没有上司的舞会

Ural大学有N个职员，编号为1~N。他们有从属关系，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。每个职员有一个快乐指数。现在有个周年庆宴会，要求与会职员的快乐指数最大。但是，没有职员愿和直接上司一起与会。

输入

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。

输出

输出最大的快乐指数。

样例

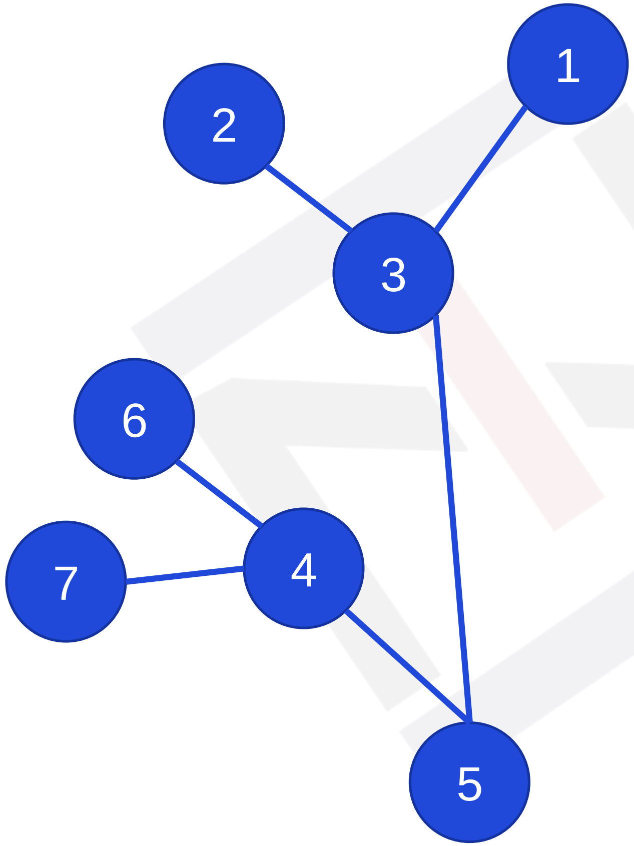
