Trie 字典树

引例:外地人　nkoj 1934

你考入大城市沙坪坝的学校，但是沙坪坝的当地人说着一种很难懂的方言，你完全听不懂。幸好你手中有本字典可以帮你。现在你有若干个听不懂的方言需要查询字典。

输入格式:

若干行表示字典的内容，每行表示一条字典的记录。每条记录包含两个空格间隔的单词，第一个单词为英文单词，第二个单词为对应的沙坪坝方言。记录条数<=100,000

接下来是一个空行

接下来又有若干每行一个单词，表示你要查询的沙坪坝方言。单词个数<=100,000

输出格式:

输出若干行，每行一个英文单词，表示翻译后的结果。

如果某个单词字典查不到，输出"eh"

样例输入:

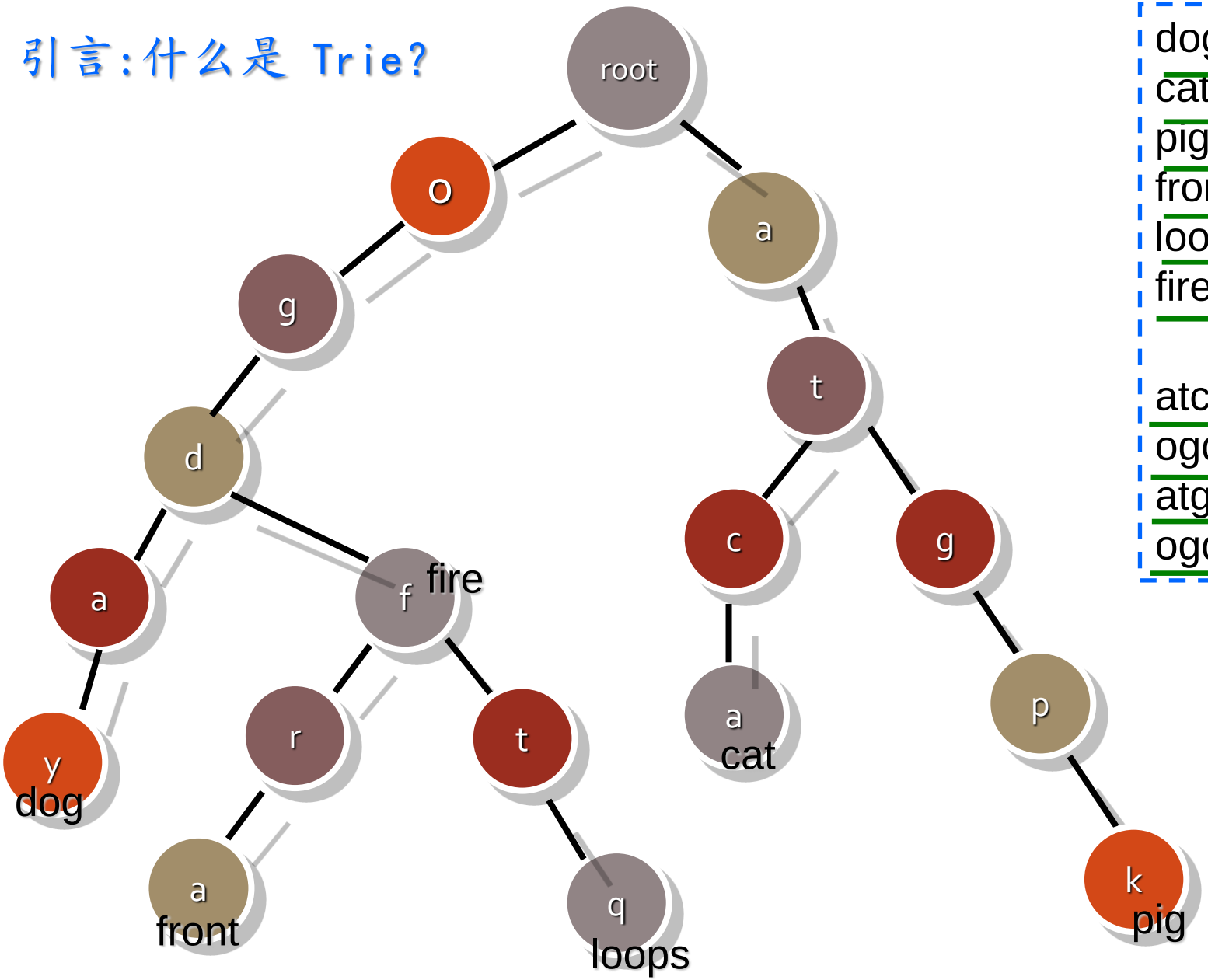
```
dog  ogday
cat  atca
pig  aigpk
front  ogdfra
loops  ogdftq
fire  ogdf

atca
ogdf
ittenkay
ogdftq
```

