

浅析“最小表示法”思想在字符串循环同构问题中的应用

安徽省芜湖市第一中学
周源

【目录】

浅析“最小表示法”思想在字符串循环同构问题中的应用..... 1

 【目录】..... 1

 【摘要】..... 2

 【关键字】..... 2

 【正文】..... 3

 一、 问题引入..... 3

 1. 明确几个记号和概念..... 3

 2. 问题..... 3

 二、 枚举算法和匹配算法..... 3

 1. 枚举算法..... 3

 2. 匹配算法..... 3

 3. 小结..... 4

 三、 最小表示法思想..... 4

 1. “最小表示法”思想的提出..... 4

 2. “最小表示法”思想的定义..... 4

 3. “最小表示法”在本题的应用..... 5

 4. 模拟算法执行..... 7

 5. 小结..... 8

 四、 总结..... 8

【摘要】

最小表示法在搜索判重、判断图的同构等很多问题中有着重要的应用。本文就围绕字符串循环同构的判断这个问题，在很容易找到 $O(N)$ 的匹配后，本文引进的“最小表示法”思想，并系统的对其下了定义，最后利用“最小表示法”思想构造出了更优秀，更自然的算法。

无论是增加“最小表示法”思想这方面的知识，提高增加竞赛中的综合素质，相信本文对同学们还是有所裨益的。

【关键字】

字符串 循环同构 匹配 最小表示法

【正文】

一、问题引入

1. 明确几个记号和概念

由于本篇论文主要讨论与字符串有关的算法，所以在本文中，一切未经说明的以 s 开头的变量均表示字符串。

(1). $|s| = \text{length}(s)$ ，即 s 的长度。

(2). $s[i]$ 为 s 的第 i 个字符。这里 $1 \leq i \leq |s|$ 。

(3). $s[i \rightarrow j] = \text{copy}(s, i, j - i + 1)$ ，即截取出 s 的第 i 个字符到第 j 个字符的子串。这里 $1 \leq i \leq j \leq |s|$ 。特别的， $s[i \rightarrow i] = s[i]$ 。

(4). 定义 s 的一次循环 $s^{(1)} = s[2 \rightarrow |s|] + s[1]$ ；而 s 的 $k(k > 1)$ 次循环 $s^{(k)} = s^{(k-1)^{(1)}$ ， s 的零次循环 $s^{(0)} = s$ 。

(5). 如果字符串 $s1$ 可以经过有限次循环得到 $s2$ ，即有 $s2 = s1^{(k)}$ ($k \in \mathbb{N}$)，则称 $s1$ 和 $s2$ 是循环同构的。

(6). 设有两个映射 $f_1, f_2: A \rightarrow A$ ，定义 f_1 和 f_2 的连接

$f_1 \bullet f_2(x) = f_1(f_2(x))$ ，这里 $x \in A$ 。——这个定义用于后文算法描述中。

2. 问题

给定两个字符串 $s1$ 和 $s2$ ， $|s1| = |s2|$ ，判断他们是否循环同构。

二、枚举算法和匹配算法

1. 枚举算法

很容易知道， $s1$ 的不同的循环串最多只有 $|s1|$ 个，即 $s1^{(0)}, s1^{(1)}, \dots, s1^{(|s1|-1)}$ ，所以只需要把他们一一枚举，然后分别与 $s2$ 比较即可。

枚举算法思维简单，易于实现，而它的时间复杂度是 $O(N^2)$ 级¹，已经可以胜任大多数问题的要求了。然而如果 N 大至几十万，几百万，枚举算法就无能为力了，有没有更优秀的算法呢？

2. 匹配算法

从枚举算法执行过程中很容易发现，枚举算法的本质就是在一个可以循环的字符串 $s1$ 中寻找 $s2$ 的匹配，于是联想到模式匹配的改进算法是 $O(N)$ 级的，那么在循环串中寻找匹配