样例输入1

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

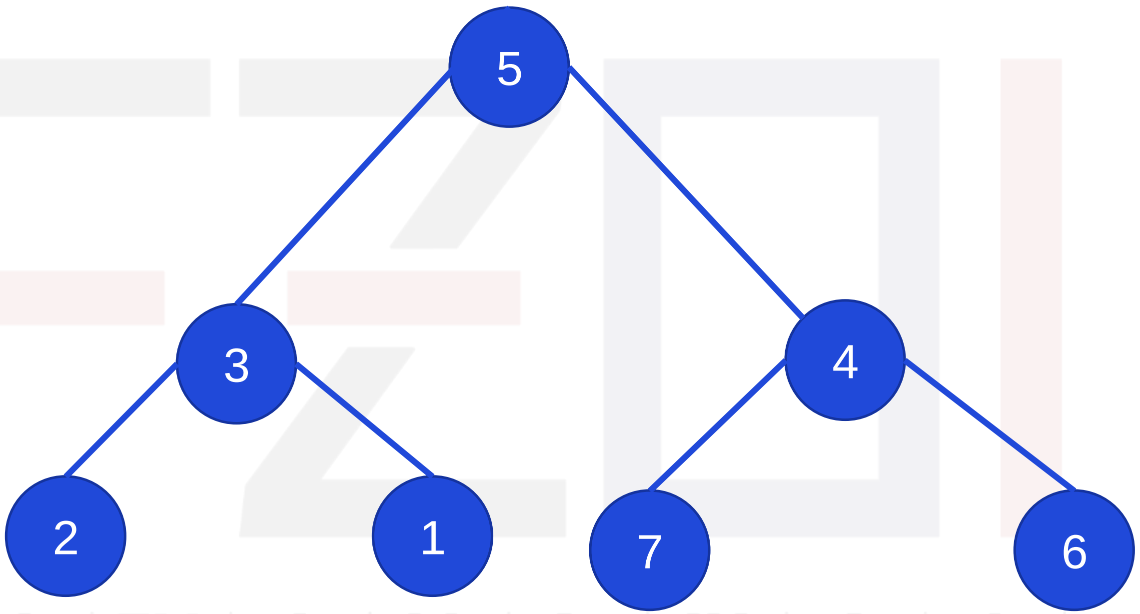
样例输出1

```
5
```

没有上司的舞会



很明显是一棵树



题目里上司和下属的影响是双向的
也可以理解为下属去不去会影响上司

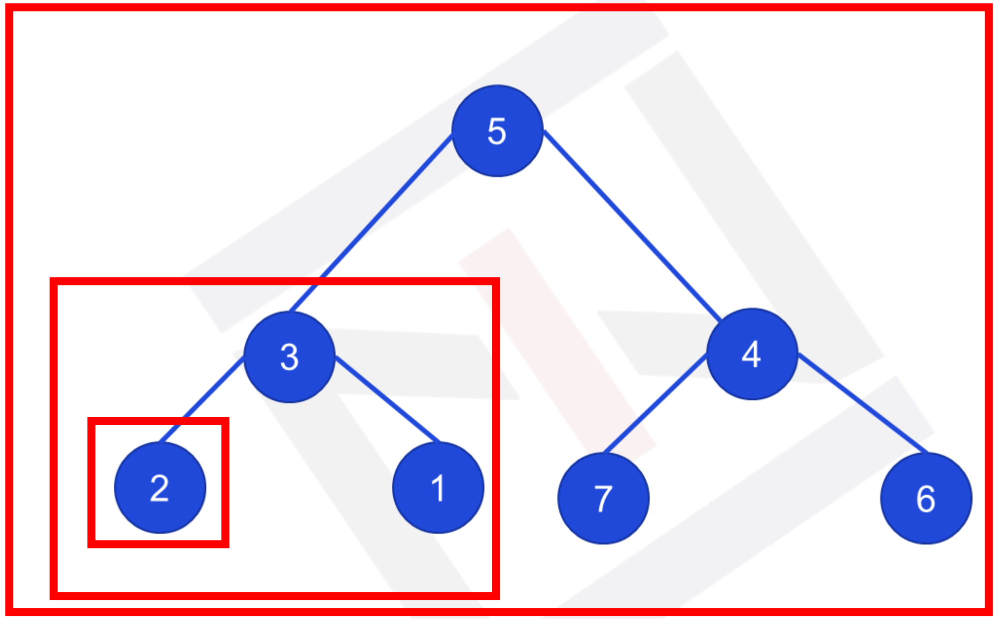
在一个树形结构里，求解全局最优解(DP)

