**THUẬT TOÁN TARJAN**

# A. Tìm các thành phân liên thông mạnh

## Input:

int n; // Số đỉnh

int g[maxn]; // g[u] - Danh sách đỉnh kề của đỉnh u

## Output:

int sLTM; // Số thành phần liên thông mạnh

int LTM[maxn] // LTM[u] - Số hiệu thành phân liên thông mạnh của u

int x[maxn]; // Thứ tự Topo các thành phần liên thông mạnh

*Code mẫu:*

int cl[maxn], s[maxn], sn;

void DFS(int u) {

cl[u]=1;

num[u]=low[u]=++id;

s[++sn]=u;

for(auto &v : g[u]) {

if (cl[v]==0) {

DFS(v);

low[u]=min(low[u],low[v]);

} else if (cl[v]==1)

low[u]=min(low[u],num[v]);

}

if (num[u]==low[u]) {

++sLTM;

int v;

do {

v=s[sn--];

x[++slx]=v;

LTM[v]=sLTM;

cl[v]=2;

} while (v!=u);

}

}

int main() {

....

for(int i=1;i<=n;++i) cl[i]=0;

id=sLTM=slx=sn=0;

for(int i=1;i<=n;++i) if (cl[i]==0) DFS(i);

....

}

*Xây dựng DAG với mỗi đỉnh là một thành phần liên thông mạnh*

vector<int> gLTM[maxn]; // g[id] - Danh sách LTM kề với LTM i

for(int i=1;i<=sLTM;++i) gLTM[i].clear();

for(int u=1;u<=n;++u)

for(auto &v : g[u]) if (LTM[u]!=LTM[v]) {

int i=LTM[u], j=LTM[v];

gLTM[i].push\_back(j);

}

# B. Tim các thành phần song liên thông

S là song liên thông nếu với mọi x, y (x!=y) thuộc S tồn tại 2 đường đi không chung cạnh

## Input:

int m; // Số cạnh

pair<int,int> E[maxm]; // Danh sách cạnh

int n; // Số đỉnh

int g[maxn]; // g[u] - Danh sách đỉnh kề của đỉnh u (lưu chỉ số cạnh)

## Output:

int sSLT; // Số thành phần song liên thông

int SLT[maxn] // SLT[u] - Số hiệu thành phân song liên thông của u

int x[maxn]; // Thứ tự Topo các thành phần song liên thông

*Code mẫu:*

int cl[maxn], s[maxn], sn;

void DFS(int u) {

cl[u]=1;

num[u]=low[u]=++id;

s[++sn]=u;

for(auto &i : g[u]) **if (Pe[u]!=abs(i))** {

int v=(i>0) ? E[i].second : E[-i].first;

if (cl[v]==0) {

Pd[v]=u, Pe[v]=abs(i);

DFS(v);

low[u]=min(low[u],low[v]);

} else low[u]=min(low[u],num[v]);

}

if (num[u]==low[u]) {

++sSLT;

int v;

**do {**

**v=s[sn--];**

**x[++slx]=v;**

**SLT[v]=sSLT;**

**} while (v!=u);**

**}**

**}**

int main() {

....

for(int i=1;i<=n;++i) cl[i]=0;

id=sSLT=slx=sn=0;

for(int i=1;i<=n;++i) if (cl[i]==0) DFS(i);

....

}

*Xây dựng rừng (nhiều cây) với mỗi đỉnh là một thành phần song liên thông*

vector<int> gSLT[maxn]; // gSLT[i]-Danh sách SLT kề với SLT i

for(int i=1;i<=m;++i) {

int u=E[i].first, v=E[i].second;

if (SLT[u]!=SLT[v]) {

int k=SLT[u], l=SLT[v];

gSLT[k].push\_back(l);

gSLT[l].push\_back(k);

}

}

# C. Tìm các thành phần Biconected

S là thành phần Biconected nếu với mọi x, y (x!=y) thuộc S tồn tại ít nhất 2 đường đi không chung đỉnh

## Input:

int n; // Số đỉnh

vector<int> g[maxn]; // Danh sách đỉnh kề

## Output:

int sBIC; // Số lượng thành phần biconected

int cBIC[maxn]; // Số đỉnh của mỗi thành phần biconected

vector<int> gBIC[maxn]; // gBIC[i] - Danh sách của biconected thứ i

*Code mẫu:*

int cl[maxn], s[maxn], sn;

void DFS(int u) {

cl[u]=1;

num[u]=low[u]=++id;

s[++sn]=u;

for(auto &v : g[u]) **if (Pd[u]!=v)** {

if (cl[v]==0) {

Pd[v]=u;

DFS(v);

low[u]=min(low[u],low[v]);

} else low[u]=min(low[u],num[v]);

}

if (Pd[u] && low[u]>=num[Pd[u]) {

int w=Pd[u];

++sBIC;

cBIC[sBIC]=0; gBIC[sBIC].clear();

int v;

do {

v=s[sn--];

++cBIC[sBIC];

gBIC[sBIC].push\_back(v);

} while (v!=u);

if (low[u]==num[w]||cBIC[sBIC]==1) {

++cBIC[sBIC];

gBIC[sBIC].push\_back(w);

}

}

}

int main() {

....

for(int i=1;i<=n;++i) cl[i]=0;

id=sBIC=slx=sn=0;

for(int i=1;i<=n;++i) if (cl[i]==0) DFS(i);

....

}

# D. Sắp xếp topo trên DAG

Một đồ thị có hướng G=(V,E) được gọi là một DAG khi đồ thị đó không có chu trình có hướng.

## Input:

int n; // Số đỉnh

vector<int> g[maxn]; // g[u] - Danh sách đỉnh kề của u, Dl đảm bảo là DAG

## Output:

int x[maxn]; // x[1], ...,x[n] là một thứ tự topo

*Code mẫu:*

void DFS(int u) {

cl[u]=1;

for(auto &v : g[u])

if (cl[v]==0) DFS(v);

x[--slx]=u;

}

int main() {

....

for(int i=1;i<=n;++i) cl[i]=0;

slx=n+1;

for(int i=1;i<=n;++i) if (cl[i]==0) DFS(i);

....

}

Chú ý

Mỗi thành phần liên thông cấu tạo bởi các thành phần song liên thông nối với nhau bằng cầu

Mỗi thành phần song liên thông cấu tạo bởi các biconected đính với nhau qua các khớp