

病毒的 DNA

——剖析一道字符匹配问题解析过程

长沙长郡中学 饶向荣

题目描述

有一种奇特的病毒，它的 DNA 序列是环状的，而一般的生物的 DNA 都是线状的，且由科学家发现：生物被此种病毒侵袭的可能性与生物和病毒的 DNA 序列最大公共长度有关，由于病毒是环状的，所以它可以循环重复的匹配。科学家们经过大量的试验发现：如果生物和病毒 DNA 序列的最大公共部分的长度还没有病毒的 DNA 长，病毒是无法安身的，也就是说这个生物被侵染的几率是 0，否则，最长公共部分的长度和被侵染的几率满足下面的关系式：

生物被侵染几率 = 最大公共部分长度 / 生物 DNA 长度。

任务

现在已知病毒的 DNA 序列和某生物的 DNA 序列，你必须求出此生物被侵染的几率是多少。

题目描述

数据范围

病毒的 DNA 序列长度 ≤ 1000 , 生物 DNA 序列长度 $\leq 10^5$ 。

样例

病毒的 DNA 序列 (环状) 为 abc , 生物的 DNA 序列为 $abbcabcabb$, 那么它们的最长的公共长度为 7 , 最长公共部分为红色部分 : $bcabcab$ 。

分析和求解

设 A 为病毒的环状 DNA 字串， A 的长度为 N 。

设 B 为生物的线状 DNA 字串， B 的长度为 M 。

那么题目所求：环串 A 和线串 B 的最大可循环公共子串长度。

根据平时的解题经验，很容易想到用动态规划来解此类求公共最大长度的题目，而且稍加分析就可设计相应的动态规划：

设 $f[i, j]$ 表示以线串 B 的第 i 位和环串 A 的第 j 位结尾的最大公共子串的长度。

分析和求解

那么动态转移方程：

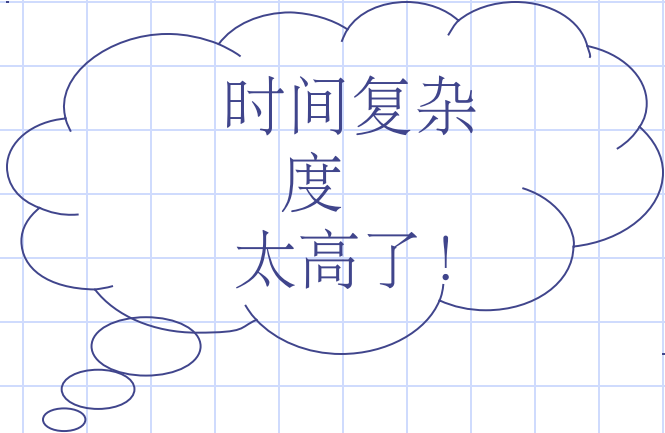
$$f[i, j] = \begin{cases} f[i-1, j-1] + 1, & A[j] = B[i] \text{ 且 } 1 < j \leq n \\ f[i-1, n] + 1, & A[j] = B[i] \text{ 且 } j = 1 \\ 0 & A[j] \neq B[i] \end{cases}$$

最后的答案：

$$Ans = \max(f[i, j], 1 \leq i \leq m, 1 \leq j \leq n)$$

空间复杂度为 $O(N)$ 。

分析和求解



时间复杂度
太高了!

时间复杂度： $O(M*N)$ 。 $n \leq 1000, m \leq 10^5$ 。

优化：

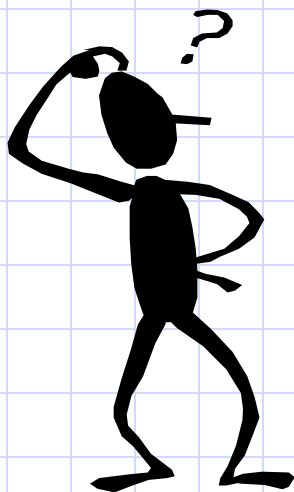
经过分析，不必求出依次所有的 $f[i, j]$ ，只有当 $B[i]=A[j]$ 时，才有必要求 $f[i, j]$ ，其余的 f 值全为 0。

