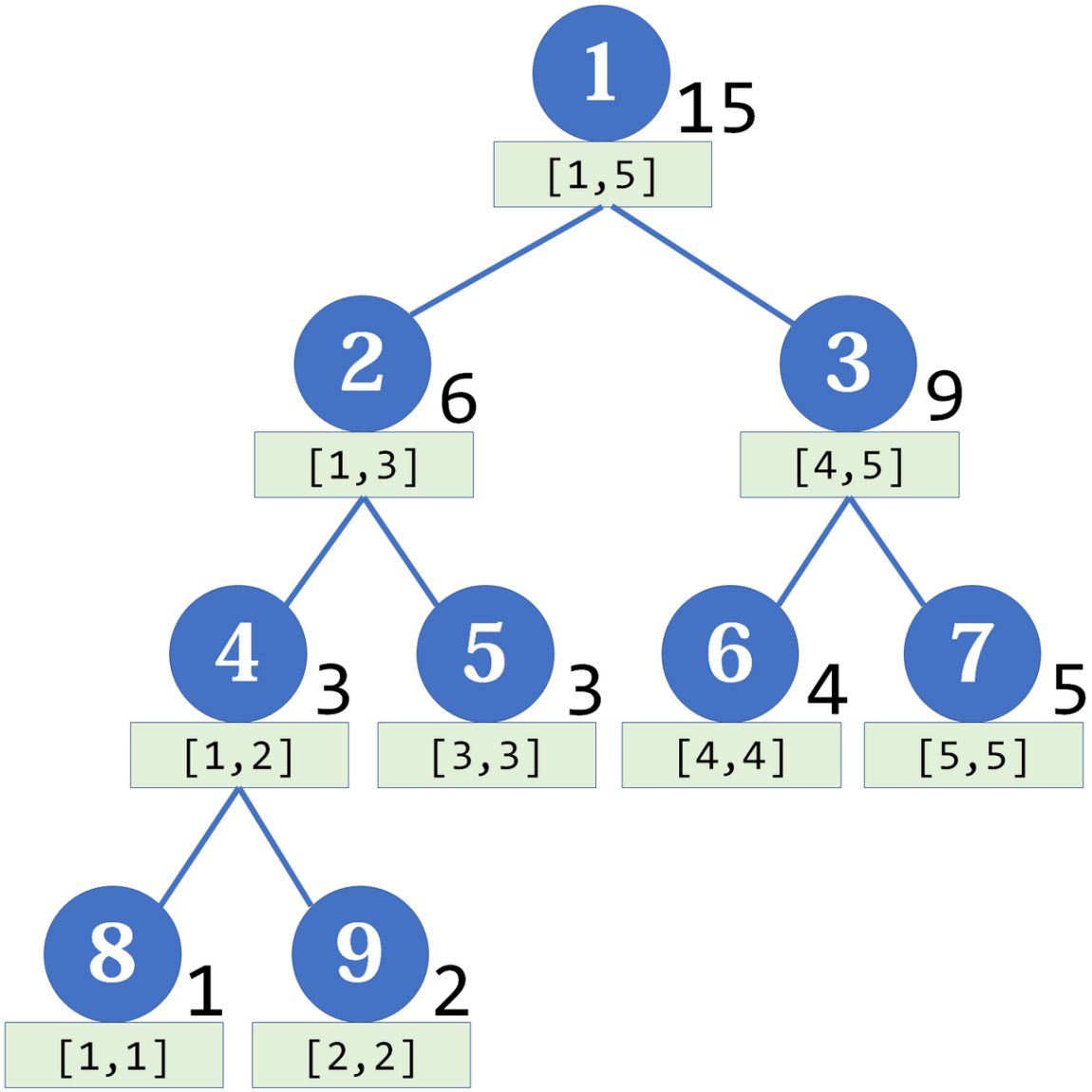
动态DP

前提知识

线段树

运用BIT的思想可以划分区间，将整个区间分为一半，一直这样操作，于是可以推出下面的图：



处理出每个节点的区间值即可，区间查询是 log_n 级别。

但是区间修改很慢，但是在这个节点的区间完全包含在需要更改的区间内，可以不用向下递归，打懒标记即可，需要使用时向下传即可。

重链剖分

树可以裂解为若干个线性表，但是直接处理每一条链是不优雅而且时间复杂度高，是否可以构造出一个dfs序列使得满足某些条件的节点在一起呢?

例题

P3384

很明显树上任意两个节点 a, b 的简单最短路径可以分解为从 a 到 $lca_{a,b}$ 和从 b 到 $lca_{a,b}$ ，很明显可以考虑转换为序列上的问题，为了满足一个子树内的节点在一起，可以考虑DFS序，但是从一个节点到祖先的节点很明显不能直接放在一起，可以退而求其次，是否可以分成数量比较少的呢?

