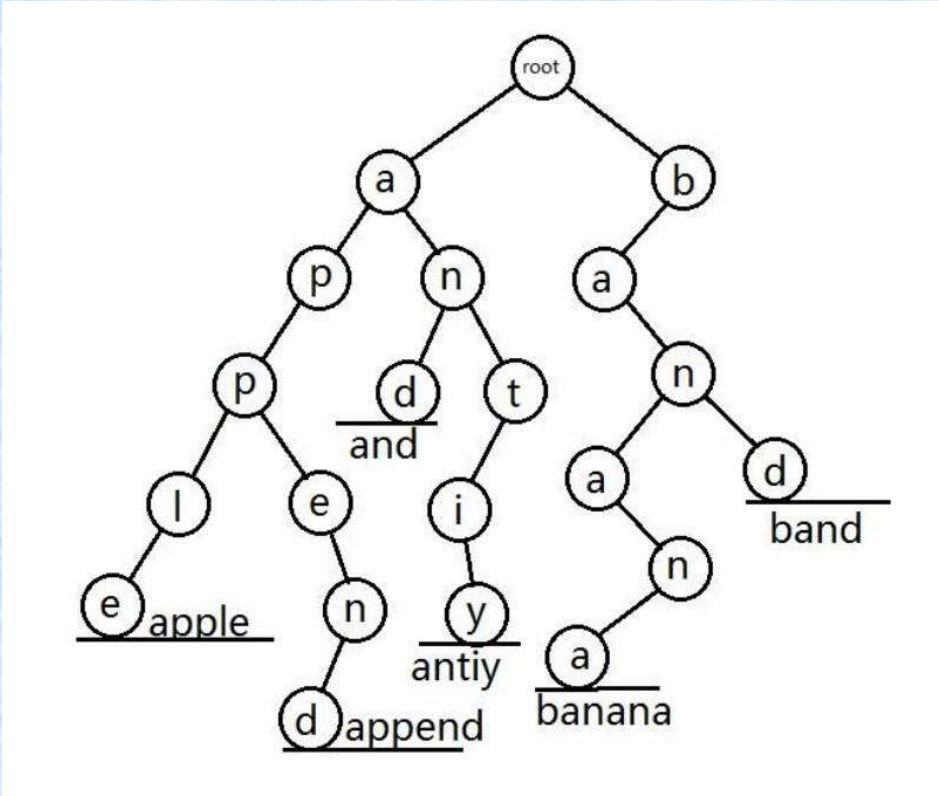
8.5 应用Trie树查询字符串的实验范例

吴永辉

ICPC Asia Programming Contest 1st Training Committee – Chair

yhwu@fudan.edu.cn

- 定义（**Trie**树）Trie树，也被称为单词查找树，前缀树，或字典树。其基本性质如下：
- 1，根节点不包含字符，除根节点外，每个节点只包含一个字符。
- 2，将从根节点到某一个节点的路上经过的节点所包含的字符连接起来，就是该节点对应的字符串。
- 3，对于每个节点，其所有子节点包含的字符是不相同的。


root
a
b
p
n
a
p
d
t
n
and
a
d
l
e
i
band
n
e
apple
n
y
a
antiy
banana
d
append

- Trie树的根节点*root*对应空字符串。一般情况下，不是所有的节点都有对应的值，只有叶节点和部分内节点所对应的字符串才有相关的值。所以，Trie树是一种用于快速检索的多叉树结构，每个节点保存一个字符，一条路可以用于表示一个字符串、一个电话号码等等信息。

- 表示一棵多叉树形式Trie树的存储方式是构建一个下标和字符一一映射的数组int *ch*[*maxnode*][*sigma_size*]来储存Trie树的节点，初始状态都为0。其中*maxnode*为节点数上限，Trie树的树根编号为0，其余节点从1开始编号；*sigma_size*为Trie树对应字符串的字符集的基数，比如，字符集是字符'a'到字符'z'的小写英文字母集，则*sigma_size*=26，而相应的下标对应相应的字符，下标0对应字符'a'，……，下标25对应字符'z'；*ch*[*i*][*j*]为节点*i*的编号为*j*的子节点，
- 比如，*ch*[0]表示根节点；*ch*[0][0]=1，表示根节点编号为0的子节点是节点1；*ch*[1][1]=2，表示节点1编号为1的子节点是节点2；*ch*[2][2]=3，表示节点2的编号为2的子节点是节点3，到这里，就是表示存储了一个"abc"字符串；而如果*ch*[2][0]=4；就表示还存了一个"aba"的串。也就是说，从根开始，通过*T*[0].*point*[0]可以找到*T*[1]，通过*T*[1]里面*point*数组第二个元素的值（索引），找到*T*[2]；再从*T*[2]的*piont*数组里面有第0个和第2个元素不为-1就表示存在字符串"aba"和"abc"。

