**AC自动机**

2008-02-25 21:41

|  |
| --- |
| 关键字:AC自动机 自动机 有限状态自动机 Trie 字母树 字符串匹配 多串匹配算法  Note:阅读本文需要有KMP算法基础，如果你不知道什么是KMP，请看这里：  <http://www.matrix67.com/blog/article.asp?id=146>   (Matrix67大牛写的)  AC自动机是用来处理多串匹配问题的，即给你很多串，再给你一篇文章，让你在文章中找这些串是否出现过，在哪出现。也许你考虑过AC自动机名字的含义，我也有过同样的想法。你现在已经知道KMP了，他之所以叫做KMP，是因为这个算法是由Knuth、Morris、Pratt三个提出来的，取了这三个人的名字的头一个字母。那么AC自动机也是同样的，他是Aho-Corasick。所以不要再YY地认为AC自动机是AC(cept)自动机，虽然他确实能帮你AC一点题目。  。。。扯远了。。。  要学会AC自动机，我们必须知道什么是Trie，即字母树。如果你会了，请跳过这一段          Trie是由字母组成的。          先看张图： http://hiphotos.baidu.com/lightxianjian/pic/item/49db91b4a21449638ad4b2c8.jpg  这就是一棵Trie树。用绿色标出的点表示一个单词的末尾（为什么这样表示？看下去就知道了）。树上一条从root到绿色节点的路径上的字母，组成了一个“单词”。         /\* 也许你看了这一段，就知道如何构建Trie了，那请跳过以下几段。\*/          那么如何来构建一棵Trie呢？就让我从一棵空树开始，一步步来构建他。  一开始，我们有一个root：  http://hiphotos.baidu.com/lightxianjian/pic/item/715550d502a2bad450da4b1a.jpg  现在，插入第一个单词，she。这就相当于在树中插入一条链。过程很简单。插完以后，我们在最后一个字母’e’上加一个绿色标记，结果如图：  http://hiphotos.baidu.com/lightxianjian/pic/item/fd8784fcc4eaed92b801a0d4.jpg          再来一个单词，shr(什么词？…..右位移啊)。由于root下已经有’s’了，我们就不重复插入了，同理，由于’s’下有’h’了，我们也略过他，直接在’h’下插入’r’，并把’r’标为绿色。结果如图：  http://hiphotos.baidu.com/lightxianjian/pic/item/eb4fbb1f542b45e5e0fe0bd0.jpg         按同样的方法，我们继续把余下的元素插进树中。         最后结果:  http://hiphotos.baidu.com/lightxianjian/pic/item/d30e65ddf2191bc98c1029e3.jpg       也就是这样:  http://hiphotos.baidu.com/lightxianjian/pic/item/49db91b4a21449638ad4b2c8.jpg    好了，现在我们已经有一棵Trie了，但这还不够，我们还要在Trie上引入一个很强大的东西：失败指针或者说shift数组或者说Next函数 …..你爱怎么叫怎么叫吧，反正就是KMP的精华所在，这也是我为什么叫你看KMP的原因。  KMP中我们用两个指针i和j分别表示，A[i-j+ 1..i]与B[1..j]完全相等。也就是说，i是不断增加的，随着i的增加j相应地变化，且j满足以A[i]结尾的长度为j的字符串正好匹配B串的前 j个字符，当A[i+1]<>B[j+1]，KMP的策略是调整j的位置（减小j值）使得A[i-j+1..i]与B[1..j]保持匹配且新的B[j+1]恰好与A[i+1]匹配（从而使得i和j能继续增加）。  Trie树上的失败指针与此类似。          假设有一个节点k，他的失败指针指向j。那么k,j满足这个性质：设root到j的距离为n，则从k之上的第n个节点到k所组成的长度为n的单词，与从root到j所组成的单词相同。          比如图中she中的’e’的失败指针就应该指向her中的’e’。因为：  http://hiphotos.baidu.com/lightxianjian/pic/item/6a6f7dd3e73fde103af3cfdc.jpg  图中红框部分是完全一样的。  那么我们要怎样构建这个东西呢？其实我们可以用一个简单的BFS搞定这一切。  对于每个节点，我们可以这样处理：设这个节点上的字母为C，沿着他父亲的失败指针走，直到走到一个节点，他的儿子中也有字母为C的节点。然后把当前节点的失败指针指向那个字目也为C的儿子。如果一直走到了root都没找到，那就把失败指针指向root  最开始，我们把root加入队列(root的失败指针显然指向自己)，这以后我们每处理一个点，就把它的所有儿子加入队列，直到搞完。  至于为什么这样就搞的定，我们讲下去就知道了。  好了，现在我们有了一棵带失败指针的Trie了，而我的文章也破千字了，接下来，我们就要讲AC自动机是怎么工作的了。  AC自动机是个多串匹配，也就是说会有很多串让你查找，我们先把这些串弄成一棵Trie，再搞一下失败指针，然后我们就可以开始AC自动机了。  一开始，Trie中有一个指针t1指向root，待匹配串(也就是“文章”)中有一个指针t2指向串头。  接下来的操作和KMP很相似：如果t2指向的字母，是Trie树中，t1指向的节点的儿子，那么t2+1,t1改为那个儿子的编号，否则t1顺这当前节点的失败指针向上找，直到t2是t1的一个儿子，或者t1指向根。如果t1路过了一个绿色的点，那么以这个点结尾的单词就算出现过了。或者如果t1所在的点可以顺着失败指针走到一个绿色点，那么以那个绿点结尾的单词就算出现过了。  我们现在回过来讲讲失败指针。实际上找失败指针的过程，是一个自我匹配的过程。http://hiphotos.baidu.com/lightxianjian/pic/item/49db91b4a21449638ad4b2c8.jpg  如图，现在假定我们确定了深度小于2(root深度为1)的所有点的失败指针，现在要确定e。这就相当于我们有了这样一颗Trie：http://hiphotos.baidu.com/lightxianjian/pic/item/c7af30ec0d3d51c12e2e21e8.jpg  而文章为’she’，要查找’e’在哪里出现。我们接着匹配’say’，那’y’的失败指针就确定了。  好好想想。前面讲的BFS其实就是自我匹配的过程，这也是和KMP很相似的。  好了，就写到这吧，有不明白可以留言或发邮件给我 |