重儿子

很明显如果直接按照输入顺序进行DFS，一条路径很可能每个节点都不相邻，所以是否可以考虑按照一定的顺序进行DFS，比如按照子树大小，所以在进行DFS时，先进入子树较大的节点，此为这个节点的重儿子，连接重儿子的边称为重边，若干条（可能为 0）重边首尾相接构成重链。

时间复杂度

性质1:

性质:

从根节点到叶子结点最多经过 $log(n)$ 条重链。

证明:

重链的切换在于经过非重边，每次经过非重边子树大小必然至少除以 2。

代码实现

```
void dfs1(int now,int fa){
    siz[now]=1;
    dep[now]=dep[fa]+1;
    ::fa[now]=fa;
    bson[now]=0;
    for(int i=fir[now];i!=-1;i=nxt[i]){
        if(v[i]==fa){
            continue;
        }
        dfs1(v[i],now);
        siz[now]+=siz[v[i]];
        if(siz[v[i]]>siz[bson[now]]){
            bson[now]=v[i];
        }
    }
    return ;
}
int id[100005];
int cnt=0;
int top[100005];
void dfs2(int now,int topf){
    id[now]=++cnt;
    init(cnt,cnt,w[now]);//处理线性表信息，转换为线性表
    top[now]=topf;
    if(bson[now]==0){
        return ;
    }
    dfs2(bson[now],topf);
    for(int i=fir[now];i!=-1;i=nxt[i]){
        if(v[i]==fa[now]||v[i]==bson[now]){
            continue;
        }
        dfs2(v[i],v[i]);
    }
    return ;
}
```

广义矩阵乘法

矩阵乘法的基本形式是 $A \times B = C$ 是：$C_{i,j} = \sum_{k=1}^{n}(A_{i,k} \times B_{k,j})$

其实求和以及乘法都可以被替换为其他运算。

满足结合律，证明略。

引入

考虑求一颗树的最大权独立集。

P4719

不考虑修改操作

很明显非常模版，定义 $dp_{i,0}$ 表示不选择 i ，考虑 i 的子树的答案，$dp_{i,1}$ 表示选择 i ，考虑 i 的子树的答案。

很明显可以推出：

$$dp_{i,0} = \sum \max(dp_{son,0}, dp_{son,1})$$

$$dp_{i,1} = w_i + \sum dp_{son,0}$$

$$Ans = \max(dp_{root,0}, dp_{root,1})$$

考虑修改操作

首先考虑如何让dp式更好看，又是求和又是最大值，一看就不好优化，可否把求和弄点呢，考虑树链剖分中的重儿子的思想，将 dp 转移式分成两部分如下，$bson_i$ 表示 i 的重儿子：

$$dp_{i,0} = g_{i,0} + \max(f_{bson_i,0}, f_{bson_i,1})$$

$$dp_{i,1} = g_{i,1} + f_{bson_i,0}$$

接下来我们根据这个来推出 g 的定义。

很明显 $g_{i,0}$ 表示不选择 i 不考虑重儿子 $bson_i$ 时的答案，$g_{i,1}$ 表示选择 i 时不考虑 $bson_i$ 时的答案。

但是max操作还是在，直接放进广义矩阵里面。

于是定义广义矩阵乘法为：$A \times B = C$ 是：$C_{i,j} = \max_{k=1}^{n}(A_{i,k} + B_{k,j})$

于是有：

$$\begin{bmatrix} g_{i,0} & g_{i,0} \\ g_{i,1} & -\infty \end{bmatrix} \times \begin{bmatrix} dp_{bson_i,0} \\ dp_{bson_i,1} \end{bmatrix} = \begin{bmatrix} dp_{i,0} \\ dp_{i,1} \end{bmatrix}$$

修改操作只用修改所经过的重链的头部的父亲的 $g_{i,1}$ 即可。

因为广义矩阵乘法同样满足结合律，所以线段树维护 g 即可。

注意为单点修改，所以完全不用加懒标记。

总结

动态DP可以用于一些树上DP问题需要在线更改树的形态并且随时求出dp答案的题目，对于树上问题启发很大，比如修改为特殊的点权以及边权可以将题目中一些操作统一化使用动态DP求解。

动态DP的思想同样可以运用的具有区间修改的DP题，本质上是将DP转移划归为广义的矩阵乘法然后套用高级数据结构实现修改。

推荐题目

P4719

SP1043

SP1716

SP6779

P5024

P6021