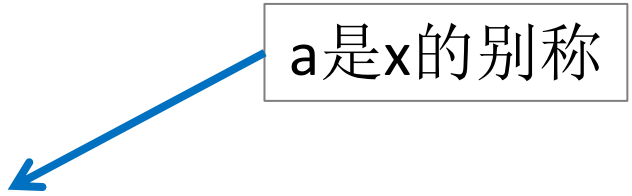
C++算法

引用
别名

预测结果

```
4    int x=1;
5    int&a=x;
6    cout<<a<<endl;
7    x=2;
8    cout<<a<<endl;
9    return 0;
```

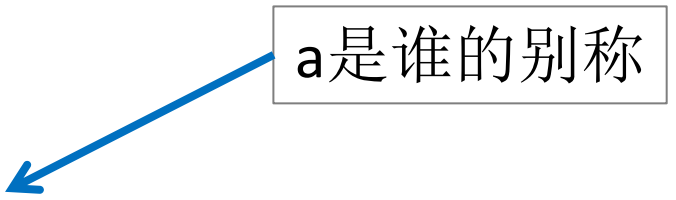


1

2

预测结果

```
4     int x=1;
5     int y=2;
6     int&a=(x>y?x:y);
7     cout<<a<<endl;
8     x*=2;
9     y*=2;
10    cout<<a<<endl;
```



2

4

快快编程1887

树上两种操作

1. 点到根路径和查询
2. 点更新

树的序列化
搭配数据结构

重链剖分

路径涉及最多
O(logn)条重链

+

树状数组

```
75      scanf("%d %d",&n,&m);
76      for(int i=1;i<=n-1;++i){
77          int u,v;
78          scanf("%d %d",&u,&v);
79          addedge(u,v);
80          addedge(v,u);
81      }
82      dfs_son(1,0);
83      dfs_top_tIO(1,0);
84      for(int i=1;i<=m;++i){
85          int x,w;
86          scanf("%d %d",&x,&w);
87          add(tI[x],w);
88          cout<<query(x)<<endl;
89      }
```

重儿子先行DFS

重链剖分
两次DFS

点更新

路径查询

tI[x]容易
误写成x

第一步	重链剖分

第1次DFS

从下往上

算son[]

```
14 void dfs_son(int u,int fa){
15     d[u]=d[fa]+1;
16     p[u]=fa;
17     sz[u]=1;
18     son[u]=0;
19     for(int i=hd[u];i;i=e[i].nxt){
20         int v=e[i].to;
21         if(v==fa) continue;
22         dfs_son(v,u);
23         
24         if(                    )
25             
26     }
27 }
```

请同学解释
变量含义

第2次DFS

从上往下

重儿子先行

算top[]

算tI[]tO[]

```
28 void dfs_top_tIO(int u,int fa){
29     tI[u]=++timer;
30     tO[u]=
31     if(son[fa]==u)
32     else top[u]=u;
33     if(!son[u])return;
34
35     for(int i=hd[u];i;i=e[i].nxt){
36         int v=e[i].to;
37         if(                    ) continue;
38         dfs_top_tIO(v,u);
39     }
40 }
```

