提示一：KMP的思路

小Hi和小Ho回到了学校，为了完成河蟹托付的伟大使命，小Hi立马把小Ho抓到了机房开始上课。

“小Ho，你来看看这样一段原串和模式串~”小Hi说着递上了一张纸条。

|  |  |
| --- | --- |
| 原串： | bababababababababb |
| 模式串： | bababb |

“嗯，这个例子中**模式串bababb在原串中第13个字符开始的地方出现了**”小Ho看了看，回答道。

“我们假设仍然使用最普通的方法来进行判断，即我们先枚举原串中的一个起始位置，然后判断从这个位置开始的字符串是否能和模式串进行完匹配。”小HI说道，“然后我们来看这个过程中有没有什么可以缩减的计算量。”

“好的！”小Ho点点头。

“你看，**在起始点为1的时候，匹配到第6个字符的时候发生了失败，这个时候我们应当做的是是不是将模式串右移一位，然后从头开始判断，**就像这样？”小Hi又在纸上画了画，递给了小Ho。“

|  |  |
| --- | --- |
| 原串： | babab**a**babababababb |
| 模式串： | babab**b** |
| 原串： | b**a**bab**a**babababababb |
| 模式串： | **b**ababb |

”是的，然后我们发现第一位就发现不能进行匹配。“小Ho老老实实的回答。

”然后我们再将模式串右移一位，然后再从头开始判断，这次我们成功的越过了原串的第7个字符，在第8个字符产生了不同。“小Hi继续往下推演。

|  |  |
| --- | --- |
| 原串： | bababab**a**bababababb |
| 模式串： | babab**b** |

”然后之后的剧情非常的相似，都是要么最后一个字符匹配不成功，要么就是第一个字符就匹配不成功，一直到了最后一次机会的时候才匹配成功。“小Ho做了总结。

”那你觉得这个过程中有没有什么没有必要计算的呢？“小Hi于是问道。

”我是这么认为的，你看这条线。“小Ho在两个串上对着的一个位置画了一条线。

|  |  |
| --- | --- |
| 原串： | babab | **a**babababababb |
| 模式串： | babab | **b** |

”嗯？”

“这是我们第一次产生了**字符不匹配的情况**，那么接下来的过程中**一定会出现两种情况之一：一种情况是模式串与原串的对齐点（即枚举的原串中的起点位置）越过了这条线，仍然没能匹配成功，而另一种情况是原串中这个位置的字符与模式串中某个位置的字符匹配上了。**”小Ho分析道：”我们先不考虑第一种情况，而来看看第二种情况会发生什么。“

|  |  |
| --- | --- |
| 原串： | babab | **a**babababababb |
| 模式串(对齐点=1)： | babab | **b** |
| 模式串(对齐点=3)： | bab | a |

”看不出嘛，小Ho你今天变成聪明了嘛！~”小Hi由衷的赞叹道。

“那当然，毕竟我最近在讨论区解答了很多问题，这很锻炼人的好么！“小Ho笑嘻嘻的回答道。

”那我也得表现下，接下来换我来说吧，反正你肯定也就差不多想到这么多是吧！“小Hi也是看破了小Ho的底细，这般说道。于是小Ho点了点头，让小Hi接着说。

”我相信一个很容易注意到的事实就在于，如果我用i表示原串和模式串产生分歧的位置（模式串上的位置，注意！这个和对齐点是不一样的东西，一个在原串上，一个在模式串上），用j表示为了匹配掉位置i上产生分歧的字符而将模式串的对齐点移动到的位置，我们会发现，模式串[1, i-j]的这一段和[j, i - 1]这一段是相同的。比如在这个例子中i=6,j=3，我们会发现模式串[1, 3]和[3,5]是相同的。“小Hi整理了下思路，如是说道。

|  |  |
| --- | --- |
| 原串： | ba | bab | **a** babababababb |
| 模式串(i=1)： | ba | bab | **b** |
| 模式串(i=3)： | | bab | a |

”而我们同时也会发现，只有在存在一个长度k，使得模式串[1, i-k]和[k, i-1]这两段相同的情况下，将模式串对其到位置k，才能保证原串和模式串的匹配过程能够进入到原串的位置i是否和模式串的对应字符相同的判定，在别的情况下，根本都进入不到位置i的判断就会发生不一致的情况了。”说着小Hi又抛出了另外一个命题。

“我已经开始有点晕了！”小Ho提出了抗议。

“那你就好好的读一遍我刚才说的话！然后自己在草稿纸上演算一下这个样例，很快就可以得出结果的！”小Hi如是说道。”总而言之我们现在需要的一个数据是，这个长度k最长是多少，而且我们对于模式串的每一个位置i，都要计算这个值。”而这就是KMP中最为重要的一个点——NEXT数组。

提示二：NEXT数组的使用

“那么，为了能够充分理解NEXT数组，我们再来回顾一下如何使用NEXT数组~"小Hi摆出一副老师的样子，说道。”首先我们来给出NEXT数组的数学定义~“

NEXT[0] = -1

NEXT[i] = max{ 0<=k< i | str.substring(1, k) == str.substring(i - k +1 , i) } 其中str.substring(i, j)表示str从位置i到位置j的子串，如果i>j则,substring为空

