## 题目大意

交互库有一个以1为根的有根树，每次可以询问若干个点，令$V$为这些点对应子树的并，交互库会返回$V$内所有不同的无序点对$(u,v)$的$d(u,v)$之和，$d(u,v)$表示$u$到$v$路径上的边数。

## 数据范围

极限数据满足：$n \le 1000, T \ge 30n$，其中$T$为你能询问的次数，具体如下：

| n | T | 分数 | 特殊性质 |
|---|---|---|---|
| 5 | 1000 | 5 | |
| 100 | 100000 | 5 | |
| | 5000 | 10 | |
| 1000 | 200000 | 10 | |
| | 100000 | 10 | A |
| | | 10 | B |
| | | 10 | |
| | 50000 | 10 | A |
| | | 10 | B |
| | | 10 | |
| | 30000 | 10 | |

$A$特殊性质满足：这是一棵二叉树。

$B$特殊性质满足：树随机，随机方式为：随机生成一个排列$p$，满足$p_1=1$，然后令$f_{p_i}$在$p_{1\sim i-1}$内随机生成。

如果你返回的树和交互库同构，你将获得$40\%$的分数。

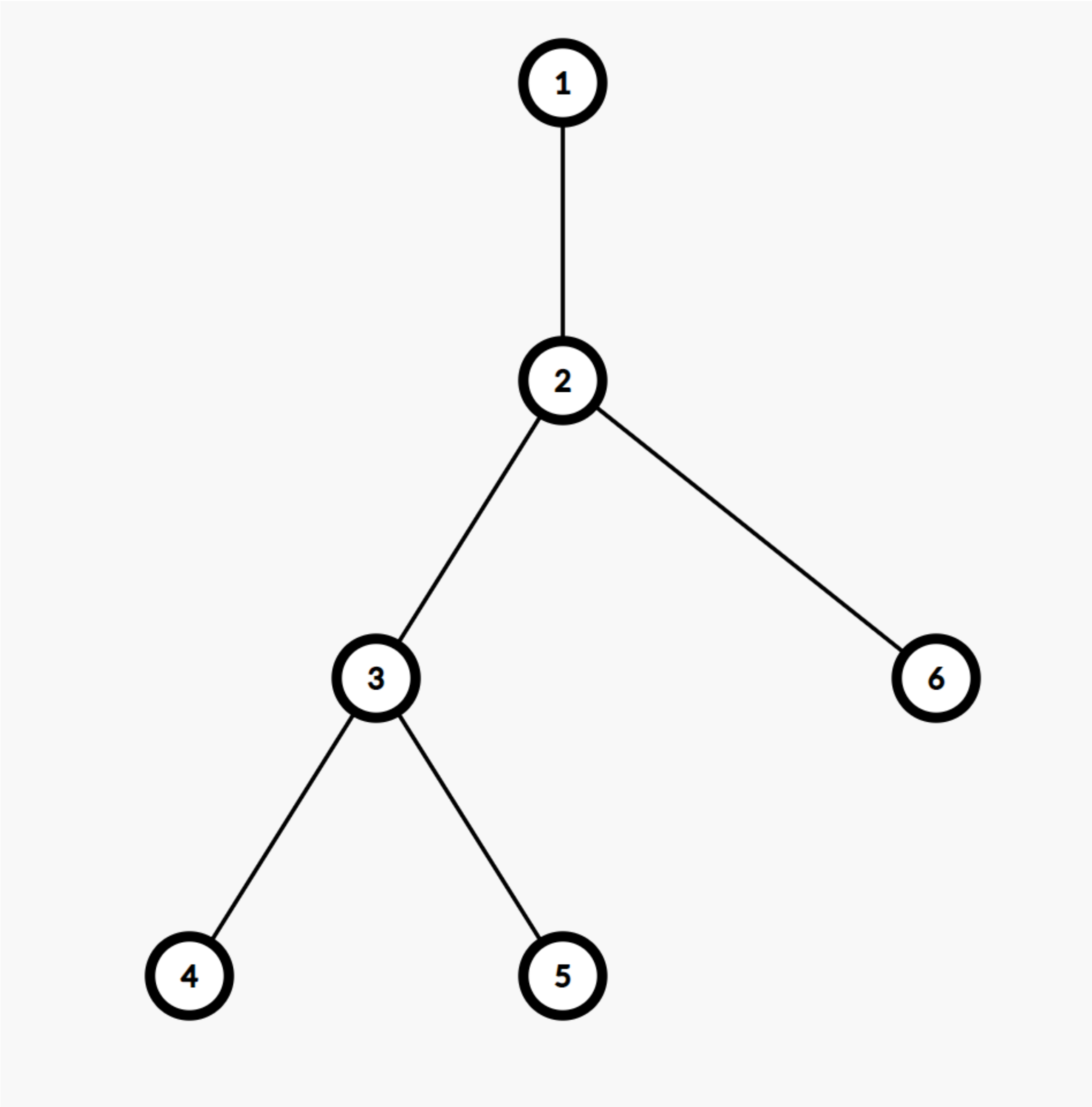
## 解题过程

这里先介绍一种符合直觉的做法：

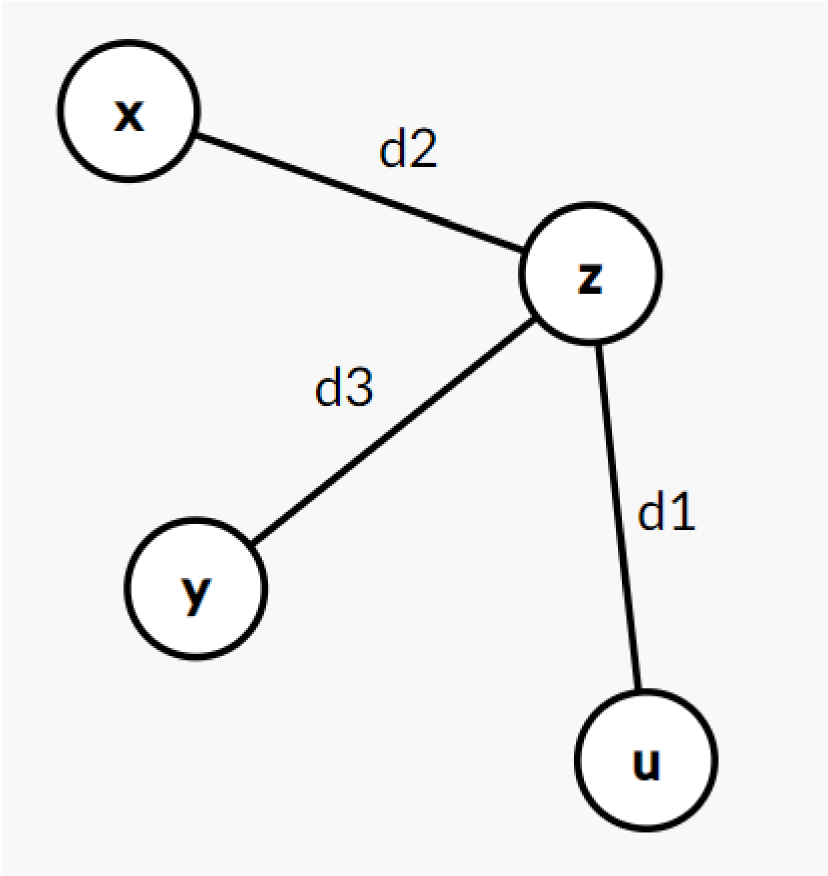
如果$D(\{x\})=0$，那么$x$是叶子，如果$x,y$都是叶子，$D(\{x,y\})=d(x,y)$，如果对于$x,y,z$知道$d(x,y),d(y,z),d(x,z)$，那么$x,y,z$所对于的无根虚树的形态可以确定，于是有以下做法：

