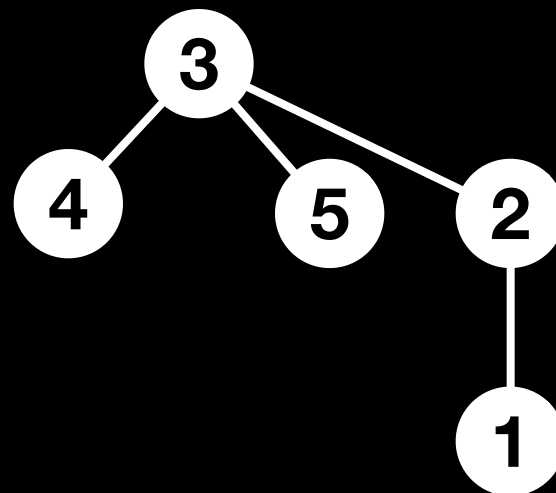
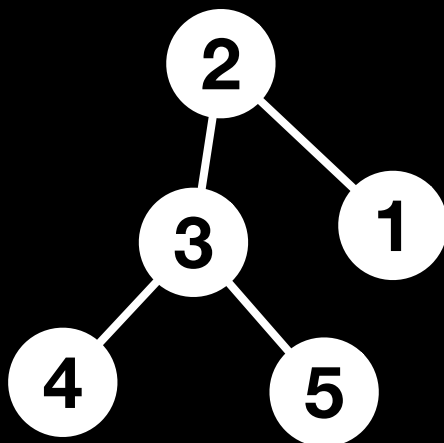
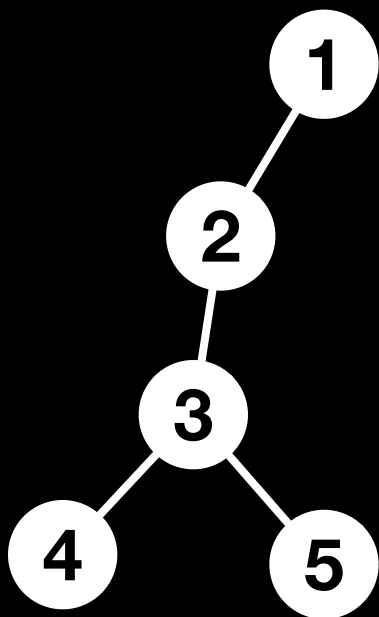


C104

树上换根



P870 战略轰炸6

树的半径

建模

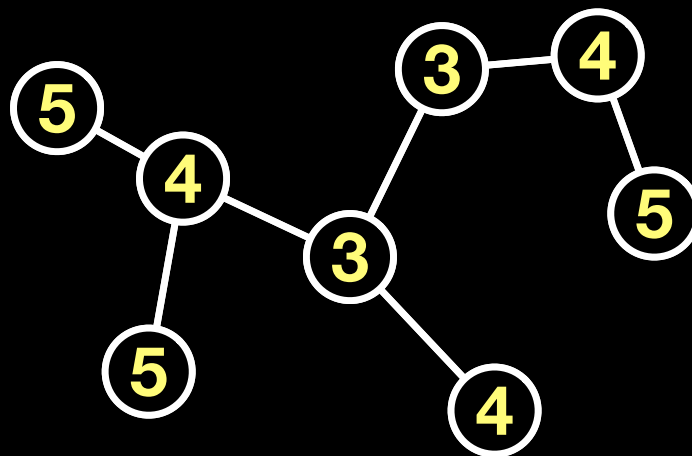
→ 破题（树上离线问询）

求到**无根树**（结果与根无关）上**每个点到其他点的最远距离**
（该距离最小值称为树的**半径**）

→ 暴力算法

枚举点，计算最远距离（DFS/BFS）
（也可认为**以每个点为根**时的高度）

复杂度： $O(n^2)$



先考虑单问询

→ 如只求1个起点，显然可以该点为根
到根节点最远的点，就是最深子树的高度
 $h[x] = 1 + \max\{h[\text{子节点}y]\}$

→ 求多个起点

即求每个点为（全树的）根时的信息

注意区分“求每个点为根的子树”的信息

此类问题称为换根问题，有人称其解法为换根DP

→ 暴力算法

以每个点为起点分别DFS/BFS求高度： $O(n^2)$

优化目标：一次DFS求出所有点的答案🤔



非根节点的问询

→ 当点 x 不是根的时候，从 x 出发的路径有几种走法？

1. 往下走，进入 x 的子树
2. 往上走，到 x 的父节点 p

$ans[x] = \max\{\text{往下走最远}, \text{往上走最远}\}$ (废话)

→ 1. 设 $f[x] = x$ 往下走最远距离 (简单)

