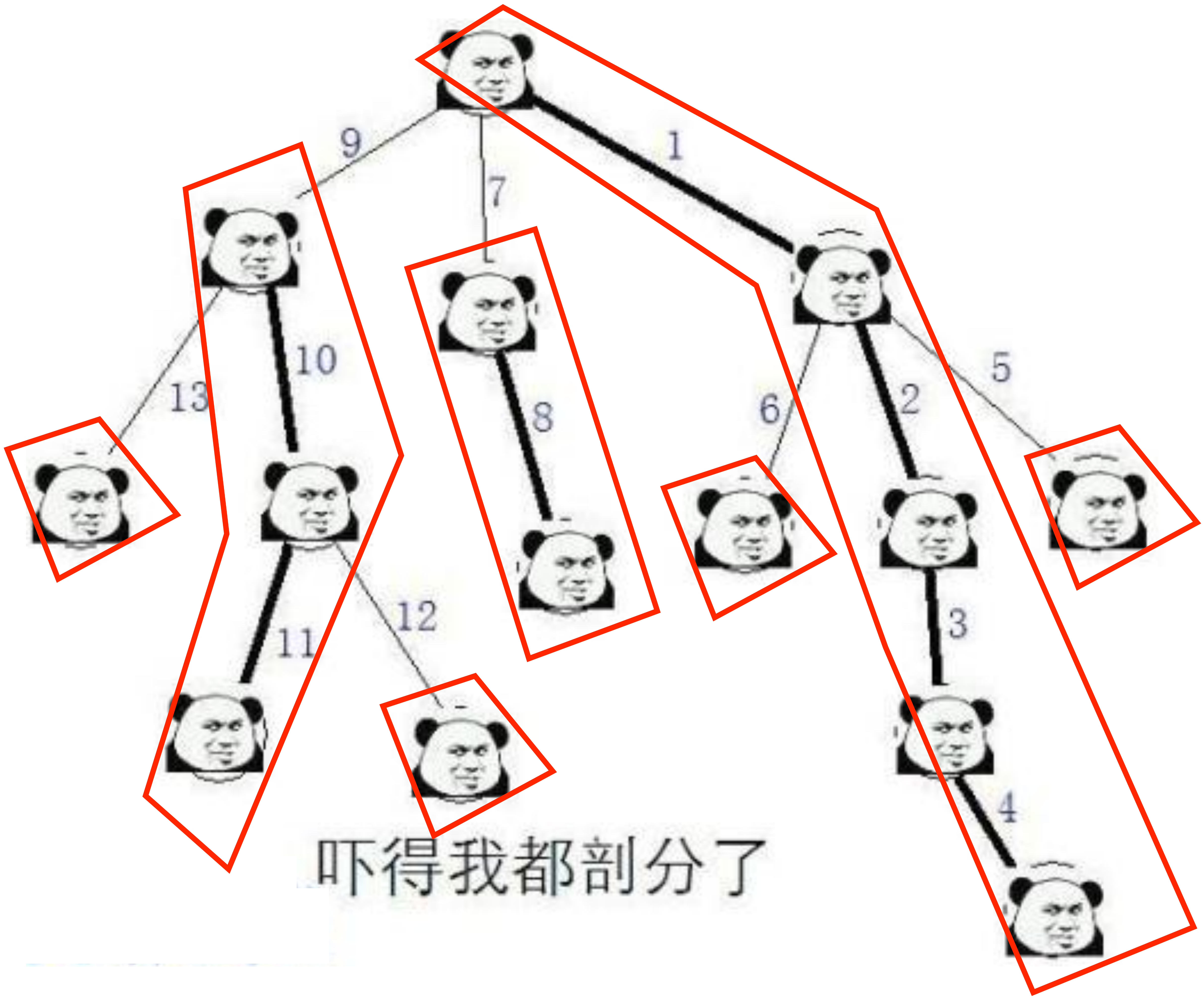
CS106 名字无所谓了

树链剖分

省选算法三大算法门类

网络流	最大流、费用流、最小割…
高级字符串	KMP、自动机、回文树…
高级数据结构	树套树、链剖、主席树、LCT…



9
7
1
10
13
8
6
2
5
12
11
3
4
吓得我都剖分了

P763 二巨头

破题

→ **离线问询**

求树上的最短路径

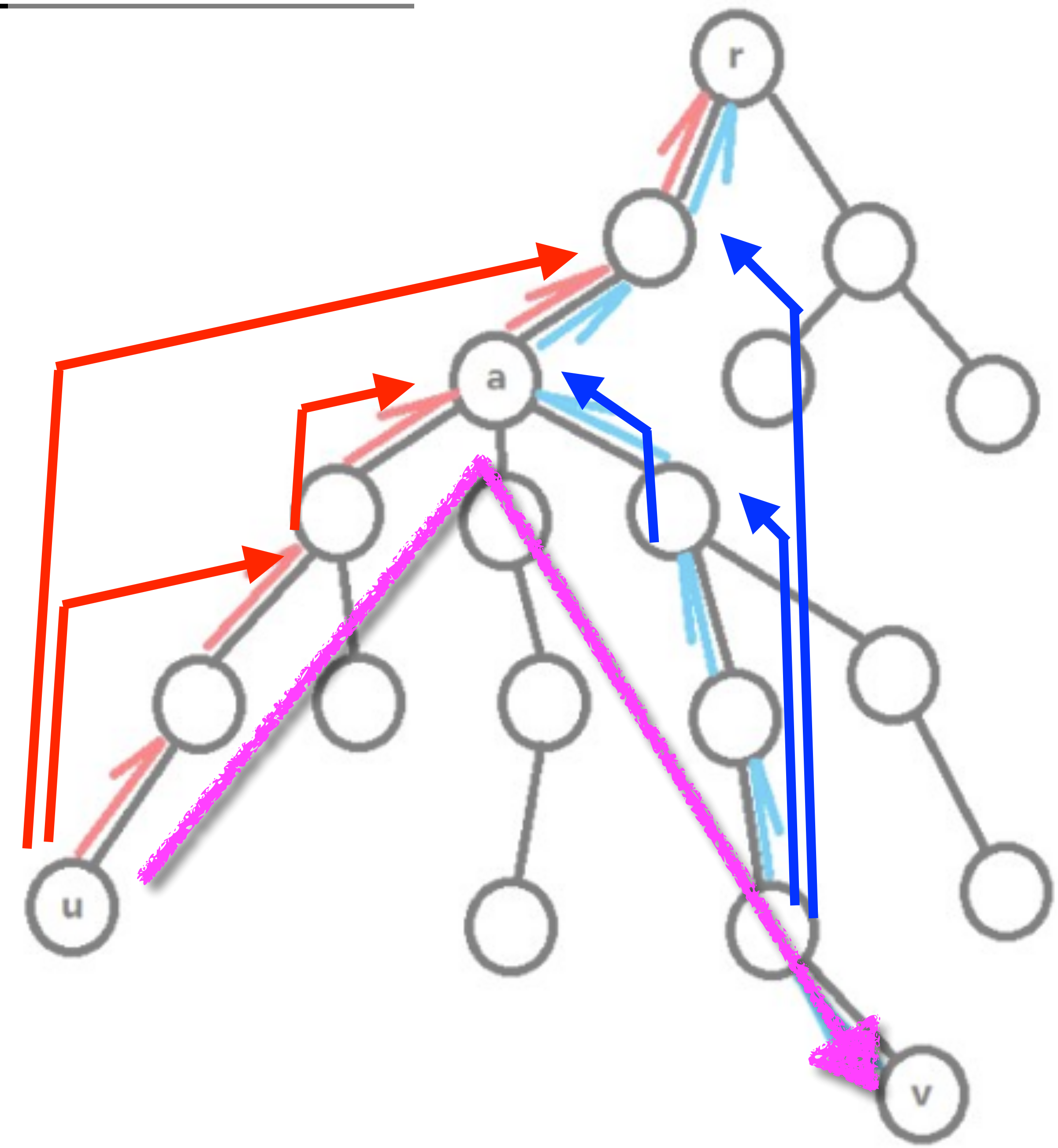
长度=**d(u)+d(v)-2d(lca(u,v))**

→ **计算LCA**

先单侧走，再一起走

走过头就不走，没过头就走

每次步长减半



树形DP（预计算）

```
void dfs(int x) {
    vis[x]=1;
    for(int i=1;i<=H && (1<<i)<=d[x];i++)
        fa[x][i]=fa[fa[x][i-1]][i-1];
    for(int i=0,y;i<es[x].size();i++) {
        y=es[x][i];
        if (vis[y]) continue;
        d[y]=d[x]+1,fa[y][0]=x;
