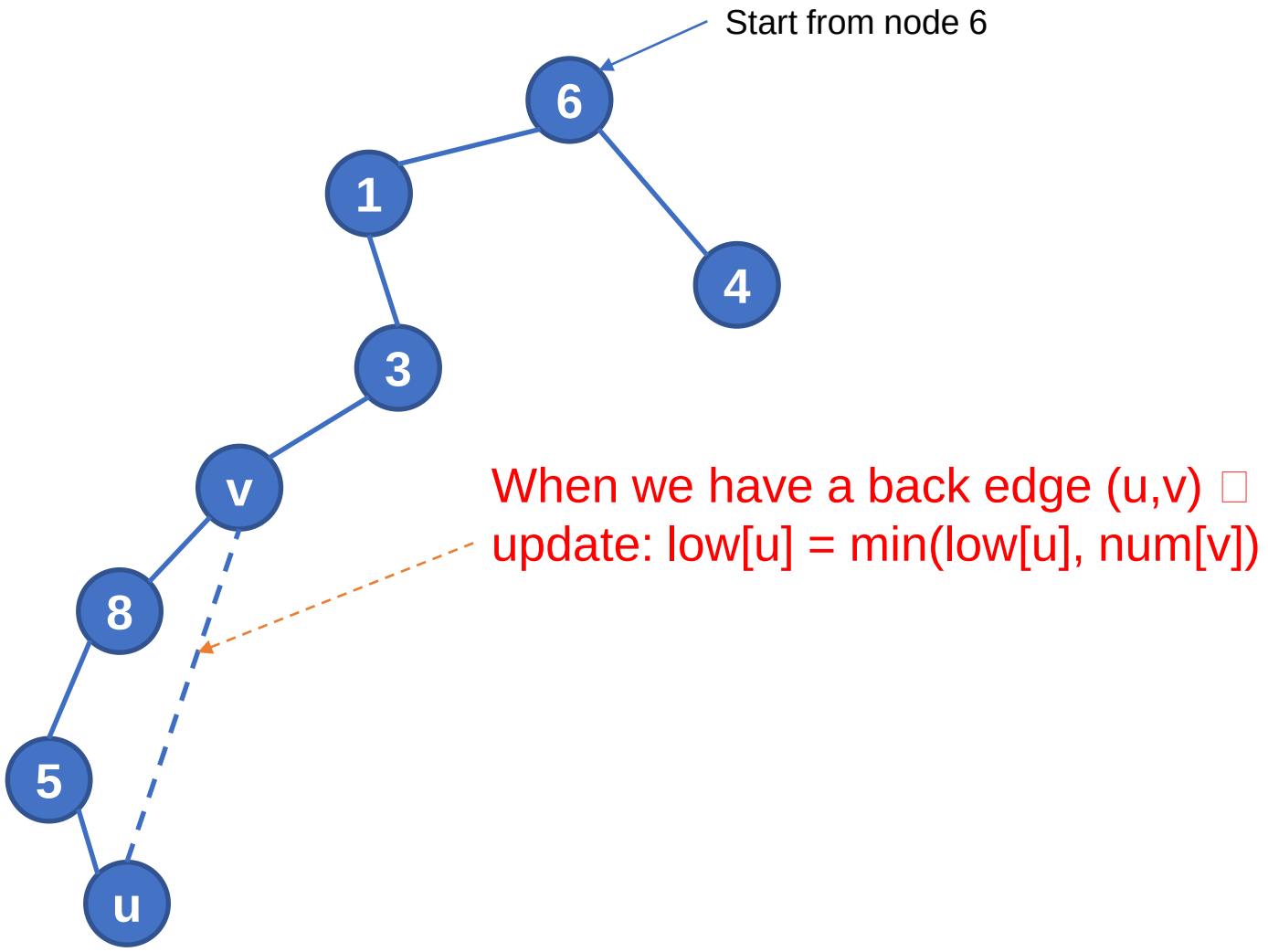
- DFS tree
  - DFS starts from a node u visits a descendants of u on the DFS tree
- Maintain data structures:
  - num[v]: time point node v is visited
  - low[v]: minimal num of some node x such that v is equal to x or there is a back end (u,x) in which u is the node v or some descendant of v