信息学

初识树形DP

西南大学附属中学校

信息奥赛教练组



树形DP的求解过程一般为自底向上，将子树从小到大作为DP的**阶段**。

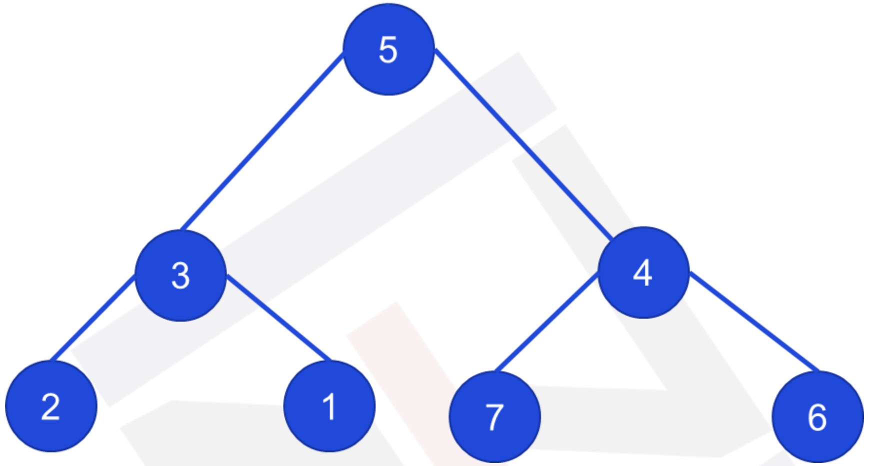
DP**状态**第一维通常是节点编号，代表该节点为根的子树。

树本身就是递归定义的，树形DP采用**递归**的方式实现。

先访问子节点，再访问根，

使用树的后序遍历进行访问

树形DP过程



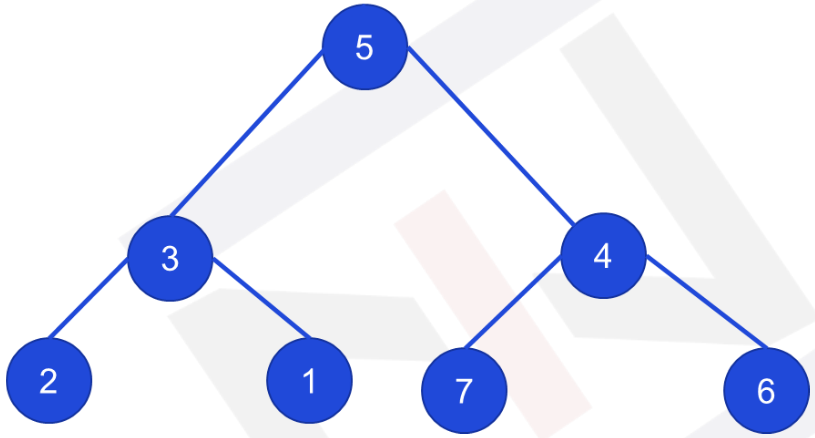
通常情况下，就是就是从根节点开始DFS，并进行DP。

状态转移一般是从子节点向上转移，具体怎么转移看题目条件。

简单来说在树上进行动态规划的转移

对于无根树，任选一个节点作为根节点。对于每个节点 u，先递归它的子节点v，**回溯时，从子节点v向父节点 u 进行状态转移**。

没有上司的舞会



设计状态

f[u]表示以u为根的子树最大快乐值　有后效性

f[u][j]表示以u为根的子树且u是否参加舞会的最大快乐值 ，j取0或1

转移方程

f[u][0]+=max(f[v][0],f[v][1])　//u不去，子节点可去可不去
f[u][1]+=f[v][0] //u去了，u的子节点肯定不能去

实现

找到根节点root，从root开始dfs
回溯时，实现转移

代码


```
#include <bits/stdc++.h>
using namespace std;
int a[60005],dp[60005][2],head[60005],cnt,n,ru[60005],root;
struct node{
		int to,nxt;
}e[60005];
void insert(int u,int v){
		e[++cnt].nxt=head[u];
		e[cnt].to=v;
		head[u]=cnt;
}
void dfs(int u){
		for(int i=head[u];i;i=e[i].nxt){
			int v=e[i].to;
			dfs(v);
			dp[u][0]+=max(dp[v][0],dp[v][1]);
			dp[u][1]+=dp[v][0];
		}
		dp[u][1]+=a[u];
}
```

```
int main(){
		scanf("%d",&n);
		for(int i=1;i<=n;i++)scanf("%d",&a[i]);
		for(int i=1;i<=n-1;i++){
			int u,v;
			cin >> v >> u;
			ru[v]++;
			insert(u,v);
		}
		for(int i=1;i<=n;i++){
			//没有入度即为根
			if(ru[i]==0)root=i;
		}
		dfs(root);
		cout << max(dp[root][0],dp[root][1]) << endl;
		return 0;
}
```

在这棵树上不存在任何一条边使得连接的两个点都来参会，换句话说这道题其实要我们求的是**树的最大的独立集**。

战略游戏

BOB喜欢玩电脑游戏，特别是战略游戏。但是他经常无法找到快速玩过游戏的方法。现在他有个问题。

他要建立一个古堡，古堡中的路形成一棵树。他要在这棵树的结点上放置最少数目的士兵，使得这些士兵能够瞭望到所有的路。某一个士兵在一个结点上时，与该结点相连的所有边都可以被瞭望到。