$f[x] = h[x] - 1$ (高度减一)

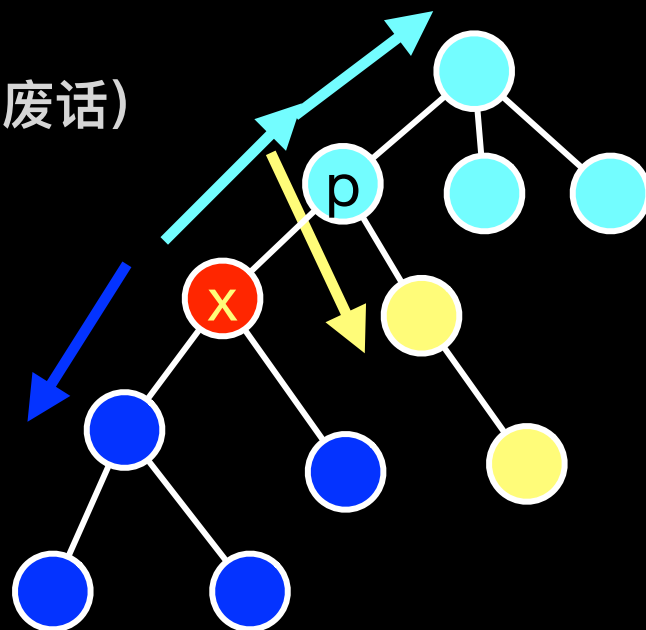
$h[x] = 1 + \max\{h[\text{子节点}y]\}$

→ 2. 设 $g[x] = x$ 往上走最远距离 🤔

2.1. 继续往上走: $1 + g[p]$

2. 往上后折返: $1 + \max\{h[\text{兄弟节点}]\}$

最高的兄弟子树+1



$$\text{ans}[x] = \max\{h[x]-1, g[x]\}$$

$$g[x] = 1 + \max\{g[p], h[\text{兄弟节点}]\}$$

最高的兄弟节点

→ 又分2种情况（纳尼？）

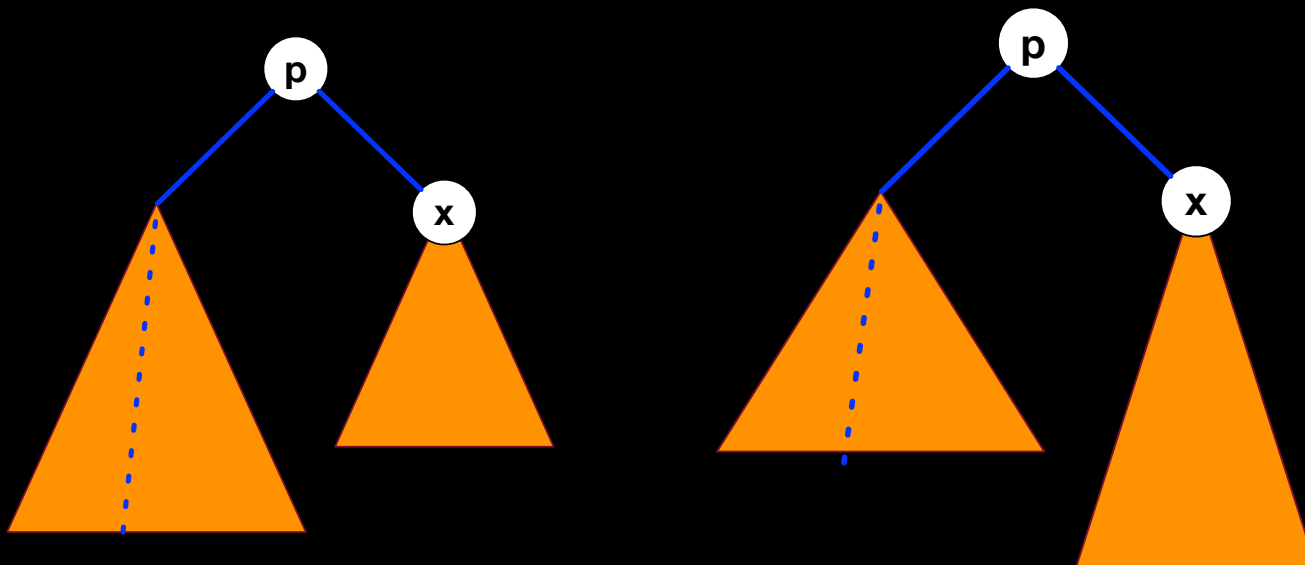
情况1: **x不是p的最高子树**（非“长子”）

$$\max h[\text{兄弟节点}] = h[p\text{的最高子树}] = h[p] - 1$$

情况2: **x是p的最高子树**

$$\max h[\text{兄弟节点}] = h[p\text{的次高子树}]$$

次高子树是可以直接求的（求h时打2个擂台即可）



```

9 void dfs(int x,int fa){
10     for (int i=0,y;i<es[x].size();i++){
11         y=es[x][i];
12         if (y!=fa) {
13             dfs(y,x);
14             if (h[y][0]+1>h[x][1]) h[x][1]=h[y][0]+1;
15             if (h[x][1]>h[x][0]) swap(h[x][1], h[x][0]);
16         }
17     }
18 }