num[6] = 1, low[6] = 1
num[1] = 2, low[1] = 1
num[3] = 3, low[3] = 1
num[2] = 4, low[2] = 4
num[8] = 5, low[8] = 4
num[5] = 6, low[5] = 4
num[9] = 7, low[9] = 4
num[7] = 8, low[7] = 5
num[4] = 9, low[4] = 9

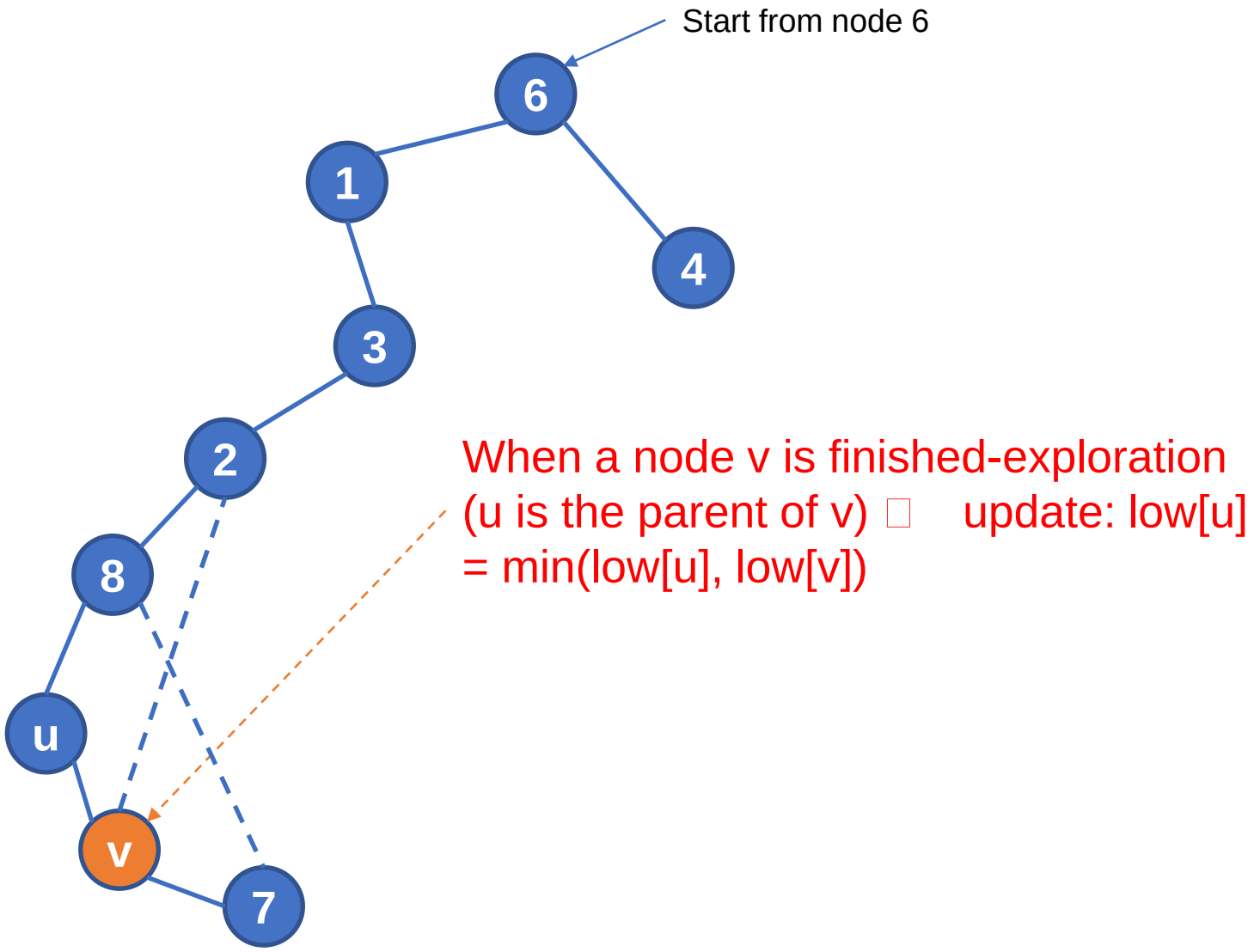
Bridges

- DFS tree
  - DFS starts from a node u visits a descendants of u on the DFS tree
- Maintain data structures:
  - num[v]: time point node v is visited
  - low[v]: minimal num of some node x such that v is equal to x or there is a back end (u,x) in which u is the node v or some descendant of v



