Chuyên đề:
DFS VÀ ỨNG DỤNG

A. Các templates cơ bản

1. Template DFS Cơ sở trên đa đồ thị vô hướng (xây dựng num, low, parent)

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

using namespace std;
typedef pair<int, int> II;

int n;
vector<vector<pair<int, int> > > adj;

vector<int> num, low, color, parent;
int Time = 0;

void DFS(int u, int edge_dad) {
    low[u] = num[u] = ++Time;
    color[u] = 1;
    for(auto e : adj[u]) {
        int v = e.first, edge_id = e.second;
        if (edge_id != edge_dad) {
            if (color[v] == 0) {
                parent[v] = u;
                DFS(v, edge_id);
                low[u] = min(low[u], low[v]);
            } else low[u] = min(low[u], num[v]);
        }
    }
}

int main() {
    // Dung da do thi
    cin >> n;
    adj.resize(n + 1);
    int m;
    cin >> m;
    for(int i = 1; i <= m; ++i) {
        int u, v;
        cin >> u >> v;
        adj[u].push_back({v, i});
        adj[v].push_back({u, i});
    }

    // DFS Base
    Time = 0;
    num.resize(n + 1, 0);
    low.resize(n + 1, 0);
```

```
    color.resize(n + 1, 0);
    parent.resize(n + 1, 0);
    for(int u=1; u <= n; ++ u) if (color[u] == 0) DFS(u, 0);
}
```

2. Templates tính bậc khớp của các đỉnh

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

using namespace std;

int n;
vector<vector<pair<int, int> > > adj;
vector<int> num, low, hint, color;
int Time = 0;

void DFS(int u, int edge_dad) {
    num[u] = low[u] = ++Time;
    color[u] = 1;
    for(auto e : adj[u]) {
        int v = e.first, edge_id = e.second;
        if (edge_id != edge_dad) {
            if (color[v] == 0) {
                DFS(v, edge_id);
                low[u] = min(low[u], low[v]);
                if (low[v] >= num[u]) ++hint[u];
            } else low[u] = min(low[u], num[v]);
        }
    }
    if (edge_dad == 0 && hint[u]) --hint[u];
}


int main() {
    // Read input
    cin >> n;
    adj.resize(n + 1);
    int m;
    for(int i = 1; i <= m; ++i) {
        int u, v;
        cin >> u >> v;
        adj[u].push_back({v, i});
        adj[v].push_back({u, i});
    }

    // Tim mang hint
    num.resize(n + 1, 0);
    low.resize(n + 1, 0);
    hint.resize(n + 1, 0);
    color.resize(n + 1, 0);
    Time = 0;
```

```
    for(int u = 1; u <= n; ++u)
        if (color[u] == 0) DFS(u, 0);
}
```

3. Template tìm cạnh cầu

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

using namespace std;

int n;
vector<vector<pair<int, int> > > adj;
vector<int> num, low, color, bridges;
int Time = 0;

void DFS(int u, int edge_dad) {
    num[u] = low[u] = ++Time;
    color[u] = 1;
    for(auto e : adj[u]) {
        int v = e.first, edge_id = e.second;
        if (edge_id != edge_dad) {
            if (color[v] == 0) {
                DFS(v, edge_id);
                low[u] = min(low[u], low[v]);
                if (low[v] > num[u]) bridges[edge_id] = 1;
            } else low[u] = min(low[u], num[v]);
        }
    }
}

int main() {
    // Read input
    cin >> n;
    adj.resize(n + 1);
    int m;
    cin >> m;
    for(int i = 1; i <= m; ++i) {
        int u, v;
        cin >> u >> v;
        adj[u].push_back({v, i});
        adj[v].push_back({u, i});
    }

    // Check bridger
    num.resize(n + 1, 0);
    low.resize(n + 1, 0);
    color.resize(n + 1, 0);
    bridges.resize(m + 1, 0);
    for(int u = 1; u <= n; ++u)
        if (color[u] == 0) DFS(u, 0);
}
```

4. Template tìm các thành phần liên thông mạnh (Tarjan algorithm)

```
#include <bits/stdc++.h>
using namespace std;
int n;
vector<vector<int> > adj;

vector<int> num, low, color, id_stconected;
vector<vector<int> > st_conected;
int cnt_stconected;
int Time = 0;

stack<int> st;
void DFS(int u) {
    num[u] = low[u] = ++Time;
    color[u] = 1;
    st.push(u);
    for(int v : adj[u]) {
        if (color[v] == 0) {
            DFS(v);
            low[u] = min(low[u], low[v]);
        } else
            if (color[v] == 1) low[u] = min(low[u], num[v]);
    }
    if (num[u] == low[u]) {
        ++cnt_stconected;
        int v;
        do {
            v = st.top();
            st.pop();
            id_stconected[v] = cnt_stconected;
            st_conected[cnt_stconected].push_back(v);
            color[v] = 2;
        } while (v != u);
    }
}
int main() {
    // Read input
    cin >> n;
    adj.resize(n + 1);
    int m;
    cin >> m;
    for(int i = 1; i <= m; ++i) {
        int u, v;
        cin >> u >> v;
        adj[u].push_back(v);
    }
    num.resize(n + 1, 0);
    low.resize(n + 1, 0);
    color.resize(n + 1, 0);
    id_stconected.resize(n + 1, 0);
    st_conected.resize(n + 1);
    cnt_stconected = 0;
    Time = 0;
    for(int u = 1; u <= n; ++u)
        if (color[u] == 0) DFS(u);
}
```

5. Template Tim các thành phần song liên thông cạnh

