**DFS Trên cây**

# A. DFS Cơ bản

Cấu trúc cây là sự mở rộng của cấu trúc mảng (mảng có thể xem như là một cây tuyến tính), với việc thực hiện DFS(1) ta luôn định hướng lại cây và thu được một sắp xếp topo:

## Input:

int n; // Số đỉnh

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

## Outut:

int x[maxn]; // Mảng thứ tự DFS

int Pd[maxn]; // Pd[u] - Đỉnh cha của u

int start[maxn]; // start[u] - Thời điểm bắt đầu DFS(u)

int stop[maxn]; // stop[u] - Thời điểm kết thúc DFS(u)

*Code cơ bản:*

void DFS(int u) {

cl[u]=1;

start[u]=++id;

x[id]=u;

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

if (cl[v]==0) {

Pd[v]=u;

DFS(v);

}

}

stop[u]=id;

}

Một trong những kết quả quan trọng nhất của sắp xếp topo trên là với một đỉnh bất kỳ, tập hợp các đỉnh cùng các đỉnh trong cây con DFS gốc nằm trong một đoạn liên tục trên sắp xếp topo. Do vậy **các bài toán về tổng hợp thông tin trong các cây con có thể được đưa về các bài toán truy vấn đoạn liên tục trên mảng** và do đó có thể sử dụng các cấu trúc dữ liệu tăng tốc được trang bị tên mảng như Segment Tree, Binary Indexed Tree (BIT), RMQ,...

# B. Qui hoạch động trên cây

Giả sử cần tổng hợp thông tin lấy từ các con, cháu... của ta có thể lập công thức qui hoạch động kiểu như:

* Nếu là lá thì tính trực tiếp
* Ngược lại, gọi là các con của u (ta có thể xây dựng một hàm tùy theo từng bài toán

Do vậy việc tính công thức trên mọi đỉnh có thể thực hiện theo phương pháp qui hoạch động trên topo ngược:

for(int i=n;i>=1;i--) {

u=x[i];

khởi tạo f[u];

for (v là con của u) tính f[u] qua f[v];

}

*Ví dụ 1:* Tính s[u] là số đỉnh trong cây con của cây DFS vơi gốc là đỉnh u

for(int i=n;i>=1;--i) {

int u=x[i];

s[u]=1;

for(auto &v : g[u])

if (Pd[v]==u) s[u] += s[v];

}

Tổng quát, với một đỉnh bất kỳ ta có thể chia tập hợp các đỉnh của cây thành hai loại

- Tập hợp các đỉnh của cây con gốc

- Tập hợp các đỉnh còn lại.

Với các đỉnh của ta cũng chia nó thành hai loại:

+ Các đỉnh

+Các đỉnh - là các đỉnh thuộc ngoại trừ các đỉnh thuộc và (các cây con "anh em" với )

Ta có thể lập các hàm qui hoạch động trên các tập hợp dựa theo sắp xếp topo DFS. Chú ý tính trên trước (topo ngược) sau đó mới tính trên (topo xuôi)

*Ví du 2:* Với mỗi đỉnh u tìm khoảng cách xa nhất từ u đến các đỉnh khác

Đặt f[u] là khoảng cách lớn nhất từ đỉnh u đến một đỉnh trong cây con DFS gốc u

Đặt g[u] là khoảng cách lớn nhất từ đỉnh u đến một đỉnh không nằm trong cây con gốc u

Nếu tính được f[u], g[u] thì khoảng cách xa nhất từ u đến đỉnh khác là max(g[u],f[u])

Ta có các cộng thức qui hoạch động:

f[u]=max(f[v1],f[v2],...,f[vk])+1 với v1, v2, ..., vk là con của u (nếu u không con f[u]=0)

Đặt w=Pd[u] khi đó

g[u]=0 nếu w=0 (u là đỉnh xuất phát)

g[u]=max(g[w]+1, D(w,u)}

Trong đó

Ở đây f2[u] là độ dài của "đường đi lớn thứ nhì" từ u đến đỉnh trong cây con gốc u

(Việc tính f2 có thể đi kèm với việc tính f)

*Tham khảo đoạn code dưới đây:*

// Qui hoạch ngược

for(int i=n;i>=1;--i) {

int u=x[i];

f[u]=f2[u]=0;

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

if (f[u]<f[v]+1) {

f2[u]=f[u];

f[u]=f[v]+1;

} else if (f2[u]<f[v]+1)

f2[u]=f[v]+1;

}

}

// Qui hoạch xuôi

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

int u=x[i];

int w=Pd[u];

if (w==0) g[u]=0; else {

g[u]=g[w]+1;

if (f[w]==f[u]+1) g[u]=max(g[u],f2[w]+1);

else g[u]=max(g[u],f[w]+1);

}

}