```
        if (low[v] >= dis[u] && pre != 0) art[u] = 1;
        ++child;
      }
      else if (v != pre) low[u] = min(low[u], dis[v]);
    }
    if (pre == 0 && child > 1) art[u] = 1;
}
```

Bridge:

```
const int N = 3e5 + 9;
int T, low[N], dis[N];
vector<int> g[N];
vector<pair<int, int>> bridge;
int n, m;

void dfs(int u, int pre = 0) {
    low[u] = dis[u] = ++T;
    int child = 0;
    for (auto &v : g[u]) {
        if (!dis[v])
        {
            dfs(v, u);
            low[u] = min(low[u], low[v]);
            if (low[v] > dis[u]) bridge.push_back({u, v});
            ++child;
        }
        else if (v != pre) low[u] = min(low[u], dis[v]);
    }
}
```

Bipartite:

```
const int N = 2e5 + 7;
vector<int> g[N];
vector<short> clr(N, -1); // 2 color(0 and 1)
bool isBipartite = 0; // Odd length cycle are
Bipartite graph

void dfs(int u, int c) {
    if(isBipartite) return;
    clr[u] = c;
    for(auto &v: g[u]) {
        if(clr[v] != -1) {
            if(clr[v] == c) {
                isBipartite = 1;
                return;
            }
            continue;
        }
        dfs(v, c ^ 1);
    }
}
```

Strongly Connected Components:

```
const int N = 1e5 + 7;
vector<int> g[N], r[N], comp[N]; // 1'Base
Indexing
bool vis[N];
stack<int> st;
int u, v, n, m, numOfComp;
void dfs(int i)
{
    vis[i] = 1;
    for(auto &j: g[i])
    {
        if(vis[j]) continue;
        dfs(j);
    }
    st.push(i);
}
void dfs2(int i)
{
    vis[i] = 1;
    comp[numOfComp].push_back(i);

    for(auto &j: r[i])
    {
        if(vis[j]) continue;
        dfs2(j);
    }
}
void solve()
{
    cin >> n >> m;

    for(int i = 1; i <= m; i++)
    {
        cin >> u >> v;
        g[u].push_back(v);
        r[v].push_back(u);
    }
    for(int i = 1; i <= n; i++)
    {
        if(vis[i]) continue;
        dfs(i);
    }
    memset(vis, 0, sizeof vis);
    numOfComp = 0;
    while(!st.empty())
    {
        int x = st.top();
        if(!vis[x])
        {
            ++numOfComp;
            dfs2(x);
        }
        st.pop();
    }
    for(int i = 1; i <= n; i++)
    {
        if(comp[i].size() == 0) continue;
        cout << i << "=> ";
```