        dfs(y);
    }
}
```

倍增索引
（自顶向下）

```
int lca(int x,int y) {
    if (d[x]<d[y]) swap(x,y);
    int dt=d[x]-d[y];
    for(int i=0;i<=H;i++)
        if ((1<<i)&dt) x=fa[x][i];
    for(int i=H;i>=0;i--)
        if (fa[x][i]!=fa[y][i])
            x=fa[x][i],y=fa[y][i];
    return x==y ? x :fa[x][0];
}
```

单侧走

一起走

求LCA的各种方法

→ **此处离线指问询可以交换顺序**

→ **值不变化如何强制在线?**

后一次问询的参数与前一次问询的结果有关

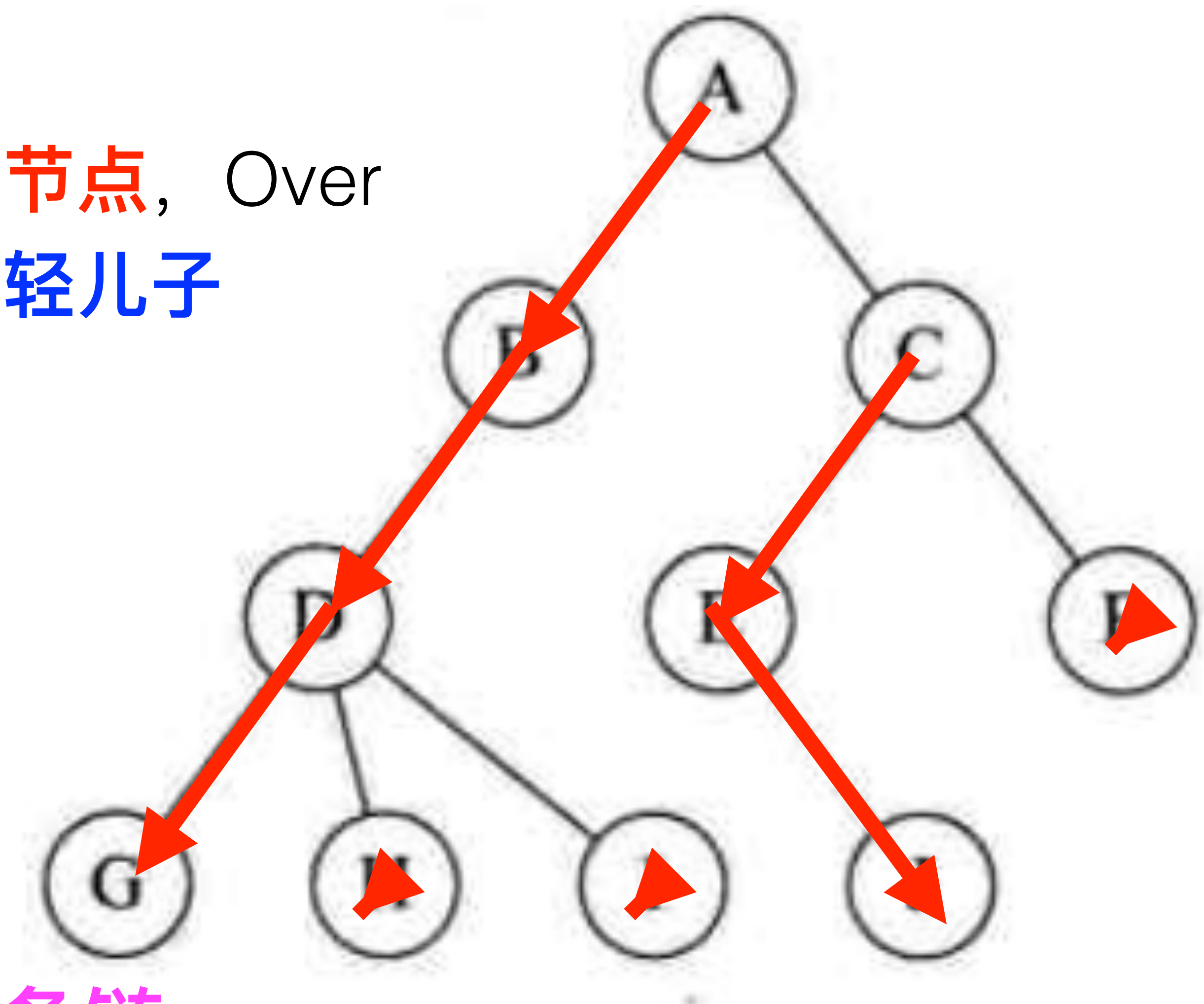
	场景	预处理	均摊查询	空间
倍增	在线	O(nlogn)	O(logn)	O(nlogn)
Tarjan	离线	O(n)	O(1)	O(n)
树剖	在线	O(n)	O(logn)	O(n)
ST表	在线	O(nlogn)	O(logn)	O(nlogn)

