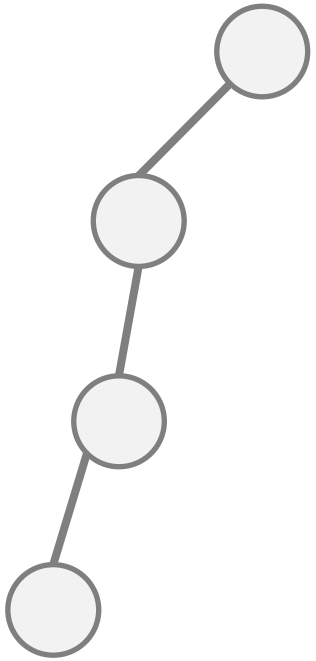
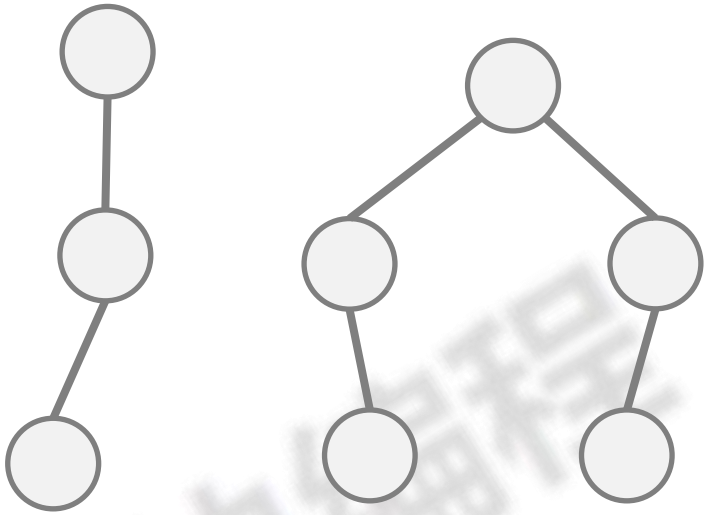
# C++算法

# 链：特指直线型路径



链



路径

# 树链剖分

将树分成若干条链

# 重链剖分

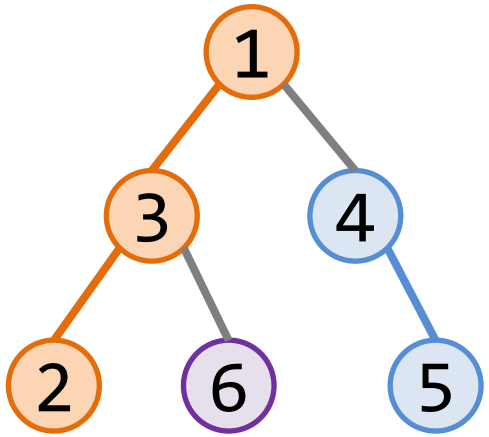
heavy-light decomposition

# 重儿子

子树节点最多的儿子叫做“重儿子”

如果两个儿子对应子树大小相同
约定选取编号较小的儿子为“重儿子”

其他儿子
是轻儿子

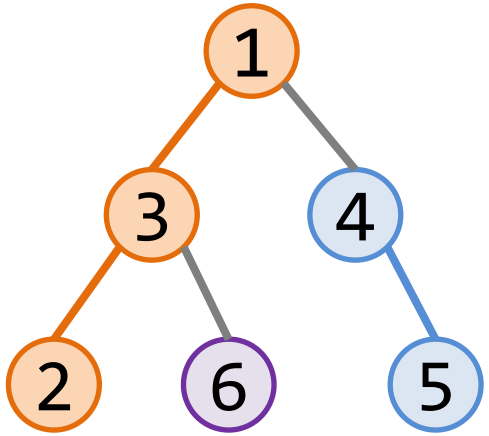


# 重儿子

子树节点最多的儿子叫做“重儿子”

如果两个儿子对应子树大小相同
约定选取编号较小的儿子为“重儿子”

其他儿子
是轻儿子

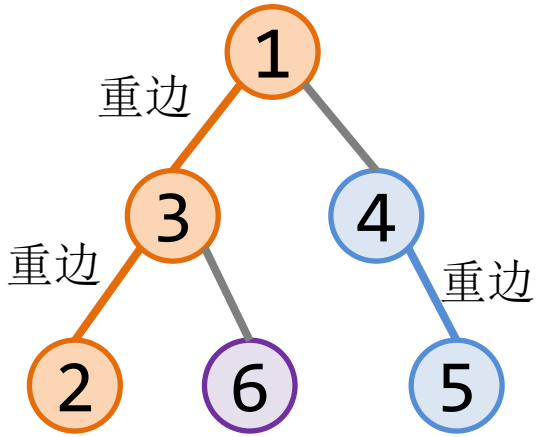



son[u]表示u的重儿子节点编号

| | u=1 | u=2 | u=3 | u=4 | u=5 | u=6 |
|---|---|---|---|---|---|---|
| son[u]= | | 0 | | | 0 | 0 |