样例输出:

```
cat
fire
eh
loops
```

注: 所有单词都用小写字母表示，且长度不超过10

引言:什么是 Trie?



dog ogday
cat atca
pig atgpk
front ogdfra
loops ogdftq
fire ogdf

atca cat
ogdf fire
atgp 没有
ogdftw 没有

Trie是一种存储大量字符串的多叉树，优点是利用字符串的公共前缀来节约查找时间和存储空间。树高与字符串的长度有关，引例这个树高度最大为10。

Trie的特点：

1. 根节点不包含字符，除根节点外每一个节点都只包含一个字符。

2. 从根节点到某一节点，路径上经过的字符连接起来，为该节点对应的字符串。

3. 在trie树中查找一个关键字的时间和树中包含的结点数无关，而取决于组成关键字的字符数。也就是查找字符串s的时间为O(s.length())

4. 如果要查找的关键字可以分解成字符序列且不是很长，利用trie树查找速度优于二叉查找树。如：
若关键字长度最大是5，则利用trie树，利用5次比较可以从26^5＝11881376个可能的关键字中检索出指定的关键字。而利用二叉查找树至少要进行$\log_2 26^5$=23.5次比较。

Trie的实现1：nkoi1934

```
struct node{
        int Num;        //如果该节点是一个单词的结尾，记录对应单词的编号
        int Next[26];   //儿子节点的编号
    };
node trie[1000001];

string s[100001],a;

int main()
{
    cin>>n>>m;
    for(k=1;k<=n;k++){   cin>>s[k]>>a;  Insert(a,k);   }
    for(k=1;k<=m;k++)
    {
        cin>>a;
        ans=Find(a);
        if(ans)cout<<s[ans];
        else cout<<"eh"<<endl;
    }
    return 0;
}
```

Trie的实现2：

```
void Insert(string c,int k)
{
    int i,t,len,p=1;
    len=c.length();
    for(i=0;i<len;i++)
    {
        t=c[i]-'a';//将字符c[i]转换成值为0到25的数字，比如'a'转换为0，'b'转换为1， ‘c’ 转换为2.....
        if(trie[p].Next[t]==0)//若p没有值为t的儿子
        {
            tot++;                  //新增一个编号为tot的节点
            trie[p].Next[t]=tot;    //记下p的值为t的孩子节点的编号
            p=trie[p].Next[t];      //p指向新添加的节点
            trie[p].Num=0;          //初始化新添加的节点，将其标记为不是单词的结尾
        }
        else p=trie[p].Next[t];     //若p存在值为t的儿子，p指向该儿子,继续讨论
    }
    trie[p].Num=k; //for循环已执行完，说明第k个单词已加入，在单词结尾做上标记
}
```

Trie的实现3：

```
int Find(string c)
{
    int i,t,len,p=1;
    len=c.length();
    for(i=0;i<len;i++)
    {
        t=c[i]-'a';
        if(trie[p].Next[t]==0)return 0;      //当前要匹配值为t的字母,若没有则结束
        p=trie[p].Next[t];                   //若存在值为t的字母，则继续匹配
    }
    return trie[p].Num;                      //若for循环执行完毕，说明找到了需要的单词，返回其编号
}
```

例1:图书管理员　nkoj1935

你是一所学校的图书管理员，你的工作有如下几种操作:

A. 添加一本书
B. 借出一本书
C. 查找一本书

你这学期要完成n次操作，非常辛苦，你想编写一个程序来帮助你。

输入格式:

第一行一个整数n，表示操作的次数。

接下来n行，每行由一个大写字母和一个单词构成，大写字母表示操作的种类，单词表示对应的书名

输出格式:

对于每次查询，输出查询的结果，找到输出"yes",没找到输出"no"。

样例输入:

```
10
A newton
A alien
A newton
B newton
A nicole
C newton
B nicole
A pugna
C alien
C nicole
```

平衡树?

样例输出:

```
yes
yes
no
```

0<=n<=200000　书名由小写字母构成且长度不超过10，相同的书可能有多本

Trie的应用

（1） 字符串检索
（2） 字符串最长公共前缀

习题：1931, 1932, 1933， 1934, 1935