重链剖分

→ **将树分成若干条链（一维）**

每个节点指向（任意）**最大子树的子节点**，Over

最大子树称为**重儿子**，其他子树称为**轻儿子**



→ **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）

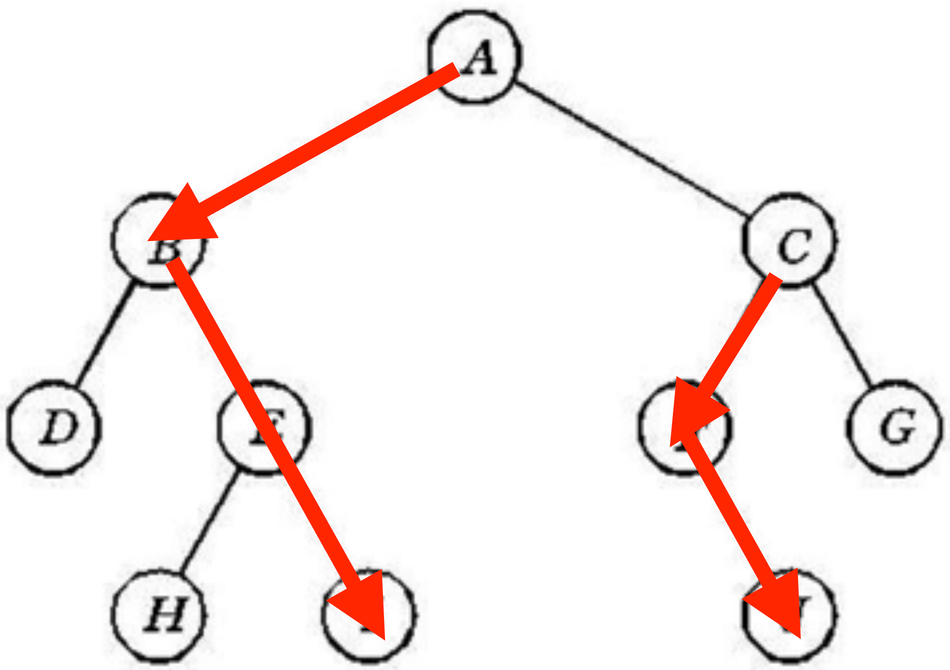
4.链的**顶端都是轻儿子**（根除外）

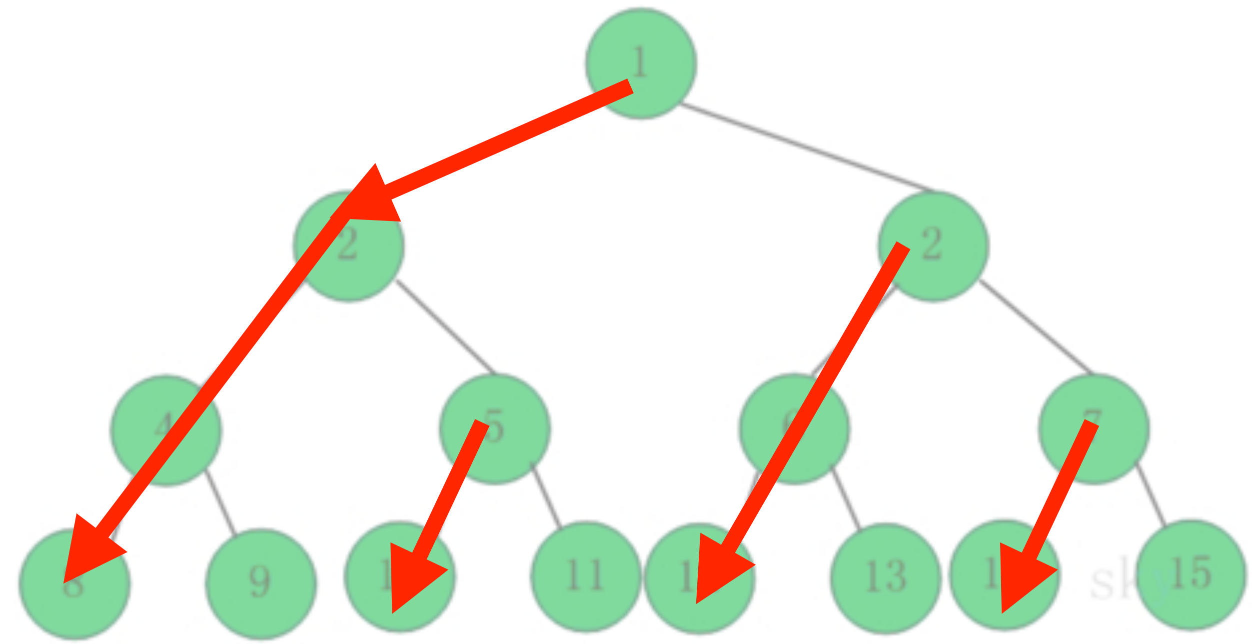
5.链的底端都是叶节点

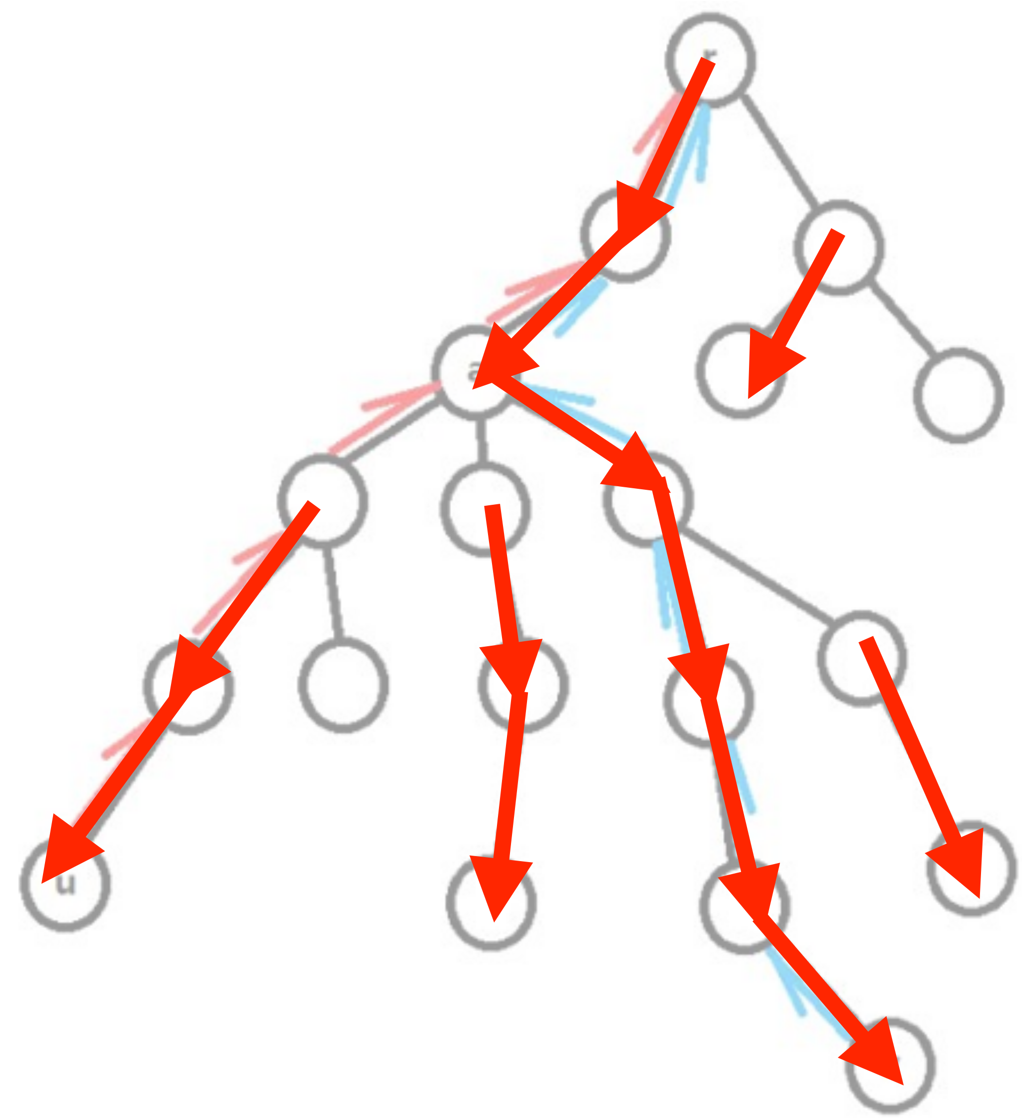
6.从任意点到根，至多经过logn条链

证明： 跨链必须经过轻儿子

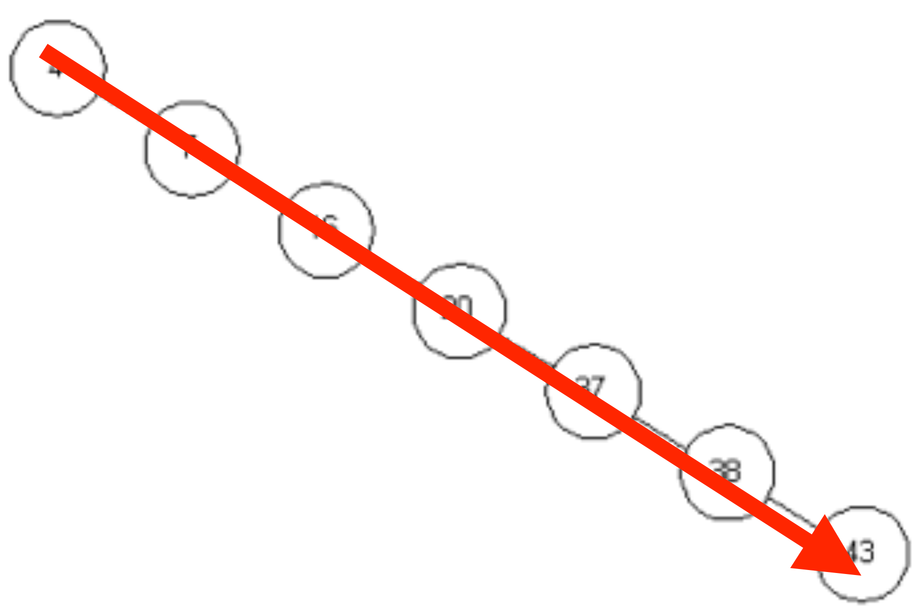
每过一个轻儿子，当前子树节点数至少加倍，证毕







我剖！



LCA

→ **对节点x,y**

如x,y不在一条链上（top[x]!=top[y]）

设d[top[x]]≥d[top[y]]（否则调换x,y）

则d[top[x]]>d[lca]**（较低的链顶低于LCA）**

→ **证明**

如d[top[x]]≤d[lca]，则lca与x同链

从y沿链到top[y]，不可能经过lca

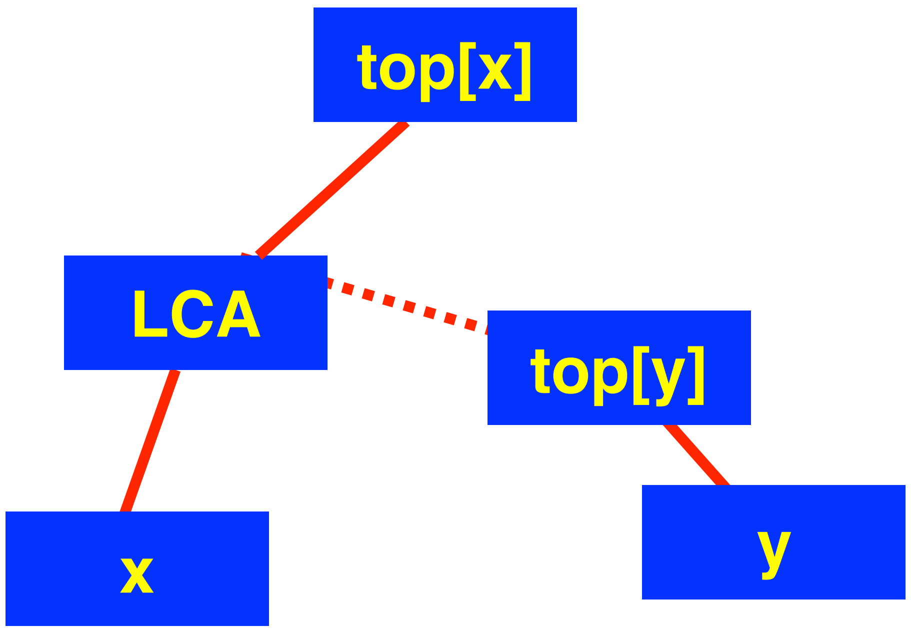