```

求最大+次大的打擂台写法

求最高/次高子树（自底向上）

h[x][0]=x的最高子树高度
h[x][1]=x的次高子树高度



```

20 void dfs_up(int x,int fa){
21     for (int i=0,y;i<es[x].size();i++){
22         y=es[x][i];
23         if (y!=fa) {
24             int d=h[y][0]+1==h[x][0];
25             g[y]=max(g[x],h[x][d])+1;
26             dfs_up(y,x);
27         }
28     }

```

d=y是不是x的最高子树

求g[x]（自顶向下）

思考题：为何不能合一个函数？

暴力对拍（重要！）

→ 对拍是用**两个程序**运行同一输入数据

通常其中一个正确性容易保证（如暴力）

对比输出的一致性来判断另一个程序的正确性

→ 如本题中可以使用枚举起点+BFS来进行对拍

→ 生成数据

手造数据难以很大，未测试中等/较大规模数据，可用程序生成

```
void genData(){
    n=2000;
    for (int i=2,u,v;i<=n;i++){
        u=i,v=rand()%(i-1)+1;
        es[u].push_back(v);
        es[v].push_back(u);
    }
}
```

生成一棵树的边表，**顶点数=2000**

$u=i, v \in [1..i-1]$
保证连通+无环

我们的目标是**没有蛀牙**
让程序搞定一切

赋值表达式

暴力枚举+DFS最远点

正解

```
50 int dfsBf(int x,int fa,int d,int L=0){
51     for (int i=0,y;i<es[x].size();i++)
52         if ((y=es[x][i])!=fa)
53             L=max(L,dfsBf(y,x,d+1)+1);
54     return L;
55 }
56
57 void solveBf(){
58     for (int i=1;i<=n;i++)
59         cout<<dfsBf(i,-1,0)<<" ";
60     cout<<endl;
61 }
62
63 void solve(){
64     dfs(1,0);
65     dfs_up(1,0);
66     for (int i=1;i<=n;i++)
67         cout<<max(g[i],h[i][0])<<" ";
68     cout<<endl;
69 }
```

见证奇迹的时刻

```
71 int main(){
72     genData();
73     //    input();
74     solve();
75     solveBf();
76     return 0;
77 }
```

```
int main(){
//    genData();
    input();
    solve();
//    solveBf();
    return 0;
}
```

```
int main(){
//    genData();
    input();
//    solve();
    solveBf();
    return 0;
}
```

```
7 8 8 8 8 9 9 9 10 10 9 9 9 10 10 11 12 9 11 13 9 12 11 11 9 12 9 10 10 9 11 10 10 12 11
10 12 12 11 11 11 11 13 11 11 10 13 12 13 11 12 9 12 11 9 10 13 12 9 11 12 10 14 14 9
7 8 8 8 8 9 9 9 10 10 9 9 9 10 10 11 12 9 11 13 9 12 11 11 9 12 9 10 10 9 11 10 10 12 11
10 12 12 11 11 11 11 13 11 11 10 13 12 13 11 12 9 12 11 9 10 13 12 9 11 12 10 14 14 9
Program ended with exit code: 0
```

测试版本
测了一组 $n=100$ 数据
(手算能做到吗😁)

提交版本
(模块化的好处)

骗分版本
(正解没搞定)

P1708 快递选择4

建模


→ 破题（最优化问题，树上离线问询）

无根树上**选一个根节点，使所有节点深度总和最大**

（找一个点，到其他点的距离之和最大）

→ **先考虑1个节点（固定根）怎么求（敲简单）**

求节点深度（敲基本树上DP，5分钟完成）

```
vector<ll> es[N];  
ll d[N];  
void dfs(ll u, ll fa){  
      
}
```