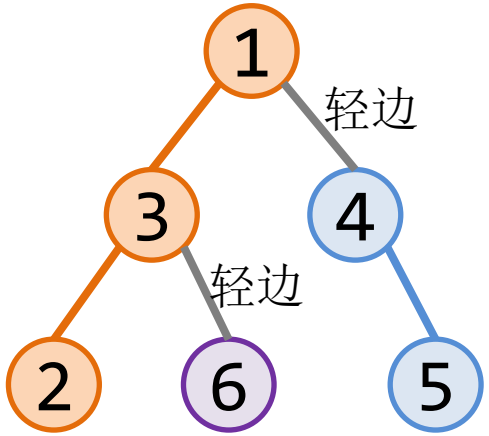
# 重边

每个节点和它“重儿子”之间的连边称作“重边”



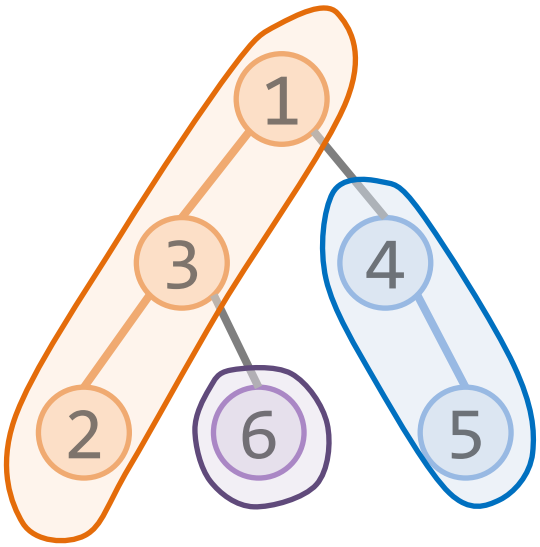
# 轻边

每个节点和它“轻儿子”之间的连边称作“轻边”



# 重链

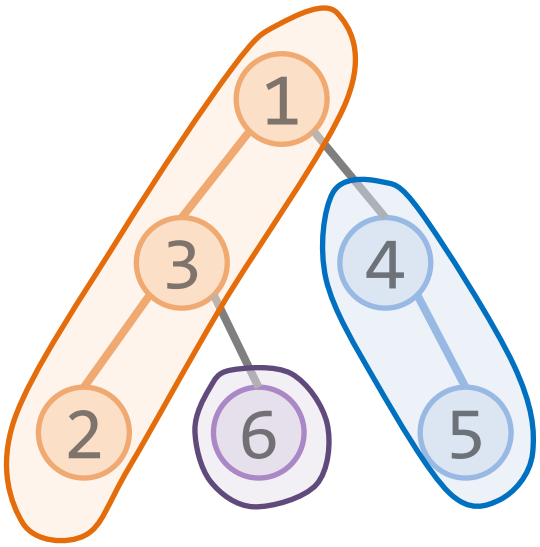
将相邻的“重边”连接起来，就形成了“重链”




注意：还有种特殊的“重链”只包含1个节点

# 重链

将相邻的“重边”连接起来，就形成了“重链”




top[u]表示u所在重链的链头(最高点)

| u=1 | u=2 | u=3 | u=4 | u=5 | u=6 |
|---|---|---|---|---|---|
| top[u]= | | | | | |

# 重链剖分的实现

1. dfs_son()

son[u]表示u的重儿子节点编号

从下往上计算

依赖sz[]

2. dfs_top()

top[u]表示u所在重链的链头(最高点)

从上往下计算

需要跑
两次DFS

第1次DFS

从下往上

算son[]

```
11 void dfs_son(int u,int fa){
12     sz[u]=1;
13     son[u]=0;
14     for(int i=hd[u];i;i=e[i].nxt){
15         int v=e[i].to;
16         if(v==fa) continue;
17         dfs_son(v,u);
18
19         if(sz[son[u]]<sz[v]||
20            sz[son[u]]==sz[v]&&v<son[u])
21
22     }
23 }
```

第2次DFS

从上往下

算top[]

```
24 void dfs_top(int u,int fa){
25     if(son[fa]==u)
26     else top[u]=u;
27     for(int i=hd[u];i;i=e[i].nxt){
28         int v=e[i].to;
29         if(v==fa) continue;
30
31     }
32 }
```