则d[top[y]]>d[lca(x,y)]≥d[top[x]]，矛盾

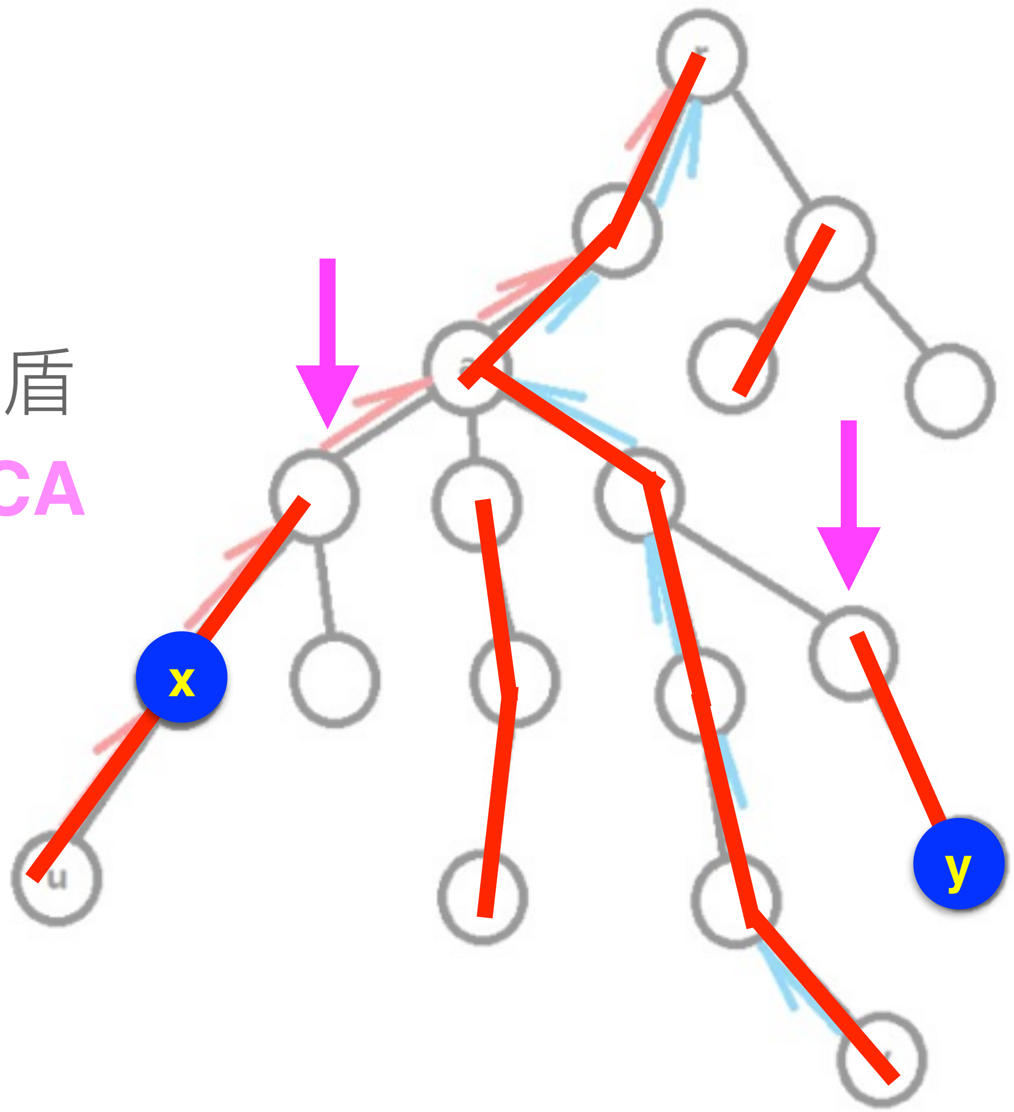
→ **结论：走到较低的链顶，不会越过LCA**

→ **重复以上过程**

直到x,y在同一链上

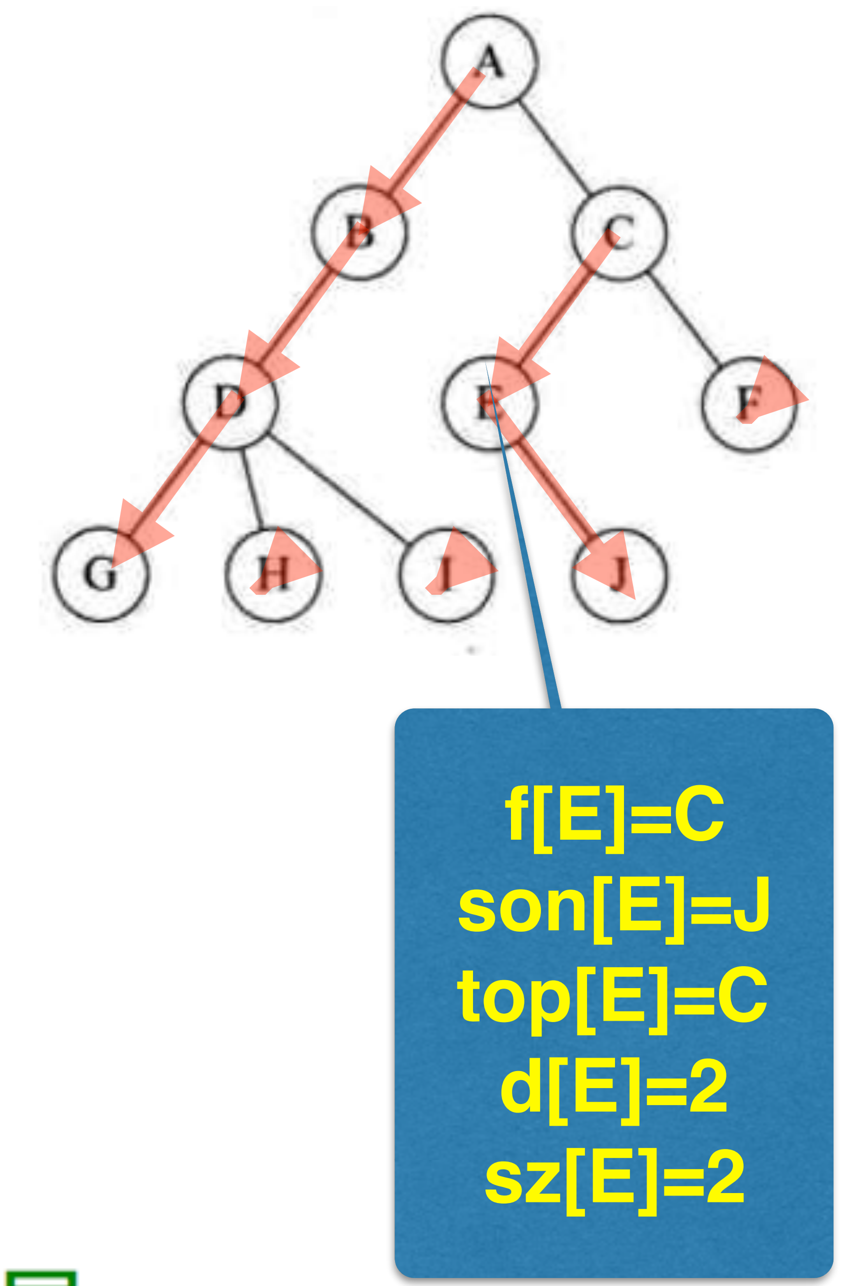
此时x,y中深度小的就是LCA





怎么存

```
int f[N], // 父节点
    son[N], // 重儿子
    top[N], // 重链顶端
    d[N], // 深度
    sz[N]; // 子树大小

int hd[N],cnt;  // 前向星
struct edge {int t,nxt;} es[N<<1];
```



```
void dfs1(int u,int fa){
    f[u]=fa,d[u]=d[fa]+1,sz[u]=1;
    for (int i=hd[u],v;i;i=es[i].nxt) {
        if ((v=es[i].t)==f[u]) continue;
        dfs1(v,u);
        sz[u]+=sz[v];
        if [redacted] son[u]=v;
    }
}
```

打擂台求重儿子

计算f,d,sz,son（自底向上）

```
void dfs2(int x,int tp){
    top[x]=tp;
    if (son[x]) dfs2(son[x],tp);
    for (int i=hd[x],v;i;i=es[i].nxt)
        if ((v=es[i].t)!=son[x] && v!=f[x])
            [redacted];
}
```

计算top（自顶向下）

递归重儿子

递归轻儿子

```
int lca(int x,int y){
    for (;top[x]!=top[y];x=f[top[x]])
        if (d[top[x]]<d[top[y]]) swap(x,y);
    return d[x]<d[y] ? x:y;
}
```

较低的链顶低于LCA

走到较低的链顶，不会越过LCA

```
for(int i=1,u,v;i<n;i++)
    scanf("%d%d",&u,&v),add(u,v),add(v,u);
dfs1(1,0),dfs2(1,1);
for(int i=1,x,y,z;i<=q;i++) {
    scanf("%d%d%d",&x,&y,&z);
    printf("%d\n",d[x]+d[y]-2*d[lca(x,y)]);
}
```