利用离线

→ 可以自行选择换根的顺序

如DFS顺序（先父节点再子节点）

→ 假设已知父节点u的答案 $f[u]$

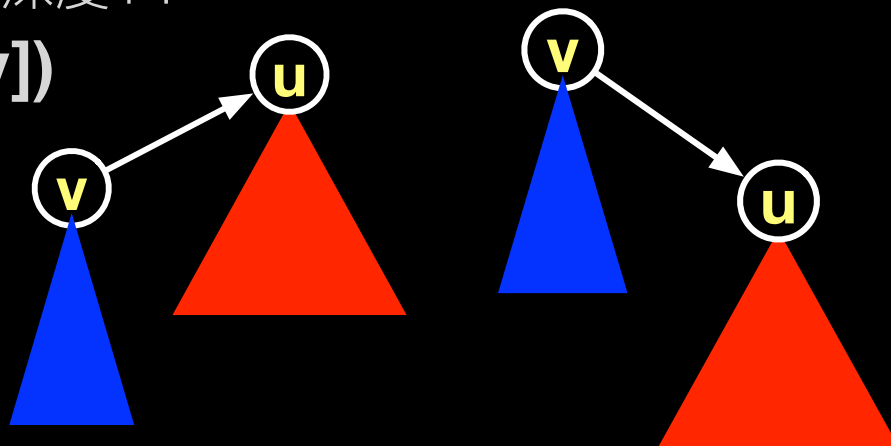
$f[u]$ =以u为根其他点的深度和

→ 现将u的子节点v作为根，答案有多大变化？

1.v及其子树深度-1

2.其他节点(u及v的兄弟子树)深度+1

→ $f[v]=f[u]-sz[v]+(n-sz[v])$



$$f[v]=f[u]-sz[v]+(n-sz[v])$$

有没有问题? 😊

→ **sz[v]**是什么? 这是一个哲学问题

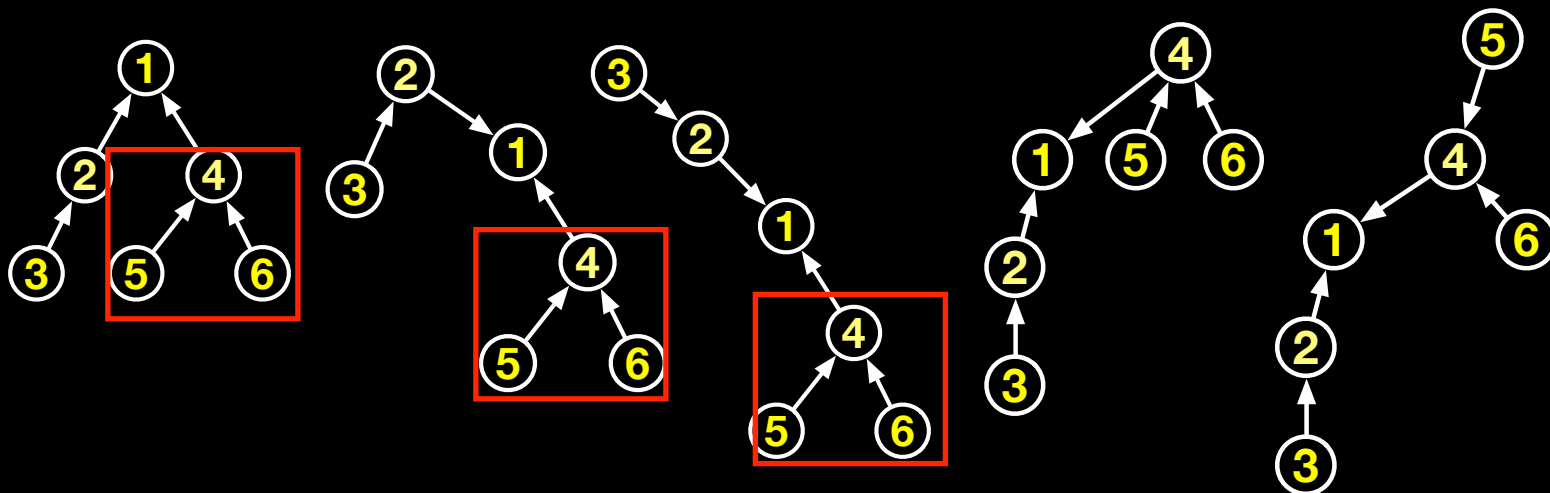
v的子树大小, 是换根前还是换根后? 🤔

→ 按说换根后子树大小应该重算, 则复杂度与暴力无异

但此处是**DFS顺序**换根

换到v前, 其**(原始的)**子树并未动过

所以此时sz[v]就是原始 (以1为根) 的sz



```
15 void calc(ll u, ll fa){
16     for (ll i=0, v; i<es[u].size(); i++)
17         if ((v=es[u][i])!=fa) {
18             f[v]=f[u]-sz[v]+(n-sz[v]);
19             calc(v, u);
20         }
21 }
```