# 重链剖分

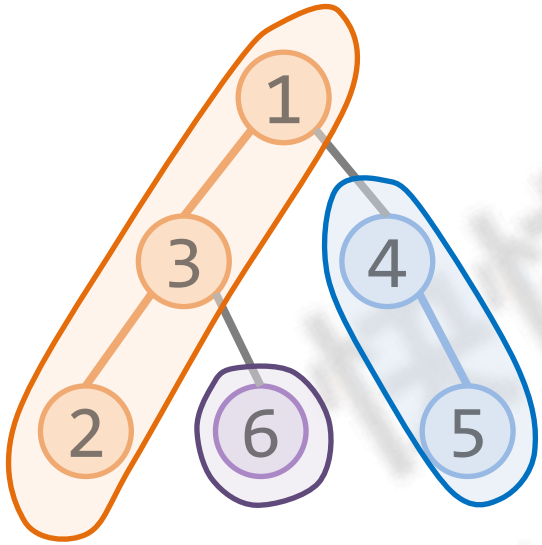
## 性质讨论

# 重链数

**重链数量==叶节点数量**

## 证明

| | |
|---|---|
| 1 | 每个叶节点一定是某条重链的链尾 |
| 2 | 每条重链的结尾一定是某个叶节点 |



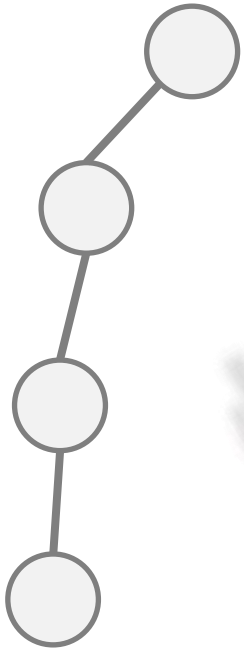
# 重链数

## 重链数量==叶节点数量

对于n个节点的树，选什么形态时
重链数量最少?

叶节点数量最少?

单链条



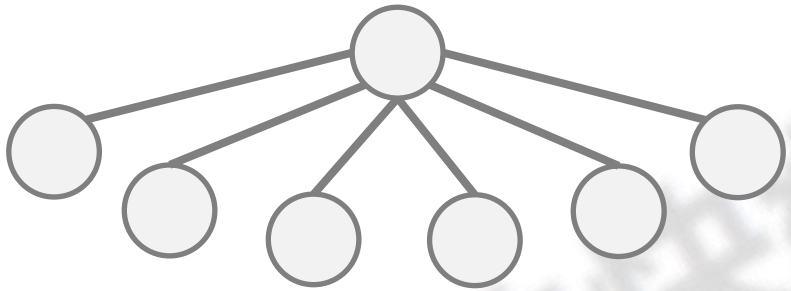
# 重链数

## 重链数量==叶节点数量

对于n个节点的树，选什么形态时
重链数量最多?

叶节点数量最多?

菊花图



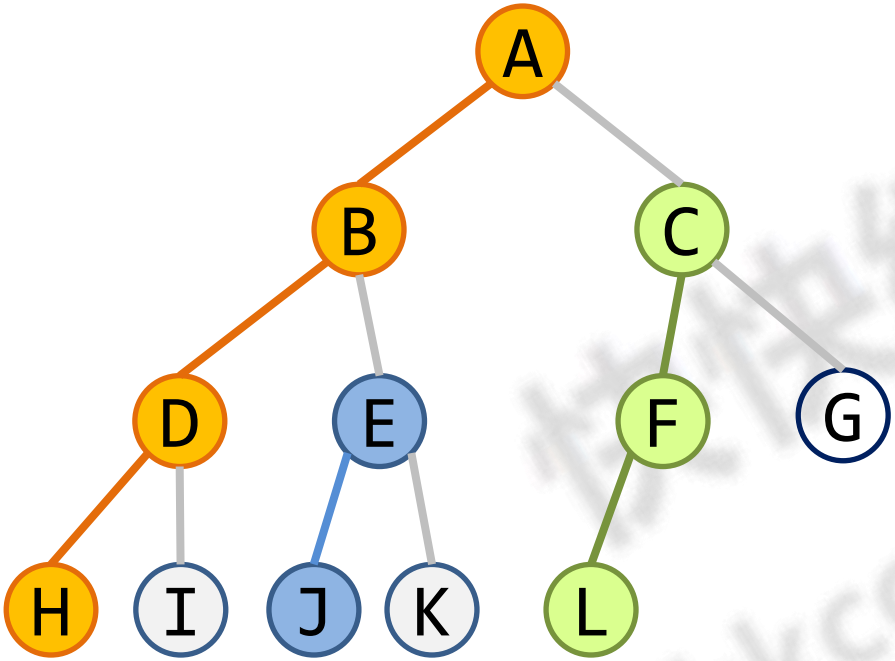
# 重链数

## 重链数量==叶节点数量

对于n个节点的二叉树，选什么形态时重链数量最多?

完全二叉树

叶节点数量最多?