P976 公司破事3

建模

→ **破题（树上的在线问询）**

求**树上某路径**的点权总和（段查询）

更新树上某路径的点权（段更新）

→ **ST表?**

不支持更新

→ **线段树?**

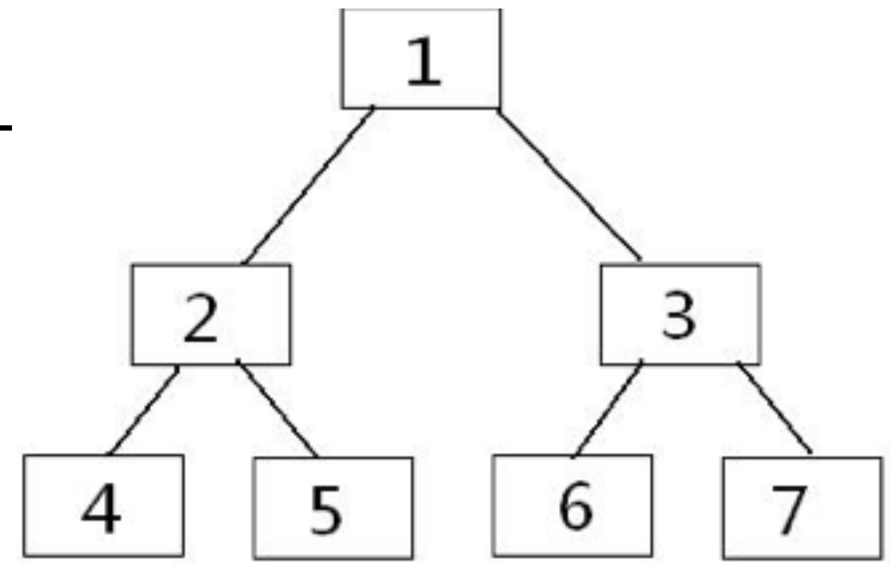
线段树是**链上**的分级索引

→ **把树化归成链（节点编号）** 🤔

1.堆式编号

2.先序遍历（dfn）

3.欧拉遍历



1 2 4 5 3 6 7

+1 +2 +4 -4 +5 -5 -2
+3 +6 -6 +7 -7 -3 -1

树剖的编号

→ **先序遍历基础上**

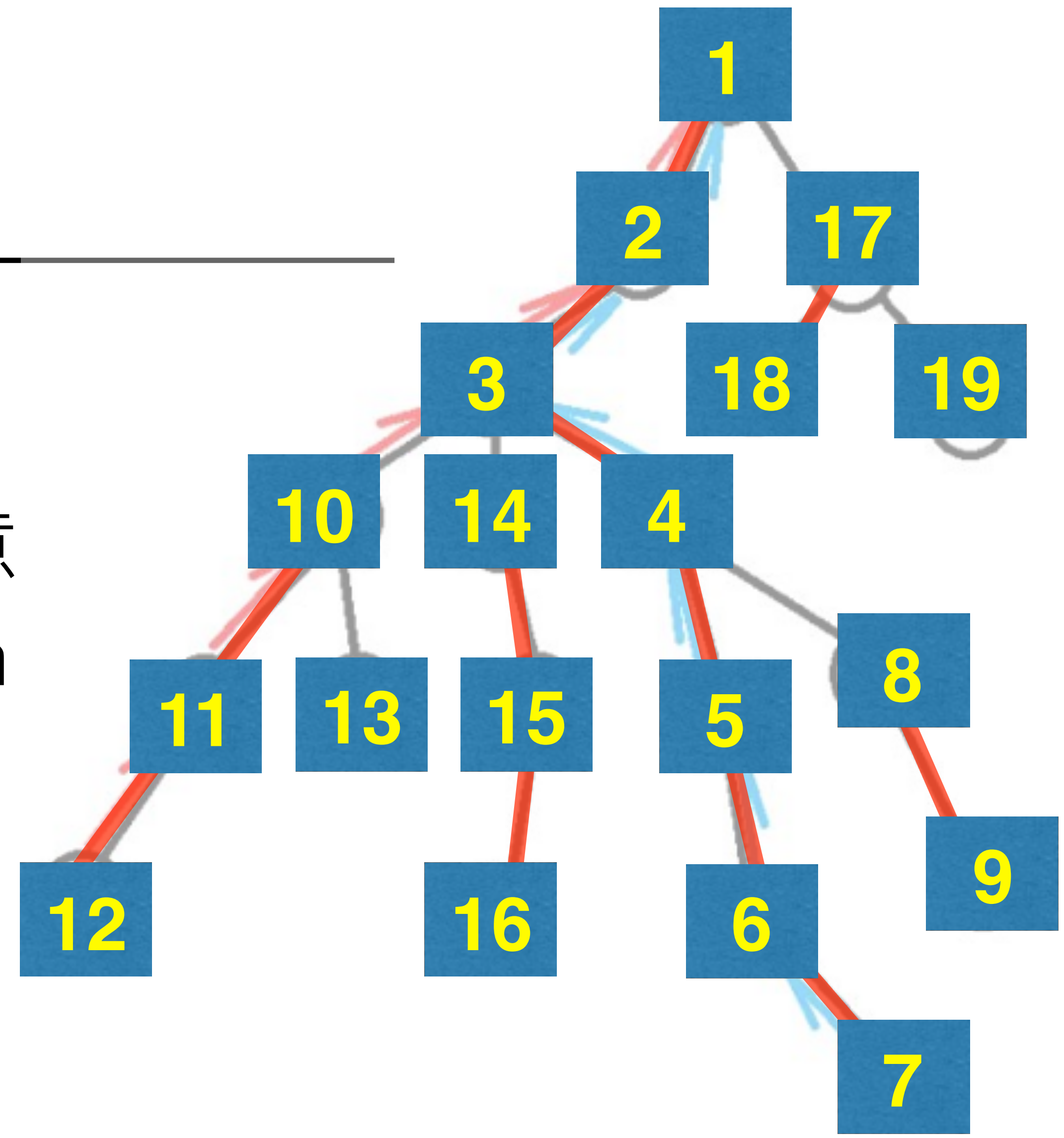
优先遍历重儿子，轻儿子顺序随意

得到树剖意义下的dfn，仍记为dfn

→ **特征**

每个子树都是**连续**的一段

每条链都是**连续**的一段




```
void dfs2(ll u,ll tp) {
    top[u]=tp,dfn[u]=++cnt,rk[cnt]=u;
    if (son[u]) dfs2(son[u],tp);
    for (ll i=hd[u],v;i;i=e[i].nxt)
        if ((v=e[i].t)!=son[u]&&v!=f[u])
            dfs2(v,v);
}
```