- Trie树主要有两个操作：
- 1.将字符集构造成Trie树，简称插入操作；
- 2.在Trie树中查询一个字符串，简称查询操作。

8.5.1 Shortest Prefixes

- 试题来源：**ACM Rocky Mountain 2004**
- 在线测试：**POJ 2001**

- 字符串的前缀是从给出的字符串的开头开始的子字符串。"carbon"的前缀是："c"，"ca"，"car"，"carb"，"carbo"和"carbon"。在本题中，空串不被视为前缀，但是每个非空字符串都被视为其自身的前缀。在日常语言中，我们会用前缀来缩写单词。例如，"carbohydrate"（“碳水化合物”）通常被缩写为"carb"。在本题中，给出一组单词，请您为每个单词找出能唯一标识该单词的最短前缀。

- 在给出的样例输入中，"carbohydrate"可以被缩写为"carboh"，但不能被缩写为"carbo"（或者更短），因为有其他单词以"carbo"开头。
- 完全匹配也可以作为前缀匹配。例如，给出的单词"car"，其前缀"car"与"car"完全匹配。因此，"car"是"car"的缩写，而不是"carriage"或列表中以"car"开头的其他任何其他词的缩写。

- 输入
- 　输入至少有两行，最多不超过1000行。每行给出一个由1到20个小写字母组成的单词。
- 输出
- 输出的行数与输入的行数相同。输出的每一行先给出输入对应行中的单词，然后给出一个空格，最后给出唯一（无歧义）标识该单词的最短前缀。

试题解析

- 本题就是找能标识每个字符串自身的最短前缀，是一道基础的Trie树的试题。数组*val*记录每个节点的访问次数。则每个字符串的最短前缀，或者是到访问次数为1的那个字符节点为止的字符串，或者是遍历完毕还没有遇到访问次数为1的字符节点时，最短前缀就是其自身。

- 1．构建字符串*s*对应的Trie树
- 设当前节点编号为*u*，初始时为0，表示从根节点开始构建Trie树；子节点编号为*sz*，初始时为1；
- 依次枚举字符串*s*的每个字母*s*[*i*]（0≤*i*≤*s*的串长-1），按下述方法将之插入Trie树：
- 计算*s*[*i*]的序数值*c*（*c*= *s*[*i*]-‘a’）；
- 若节点*u*编号为*c*的子节点空（*ch*[*u*][*c*]==0），则节点*sz*为叶节点（memset(*ch*[*sz*], 0, sizeof(*ch*[*sz*])）；访问次数为0（*val*[*sz*] = 0）；且作为节点*u*编号为*c*的子节点，下一个子节点编号为*sz*+1（*ch*[*u*][*c*] = *sz*++）；
- 访问节点*u*的序值为*c*的子节点，沿该节点继续构建下去（*val*[*u*]++；*u*= *ch*[*u*][*c*]）。

- 2. 计算和输出单词*s*的最短前缀
- 从根出发（*u* = 0），依次枚举字符串*s*的每个字母*s*[*i*]（0≤*i*≤*s*的串长-1）：
- 输出前缀字母*s*[*i*]；
- 计算*s*[*i*]的序数值*c*（c= *s*[*i*]-‘a’）；
- 若节点*u*编号为*c*的子节点仅被访问1次（*val*[*ch*[*u*][*c*]] == 1），则说明*s*[*i*]是最短前缀的尾字符，退出计算；否则继续沿序数值*c*的子节点搜索下去（*u* = *ch*[*u*][*c*]）。

8.5.2 Phone List

- 试题来源：**Nordic 2007**
- 在线测试：**POJ 3630**

- 给定一个电话号码列表，确定它是否是一致的，即没有一个号码是另一个号码的前缀。假设电话目录中列出了这些号码：
- Emergency 911
- Alice 97 625 999
- Bob 91 12 54 26
- 在这种情况下，不可能给Bob打电话，因为只要您拨了Bob的电话号码的前三位，程控交换机就会把您的电话转到911。所以这份列表是不一致的。

- 输入
- 输入的第一行给出一个整数，$1 \le t \le 40$，表示测试用例的数目。每个测试用例首先的一行给出n，表示电话号码的数目，$1 \le n \le 10000$。接下来的n行，每行给出一个电话号码。电话号码最多是十位数字的序列。
- 输出
- 对于每个测试用例，如果列表是一致的，则输出“YES”，否则输出“NO”。

试题解析

- 本题是Trie前缀树的基础训练题。算法如下：
- 每次输入一个字符串，则插入Trie树中。边插入边判断，则会有下面的三种情况之一：
- 1. 当前插入的字符串从来没有被插入过，返回未冲突标志，继续插入下一条字符串；
- 2. 当前插入的字符串是已经插入过的字符串的前缀，停止插入，返回冲突标志，输出NO；
- 3. 当前插入的字符的前缀已经作为单独的字符串插入过，停止插入，返回冲突标志，输出NO。