P4913 【深基16.例3】二叉树深度

题目描述

给出每个节点的两个儿子节点，建立一棵二叉树（根节点为 1），如果是叶子节点，则输入0 0。建好树后希望知道这棵二叉树的深度。二叉树的深度是指从根节点到叶子结点时，最多经过了几层。

最多有 10^6个结点。

输入格式

第一行一个整数 n，表示节点数。

之后 n 行，第 i 行两个整数 l、r，分别表示节点 i 的左右子节点。若 l=0则表示无左子节点，r=0 同理。

输出格式

一个整数，表示最大节点深度。

输入样例

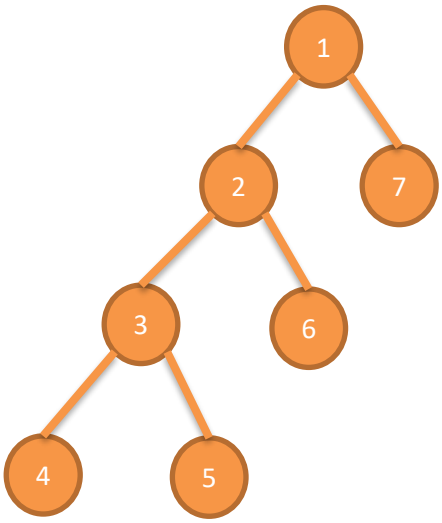
```
7
2 7
3 6
4 5
0 0
0 0
0 0
0 0
```

输出样例

```
4
```

解析

根据题意我们可以构建出这棵二叉树。



通过观察图像，我们很容易得出这棵树的最大深度是4，那么我们该如何计算呢？

在这里我们使用了一个逆推法。

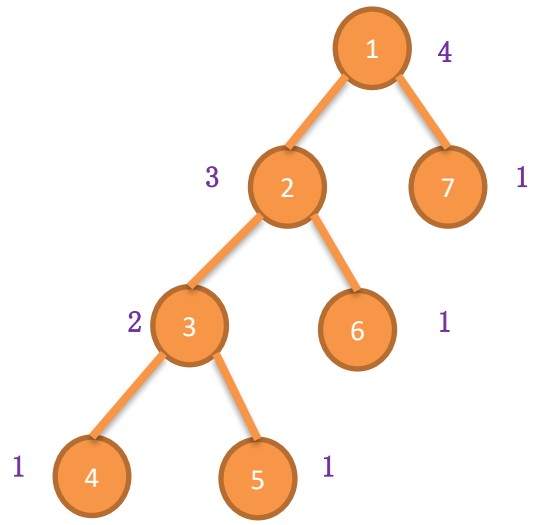
我们设空节点的深度为0，即不存在的节点为0。很明显节点4,5,6,7可以被看作各有两个深度为0的空节点。

接下来我们开始标记4号节点，因为它的左右两个节点的深度都为0，因此4号节点的深度就是0+1（本身）。同理，5，6,7号3个节点的深度也是1。

此时，作为4,5号节点的父节点3，深度则是1（4,5节点的最大深度）+1（自身）=2。

这时，作为3号和6号节点父节点的2号节点，就需要在3号和6号节点中选择最大的一个，再加上自身，那么3号节点的深度就是3.同理，1号节点的深度则为4。

图像如下，紫色即为深度：



编码

```cpp
#include <bits/stdc++.h>

using namespace std;

//10的6次方个节点
const int N = 1e6 + 10;

//树的结构体+存储数组
struct Node {
    int left; // 左结点ID
    int right;// 右结点ID
} t[N];        //i:当前结点ID

//n个节点
int n;

//求以x为根的树的高度,注意递归函数的定义
int dfs(int x) {
    //递归终止条件，是遇到左结点或右结点标识的是0。
    if (x == 0) {
        return 0;
    }
    //左结点报告它的高度，右结点报告它的高度，
    //我只选择一个大的，然后把我自己的1加上去，再向上级领导汇报！
    return max(dfs(t[x].left), dfs(t[x].right)) + 1;
}
int main() {
    //读入
    cin >> n;
    //创建二叉树 build
    for (int i = 1; i <= n; i++) {
        cin >> t[i].left >> t[i].right;
    }
    //求根开始的树的高度
    cout << dfs(1);
    return 0;
}
```