输入

第一行n，表示树的结点数目

接下来n+1行，每行描述每个结点信息，依次是：该结点的标号i，k（后面有k条边与i相连），接下来k个数，分别是每条边的另一个结点。

对于n个结点的树，结点标号在0~n-1之间，文件每条边只出现一次。0<=n<=1500

输出

输出一个数，需要的最少的士兵数目

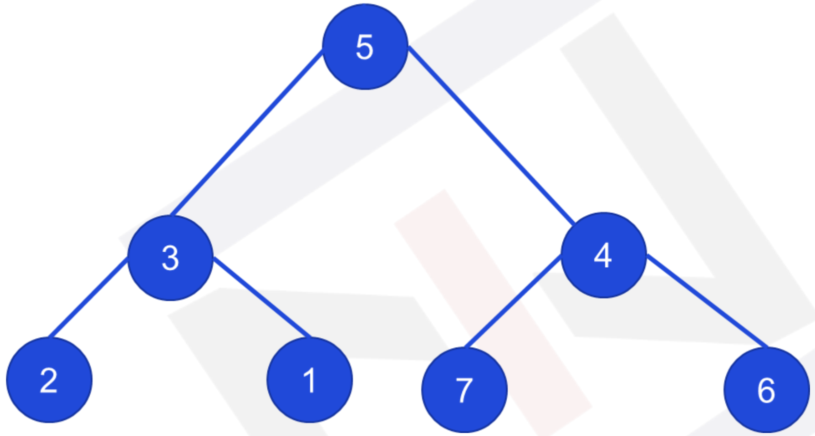
样例

样例输入1

```
4
0 1 1
1 2 2 3
2 0
3 0
```

样例输出1

```
1
```



选取最少的点，覆盖树所有的边

树的最小点覆盖

设计状态

f[i][j]表示第 i 个节点有/无士兵，观测 i 与 i 的子节点所需要的士兵的数量的最小值

f[i][0] 表示第 i 个节点无士兵的最小值
f[i][1] 表示第 i 个节点有士兵的最小值

转移方程

当 u 点不放士兵时，它的儿子就必须都要放士兵
f[u][0] += f[v][1] ;

当 u 点放士兵时，它的儿子可以放士兵，也可以不放士兵
f[u][1] += min(f[v][0],f[v][1])

答案：ans = min(f[root][0],f[root][1])

代码


```
void dfs(int u,int fa) {
    f[u][1]=1,f[u][0]=0;
    for(int i=head[u];i;i=nex[i]) {
        int v=to[i];
        if(v==fa) continue;
        dfs(v,u);
        f[u][0]+=f[v][1];
        f[u][1]+=Min(f[v][1],f[v][0]);
    }
}
```

二叉苹果树

题目描述

有一棵苹果树，如果树枝有分叉，一定是分2叉（就是说没有只有1个儿子的结点）

这棵树共有N个结点（叶子点或者树枝分叉点），编号为1-N,树根编号一定是1。

我们用一根树枝两端连接的结点的编号来描述一根树枝的位置。下面是一颗有4个树枝的树

```
2   5
 \ /
  3   4
   \ /
    1
```