rk[i]=dfn为i的节点原始编号

树上的线段树

→ **在dfn序列上建线段树**

相当于把**各链首尾相接**

→ **特征**

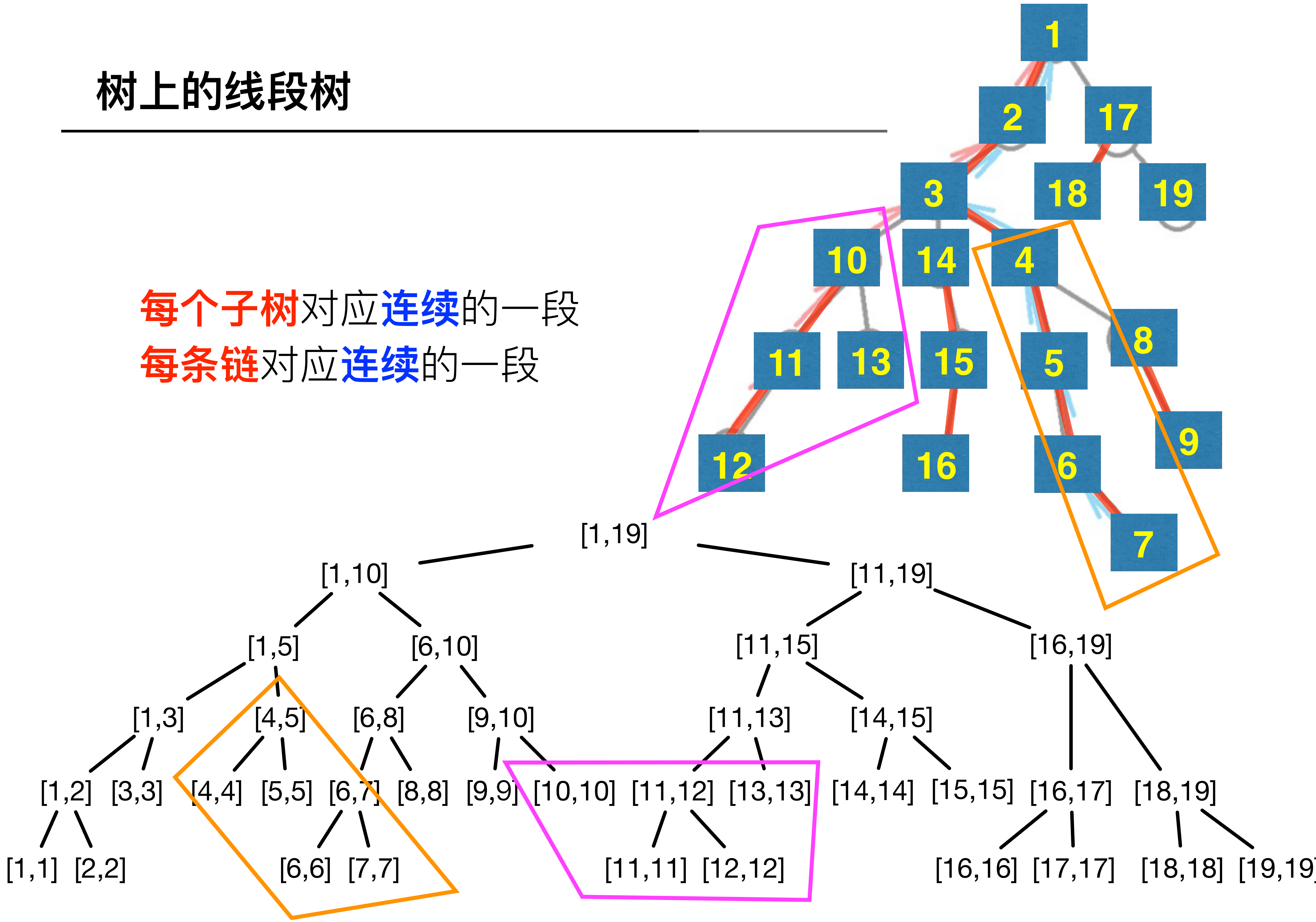
每个子树都是**连续**的一段

每条链都是**连续**的一段

```
void build(ll l,ll r,ll x) {
    tr[x].l=l,tr[x].r=r;
    if (l==r) { tr[x].sum=a[rk[l]]; return; }
    ll m=(l+r)>>1;
    build(l,m,x<<1);
    build(m+1,r,x<<1|1);
    tr[x].sum=(tr[x<<1].sum+tr[x<<1|1].sum)%p;
}
```

树上的线段树

每个子树对应连续的一段
每条链对应连续的一段

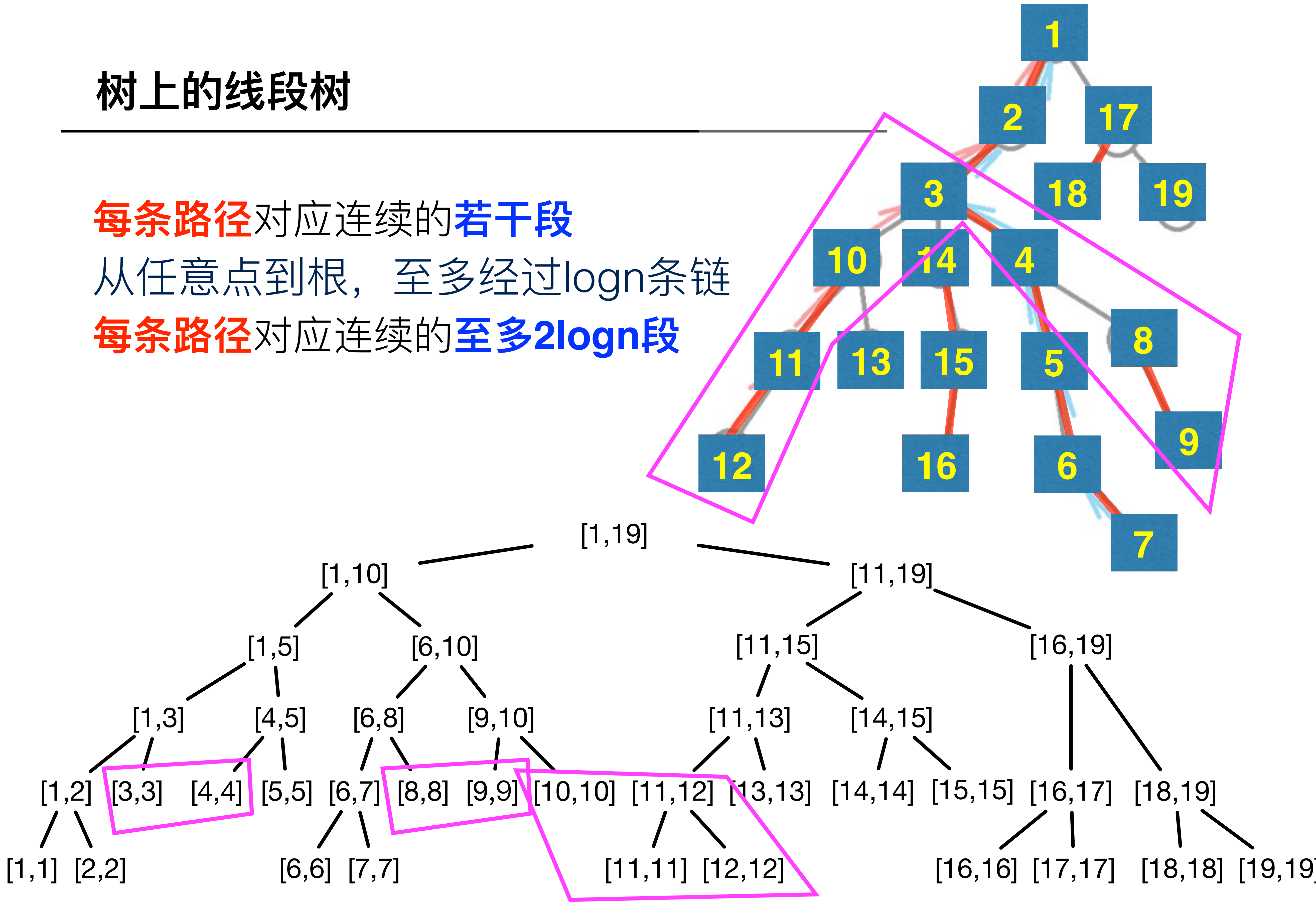


树上的线段树

每条路径对应连续的**若干段**

从任意点到根，至多经过logn条链

每条路径对应连续的**至多2logn段**



路径操作

→ **每条路径对应连续的至多2logn段**

做O(logn)次普通段更新/段查询即可

每次查询到较低的链顶

直到两端在同一链上

```
ll queryPath(ll x,ll y,ll ans=0) {
    for (;top[x]!=top[y];x=f[top[x]]) {
        if (d[top[x]]<d[top[y]]) swap(x,y);
        ans=(ans+query(dfn[top[x]],dfn[x],1))%p;
    }
    if (d[x]>d[y]) swap(x,y);
    return (ans+query(dfn[x],dfn[y],1))%p;
}
```