¹这里 $N = |s1| = |s2|$ 。

是不是也有线性的算法呢？回答是肯定的：

由于循环串与一般的字符串本质的区别就是前者是“循环”的，如果能去掉“循环”这个限制，那么就可以直接套用一般字符串的模式匹配算法了！显然，将 $s1$ 复制两次： $S = s1 + s1$ 做为主串，则任何与 $s1$ 循环同构的字符串至少都可以在 S 中出现一次，于是可以说 S 就是循环串 $s1$ 的一般字符串形式！问题成功转化为求 $s2$ 在 S 中的模式匹配。——这完全可以在 $O(N)$ 级时间内解决。

3. 小结

很容易得到的枚举算法显然不能满足大数据的要求，于是我们从算法的执行过程入手，探查到了枚举算法的实质：模式匹配。最后，通过巧妙的构造、转换模型，直接套用模式匹配算法，得到了 $O(N)$ 级别的算法。

但是问题是否已经完美解决了呢？也许你会说：以 KMP 算法为首的模式匹配改进算法，都是以难理解，难记忆著称的！这的确是 KMP 算法的缺点，而且其 next 数组繁琐的计算严重制约着算法的可扩展性，看来是有必要寻求更简洁的算法了。

三、最小表示法思想

1. “最小表示法”思想的提出

首先来看一个引例：

[引例] 有两列数， a_1, a_2, \dots, a_n 和 b_1, b_2, \dots, b_n ，不记顺序，判断它们是否相同。

[分析] 由于题目要求“不记顺序”，因此每一列数的不同形式高达 $n!$ 种之多！如果要一一枚举，显然是不科学的。于是一种新的思想提出了：如果两列数是相同的，那么将它们排序之后得到的数列一定也是相同的。于是，算法复杂度迅速降为 $O(N \log_2 N)$ 级。

这道题虽然简单，却给了我们一个重要的启示：当某两个对象有多种表达形式，且需要判断它们在某种变化规则下是否能够达到一个相同的形式时，可以将它们都按一定规则变化成其所有表达形式中的最小者，然后只需要比较两个“最小者”是否相等即可！

下面我们系统的给出“最小表示法”思想的定义。

2. “最小表示法”思想的定义

设有事物集合 $T = \{t_1, t_2, \dots, t_n\}$ 和映射集合 $F = \{f_1, f_2, \dots, f_m\}$ ，其中 $f_i (1 \leq i \leq m)$ 是 T 到 T 的映射： $f_i: T \rightarrow T$ 。如果两个事物 $s, t \in T$ ，有一系列 F 映射的连接使 $f_{i_1} \cdot f_{i_2} \cdot \dots \cdot f_{i_k}(t) = s$ ，则说 s 和 t 是 F 本质相同的。

这里 F 满足：

- (1). 任意 $t \in T$ ，一定能在 F 中一系列映射的连接的作用下，仍被映射至 t 。
- (2). 任意 $s, t \in T$ ，若有 $f \in F$ 使 $f(s) = t$ ，则一定存在一个或一系列映射

$$f_{i_1}, f_{i_2}, \dots, f_{i_k} \in F, \text{ 他们的连接 } f_{i_1} \cdot f_{i_2} \cdot \dots \cdot f_{i_k}(t) = s.$$

由 F 的性质(1)可知, s 和 s 是 F 本质相同的, 即“本质相同”这个概念具有自反性。

从性质(2)可知, 如果 s 和 t 是 F 本质相同的, 那么 t 和 s 也一定是 F 本质相同的 ($s, t \in T$)。即“本质相同”这个概念具有对称性。

另外, 根据“本质相同”概念的定义很容易知道, “本质相同”这个概念具有传递性。

即若 t_1 和 t_2 是 F 本质相同, t_2 和 t_3 是 F 本质相同, 那么一定有 t_1 和 t_3 是 F 本质相同的。

给定 T 和 F , 如何判断 T 中的两个事物 s 和 t 是否互为 F 本质相同的呢? “最小表示法”就是可以应用于此类题目的一种思想。它规定 T 中的所有事物均有一种特殊的大小关系。