又因为 A, B 中的字符只有 $['a'..'z', 'A'..'Z']$ ，那么只需在开始时用链表记录 $'a'..'z', 'A'..'Z'$ 出现的位置，动态规划的过程中就可以实现这个优化。

分析和求解

问题的结症没有解决：

算法的时间复杂度还是没有降低。



很难找到他的更复杂的动态规划！
的飞跃的其它的动态规划。

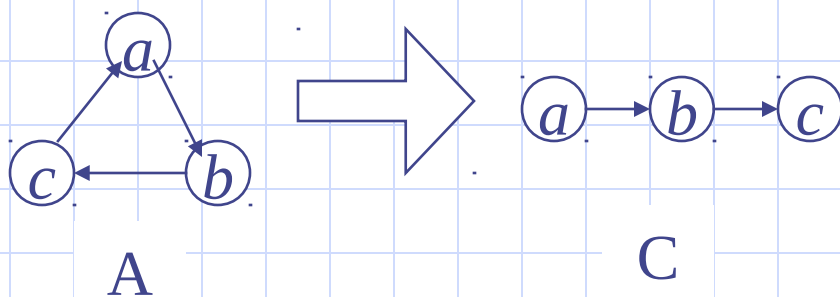
问题转化

动态规划未用到另一条件：

只有最大公共子串的长度大于等于 N 时，才有必要计算这个长度。

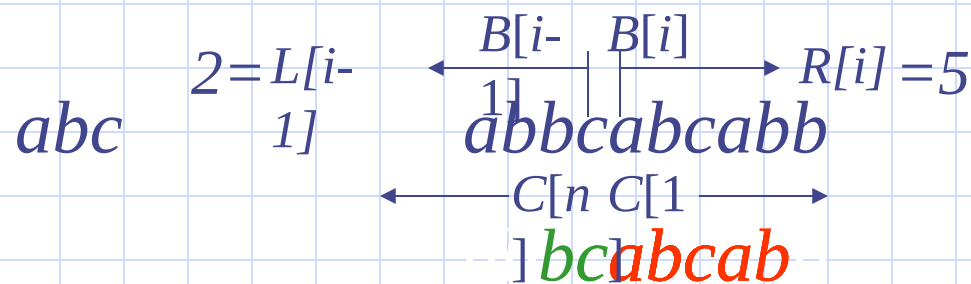
如前所述，可以发现：此题实际上比较类似一般字符匹配问题，不同点在于此题有环串存在！

因为环串的匹配起始位置是不定的，与一般的字符匹配问题是不同的。所以不妨先将环串 A 断开，设为线串 C 。



求解的变换

如果最长公共子序列长度大于等于 n ，直接输出包含 C 中所有字符。



那么如果求出从 B 的第 i 位和 C 的第一位起向后循环匹配的最大长度，记为 $R[i]$ 。

再求出从 B 的第 $i-1$ 位和 C 的最后一位起向前循环匹配的最大长度，记为 $L[i-1]$ 。

$$Ans = \max(L[i-1] + R[i], 1 \leq i \leq M)$$

$R[i]$ 的求解

$R[i]$ 的初步求法为：

枚举 B 中位置 i ，和 C 向后匹配到不能匹配为止，显然匹配的长度可达到 M ，那么此法的复杂度为 $O(M^2)$ ，复杂度有增无减！

必须考虑避免不必要的匹配：

abc $abbcabcabb$
 $\begin{matrix} i & & i+n \\ | & & | \end{matrix}$
 $R[i] \xrightarrow{n} R[i+n]$