现在这颗树枝条太多了，需要剪枝。但是一些树枝上长有苹果。

给定需要保留的树枝数量，求出最多能留住多少苹果。

输入输出格式

第1行2个数，N和Q $(1 \leq Q \leq N, 1 < N \leq 100)$。

N表示树的结点数，Q表示要保留的树枝数量。接下来N-1行描述树枝的信息。

每行3个整数，前两个是它连接的结点的编号。第3个数是这根树枝上苹果的数量。

每根树枝上的苹果不超过30000个。

输出格式

一个数，最多能留住的苹果的数量。

输入样例

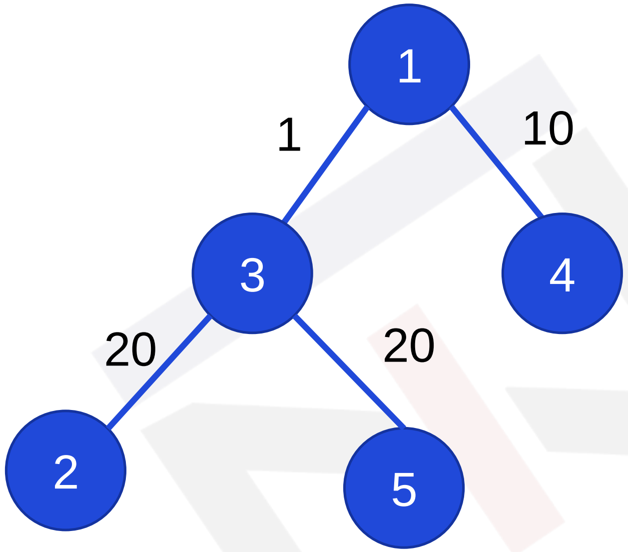
```
5 2
1 3 1
1 4 10
2 3 20
3 5 20
```

输出样例

```
21
```

二叉苹果树

西南大学附属中学
High School Affiliated to Southwest University



设计状态

f[i][j] 表示以i为根的子树保留j个树枝时可以得到的最大的苹果数量

转移方程

从左右子树的选择来考虑

1.只选左子树

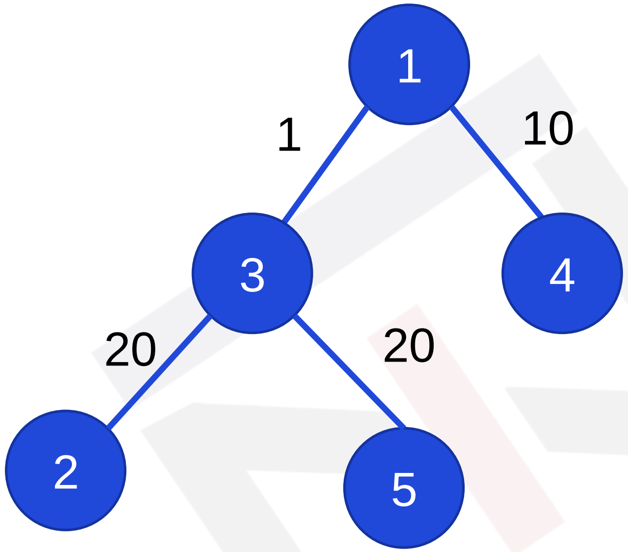
左子树需要保留j-1条，因为节点i跟左子树必须有一条边相连

2.只选右子树

右子树需要保留j-1条，因为节点i跟右子树必须有一条边相连

3.两个都选

左边保留k条边，右边就保留j-k-1条边



设计状态

f[i][j] 表示以 i为根的子树保留 j个树枝时可以得到的最大的苹果数量

转移方程

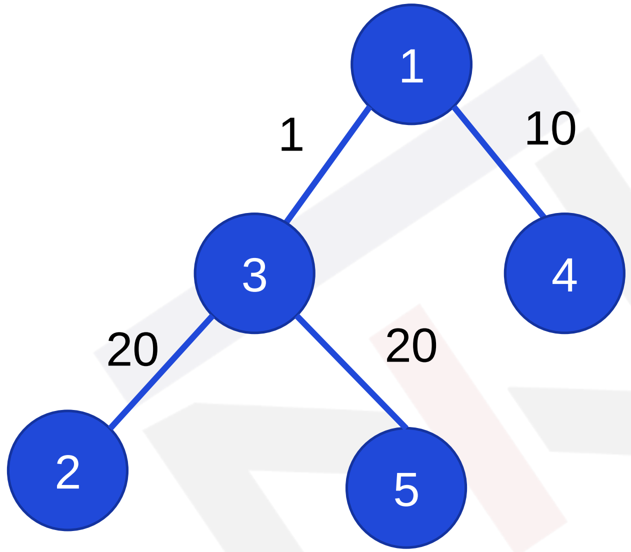
设树根为i，左孩子为 lson[i]，右孩子为rson[i] 。对于根节点为i的子树，保留j个边，左子树保留k个边，右子树保留j-k-1个边