计算顺序：自顶向下

```
23 void solve(){
24     d[1]=1;
25     dfs(1, 0);
26     for (ll i=1; i<=n; i++) f[1]+=d[i];
27     calc(1, 0);
28     for (ll i=1; i<=n; i++)
29         if (f[i]>f[ans]) ans=i;
30     cout<<ans<<endl;
31 }
```

预计算DFS，得到f[1]

换根算DFS，得到所有f

暴力对拍（枚举+暴力求深度，请自己实现）

```
int main(){
    genData();
    //    input();
    solve();
    solveBf();
    return 0;
}
```

→ 作业合格要求

后几行必须如图

保留input+solve，无WA

保留input+solveBf，无WA

换根问题思路初步小结

→ 先考虑固定根节点怎么做

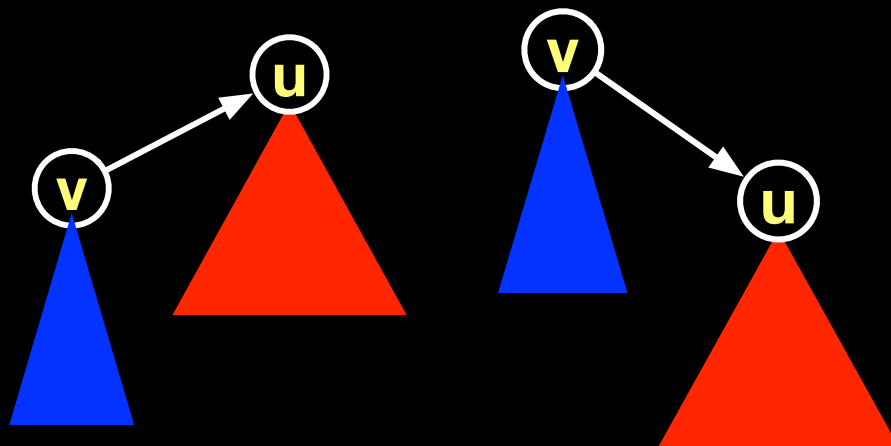
如想不出更好办法，则枚举根节点（暴力换根，并可用来对拍）

→ 利用离线选择合适的换根顺序（如DFS）