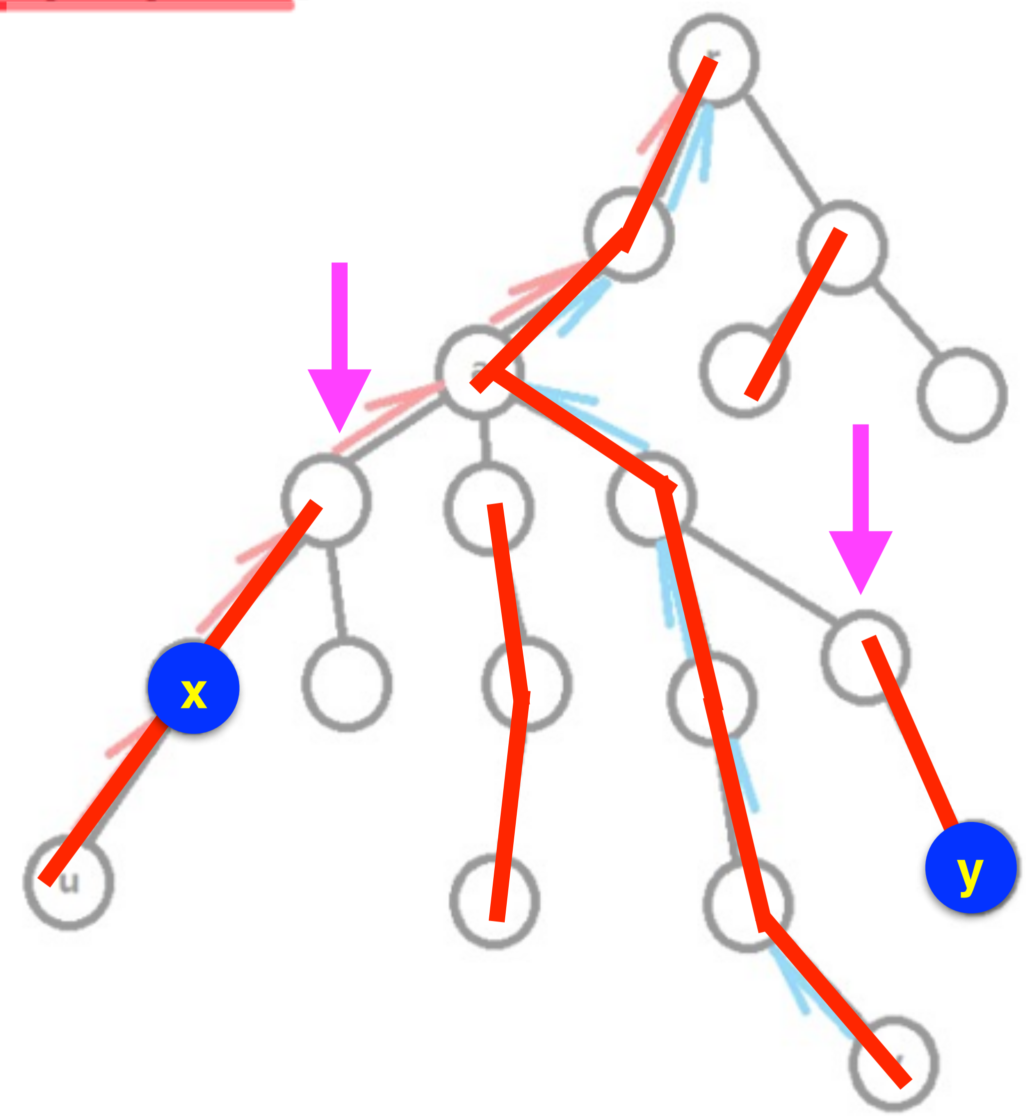
再查一把最后那条链

```
void pushdown(ll x) {
    tr[x<<1].sum=(tr[x<<1].sum+tr[x].tag*len(x<<1))%p;
    tr[x<<1].tag=(tr[x<<1].tag+tr[x].tag)%p;
    tr[x<<1|1].sum=(tr[x<<1|1].sum+tr[x].tag*len(x<<1|1))%p;
    tr[x<<1|1].tag=(tr[x<<1|1].tag+tr[x].tag)%p;
    tr[x].tag=0;
}

void update(ll l,ll r,ll c,ll x) { // 段更新
    if (l>tr[x].r || r<tr[x].l) return;
    if (l<=tr[x].l && r>=tr[x].r) {
        tr[x].sum=(tr[x].sum+c*len(x))%p;
        tr[x].tag=(tr[x].tag+c)%p;
        return;
    }
    pushdown(x);
    update(l,r,c,x<<1),update(l,r,c,x<<1|1);
    tr[x].sum=(tr[x<<1].sum+tr[x<<1|1].sum)%p;
}

ll query(ll l,ll r,ll x) { // 段查询
    if (l>tr[x].r || r<tr[x].l) return 0;
    if (l<=tr[x].l && r>=tr[x].r) return tr[x].sum;
    pushdown(x);
    return (query(l,r,x<<1)+query(l,r,x<<1|1))%p;
}
```

你不该
看这页

```
void updatePath(ll x,ll y,ll c) {
    for (;top[x]!=top[y];x=f[top[x]]) {
        if (d[top[x]]<d[top[y]]) swap(x,y);
        update(dfn[top[x]],dfn[x],c,1);
    }
    if (d[x]>d[y]) swap(x,y);
    update(dfn[x],dfn[y],c,1);
}
```

→ 复杂度：O(m(logn)^2)



P974 公司破事2

建模

→ **破题（树上的在线问询）**

工作/休假状态记为点权0/1

更新某点到根节点的点权为1（路径更新）

更新某子树上点权为0（子树更新）

查询更新前后全树点权总和之差

→ **子树操作更简单**

子树对应连续的**一段**

```
for (int i=1,x,t;i<=q;i++) {
    scanf("%d%d",&t,&x),x++;
    int t1=tr[1].sum;
    if (t==1) {
        updatePath(1,x,1);
        printf("%d\n",abs(tr[1].sum-t1));
    } else if(t==2) {
        update(dfn[x],dfn[x]+sz[x]-1,0,1);
        printf("%d\n",abs(t1-tr[1].sum));
    }
}
```

路径更新到根

子树更新

一些细节

```
void pushdown(int x) {
    tr[x<<1].sum=len(x<<1)*tr[x].tag;
    tr[x<<1|1].sum=len(x<<1|1)*tr[x].tag;
    tr[x<<1].tag=tr[x<<1|1].tag=tr[x].tag;
    tr[x].tag=-1;
}

void update(int l,int r,int v,int x) {
    if (tr[x].r<l || tr[x].l>r) return;
    if (tr[x].r<=r && tr[x].l>=l) {
        tr[x].sum=(tr[x].r-tr[x].l+1)*v;
        tr[x].tag=v;
        return;
    }
    if (tr[x].tag!=-1) pushdown(x);
    update(l,r,v,x<<1); update(l,r,v,x<<1|1);
    tr[x].sum=tr[x<<1].sum+tr[x<<1|1].sum;
}
```

以-1表示无tag

段更新

注意v不是增量

作业

1.二巨头（P763，树链剖分）

2.公司破事2（P974，子树更新，路径更新）

可用暴力方法