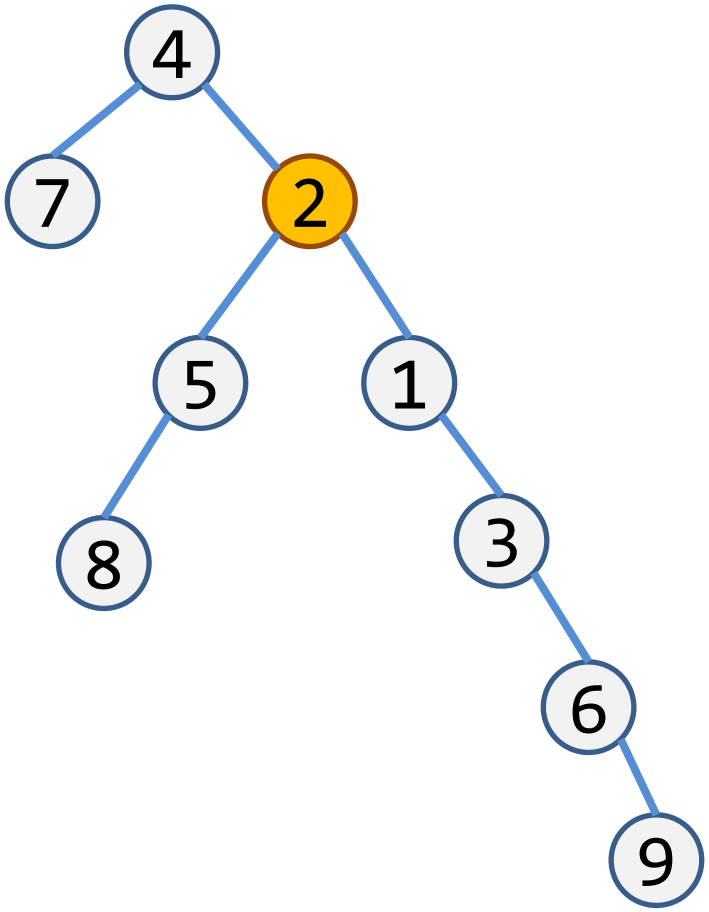


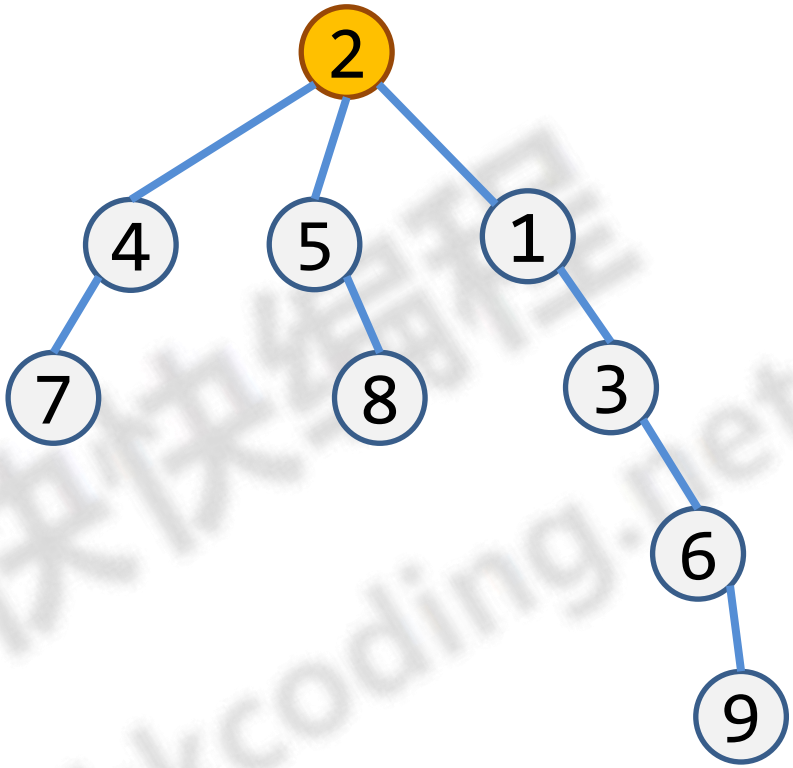

O(n)等级

# 重心

取重心为根时
最大子树最小

树的重心一定在
根出发的重链上



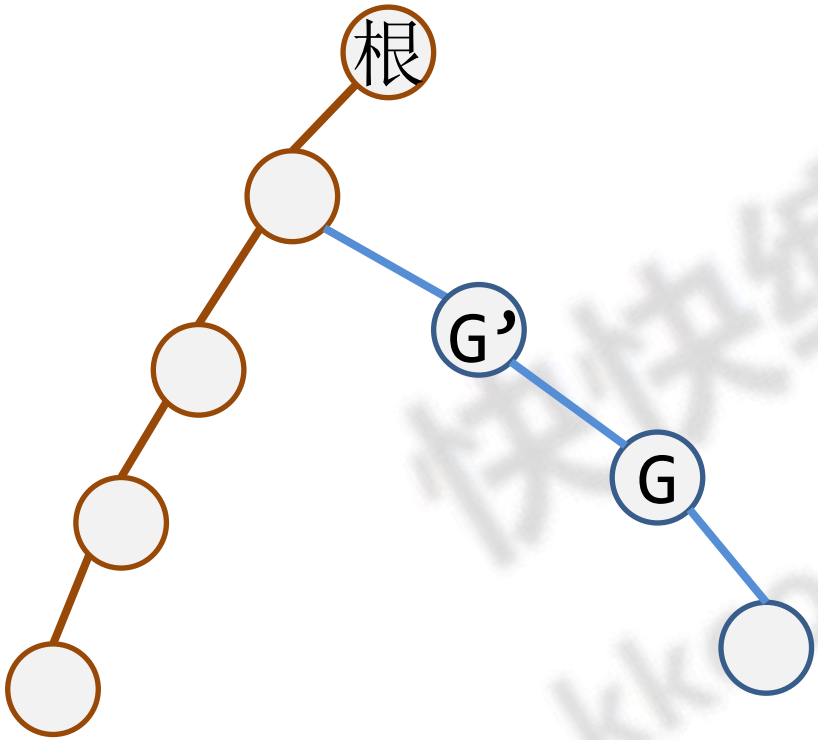



| 重心 |
| --- |
| 取重心为根时<br>最大子树最小 |

# 树的重心一定在根出发的重链上

| 证明 |
| --- |
| 反证法 |
| 局部调整 |

| 假设重心G不在根出发的重链P上 |
| --- |
| 将G改成G的邻居G’,使其更靠近重链P |
| 取G’为根时,最大子树变小了 |

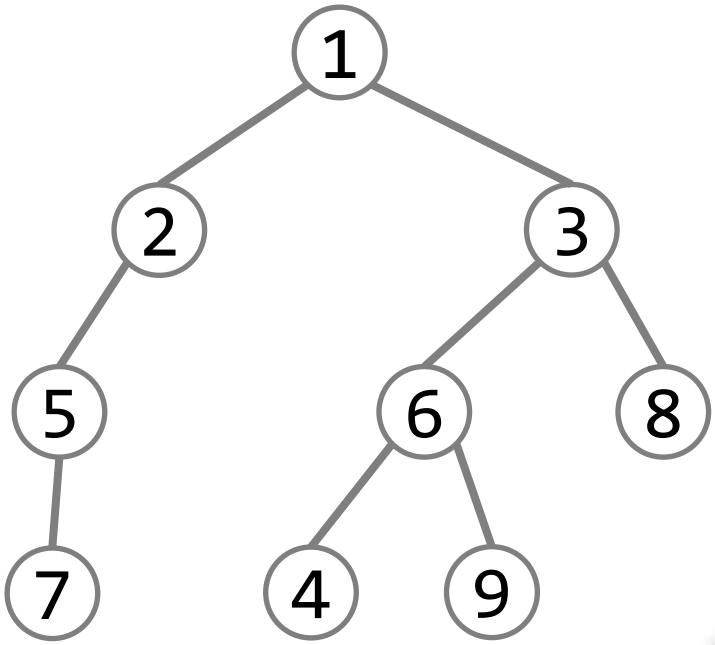


# 快快编程2562

SFS序列化

size
first
search

重儿子先行DFS




| SFS序列 | 136498257 | |
|---|---|---|
| | 每条重链在SFS序列中都连续 | 区间 |

| 每个子树在SFS序列中都连续 | 区间 |
|---|---|

第1次DFS

从下往上

算son[]

```
12 void dfs_son(int u,int fa){
13     sz[u]=1;
14     son[u]=0;
15     for(int i=hd[u];i;i=e[i].nxt){
16         int v=e[i].to;
17         if(v==fa) continue;
18         dfs_son(v,u);
19         sz[u]+=sz[v];
20         if(
21                                    )
22             son[u]=v;
23     }
24 }
```

第2次DFS

从上往下

算top[]

```
25 void dfs_top(int u,int fa){
26     ++timer;
27     dfn[u]=
28     id[timer]=u;
29     if(son[fa]==u) top[u]=top[fa];
30     else top[u]=u;
31     if(          ) return;
32     
33     for(int i=hd[u];i;i=e[i].nxt){
34         int v=e[i].to;
35         if(                    ) continue;
36         dfs_top(v,u);
37     }
38 }
```