可以先把这棵树的形态确定，如果询问两棵叶子，返回的是就是距离，这样可以先确定下两个叶子的距离。由这两颗叶子开始推导所有叶子的相对位置。

因为1号点度数可能为1（如下图），现在的目的仅仅是找到$\{4,5,6\}$它们构成的无根虚树。

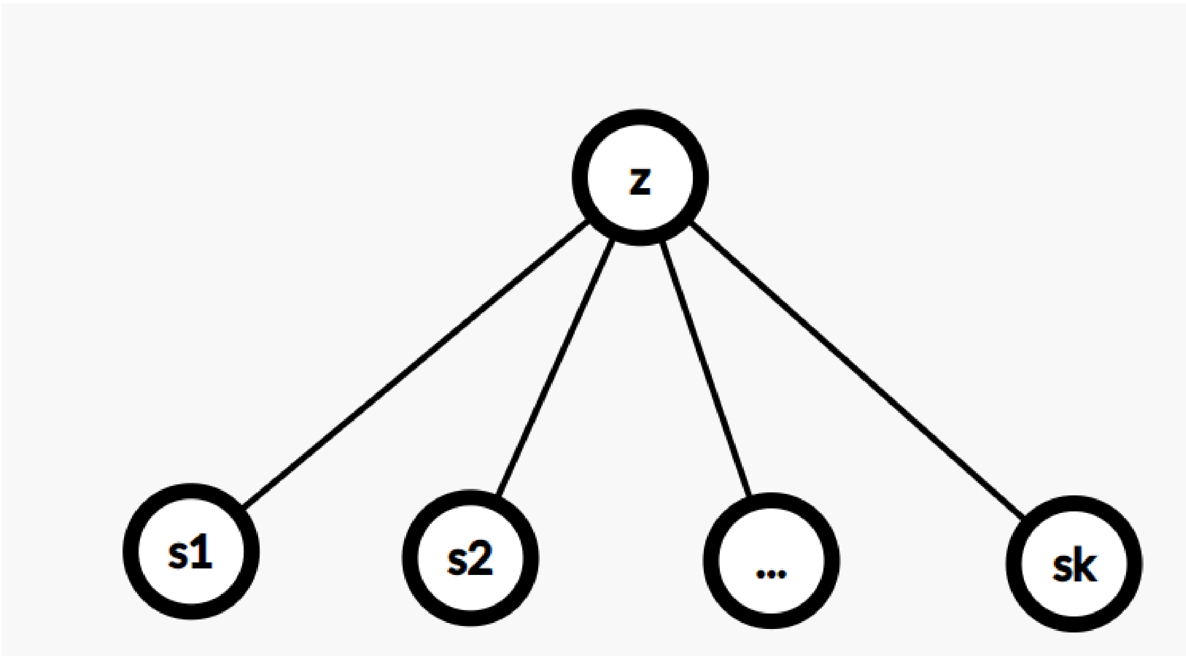



现在已经有了若干叶子构成的虚树$A$了，如果要新加入一个点$u$，可以先找到$A$的重心和在重心两侧的点$x, y$，然后询问出$d(u, x), d(u, y)$，就可以解出：




此时如果$z$不是重心，那么可以知道$u$在重心的哪一侧，类似点分治递归求解即可。

但如果$z$是重心，可以先排除在$x,y$侧，然后：




可以在$s_1,s_2\dots s_p$中各找一个点$q_1,q_2,\dots,q_p$，然后$D(\{q_1,q_2,\dots q_p\})$是可以自己算出来的，只要询问一下$D(\{q_1,q_2,\dots q_p,u\})$就可以知道$u$是否在这些子树内了。

二分求解就可以在$O(n\log^2 n)$次询问求出上面的问题。

考虑不正好二分在$\frac{k}{2}$处，可以找到最长的前缀$i$使得$\sum_{j\le i}|s_i|<\sum_{j>i}|s_i|\vee i=1$（$|s_i$表示子树$s_i$的大小），这样每两次二分都会使剩下的子树的总大小除以$2$，就可以做到$O(n\log n)$。

现在要找到根的位置，如果询问$D(\{u,v\})=D(\{v\})$，那么说明$v$是$u$的祖先，这样，可以用$n$次询问求出$u$的所有祖先，类似上面的，将1当成一个叶子，只不过单次询问需要$n$次操作。

现在求出了整个树的虚树，复杂度是$O(n \times \log n + 1 \times n \log n) = O(n \log n)$，这是一种返回同构的树的做法。