考虑换根后答案的变化

DFS序好处是父子换根不影响其他点

→ 有些时候在原树上区分“外部”/“内部”即可解决问题



P1709 换根重剖

树链剖分 (C105) 预备知识

→ 树剖是一种（听上去）高大上的算法

将**树分成若干条（一维的）链**

有多种方案：重链剖分、实链剖分、长链剖分，etc

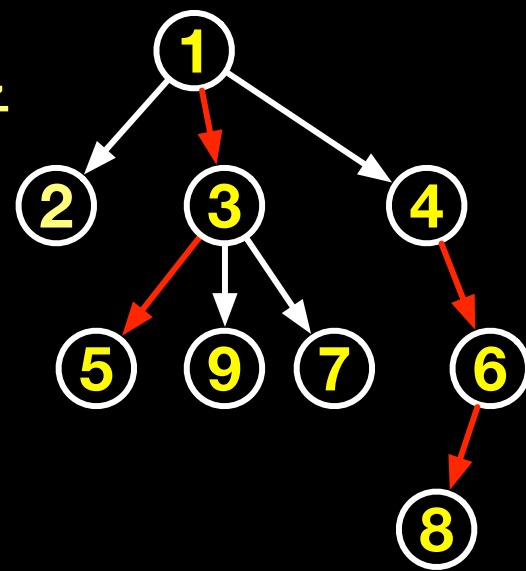
→ **重链剖分**

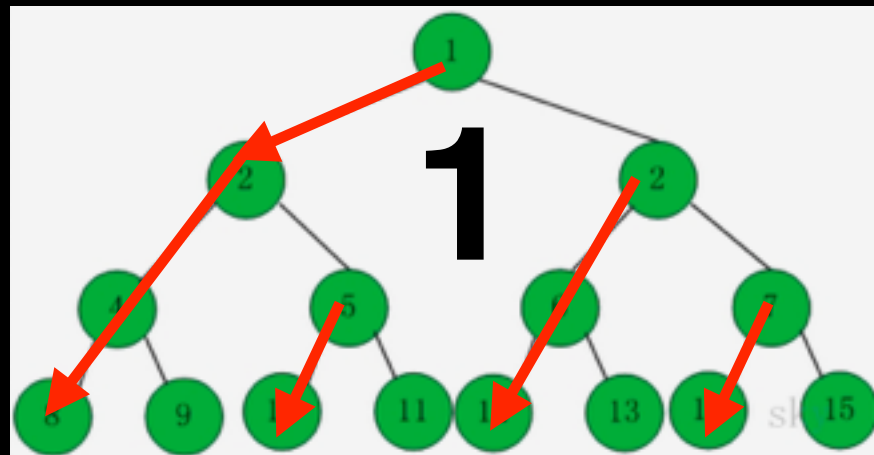
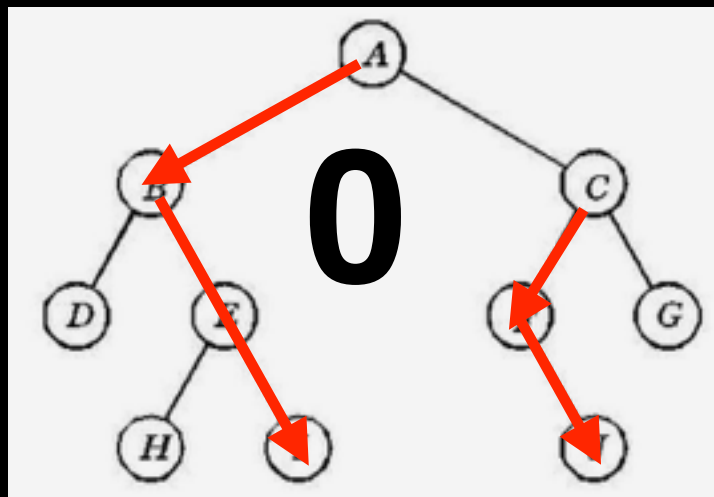
每个节点指向节点数最多的子树，Over

最大子树称为**重儿子**，其他子树称为**轻儿子**

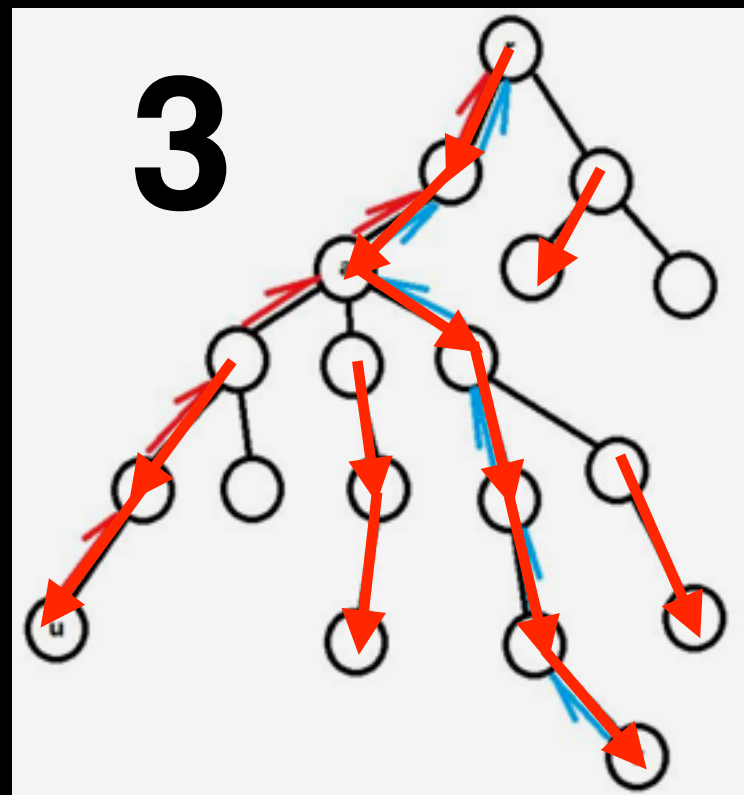
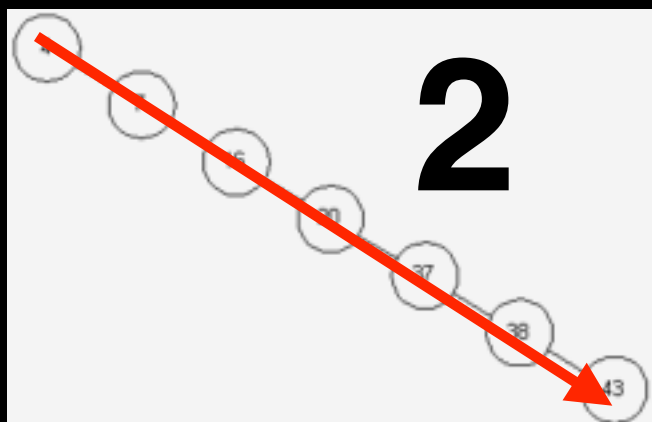
→ **简单特征**

- 1.链上都是直系关系（废话）
- 2.任何两条链不会相交（废话）
- 3.任何节点属于唯一的链（废话）
- 4.链的底端都是叶节点（废话）
- 5.根除外，链的顶端都是轻儿子（废话）





我剖!