# 重链剖分

## 性质讨论

任意点v到根路径上

轻边

重边

连续重边合并成重链

重链不可以连续出现
重链之间一定是轻边

以上两种情况混合出现

路径上轻边数

树上共n个节点

# 任意点v到根路径上轻边数不超过log(n)

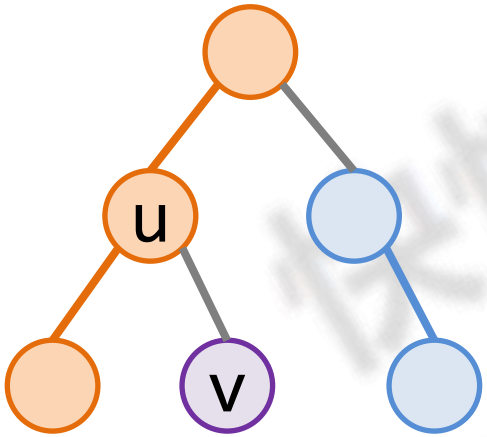
证明

爬树：不断将v改到p[v]

若v是轻儿子（v到p[v]是轻边）

p[v]子树大小 > v子树大小*2

从v到根，每过1条轻边，子树大小至少翻倍



路径上重链数

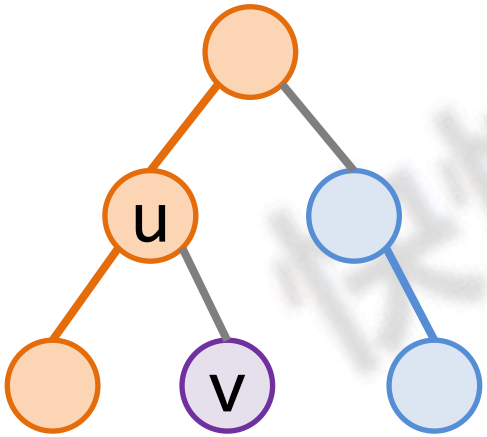
树上共n个节点

# 任意点v到根路径上重链数不超过log(n)

证明

从v到根，任意2条重链间一定有轻边

任意点v到根路径上轻边数不超过log(n)



路径上重链数

树上共n个节点

任意u到v的路径上
重链数不超过2*log(n)

应用

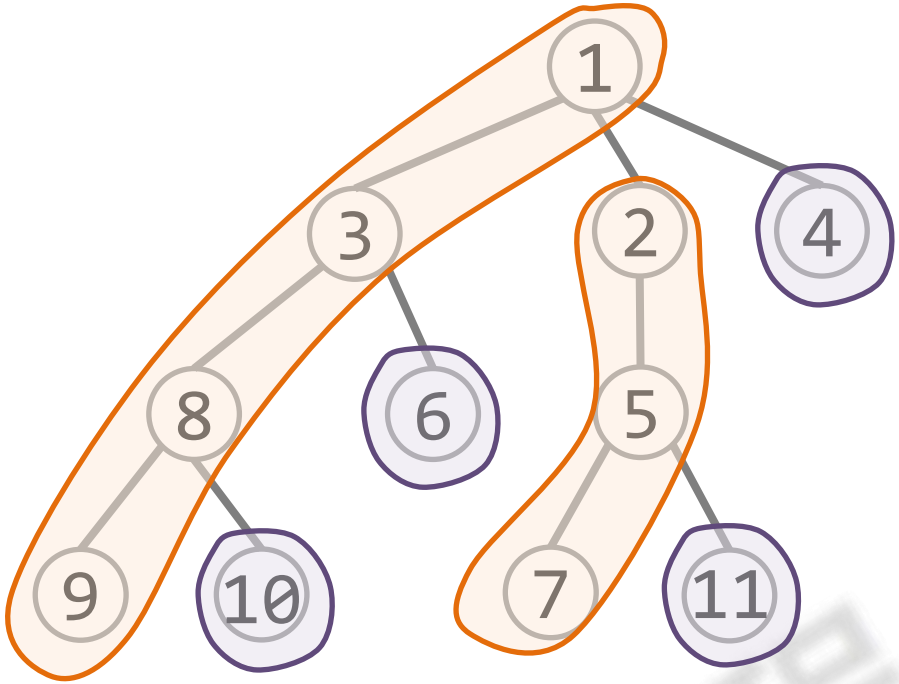
将树上路径拆分成O(logn)条重链

每条重链在SFS序列中都是区间

1条路径问询可以拆分成
O(logn)个区间问询

# 快快编程2563

| 6个问询： | |
|---|---|
| 9 1 | 1 |
| 10 9 | 2 |
| 9 4 | 2 |
| 7 2 | 1 |
| 7 4 | 3 |
| 10 11 | 4 |