然后可以求出每一个点$u$的$D(\{u\})$，使用随机询问区分相同的$D(\{u\})$即可，这一部分远低于$O(n \log n)$。

在考虑区分$D(\{u\})$相同时，发现整个算法流程可以简化，即下面做法不依赖于树的形态，可以直接解决本题：

对于每一个节点$u$和一个集合$T$，如果$D(T) = D(T \cup \{u\})$，那么$T$内有$u$的祖先。

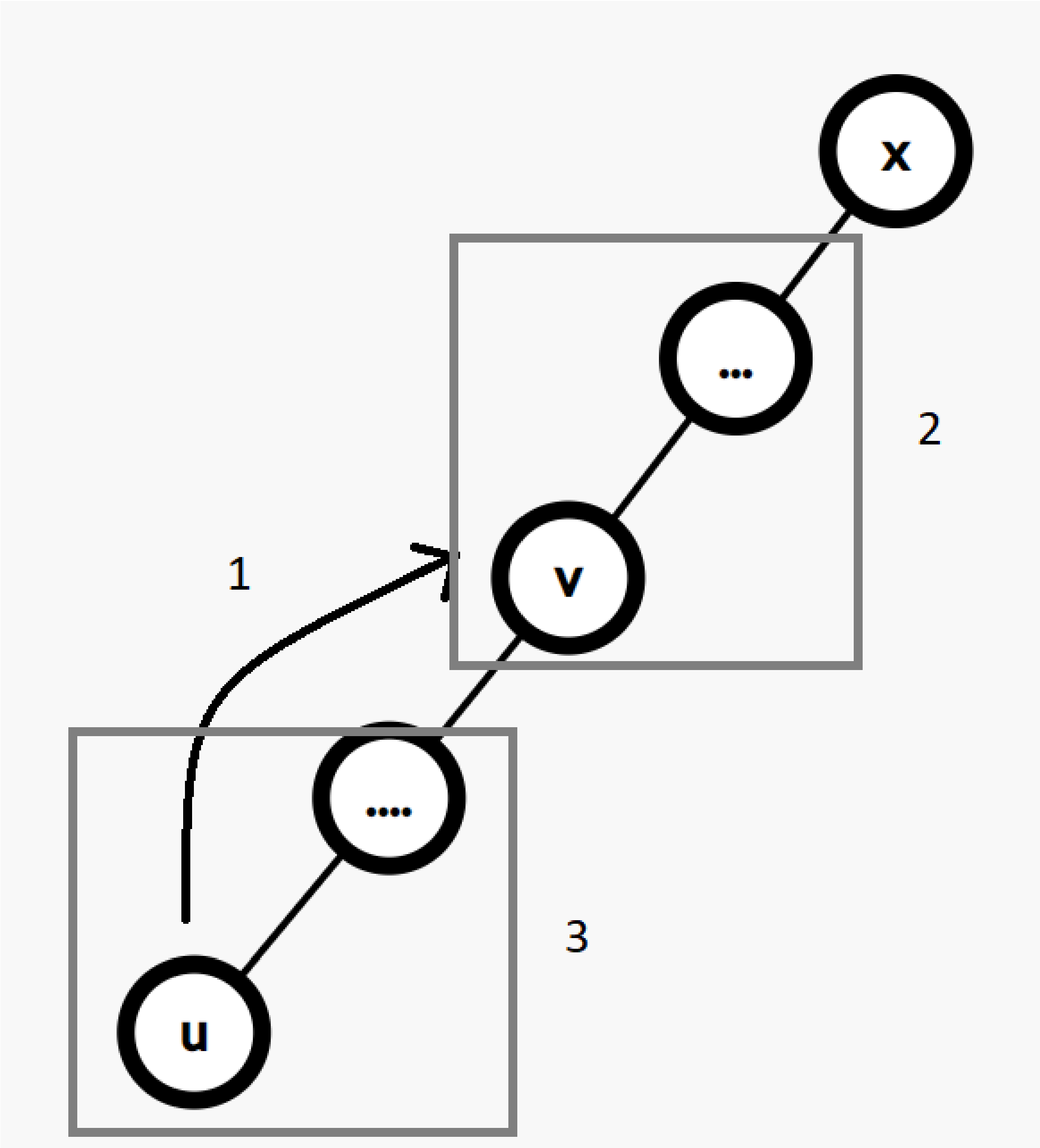
对于一个节点$u$和一个集合$T$，$u \in T$，令$u$的祖先集合为$F_u(T)$，且$F_u(T) \neq \{u\}$时，总是有$D(T) = D(T/\{u\})$，