请同学解释
变量含义

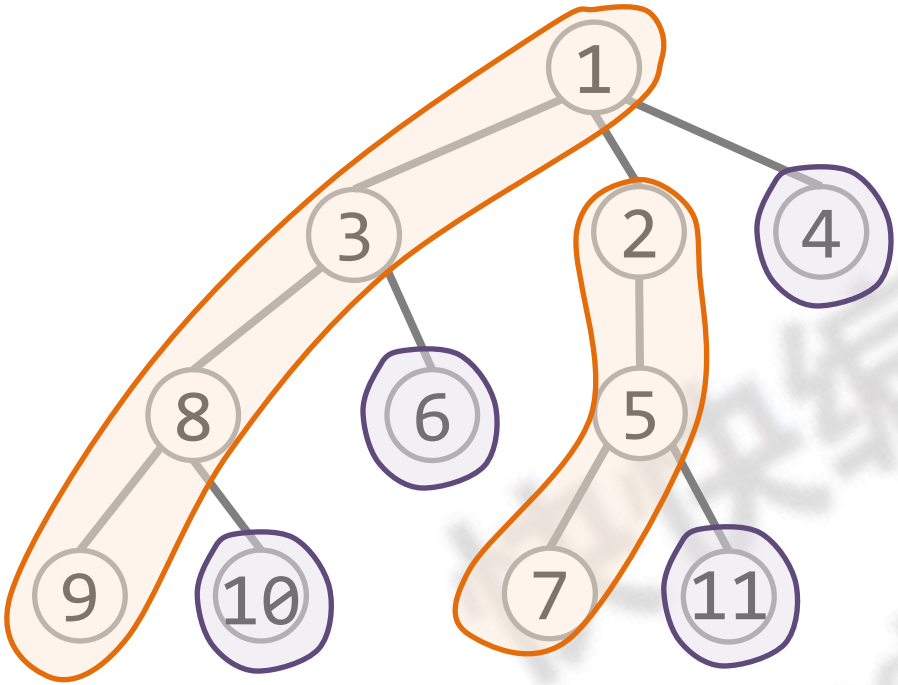
另类代码

错误

```
5  void dfs_top_tIO(int u,int fa){
6      tI[u]=++timer;
7      if(son[fa]==u) top[u]=top[fa];
8      else top[u]=u;
9      if(!son[u])return;
10     dfs_top_tIO(son[u],u);
11     for(int i=hd[u];i;i=e[i].nxt){
12         int v=e[i].to;
13         if(v==fa||v==son[u]) continue;
14         dfs_top_tIO(v,u);
15     }
16     tO[u]=timer;
17 }
```

叶节点
tO值遗漏

第二步	点到根路径拆分成 O(logn)条重链	不断 上跳



```
63 int query(int x){
64     int ans=0;
65     while(x){
66         ans+=rsq(                    );
67         ans%=MOD;
68         
69     }
70     return ans;
71 }
```

区间求和问询

为什么不可以用
rsq(tI[x],tI[top[x]]);

易错点	树序列化后区间端点左右不能混淆

第三步	树状数组

区间求和转换为
前缀和做差

```
59 int rsq(int l,int r){
60     int res=psq(r)-psq(l-1);
61     return
62 }
```

```
51 int psq(int i){
52     int sum=0;
53     while
54
55
56     }
57     return sum;
58 }
```

点更新

```
44 void add(int i,int w){
45     while
46         
47         bit[i]%=MOD;
48         
49     }
50 }
```

快快编程976

树上两种操作

1	路径和查询	
2	路径更新	整体增加

树的序列化
搭配数据结构

重链剖分

路径涉及最多
O(logn)条重链

分块

```
102     scanf("%lld %lld",&n,&m);
103     for(ll u=1;u<=n;++u) scanf("%lld",&x[u]);
104     for(ll i=1;i<=n-1;++i){
105         ll u,v;
106         scanf("%lld %lld",&u,&v);
107         addedge(u,v);
108         p[v]=u;
109     }
110     dfs_son(1);
111     dfs_top_tIO(1);
112     for(ll u=1;u<=n;++u)a[tI[u]]=x[u];
113     B=sqrt(n)+1;
114     for(ll i=1;i<=n;i++)b[i]=(i-1)/B+1;
115     for(ll i=1;i<=n;++i)bs[b[i]]=(bs[b[i]]+a[i])%MOD;
116     for(ll i=1;i<=m;++i){
117         ll u,v,w;
118         scanf("%lld %lld %lld",&u,&v,&w);
119         solve(u,v,w);
120     }
```

只记录父亲到儿子的连边

重儿子先行DFS

重链剖分
两次DFS

新序列a[]

易错点
不能将x[]
代替a[]

