字符串匹配

KMP

KMP

KMP算法是一种改进的字符串匹配算法，由D.E.Knuth，J.H.Morris和V.R.Pratt同时发现，因此人们称它为克努特——莫里斯——普拉特操作（简称KMP算法）。KMP算法的关键是利用匹配失败后的信息，尽量减少模式串与主串的匹配次数以达到快速匹配的目的。具体实现就是实现一个next()函数，函数本身包含了模式串的局部匹配信息。时间复杂度O(m+n)。

朴素的匹配

举个例子

原串:
bababababababababb
模式串:
bababb
第一次匹配
bababababababababb
bababb
在起始点为1的时候，匹配到第6个字符失败了，按照朴素的匹配方法，应该把模式串整体右移一位

第二次匹配
bababababababababb
 bababb
此时第一个字符就匹配失败了，这个时候就要再移一位这样匹配下去。。。

直到最后一次匹配才能成功

bababababababababb

bababb

那么朴素的匹配算法很慢 如果原串和模式串的长度分别为n,m的话

匹配的复杂度为O(n*m)

那么能不能考虑一种更优的匹配算法

观察第一次匹配的时候

babab|**a**bababababababb

babab|**b**

这样的话

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。

我们先来考虑第二种情况是怎么样的

babab|**a**bababababababb
babab|**b**
原串：　　babab|abababababababb
模式串(i=1)：babab|b
模式串(i=3)：　　bab|ab
用i表示原串与模式串产生分歧的地方，用j表示为了消除掉分歧而将模式串的点对其到的位置
我们发现模式串[1,i-j]与[j,i-1]的这一段相同，比如这次的i=6，j=3，模式串的[1,3]和[3,5]是相同的。

也就是必须找到一个K

使得模式串的[1,i-k]与[k,i-1]这一段相同，

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，如果没有的话，根本进入不到i的位置就会发生不能匹配的问题了

我们要求的就是对于模式串的每个位置i所对应的最长长度的k，这是KMP最为重要的next数组

NEXT数组的使用

next数组的定义

NEXT[0]=-1;

NEXT[i]=max{0<=k<=i|str.substring(1,k)==str.substring(i-k+1,i)}

substring(i,j)代表str这个字符串(i,j)这一段，若i>j，则代表空

根据定义写出之前字符串的NEXT数组

NEXT数组的使用

字符串：ｂａｂａｂｂ

NEXT ：００１２３１

NEXT数组的使用

换一个原串bababababcbabababababb
用ori表示原串，par表示模式串，p表示原串的下标
q表示模式串的下标
原串(p=5):　　babab|abcbabababababb
模式串(q=5):　babab|b
第一次匹配失败，因为这个时候
ori[q+1]!=par[p+1]
此时令q=nex[q],q=3

NEXT数组的使用

q=3 移动模式串并且对齐

原串(p=5):　　babab|abcbabababababb

模式串(q=3):　　bab|abb

这个时候发现ori[1...p]=par[1...q]还是一一对应，

这个时候ori[p+1]=par[q+1]相同了，可以继续匹配

原串(p=7):　　bababab|cbabababababb

模式串(q=5):　　babab|b

q=next[q],q=3

原串(p=7):　　bababab|cbabababababb

模式串(q=3):　　bab|abb

NEXT数组的使用

q=3,q=next[q],q=1
原串(p=7):　　bababab|cbabababababb
模式串(q=1):　　b|ababb
q=1,q=next[q],q=0
原串(p=7):　　bababab|cbabababababb
模式串(q=0):　　|bababb
q=0,q=next[q],q=-1
原串(p=7):　　bababab|cbabababababb
模式串(q=0):　　| bababb
这个时候枚举原串的对起点超过了这条线，也就是之前说过的第一种情况，这个时候应该p和q均+1然后继续

NEXT数组的使用

q=3,q=next[q],q=1
原串(p=7):　　bababab|cbabababababb
模式串(q=1):　　b|ababb
q=1,q=next[q],q=0
原串(p=7):　　bababab|cbabababababb
模式串(q=0):　　|bababb
q=0,q=next[q],q=-1
原串(p=7):　　bababab|cbabababababb
模式串(q=0):　　| bababb
这个时候枚举原串的对起点超过了这条线，也就是之前说过的第一种情况，这个时候应该p和q均+1然后继续

NEXT数组的求解

假设我们已经知道了之前的模式串NEXT[1...4]，现在要求解NEXT[5]的

NEXT数组的求解

把模式串的substring(1,4)当作新的模式串，把模式串的substring(1,5)当作原串

原串(p=4)：　baba|b

模式串(q=4)：baba|

这个时候par[q+1] 是空字符肯定是匹配不上的，也就是q=next[q],q=2;

原串(p=4)：　baba|b

模式串(q=2)：　ba|ba

这个时候匹配上了，也就是NEXT[5]=2;

如果要求解NEXT[6]呢?

NEXT数组的求解

如果要求解NEXT[6]，没有必要重新开始匹配直接在字符串的后面加上字母就行了

原串(p=5)：　　babab|b

模式串(q=3)：　　bab|a

没法继续匹配

q=next[q],q=1

原串(p=5)：　　babab|b

模式串(q=3)：　　b|aba

没法继续匹配

q=next[q],q=0

原串(p=5)：　　babab|b

模式串(q=0)：　　|baba

匹配成功

NEXT[6]=1;