然后, 根据 F 中的变化规则, 将 s 和 t 均化为规定大小关系中的最小者 m_1 和 m_2 , 如果两者相同, 则易知, s 和 m_1 本质相同, m_1 和 m_2 本质相同, m_2 和 t 本质相同, 所以 s 和 t 是本质相同的。否则, 可以证明, s 和 t 不是本质相同的。

3. “最小表示法”在本题的应用

在本题中, 事物集合 T 表示的是不同的字符串, 而映射集合 F 则表示字符串的循环法则, “事物中的大小关系”就是字符串间的大小关系。

那么, 如何将最小表示法应用于本题呢? 最简单的方法就是根据上文, 分别求出 $s1$ 和 $s2$ 的最小表示, 比较它们是否相同。如果要是很简单的这么做, 问题就非常麻烦了: 求字符串的最小表示法虽然有 $O(N)$ 级算法, 但是思路十分复杂, 还不如匹配算法——如果单纯得这么做, 就违背了我们寻找更好算法的初衷。这样, 看上去“最小表示法”是无能为力了。

然而我们换一种思路:

和匹配算法相似, 将 $s1$ 和 $s2$ 各复制一次: $u = s1 + s1$, $w = s2 + s2$; 并设两个指针 i 和 j 分别指向 u 和 w 的第一个字符。

设 $M(s) = \min_{1 \leq k \leq |s|} \{ \text{任意 } 0 \leq i < |s|, \text{ 均有 } s^{(k-1)} \geq s^{(i)} \}$, 也就是说函数 $M(s)$ 返回的

是 s 最小表示串的第一个字符在原串中的位置², 如果有多个最小表示串, 则取在原串中位置最小的一个。

显然如果 $s1$ 和 $s2$ 是循环同构的, 且当前两指针 $i = M(s1)$ 且 $j = M(s2)$ 的时候, 一定可以得到 $u[i \rightarrow i + |s1| - 1] = w[j \rightarrow j + |s2| - 1]$, 迅速得到 $s1$ 和 $s2$ 是循环同构的。

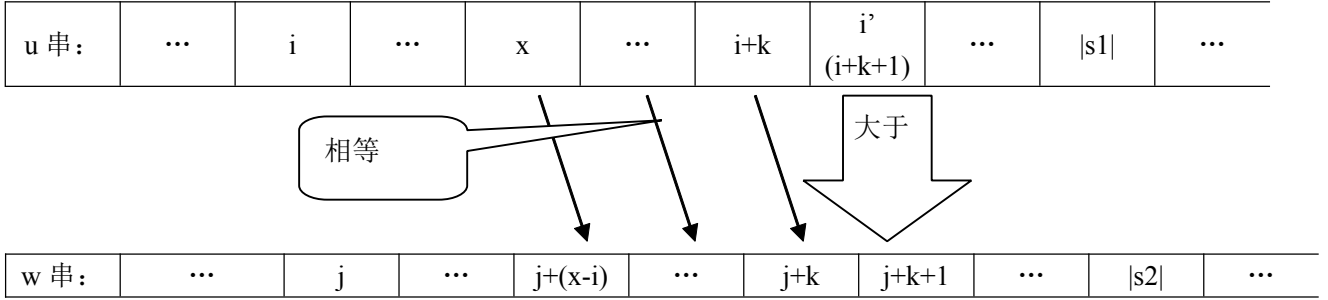
当 $i \leq M(s1)$ 且 $j \leq M(s2)$ 时, 两指针仍然有机会达到 $i = M(s1)$, $j = M(s2)$ 这个状态。于是问题转化成: 仍然设 $s1$ 和 $s2$ 是循环同构的, 当前两指针分别向后滑动比较, 如果发现比较失败, 有 $u[i+k] \neq w[j+k] (k \geq 0)$, 如何向后滑动指针, 使新指针仍然满足 i' 和