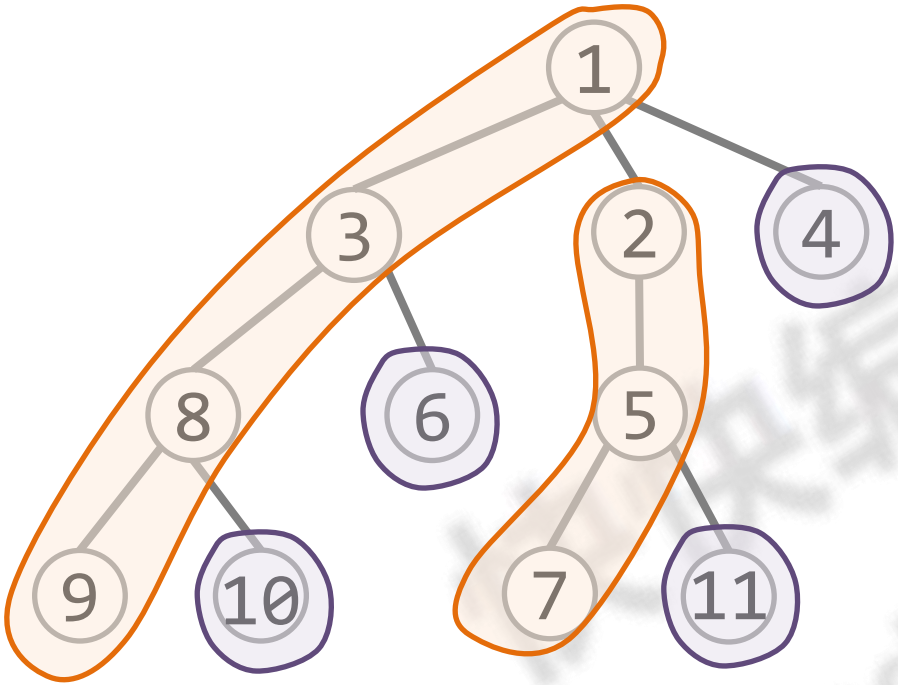
分块预处理

第一步	重链剖分

第二步

路径拆分成O(logn)条重链

分两个阶段
两端点所在重链头不同
两端点所在重链头相同



路径拆分
分两个阶段

```
76 void solve(ll u,ll v,ll w){
77     ll ans=0;
78     while(top[u]!=top[v]){
79         ll&x=(d[top[u]]>d[top[v]]?u:v);
80         ans+=rsq(                          );
81         ans%=MOD;
82         add(                          );
83                                    
84     }
85     if(d[u]>d[v])                 
86     ans+=                         
87     ans%=MOD;
88                                   
89     cout<<ans<<endl;
90 }
```

两端点所在
重链头不同

两端点所在
重链头相同

i号元素的真实值	a[i]+c[b[i]]

第三步	分块

维护各个数组信息

a[i]	序列i号元素已更新的数值
b[i]	序列i号元素所在块号
c[iB]	iB号块的延迟增量
bs[iB]	iB号块的块内总和

快快编程974

0表示休息
1表示工作

树上两种操作

01数值

1	路径和查询	1的个数
2	路径更新	整体修改

重链剖分

路径涉及最多O(logn)条重链

树的序列化
搭配数据结构

分块

i号元素的真实值	tag[b[i]]==2?a[i]:tag[b[i]]

分块

维护各个数组信息

a[i]	序列i号元素已更新的数值
b[i]	序列i号元素所在块号
tag[iB]	iB号块的延迟修改目标
bs[iB]	iB号块的块内总和

0或1或2
2表示已没有延迟修改

```
114     scanf("%lld",&n);
115     for(ll u=2;u<=n;++u){
116         scanf("%lld",&p[u]);
117         p[u]++;
118         addedge(p[u],u);
119     }
120     dfs_son(1);
121     dfs_top_tIO(1);
122     for(ll i=1;i<=n;++i)a[i]=1;
123     B=sqrt(n)+1;
124     for(ll i=1;i<=n;i++)b[i]=(i-1)/B+1;
125     for(ll i=1;i<=n;++i)bs[b[i]]+=a[i];
126     for(ll iB=1;iB<=b[n];++iB)tag[iB]=2;
127     scanf("%lld",&m);
128     for(ll i=1;i<=m;++i){
129         ll t,x;
130         scanf("%lld %lld",&t,&x);
131         x++;
132         if(t==1) holiday(x);
133         else work(x);
134     }
```