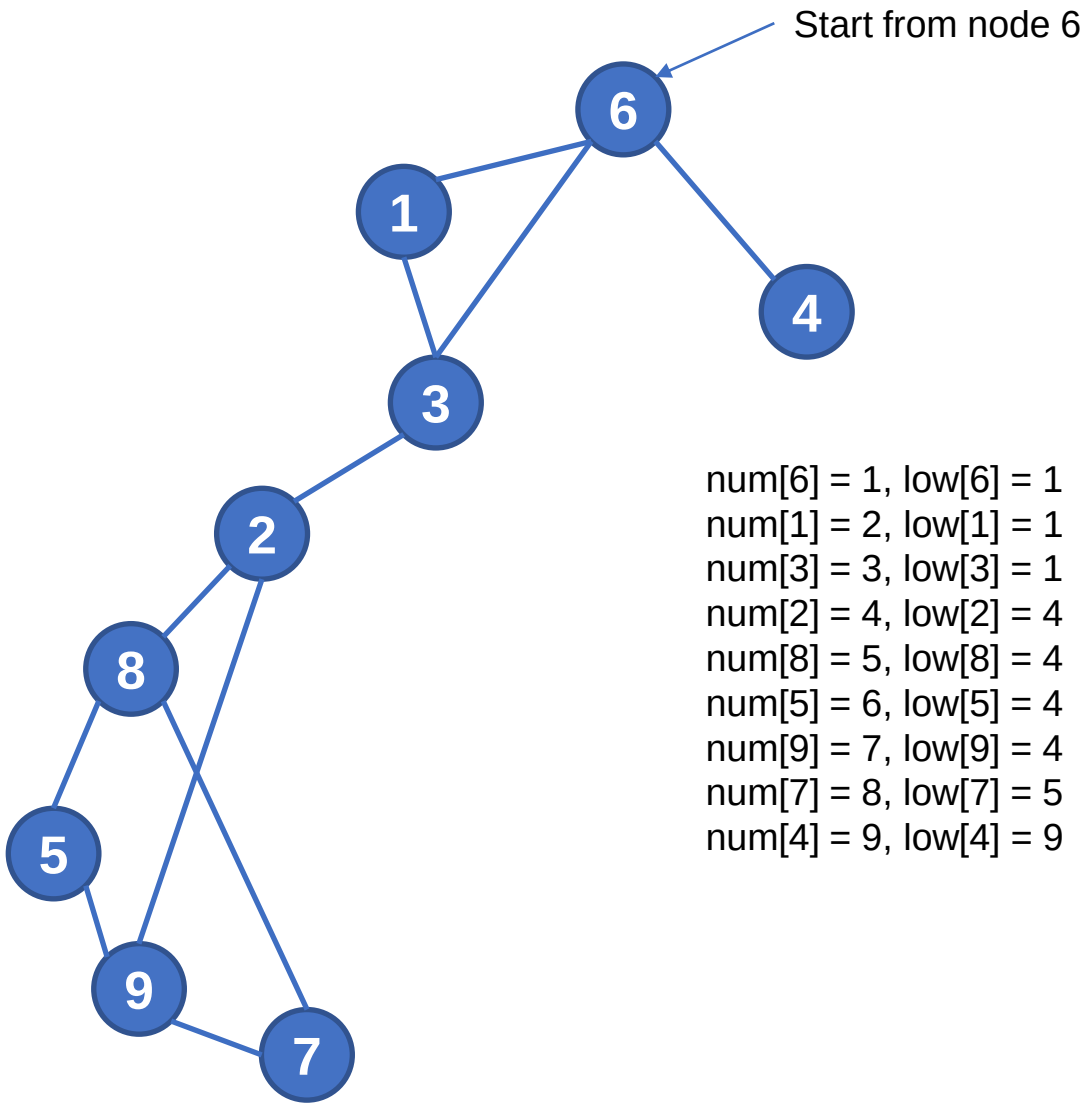
Bridges

- DFS tree
  - DFS starts from a node u visits a descendants of u on the DFS tree
- Maintain data structures:
  - num[v]: time point node v is visited
  - low[v]: minimal num of some node x such that v is equal to x or there is a back end (u,x) in which u is the node v or some descendant of v



Bridges

- Bridge:
  - Forward edge (u,v) having low[v] > num[u] is a bridge
- Articulation point:
  - If u is not a root of the DFS tree and forward edge (u,v) having low[v] ≥ num[u] then u is an articulation point
  - If u is the root of a DFS tree, then u is an articulation point if it has more than 1 child



num[6] = 1, low[6] = 1
num[1] = 2, low[1] = 1
num[3] = 3, low[3] = 1
num[2] = 4, low[2] = 4
num[8] = 5, low[8] = 4
num[5] = 6, low[5] = 4
num[9] = 7, low[9] = 4
num[7] = 8, low[7] = 5
num[4] = 9, low[4] = 9

Implementation – Bridges

```
#include <bits/stdc++.h>

using namespace std;
const long N = 100000 + 7;
const long INF = 1000000000 + 7;
const long MODULE = 1000000000 + 7;
typedef pair<int,int> ii;

int n, m;
int d[N], p[N], khop[N], cau[N], num[N], low[N], ca[N], child[N];
vector <int> a[N];
int t;
void input(){
        cin >> n >> m;
        for (int i = 0; i < m; i++) {
                int u, v;
                cin >> u >> v;
                a[u].push_back(v);