```
#include <bits/stdc++.h>
using namespace std;
int n;
vector<vector<pair<int, int> > > adj;
vector<int> num, low, color, id_twoconected;
vector<vector<int> > two_conected;
int cnt_twoconected, Time;

stack<int> st;
void DFS(int u, int edge_dad) {
    num[u] = low[u] = ++Time;
    color[u] = 1;
    st.push(u);
    for(auto e : adj[u]) {
        int v = e.first, edge_id = e.second;
        if (edge_id != edge_dad) {
            if (color[v] == 0) {
                DFS(v, edge_id);
                low[u] = min(low[u], low[v]);
            } else low[u] = min(low[u], num[v]);
        }
    }
    if (low[u] == num[u]) {
        ++cnt_twoconected;
        int v;
        do {
            v = st.top();
            st.pop();
            id_twoconected[v] = cnt_twoconected;
            two_conected[cnt_twoconected].push_back(v);
        } while (v != u);
    }
}
int main() {
    // Read input
    cin >> n;
    adj.resize(n + 1);
    int m;
    cin >> m;
    for(int i = 1; i <= m; ++i) {
        int u, v;
        cin >> u >> v;
        adj[u].push_back({v, i});
        adj[v].push_back({u, i});
    }
    num.resize(n + 1, 0);
    low.resize(n + 1, 0);
    color.resize(n + 1, 0);
    id_twoconected.resize(n + 1, 0);
    two_conected.resize(n + 1);
    cnt_twoconected = 0;
    Time = 0;
    for(int u = 1; u <= n; ++u)
        if (color[u] == 0) DFS(u, 0);
}
```

6. Template Tìm các thành phần song liên thông đỉnh

```
#include <bits/stdc++.h>
using namespace std;
int n;
vector<vector<pair<int, int> > > adj;
vector<int> num, low, color, siz;
vector<vector<int> > bi_conected;
int cnt_biconected = 0, Time = 0;
stack<int> st;
void DFS(int u, int edge_dad) {
    num[u] = low[u] = ++Time;
    color[u] = 1;
    st.push(u);
    siz[u] = 1;
    for(auto e : adj[u]) {
        int v = e.first, edge_id = e.second;
        if (edge_id != edge_dad) {
            if (color[v] == 0) {
                DFS(v, edge_id);
                low[u] = min(low[u], low[v]);
                siz[u] += siz[v];
                // Xu ly 1 thanh phan song lien thong canh
                if (low[v] >= num[u]) {
                    ++cnt_biconected;
                    int w;
                    do {
                        w = st.top();
                        st.pop();
                        bi_conected[cnt_biconected].push_back(w);
                    } while (v != w);
                    if (low[v] == num[u] || siz[v] == 1)
                        bi_conected[cnt_biconected].push_back(u);
                }
            } else low[u] = min(low[u], num[v]);
        }
    }
}
int main() {
    // Read input
    cin >> n;
    adj.resize(n + 1);
    int m;
    cin >> m;
    for(int i = 1; i <= m; ++i) {
        int u, v;
        cin >> u >> v;
        adj[u].push_back({v, i});
        adj[v].push_back({u, i});
    }
    num.resize(n + 1, 0); low.resize(n + 1, 0); color.resize(n + 1, 0);
    bi_conected.resize(n + 1);  siz.resize(n + 1, 0);
    cnt_biconected = 0;
    Time = 0;
    for(int u = 1; u <= n; ++u)
        if (color[u] == 0) DFS(u, 0);
}
```

7. Template sắp xếp topo trên DAG

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

using namespace std;

int n;
vector<vector<int> > adj;
vector<int> tp, color;
int Time;

void DFS(int u) {
    color[u] = 1;
    for(int v : adj[u])
        if (color[v] == 0) DFS(v);
    tp[--Time] = u;
}

int main() {
    // Read input
    cin >> n;
    adj.resize(n + 1);
    int m;
    cin >> m;
    for(int i = 1; i <= m; ++i) {
        int u, v;
        cin >> u >> v;
        adj[u].push_back(v);
    }

    tp.resize(n + 1);
    color.resize(n + 1);
    Time = n + 1;
    for(int u = 1; u <= n; ++u)
        if (color[u] == 0) DFS(u);
}
```

Áp dụng: Tìm được đi dài nhất từ đỉnh *s* đến đỉnh *t* trên DAG

Sau khi có thứ tự topo tp[1...n] việc tìm đường đi dài nhất được thực hiện như sau:

```
for(int i = 1; i <= n; ++i) dp[i] = INT_MIN;
for(int i = 1; i <= n; ++i) {
    int u = tp[i];
    if (u == s) dp[u] = 0;
    if (dp[u] > INT_MIN) {
        for(auto e : adj[u]) {
            int v = e.first, L = e.second;
            dp[v] = max(dp[v], dp[u] + L);
        }
    }
}
```

8. Qui hoạch động trên cây

8.1 Tính qua con

```
void DFS(int u, int p) {
    // Khoi dau dp[u]
    for(int v : adj[u]) {
        if (v != p) {
            DFS(v, u);
            // Cap nhat dp[u] qua dp[v]
        }
    }
}
```

8.2 Tính qua cha

```
void DFS(int u, int p) {
    // Tinh dp[u] qua dp[p]
    for(int v : adj[u]) {
        if (v != p) {
            DFS(v, u);
        }
    }
}
```