原题编号0到n-1
改成1到n

重儿子先行DFS

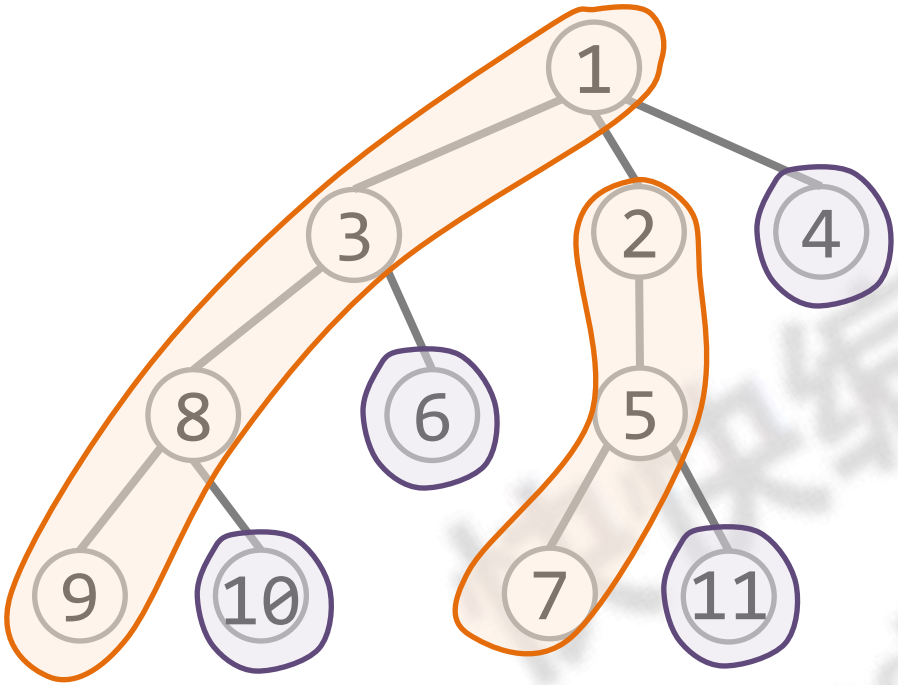
重链剖分
两次DFS

分块预处理

编号加1

路径拆分成O(logn)条重链

分两个阶段
两端点所在重链头不同
两端点所在重链头相同



```
89 void work(ll x){
90     ll ans=
91     upd(tI[x],tO[x],1);
92     cout<<ans<<endl;
93 }
```

```
94  void holiday(ll x){
95      ll ans=0;
96      while(x){
97          ans+=
98          upd(tI[top[x]],tI[x],0);
99
100     }
101     cout<<ans<<endl;
102 }
```

分块

将延迟更新的信息落地

pushdown()

```
41 void pushdown(ll iB){
42     if(tag[iB]==2)return;
43     ll l=B*(iB-1)+1;
44     ll r=min(n,B*iB);
45     for(ll i=l;i<=r;i++)
46         a[i]=tag[iB];
47     tag[iB]=2;
48 }
```

```
49 void upd(ll l,ll r,ll z){
50     pushdown(b[l]);
51     pushdown(b[r]);
52     if(b[l]==b[r]){
53         for(ll i=l;i<=r;++i){
54             bs[b[i]]+=z-a[i];
55             a[i]=z;
56         }
57         return;
58     }
59
60
61
62
63
64
65
66
67
68
69
70
71 }
```

左右两端点所在块更新落地

```
72 ll rsq(ll l,ll r){
73     pushdown(b[l]);
74     pushdown(b[r]);
75     ll ans=0;
76     if(b[l]==b[r]){
77         for(ll i=l;i<=r;++i)
78             ans+=a[i];
79         return ans;
80     }
81
82
83
84
85
86
87     return ans;
88 }
```

左右两端点所在块更新落地

查错方法

让分块退化成暴力：B=n

打印所有数组元素

快快编程作业

1887,976,974