有根树

(r_tree.cpp/c/pas)

Time Limit : 2 sec , Memory Limit : 128MB

Description

图 $G=(V, E)$ 是一种数据结构，V 是一个有限顶点集合， E 是 V 上的二元关系，表示边的集合。图 1 展示了一个图的例子。

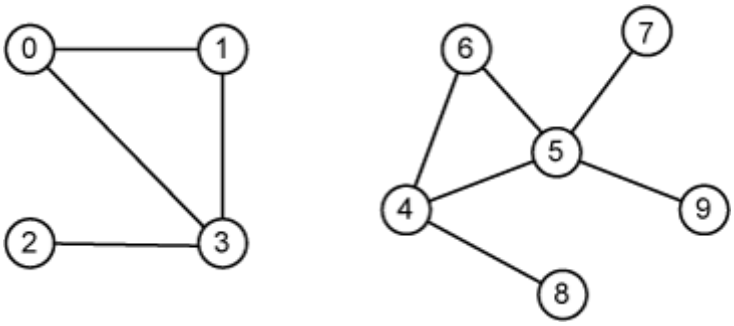

图 1

一棵自由树是连通的、无环的无向图。一棵有根树是一棵自由树，其中有一个被称为“根”的顶点和其他顶点有所不同。有根树的顶点被称为"结点" 。

你的任务是编写一个程序，输出给定的有根树 T 中每一个结点 u 的信息。信息内容如下：

- 结点 u 的 ID（编号）
- 结点 u 的父亲结点编号
- 结点 u 的深度
- 结点 u 的类型 (根, 内部结点 or 叶子结点)
- 结点 u 的孩子列表

对于有根树 T ，其根 r 到结点 x 的路径上的最后一条边连接着结点 p ，那么 p 是 x 的父亲，x 是 p 的孩子。

在有根树 T 中仅有根节点没有父亲。我们将没有子结点的结点称为外部结点或者叶子结点。除叶子结点以外的结点称为内部结点。

有根树 T 中结点 x 的孩子结点数称为 x 的度，从根 r 到结点 x 的路径长度称为 x 的深度。

这里我们设给定的树有 n 个结点，结点编号（ID）分别为 0 to n -1。

图 2 展示了一棵有根树的例子，树中每个结点的编号用圆圈里的数字来表示，这个图对应第一个样例输入。

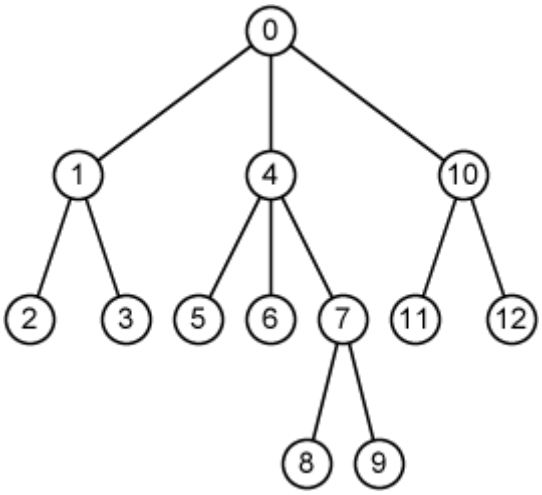

图 2

Input

第一行输入结点的数目 n。

接下来 n 行，每行输入一个结点的信息，输入格式如下：

id k c_1 c_2 ... c_k

id 是结点 u 的编号，k 是结点 u 的度，c_1 ... c_k 是结点 u 的 1 至 k 个孩子结点的编号。如果结点没有孩子，则 k 为 0。

Output

按照结点编号的升序输出每个结点的信息，内容格式如下：

node id: parent = p , depth = d, $type$, [c_1...c_k]

p 是父亲结点的编号。如果结点没有父亲，输出 -1。

d 是结点的深度。

$type$ 是结点的类型，用字符串 (root, internal node or leaf)来表示。如果根结点能被考虑为一个叶子结点或者内部结点，则输出 root.

$c_1 \ldots c_k$ 是孩子的结点的列表，按照有序树的样式（即严格按照输入的编号顺序）来输出。

请务必注意样例输出的格式，相邻信息用逗号和空格隔开。

Constraints

- $1 \le n \le 100000$

Sample Input 1

```
13
0 3 1 4 10
1 2 2 3
2 0
3 0
4 3 5 6 7
5 0
6 0
7 2 8 9
8 0
9 0
10 2 11 12
11 0
12 0
```

Sample Output 1

```
node 0: parent = -1, depth = 0, root, [1, 4, 10]
node 1: parent = 0, depth = 1, internal node, [2, 3]
node 2: parent = 1, depth = 2, leaf, []
node 3: parent = 1, depth = 2, leaf, []
node 4: parent = 0, depth = 1, internal node, [5, 6, 7]
node 5: parent = 4, depth = 2, leaf, []
node 6: parent = 4, depth = 2, leaf, []
node 7: parent = 4, depth = 2, internal node, [8, 9]
node 8: parent = 7, depth = 3, leaf, []
node 9: parent = 7, depth = 3, leaf, []
node 10: parent = 0, depth = 1, internal node, [11, 12]
node 11: parent = 10, depth = 2, leaf, []
node 12: parent = 10, depth = 2, leaf, []
```

Sample Input 2

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

Sample Output 2

```
node 0: parent = 1, depth = 1, leaf, []
node 1: parent = -1, depth = 0, root, [3, 2, 0]
node 2: parent = 1, depth = 1, leaf, []
node 3: parent = 1, depth = 1, leaf, []
```

Note

你可以利用“左孩子右兄弟表示法”（**left-child, right-sibling representation**）来实现一棵树，含有下列数据:

- 结点 u 的父亲结点
- 结点 u 最左侧的孩子结点
- 结点 u 右侧紧邻的兄弟结点