”那么我们对之前例子中的模式串进行求解，可以得到这样的NEXT数组。“小Hi在纸上写了又写，画了又画。

|  |  |
| --- | --- |
| 模式串： | b a b a b b |
| NEXT： | 0 0 1 2 3 1 |

”然后再来看这个NEXT数组是如何使用的！**为了表明NEXT的所有使用情况，我们换一个原串。然后首先，我们第一次匹配，如果用ori表示原串，用par表示模式串，用p表示原串的下标（从1开始），用q表示模式串的下标（从1开始）的话，会发现最多匹配到p=5, q=5就不能往下匹配了，因为此时ori[p +1]不等于par[q + 1]**“小Hi为了使说明更加简洁，先下了一堆定义。

”好的！小Hi老师好棒！“小Ho在一旁煽风点火道。

|  |  |
| --- | --- |
| 原串(p=5)： | babab | **a**bcbababababb |
| 模式串(q=5)： | babab | **b** |

”此时，令q = NEXT[q]，并将ori[1..p]和par[1..q]对齐，便会发现ori[1..p]和par[1..q]仍然是一一对应的。“

|  |  |
| --- | --- |
| 原串(p=5)： | babab | abcbababababb |
| 模式串(q=3)： | bab | abb |

“此时，ori[p+1]和par[q+1]相同了，于是可以继续往下匹配，但是到了p=7,q=5的时候又发现不能够接着匹配了。”

|  |  |
| --- | --- |
| 原串(p=7)： | bababab | **c**bababababb |
| 模式串(q=5)： | babab | **b** |

”此时，令q = NEXT[q]，并将ori[1..p]和par[1..q]对齐，便会发现ori[1..p]和par[1..q]仍然是一一对应的，这和之前是一样的。”

|  |  |
| --- | --- |
| 原串(p=7)： | bababab | **c**bababababb |
| 模式串(q=3)： | bab | **a**bb |

“此时，ori[p+1]和par[q+1]仍然不相同，于是还得令q=NEXT[q]。”

|  |  |
| --- | --- |
| 原串(p=7)： | bababab | **c**bababababb |
| 模式串(q=1)： | b | **a**babb |

“此时，ori[p+1]和par[q+1]仍然不相同，令q=NEXT[q]。”

|  |  |
| --- | --- |
| 原串(p=7)： | bababab | **c**bababababb |
| 模式串(q=0)： | | **b**ababb |

“此时，ori[p+1]和par[q+1]仍然不相同，令q=NEXT[q]。”

|  |  |
| --- | --- |
| 原串(p=7)： | bababab | **c**bababababb |
| 模式串(q=-1)： | |   bababb |

”到了这一步，就相当于我们之前所说的**模式串与原串的对齐点（即枚举的原串中的起点位置）越过了这条线（当时指C右侧的那条线）的情况，这种情况下，就应当p和q均+1，然后继续之前的操作。”小Hi擦了一把汗，说道。**

“这样一说，我就大致能够理解NEXT数组是怎么用来求解模式匹配问题的了，但是它是如何求的呢？一般的方法不是要O(模式串长度的立方）的么？”小Ho问道。

“这就是我接下来要和你说的啦！”小Hi笑道：“但是让我先喝口水！”

提示三：如何求解NEXT数组

“首先我们不想如何求整个NEXT数组，而是假设我们已经知道了之前例子中模式串的NEXT[1..4]，来求NEXT[5]如何？”小Hi建议道。

“好的！这样我们就只需要平方级的算法就可以算出它的值了！”小Ho高兴道。

“有点追求好不好！”小Hi深深的吸了一口气：“你这样和之前的解法有什么不同么！”

“似乎没有。。那你说怎么算吧！我反正脑子已经成浆糊了。”小Ho郁闷道。

“我们把par.substring(1, 5)当做新的原串ori\_new，然后把par.substring(1, 4)当做新的模式串par，会如何？”小Hi微微一笑。

“会。。我来试试！"小Ho接过小Hi手中的纸笔，便开始演算：“首先就直接匹配到了p=4, q=4的情况，这时候严格来说已经算匹配完成了，但是肯定不是就这么结束的，此时par\_new[q +1]因为是空字符，所以肯定和ori\_new[p+1]匹配不上。于是令q = NEXT[q]”

|  |  |
| --- | --- |
| 原串(p=4)： | baba | **b** |
| 模式串(q=4)： | baba | |
| 原串(p=4)： | baba | **b** |
| 模式串(q=2)： | ba | **b** |

”然后这时候ori\_new[p + 1]就直接和par\_new[q + 1]匹配上了，于是新的p=5，q=3，莫非……这个最后的q就是NEXT[5]！“小Ho忽然灵光一闪。

”没错，就是这样！那你想想现在如何求NEXT[6]。“小Hi继续引导小Ho。

”首先我们没有必要重新从头开始匹配，直接在原串和模式串的后面加上第6个字符就可以了。“小Ho分析道。

|  |  |
| --- | --- |
| 原串(p=5)： | babab | **b** |
| 模式串(q=3)： | bab | **a**bb |

”没法继续匹配，于是令q=NEXT[q]。“

|  |  |
| --- | --- |
| 原串(p=5)： | babab | **b** |
| 模式串(q=1)： | b | **a**babb |

”还是没法继续匹配，于是令q=NEXT[q]。“

|  |  |
| --- | --- |
| 原串(p=5)： | babab | **b** |
| 模式串(q=0)： | | **b**ababb |

”此时可以匹配了，新的p=6,q=1，所以NEXT[6]就是1！“小Ho高兴道：”没想到NEXT数组的本身会用一种递归的方式进行求解，真是太巧妙了！“

”那你要不要赶紧去写一下代码，KMP算法的代码可是可以写的很短很巧妙的哦！~“小Hi建议道。

”好！“