如果从 i 开始往后逐字比较时，已匹配的长度已经达到了 N ，那么就没有必要再往后比较了，必然有 $R[i] = R[i+n] + n$ 成立。

$Q[i]$ 的求解

定义 $Q[i]$ 表示 B 从位置 i 开始和 C 非循环匹配的最大长度。
那么求出 $Q[i]$ ，就求出了 $R[i]$ 了。

$$R[i] = \begin{cases} Q[i] & Q[i] < n \\ R[i+n] + n & Q[i] = n \end{cases}$$

$R[i]$ 那么问题等价于求 $Q[i]$ ，如何求 $Q[i]$ ？

一般的，到了这一步，我们希望通过字符匹配类问题中高效的 KMP 算法来求出 $Q[i]$ 。

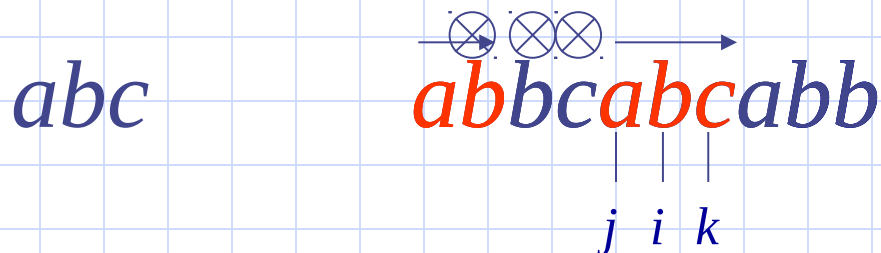
但稍加分析就会发现： KMP 的匹配过程是跳动的，不便于求出所有的 $Q[i]$ 。

分析和求解

虽然直接运用 KMP 求 $Q[i]$ 的计划落空，但是否还可以借鉴 KMP 的解题思路和特点来设计 $Q[i]$ 的求法呢？

首先，要使复杂度尽量低，总的匹配次数也就应该尽量少。而 KMP 解题的一大特点就是：尽量的利用了前面的比较结果，达到了不重复比较被匹配串中已匹配字符的目的。

在 $Q[i]$ 的计算上是否也可以做到这一点呢？



K 为求 $Q[1..i-1]$ 的过程中， B 中已被匹配到最大的位置。

J 为匹配到 K 的起始位置。

分析和求解

为了不重复比较 B 中已匹配字符，最理想的情况下，我们期望可以直接从 $k+1$ 位开始向后比较就能求出 $Q[i]$ 。

显然，如果能直接确定 $Q[i]+i \geq k$ 时，也就是从 i 位置起，可匹配到的最大位置不小于 k ，就可以直接从 $k+1$ 位开始往后逐字比较求 $Q[i]$ 。

但这样就存在两个问题：

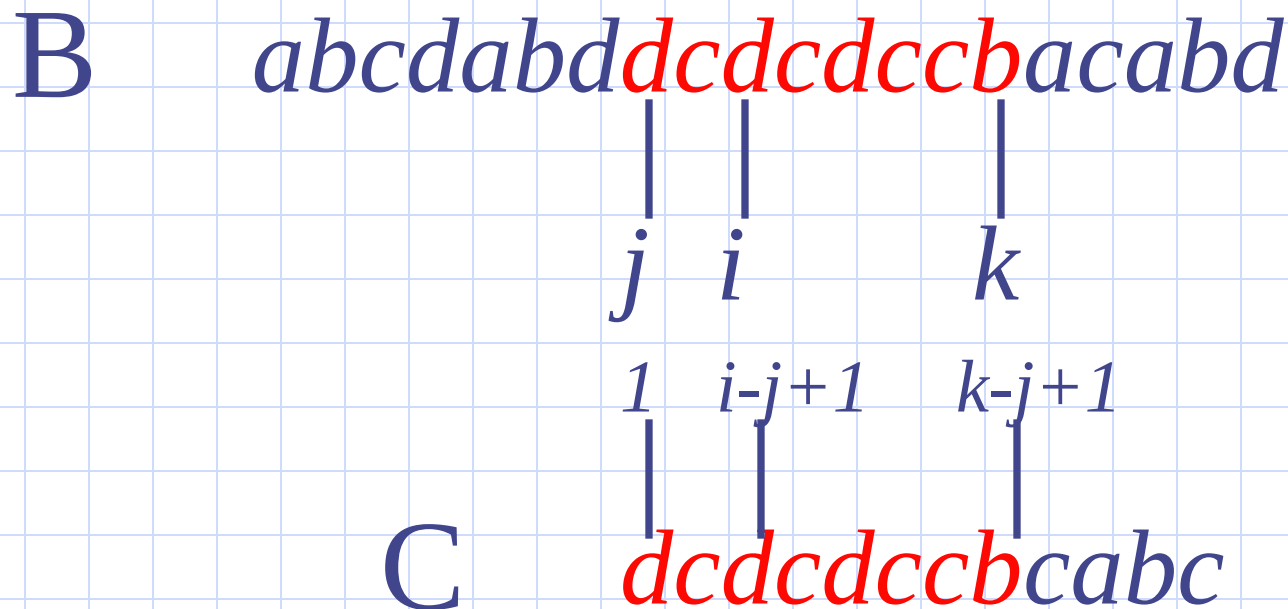
1. 怎么确定 $Q[i]+i \geq k$ 呢？
2. 如果 $Q[i]+i < k$ 时，又怎么办呢？

