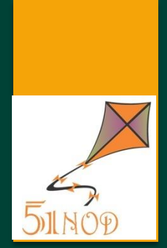
第三章：树结构

第四节：树上的动态规划

树上的动态规划



通过之前两节的学习，我们发现树上DFS总是伴随着一种状态转移，有从父节点向子节点的转移，也有子节点向父节点的转移；

这和我们一些DP的过程很相似，所有很多树上的题目我们都可以通过树上DP来解决。；树形DP 。

这是个什么东西？为什么叫这个名字？跟其他DP有什么区别？

相信很多初学者在刚刚接触一种新思想的时候都会有这种问题。

没错，树形DP准确的说是一种DP的思想，将DP建立在树状结构的基础上。

既然说了这是一种思想，那么单讲的话，也讲不出什么东西来。所以我们结合具体题目进行讲解，希望大家可以在题目中领悟这种思想。

提到树形DP入门题，很多人都会提到没有上司的舞会这道题，的确，这道题堪称树形DP的典范，下面我们来从这道题将起带领大家入门树形DP 。

2605没有上司的舞会

题目描述：

某大学有N个职员，编号为1~N，校长的编号为1，他们之间有从属关系，也就是说他们的关系就像一棵以校长为根的树，父结点就是子结点的直接上司。现在有个周年庆宴会，宴会每邀请来一个职员都会增加一定的快乐指数Ri，但是呢，如果某个职员的直接上司来参加舞会了，那么这个职员就无论如何也不肯来参加舞会了。所以，请你编程计算，邀请哪些职员可以使快乐指数最大，求最大的快乐指数。

输入描述：

第一行一个整数N。(1<=N<=100000)

接下来N行，第i+1行表示i号职员的快乐指数Ri。(-128<=Ri<=127)

接下来N-1行，每行输入一对整数L,K。表示K是L的直接上司或者L是K的直接上司。

输出描述：

输出最大的快乐指数。

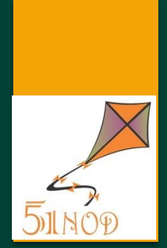
输入样例：

```
7
1
1
1
1
1
1
1
1 3
2 3
6 4
7 4
4 5
3 5
```

输出样例：

```
5
```

2605没有上司的舞会-解题思路



我们开始分析这道题，我们首先可以发现，对于某个职员来说，他只有两个状态，要么参加舞会要么不参加舞会，而且当这个职员选择参加舞会时，他的所有下属都不能来参加舞会，当这个职员不参加舞会时，他的下属要么参加舞会，要么不参加舞会。所以我们可以根据这两个状态定义DP数组的含义。

F[i][0]表示以i为根的子树当i不选择时产生的最大快乐指数

F[i][1]表示以i为根的子树当i选择时产生的最大快乐指数

2605没有上司的舞会-解题思路

之后对于每个节点的两个状态，可以从他的所有子节点状态转移过来

状态转移方程应该为：

$F[x][0] = \sum_{s \in Son(x)} Max(F[s,0],F[s,1])$

当x节点不选择的时候，对于所有子节点，可以选可以不选，所以选择快乐值大的进行转移。

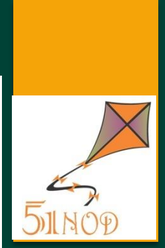
$F[x][1] = H[x] + \sum_{s \in Son(x)} F[s,0]$

当x节点选择的时候，首先应该加上x节点本身的快乐值，之后所有子节点都不能选择，所以把所有子节点不选择的状态转移过来即可。

没有上司的舞会
参考答案

```
#include <bits/stdc++.h>
using namespace std;
const int maxn = 1e5+10;
int dp[maxn][2];
int a[maxn];
vector<int> G[maxn];
void dfs(int rt,int fa)
{
    dp[rt][0]=0;
    dp[rt][1]=a[rt];
    for(int i=0;i<G[rt].size();i++)
    {
        int to=G[rt][i];
        if(to==fa) continue;
        dfs(to,rt);
        dp[rt][1]=dp[rt][1]+max(0,dp[to][0]);
        dp[rt][0]=dp[rt][0]+max(0,max(dp[to][0],dp[to][1]));
    }
    return ;
}
```

没有上司的舞会
参考答案




```
int main()
{
    int n;
    int u,v;
    scanf("%d",&n);
    for(int i=1;i<=n;i++) G[i].clear();
    for(int i=1;i<=n;i++) scanf("%d",&a[i]);
    for(int i=1;i<=n-1;i++)
    {
        scanf("%d%d",&u,&v);
        G[v].push_back(u);
        G[u].push_back(v);
    }
    dfs(1,0);
    int ans=0;
    for(int i=1;i<=n;i++) ans=max(ans,max(dp[i][0],dp[i][1]));
    printf("%d\n",ans);
    return 0;
}
```