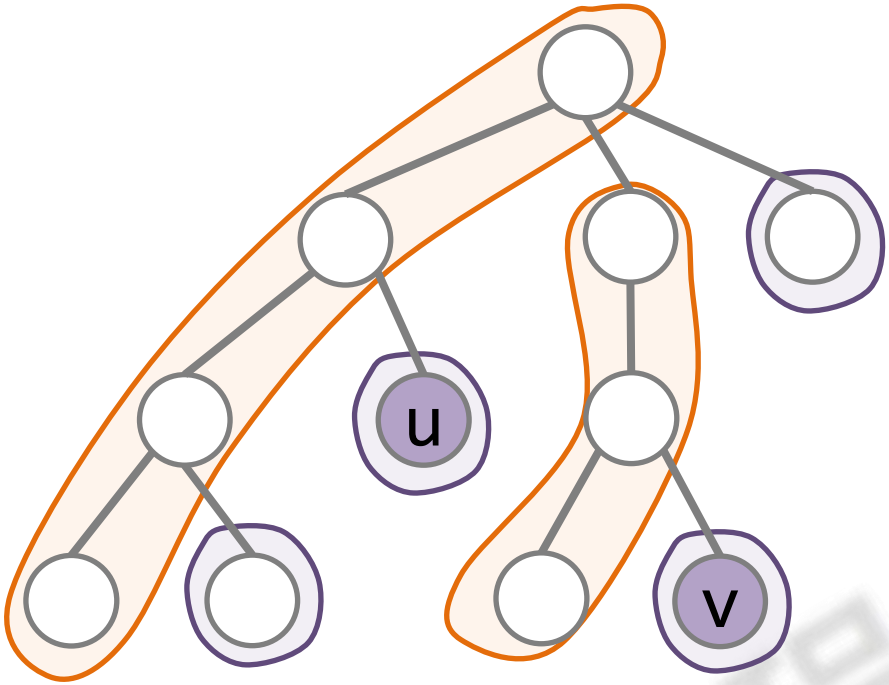



手算后识别出算法

爬树：从u和v两点逐步会合交替往上跳

| 若u和v在同一条重链,答案为1 | 如何判断? | top[u]==top[v] |
|---|---|---|
| 若u和v不在同一条重链 | 如何爬? | 比较top[u]和top[v]的深度 |