不妨考虑二分$T_0, T_1$，至少有一个满足$D(T_k) = D(T_k/\{u\})$，这样二分下去，可以找到一个$u$的祖先$v$。

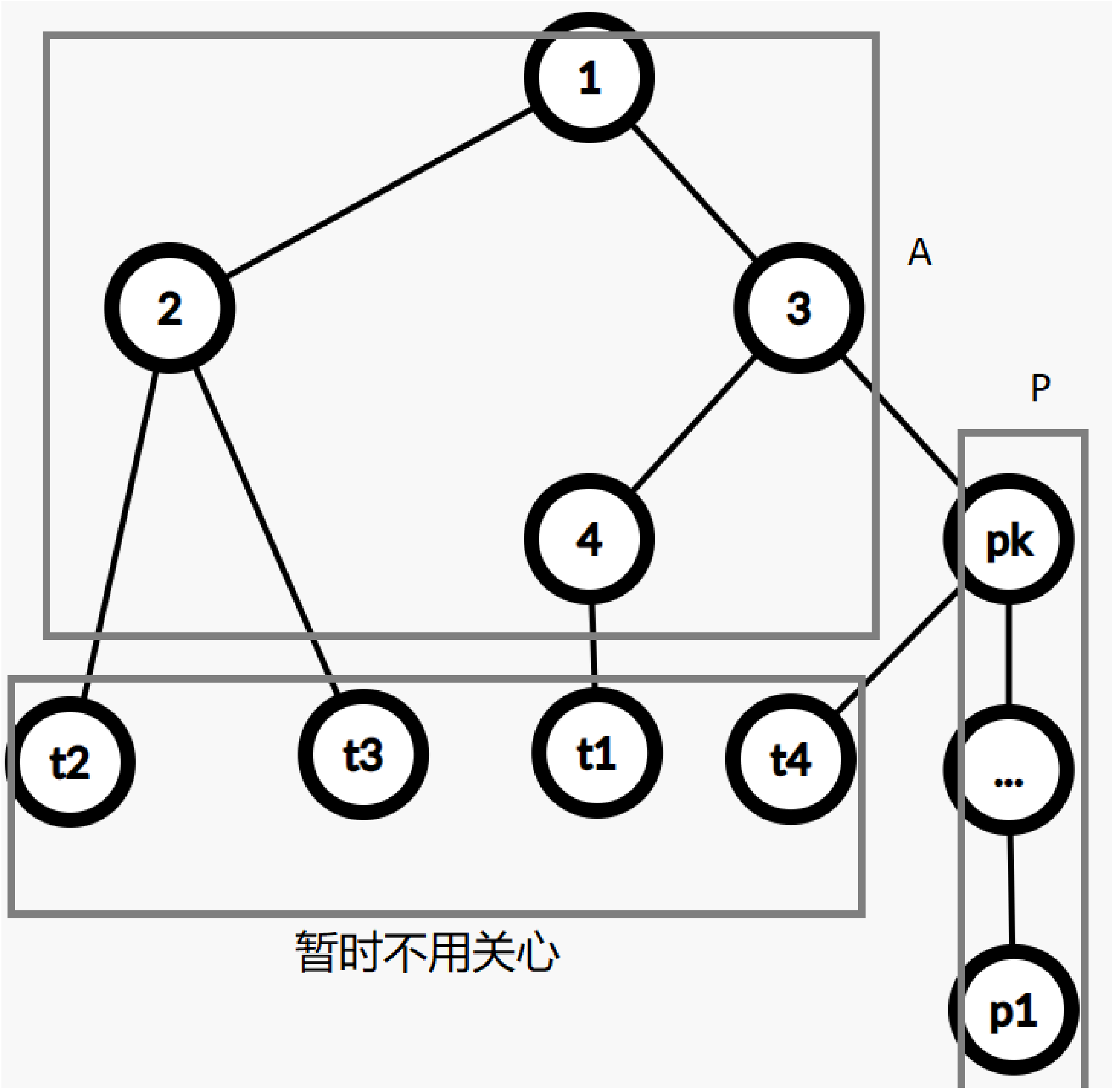
不妨令$F_u = [p_1 = u, p_2, p_3, \dots, p_m]$，其中$v = p_k$，我们不妨先处理完$p_{k \sim m}$，将这些从$T$中删去，再处理$p_{1 \sim k}$。

也就是说类似这样处理：

```
def solve(u):
    if(v=get(u)):
        solve(v)
        solve(u)
    else:
        T=T/{u}
```




然后就是找到$p_m$在全树$/T$中的父亲，不妨记全树$/T$这个集合为$A$。




考虑$A$的后序遍历$a_{1\sim k}$，它的一个前缀，就是$A$上若干子树的集合，因为如果$a_x$是$a_y$的祖先，那么$x > y$。

那么最短的前缀$a_{1\sim l}$，满足$D(a_{1\sim l}, p_m) = D(a_{1\sim l})$，就一定有$a_l$是$p_m$的祖先，又因为$a_o$是$p_m$的祖先时，由$o > l$可以推出$a_l$不是$a_o$的祖先。

所以我们要找的这个父亲，一定就是$a_l$。

对每一个叶子依次执行以上操作即可，复杂度$O(n \log n)$。

具体分析一下操作次数：因为每两次询问都可以确定一个点的位置，所以需要$2n\log_2 n + O(n)$步即可。