选重儿子

→（给定根节点）求子树大小 $sz[u]$ ，地球人都会做
重子树，无非是再求个max值而已

子树比较规则

```
9  bool better(ll u, ll v) {
10     return sz[u] > sz[v] || (sz[u] == sz[v] && u < v);
11 }
12 void dfs(ll u, ll f) {
13     sz[u] = 1, son[u] = 0;
14     for (ll i = 0, v; i < es[u].size(); ++i) {
15         if ((v = es[u][i]) == f) continue;
16         dfs(v, u);
17         sz[u] += sz[v];
18         if (better(v, son[u])) son[u] = v;
19     }
20     L[u] = L[son[u]] + 1;
21 }
```

$sz[0]=0$ 虚拟擂主

打擂台求重儿子

$L[u]=u$ 向下的重链长度

暴力对(pian)拍(fen) (80分)

这里换根略难，你可以选择枚举根节点硬算

```
void solveBf(){  
    for (ll i=1;i<=n;++i)  
        dfs(i,0),cout<<L[i]<<" ";  
    cout<<endl;  
}
```

思考题：为何多次调用dfs，son/L这些数组不需要每次清空？

换根（选学）

→ 还是选择如DFS顺序

假设已知父节点u的方案

→ 现将u的子节点v为根，子树有多大变化？

v多了一个子树u（大小 $=n-sz[v]$ ）

u少了一个子树v（大小 $=sz[v]$ ）

其他节点子树状态不变

→ 重剖有多大变化？

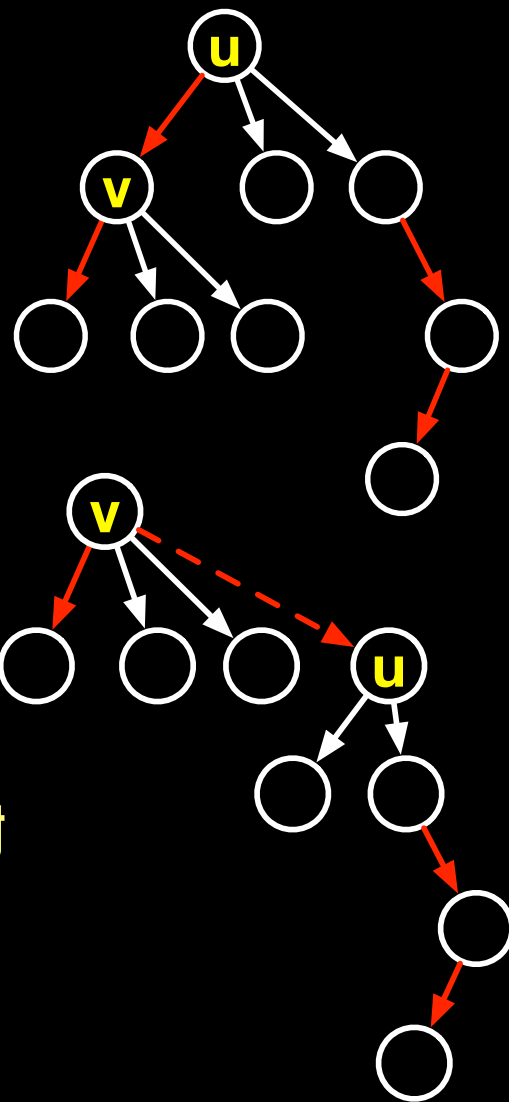
如(换根后的) $sz[u] > sz[son[v]]$ ，则 $son[v]=u$

否则 $son[v]$ 不变

如 $v=son[u]$ ，则 $son[u]=u$ (换根前的)次大子树

否则 $son[u]$ 不变

其他节点 son 不变



```
23 void calc(ll u, ll f){
```

```
24     ans[u]=L[u];
```

以u为根，记录答案

```
25     ll s1=0, s2=0;
```

```
26     for (ll i=0, v; i<es[u].size(); ++i){
```

打擂台求u的
最大/次大子树

```
27         v=es[u][i];
```

```
                if (better(v, s2)) swap(v, s2);
```

```
                if (better(s2, s1)) swap(s2, s1);
```

```
    }
```

```
31     for (ll i=0, v, x; i<es[u].size(); ++i){
```

```
32         if ((v=es[u][i])==f) continue;
```

```
33         if (v==(x=s1)) x=s2;
```

x=u新的重儿子

```
34         sz[u]-=sz[v], sz[v]+=sz[u];
```

```
35         L[u]=L[son[u]=x]+1;
```

```
36         if (better(u, son[v]))
```

```
37             L[v]=L[son[v]=u]+1;
```

```
38         calc(v, u);
```

```
39         sz[v]-=sz[u], sz[u]+=sz[v];
```

```
40         L[v]=L[son[v]=son[v]]+1;
```

```
41     }
```

```
42     L[u]=L[son[u]=son[u]]+1;
```

```
43 }
```