                a[v].push_back(u);
        }
}
```

Implementation – Bridges

```
void dfs(int u){
        t ++;
        num[u] = t;
        low[u] = num[u];
        for (int i = 0; i < a[u].size(); i++) {
                int v = a[u][i];
                if ( v == p[u] ) continue;

                if ( num[v] ) {
                        low[u] = min(low[u] , num[v] );
                }
                else {
                        p[v] = u;
                        dfs(v);
                        low[u] = min(low[u] , low[v]);
                }
        }

}
```

Implementation – Bridges

```
int main(){
        input();
        int anskhop = 0, anscau = 0;
        t = 0;
        for (int i = 1; i <= n; i++) {  if ( !num[i] ) dfs(i);  }
        for (int i = 1; i <= n; i++) {
                int v = p[i];
                if (v > 0) child[v]++;
        }
        for (int i = 1; i <= n; i++) {
                int u = p[i];
                if ( u > 0 && p[u] > 0 && low[i] >= num[u] ) khop[u] = 1;
        }
        for (int i = 1; i <= n; i++) {   if ( p[i] == 0 && child[i] >= 2 ) khop[i] = 1;     }

        for (int i = 1; i <= n; i++) anskhop += khop[i] == 1;
        for (int i = 1; i <= n; i++) {   if ( p[i] != 0 && low[i] >= num[i] ) ++anscau;     }

        cout << anskhop << " " << anscau << endl;
        return 0;
}
```