²注意, 这里所说的“在原串中的位置”, 并不是单纯的在原串中寻找到的第一个匹配, 而是以之开头的循环串是原串的最小表示。

j' 仍然满足 $i' \leq M(s1), j' \leq M(s2)$ 。

从不相等的 $u[i+k]$ 和 $u[j+k]$ 下手：如果 $u[i+k] > u[j+k]$ ，那么任意整数

$x \in [i, i+k]$ ，从 $s1$ 第 x 个字符开头的循环串是 $s1^{(x-1)}$ ，如图， $s1^{(x-1)}$ 的前 $(i'-x)$ 个字符是



$u[x \rightarrow i'-1]$ ，当然，也可以表示为 $u[x \rightarrow i+k]$ ，对称的，可以在 w 串中找到一段字符

$u[j+(x-i) \rightarrow j+k]$ ，这两段字符串长度相等，而且根据上文假设的扫描过程可以得到

$u[x \rightarrow i+k-1] = u[j+(x-i) \rightarrow j+k-1]$ 。而 $u[i+k] > u[j+k]$ ，所以得到

$u[x \rightarrow i+k] > u[j+(x-i) \rightarrow j+k]$ ，即 $s1^{(x-1)}[1 \rightarrow i'-x+1] > u[j+(x-i) \rightarrow j+k]$ 。

而根据假设 $s1$ 和 $s2$ 是循环同构的，那么一定能在 $s1$ 的所有循环串中找到一个字符串 $s1'$ ，

满足它的前 $(i'-x+1)$ 个字符是 $u[j+(x-i) \rightarrow j+k]$ 。于是有 $s1^{(x-1)} > s1'$ ，得 $s1^{(x-1)}$ 一定

不是 $s1$ 的最小表示。所以 $M(s2)$ 不可能在 $[i, i+k]$ 一段中，也就可以将指针 i 向后滑动

$(k+1)$ 个字符： $i \leftarrow i+k+1$ 。

同理得到如果 $u[i+k] < u[j+k]$ ，可将指针 j 向后滑动 $(k+1)$ 个字符： $j \leftarrow j+k+1$ 。

仍满足 $i \leq M(s1)$ ， $j \leq M(s2)$ 。

于是设计出算法：

- (1). 将 $s1$ 和 $s2$ 各复制一次： $u = s1 + s1$ ， $w = s2 + s2$ ；并设两个指针 i 和 j 分别指向 u 和 w 的第一个字符。
- (2). 如果 i 和 j 均中至少一个大于 $|s1|$ ，

则可以肯定 $s1$ 和 $s2$ 不是循环同构的，返回 *false*，算法结束；

否则找到最小的 $k \geq 0$ 使 $u[i+k] \neq u[j+k]$ ，如果这样的 k 不存在或 $k \geq |s1|$ ，

则说明 $u[i \rightarrow i + |s1| - 1] = u[j \rightarrow j + |s1| - 1]$ ，即 u 和 w 各有一段长为 $|s1|$