对于根节点 lson[i]的子树，保留k-1个边，左子树lson[lson[i]]保留k_1个边，在右子树rson[lson[i]]保留 k-1-k_1个边。

对于 rson[i]为根的子树，在该子树的左右子树中保留j- k-1个边，左子树lson[rson[i]]保留 k_2个边，右子树rson[rson[i]] 保留j-k-1-k_2个边；...... 这就是一个递归的过程

f (i,j) =max{f(lson[i],k)+f(rson[i],j-k-1)+a[i]}

二叉苹果树



设计状态

f[i][j] 表示以 i为根的子树保留 j个树枝时可以得到的最大的苹果数量

转移方程

对于每一条边，只能取一次，取后会对总的结果造成影响
01背包的思想

将保留的边的数目看成是背包的容量（最多可以保留j条边），边的权重看成是价值，每条边看成是体积为1

f[u][j]=max(f[u][j],f[u][j-1-k]+f[v[i]][k]+w[i])

v[i]:结点i的子节点，前向星建图

如果这棵树不是二叉的，多叉树怎么做?

尝试一下**[CTSC1997] 选课，提示：分组背包的思想**

做做题
休息一下

求树的重心

n 个结点的无根树，找到一个结点 u ，删除结点 u 后，最大的连通块的结点数最小，则节点 u 就为重心。

问题描述：

给定n(2≤n≤10000)个点的树，输出重心的节点编号。

问题分析：

需要知道删除节点 u 后，剩下连通块的数量，即节点 u 的孩子节点数+1。

设 **f[u]** 表示节点 u 为根的子树的节点数量，初始时全为1。

设节点 u 的孩子节点为 v ，则：

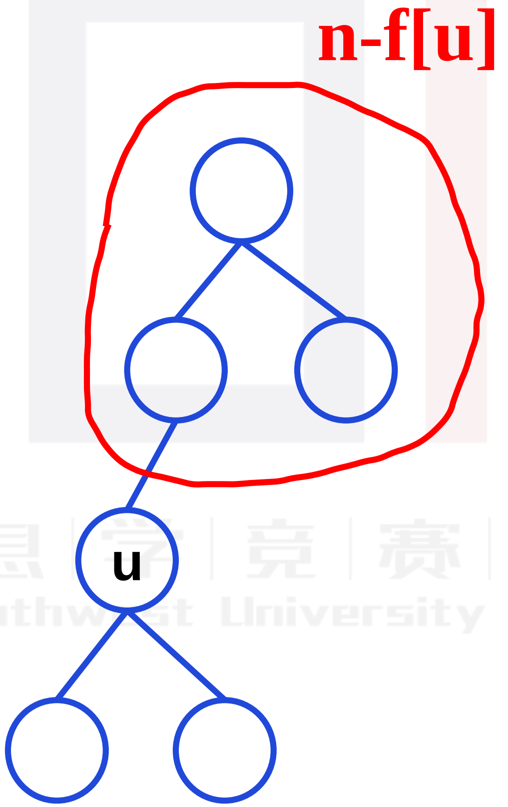
$$f[u] = f[u] + \sum_{v\in son[u]} f[v]$$

设 **mx[u]** 表示删除节点 u 后，最大的连通块数量。

$$mx[u] = max_{v\in son[u]}\{ f[v], n - f[u]\}$$

最小的 mx[u] 就是重心节点。

通过一次DFS，可以求解出 f、mx 数组，时间复杂度为O(n)。



求树上最长路径(树的直径)