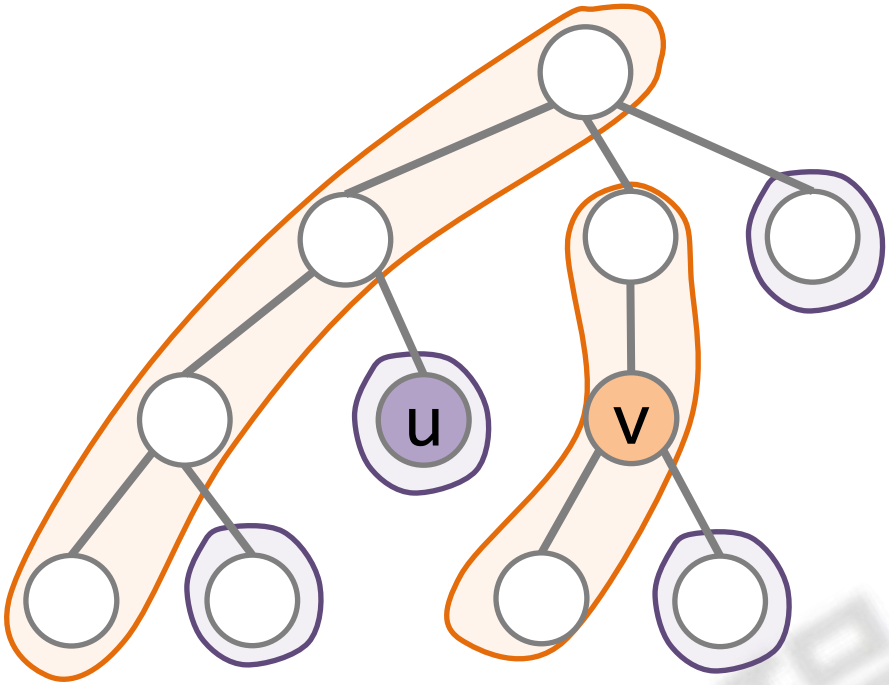
选链头较深节点
往上跳到
链头的父节点




| 若d[top[u]] < d[top[v]] | v 跳到 p[top[v]] |
|---|---|

| 若u和v不在同一条重链 | 如何爬? | 比较top[u]和top[v]的深度 |
|---|---|---|

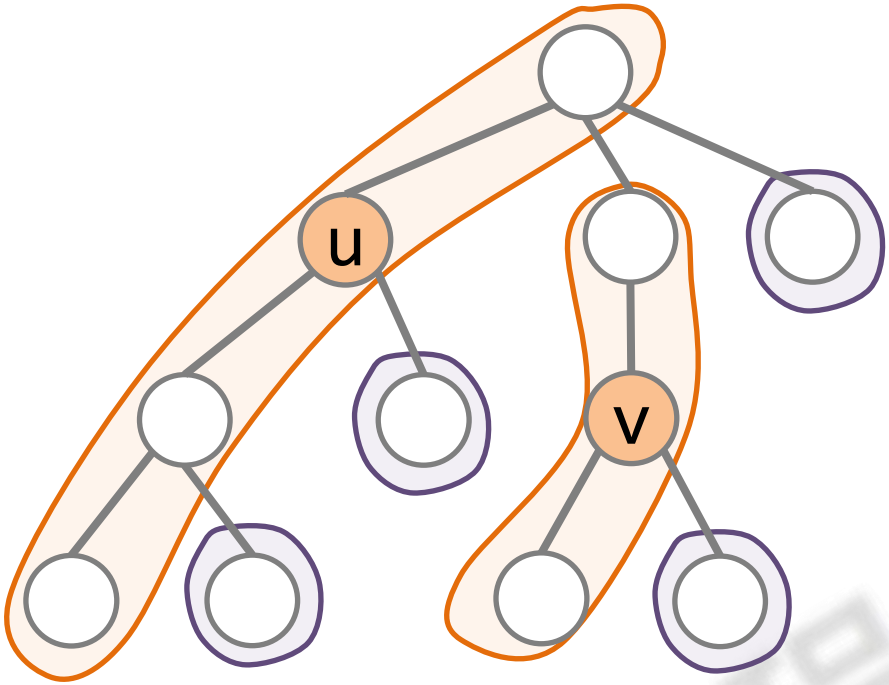
选链头较深节点
往上跳到
链头的父节点




| 若d[top[u]] >= d[top[v]] | u 跳到 p[top[u]] |
|---|---|

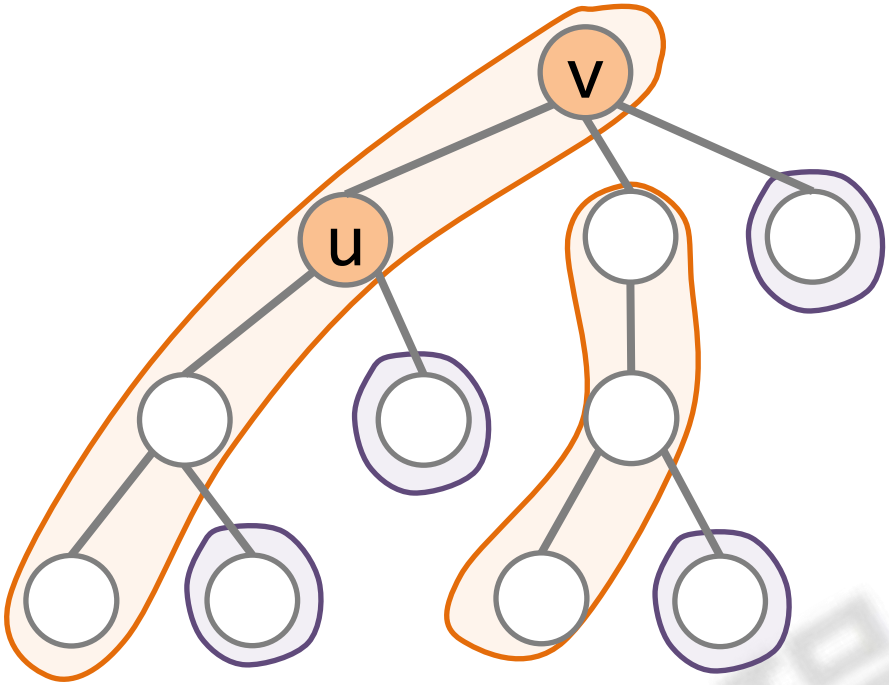
| 若u和v不在同一条重链 | 如何爬? | 比较top[u]和top[v]的深度 |
|---|---|---|

选链头较深节点
往上跳到
链头的父节点




| 若d[top[u]] < d[top[v]] | v 跳到 p[top[v]] |
|---|---|

| 若u和v不在同一条重链 | 如何爬? | 比较top[u]和top[v]的深度 |
|---|---|---|




若u和v在同一条重链,停止爬树

top[u]==top[v]

从u到v的路径共经过几条不同的重链

```
35 int query(int u,int v){
36     int cnt=1;
37     while(                    ){
38         if(d[top[u]]<d[top[v]])
39             
40         else
41             
42         ++cnt;
43     }
44     return cnt;
45 }
```

[[ ]]嵌套

[[ ]]嵌套

[[ ]]嵌套

关键句都包含top[]不要遗漏

# 重链剖分

O(logn)求解LCA

# 求u和v最近公共祖先

```
34 int lca(int u,int v){
35     while(top[u]!=top[v]){
36         if(                              )
37             u=p[top[u]];
38         else
39             v=p[top[v]];
40     }
41     return           ?u:v;
42 }
```

关键句都包含top[]不要遗漏

# 快快编程作业

| |
|---|
| 2561,2562,2563 |
| 整理详细证明发班级群 |

| | |
|---|---|
| 拓展题 | 1826,1887,974 |