思考题：
为何要重新算？

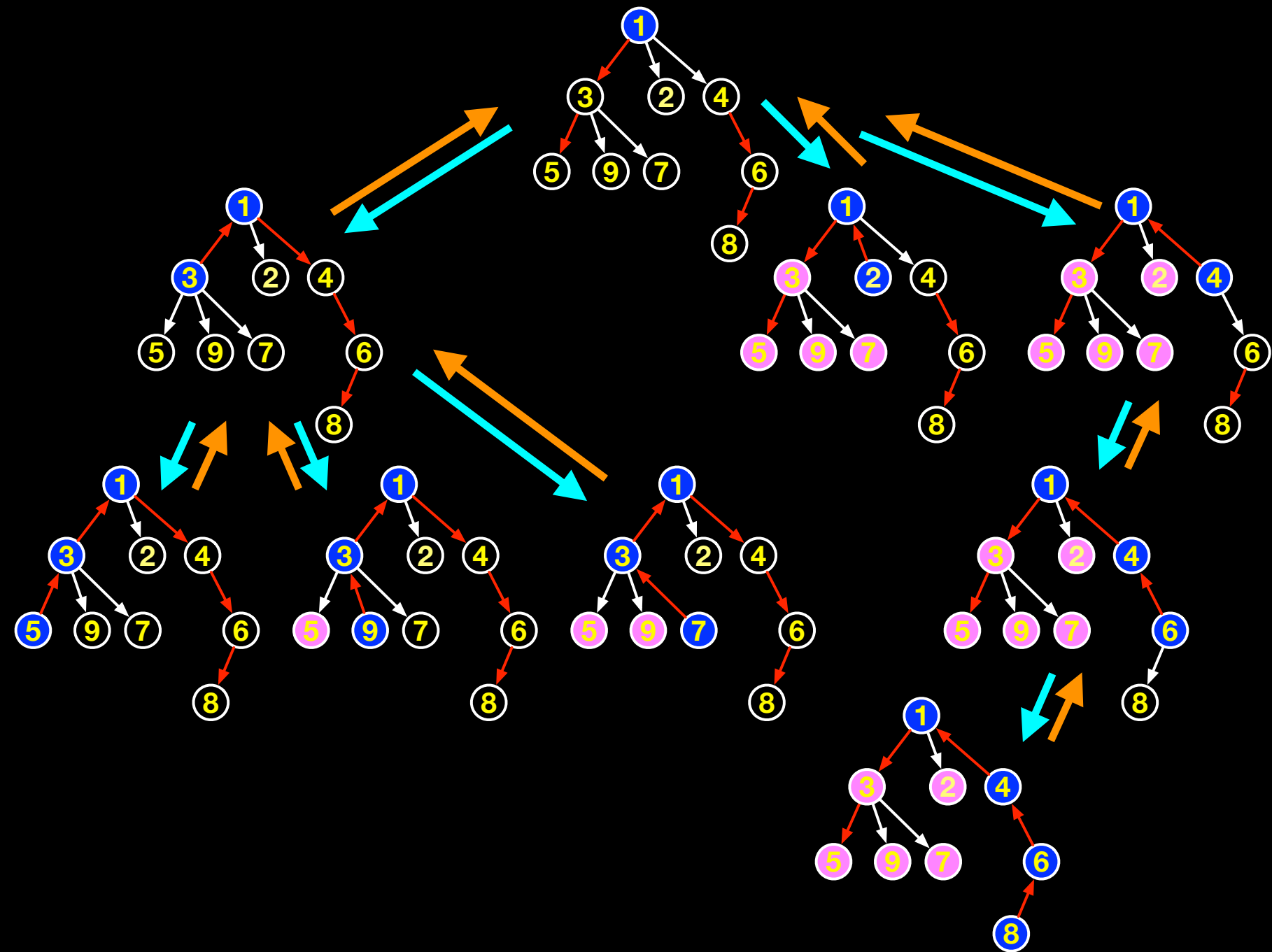
更新sz
注意顺序

更新L

此时son, L是v为根的重剖

回溯恢复v的值

回溯恢复u的值



作业

- 1.战略轰炸6 (P870)
- 2.快递选址4 (P1708。有坑, 40分即可)
C104第1个独立AC的奖励红包一个~
- 3.换根重剖 (P1709, 80分即可)
- 4.树的直径 (P1685, 拓展题)