n 个结点的无根树，找出两个点 u 、 v ，使得 u 到 v 的距离最远，这两个点之间的路径就是树的最长路径。

问题描述：

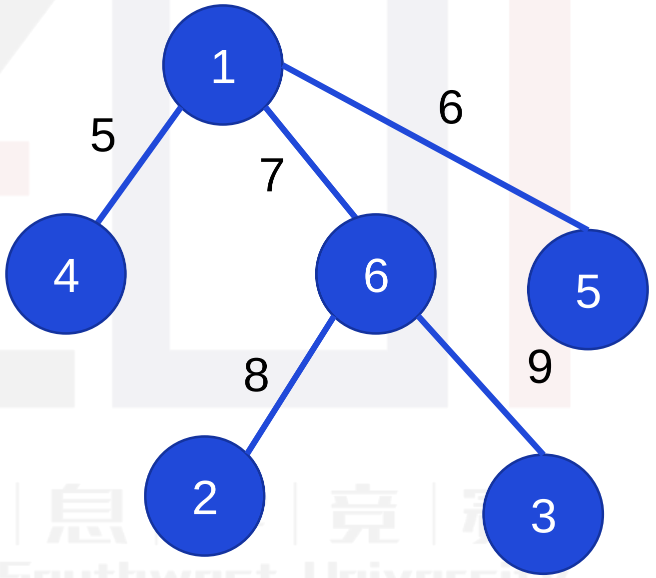
给定n(n≤10000)个点的树，依次输出节点1~n为起点的最长路径。

求树上最长路径

问题分析：

方法一：以某个点为根dfs一次，找到最远的端点x；再从x开始dfs一次，找到最远的端点y。两次dfs，x，y的距离即为答案。

缺陷：如果树边为负权

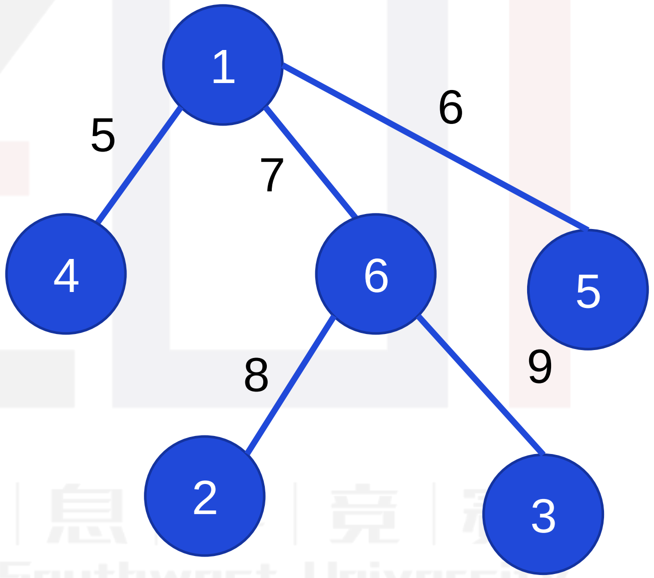


问题分析：

方法二：任取一个根节点，对于结点A上的所有路径，我们找到A点的所有直接儿子，分别求它们从上往下的最大距离（到叶子结点）。

两个值 dp1[i] 表示以 i 为根的子树中，i 到叶子节点的距离最大值
dp2[i] 表示以 i 为根的子树中，i 到叶子节点的距离次大值

答案即为max(dp1[i]+dp2[i])



小结

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。
整体的思路大致就是用树形的结构存储数据。

1.选择节点类

$$\begin{cases} f[i][0] = f[j][1] \\ f[i][1] = \max / \min(f[j][0], f[j][1]) \end{cases}$$

2.树形背包类

$$\begin{cases} f[v][k] = f[u][k] + val \\ f[u][k] = \max(f[u][k], f[v][k-1]) \end{cases}$$

过程：先找到树根root，从树根开始运用dfs递归(后序遍历)，跟dfs一样先初始化，然后递归到叶子节点上为止，把最底层的f[i][j]更新完毕，再回来往上走，自底向上地根据题意更新上层的f[][]数组，最后输出答案即可

求最大的独立集/最小点覆盖

树上背包

求树的重心

求树的最长路径

拓展学习：二次扫描/换根DP

Thanks

For Your Watching

FZOI

西大附中信息学竞赛

High School Affilinted to Southwest University