字符串多模匹配算法之AC自动机理解心得

absolute8511 总结于 2009-2-26

AC自动机算法全称Aho-Corasick算法，是一种字符串多模式匹配算法。用于在一段文本中查找多个模式字符串。最近看到这个算法的一些文章，由于理解能力有限，琢磨了许久才有一些眉目，故记下此时的理解过程，防止过久了又要琢磨许久才能理解，也希望能帮助其他人加深理解，如有理解不当之处还望指出修正。^\_^

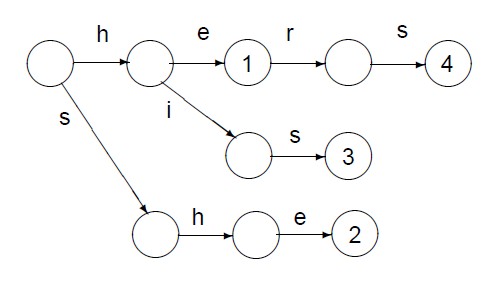
总结如下：

该算法有两个主要步骤，一个是字典树的构造，一个是搜索路径的确定。

**1. 字典树的构造**

这个比较好理解，就是把要匹配的一些字符串添加到树结构中去，树边就是单词中的字符，单词中最后一个字符的连接节点添加标志，以表示改节点路径包含1个字典中的字符串，搜索到此节点就表示找到了字典中的某个单词，可以直接输出。

例子：某字典P=｛he,she,his,hers｝对应的字典树如下图：

[](http://pic.yupoo.com/absolute8511/56592709f095/vfr6zyn9.jpg)

图中有数字的节点到根节点的路劲正好对应字典中的字符串，数字表述单词在字典中的顺序，也可以是其他标志。

【转载请注明出处：<http://hi.baidu.com/absolute8511/blog/item/73ffcbf293d86e14b17ec5e9.html>】

**2. 搜索路径的确定**

就是这部分我琢磨了很久，我的理解是 利用后缀字符串来确定。后缀字符串就是某个字符串的后面的一部分。比如abcde的后缀字符串有bcde,cde,de和e。

假定目标字符串为ushers，字典为上图所示。

搜索过程目标字符串指针指向的字符和字典中的字符会有以下几种情况：

a. 当前字符匹配，表示从当前节点沿着树边有一条路径可以到达目标字符，此时只需沿该路径走向下一个节点继续匹配即可，目标字符串指针移向下个字符继续匹配；

如：当指针指到s处，此时字典树指针处于根，要从根到s处，可以看到图中有一条从根经s连接到的节点，因此字典树节点指针指向此节点，目标字符串指针移动到下一字符h继续匹配；显然当前节点有一条经h连接到的节点，于是重复操作到有数字标志的节点2处，表示已找到，该匹配字符串就是"she"，输出该字符串的位置后，目标字符串指针增1指向"r"，字典指针指向数字2节点，进行下次匹配。

b. 当前字符无匹配，表示当前节点的任何一条边都无法达到要匹配的字符，此时不能沿现有路径前进，只能回溯，回溯到存在的最长的后缀字符串处，如果没有任何后缀字符串匹配则回溯到树根处。然后从当前回溯节点判断是否可以到达目标字符串字符。

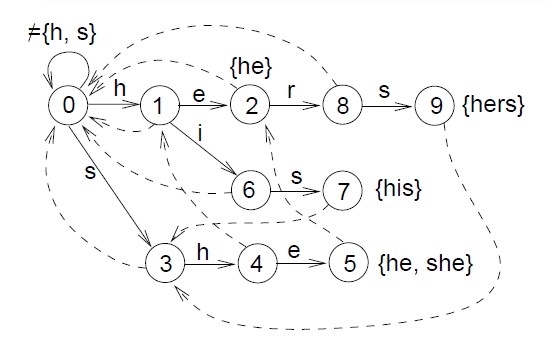
如：接上，由于数字2节点无经"r"的连接，因此回溯，she的后缀字符串he在字典树中，因此字典树指针指向带有数字1的标志节点，由于带有标志，直接输出该节点"HE"(存疑，很多文章没有提到此处需要输出，正常路径移动的字典指针节点要判断是否可以输出，那么由回溯路径改变的字典指针指向的节点要不要判断是否输出？)，然后从数字1节点判断是否有经"r"到下一节点的路径，显然图中有。因此字典树节点指向下一节点，重复以上操作，最后找到"hers"，此时匹配搜索也结束了。

以上两种情况直到目标字符串指针直到末尾结束匹配。在匹配过程中遇到有标志的节点说明找到了字典中的某个词，可以直接输出。

更新：输出说明：每次目标串指针移动前都需要判断当前节点是否可以输出，并递归的判断当前节点回溯路径上的节点是否可以输出（其实就是判断所有后缀字符串，she匹配时，其后缀he也会匹配，即使she不匹配，其后缀he也可能匹配，因此需递归判断后缀字符串），直到树根结束递归。

由于固定字典的字符串的后缀字符串都是已知的，因此可以在字典树结构中存储匹配失败的路径方向，因此只要字典树构造完毕，就可以根据字典树的路径进行匹配了，效率非常快。以上就是我对该算法的全部过程的理解，疏漏之处在所难免。

附1：含匹配失败的情况的路径选择的字典树，实线表示匹配成功的正常路径，虚线表示失败的回溯路径

[](http://pic.yupoo.com/absolute8511/2904470a0214/vwepyibw.jpg)

附2：伪代码实现

T为目标字符串，长度为m，q为字典树的节点指针，g函数返回从节点q经过路径T[i]到达的下一节点指针，f函数返回节点q的回溯节点指针。flag判断节点是否为标志节点

q := 0; // initial state (root)  
for i := 1 to m do  
    while g(q,T[i]) = NULL do  
        q := f(q); // 回溯

    q := g(q,T[i]); // 前进

    node:=q;

    while(node!=root){  
        if flag(node) exist ; then print i, out(node);

        node = f(node);   //查找回溯节点

    }  
endfor;