KMP 算法达到避免重复比较目的的关键点在于：

将匹配串和被匹配串对应起来考虑。

分析和求解

先考虑 B 串中 j 到 k 这一段。显然它和 C 串的 1 到 $k-j+1$ 这段是完全一样的；同样， B 中的 i 位置，对应于 C 串中的 $i-j+1$ 位置。



分析和求解

求出从 C 的第 $i-j+1$ 位和 C 的第一位开始可匹配的最大长度，记为 L 。

B *abcdabd**dcdcdccb**acabd*

 | | |
 j *i* *k*

 1 $i-j+1$ $k-j+1$

C

*dcdcdccb**cabc*

*dcdc**dccb**cabc*

 └───┘
 L

分析和求解



观察上图，可得出以下结论：

当 $L < k-i+1$ 时： $Q[i] = L$

当 $L \geq k-i+1$ 时： $Q[i] + i \geq k$ 。只需从 $k+1$ 向后逐字比较，就可求出 $Q[i]$ 。

所以，关键在 L 的计算上。

C

L 的值仅由 $i-j+1$ 和 C 确定。

设 $T[x](1 \leq x \leq n)$, 表示从 C 的 x 位置起和 C 匹配的最大长度。

那么求解 R 的过程中, 对 B 中任何位置 i 和对应的 j , $T[i-j+1]$ 就相当于要求的 L 。

T 的求解

接下来，求解 T :

其实 T 的计算和 Q 的计算本质上是一样：都要求从一个字符串的每个位置起和 C 可匹配的最大长度。

$Q[i]$ 可通过 $T[1..n]$ 和适当比较算出来，而 $T[x]$ 则可通过 $T[1..x-1]$ 和适当的比较求出来。

那么计算 $T[x]$ 的时间复杂度： $O(N)$ 。

复杂度分析

此算法的总时间复杂度

= 匹配的时间复杂度 + 计算 $T[x]$ 的时间复杂度

= $O(M+N)$ 。

效率上得到了极大的提高。

问题圆满解决！

上面的解决方式，看起来与 *KMP* 十分的相似，但实际上透彻的理解后，还是有很大的区别的。此解法主要是借鉴 *KMP* 的思想和特点提出的，而不是生搬硬套。

回顾解题过程

下面回顾整个解题的过程：

1. 先考虑动态规划，行不通！
2. 分析问题，发现问题要求长度不小于 N 的公共部分，这是此题转化的关键条件。
3. 求解分解为 $L[i]$ 和 $R[i]$ 。
4. 精简 $R[i]$ 的求解，求相应的 $Q[i]$ 。
5. 灵活运用 KMP 解题思路和特点，设计出 T 数组，使得匹配的时间复杂度降为 $O(M)$ 。这部分是圆满解决此问题的关键。

总结

上面 $Q[i]$ 的求解思路具有一定的可推广性，对于某些字符串匹配问题可以类似的寻求问题的解决方法。

上面讲解的题目比较的简单，只是希望大家着重注意解题的思路，但愿能够给大家一点启发。从此题的解决上，也发现解题时要牢记以下几点：

- 需要全面分析问题，抓住问题的关键
- 要有一定的知识积累量，必须牢固掌握基础知识
- 要有灵活变通运用的能力

谢

谢