的子串相等， $s1$ 和 $s2$ 是循环同构的，返回 $true$ ，算法结束；否则转第(3)步。

(3). 如果 $u[i+k] > u[j+k]$ ，则将指针 i 向后滑动 $(k+1)$ 个字符： $i \leftarrow i+k+1$ ；

否则说明 $u[i+k] < u[j+k]$ ，就将指针 j 向后滑动 $(k+1)$ 个字符： $j \leftarrow j+k+1$ 。

继续执行第(2)步。

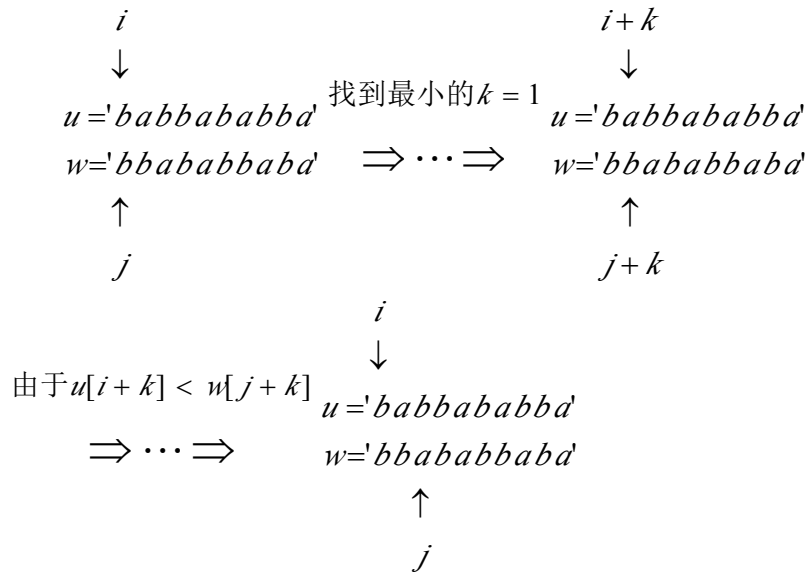
容易得出，本算法的时空复杂度也是均为 $O(N)$ 级。

更清楚一点，我们附上一个例子：

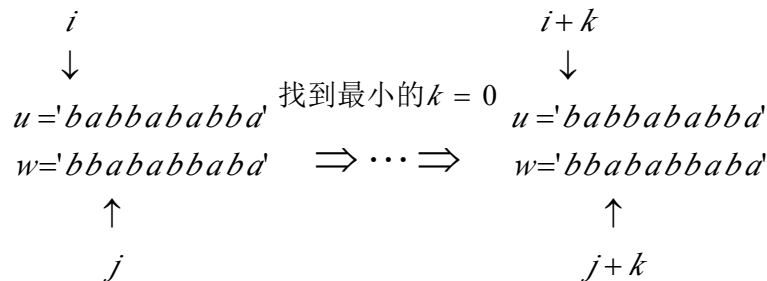
4. 模拟算法执行

设 $s1 = 'babba'$ ， $s2 = 'bbaba'$ ，显然它们是循环同构的。模拟执行算法是这样的：首先有 $u = s1 + s1 = 'babbaabba'$ ， $w = s2 + s2 = 'bbabaabba'$ ，并设指针 $i = 1, j = 1$ 。

第一次执行(2)、(3)两步：



第一次执行完毕后得到 $i = 1, j = 3$ ，下面第二次执行：



$$\begin{array}{c}
 i \\
 \downarrow \\
 \text{由于 } u[i+k] > w[j+k] \quad u='babbbababba' \\
 \Rightarrow \cdots \Rightarrow \quad w='bbababbaba' \\
 \uparrow \\
 j
 \end{array}$$

现在结果是 $i=2, j=3$ ，同理执行第三次：

$$\begin{array}{ccc}
 i & & i+k \\
 \downarrow & & \downarrow \\
 u='babbbababba' & \text{找到最小的 } k=2 & u='babbbababba' \\
 w='bbababbaba' & \Rightarrow \cdots \Rightarrow & w='bbababbaba' \\
 \uparrow & & \uparrow \\
 j & & j+k \\
 \\
 i & & \\
 \downarrow & & \\
 \text{由于 } u[i+k] > w[j+k] & u='babbbababba' & \\
 \Rightarrow \cdots \Rightarrow & w='bbababbaba' & \\
 \uparrow & & \\
 j & &
 \end{array}$$

此时已经是 $i=5, j=3$ ，执行第四次：

$$\begin{array}{ccc}
 i & & i+k \\
 \downarrow & & \downarrow \\
 u='babbbababba' & k \leq |s1|=5 \text{ 时未找 } u[i+k] \neq w[j+k]! & u='babbbababba' \\
 w='bbababbaba' & \Rightarrow \cdots \Rightarrow & w='bbababbaba' \\
 \uparrow & & \uparrow \\
 j & & j+k
 \end{array}$$

这时发现 $u[5 \rightarrow 9] = w[3 \rightarrow 7]$ ！成功得出 $s1$ 和 $s2$ 是循环同构的，算法返回值为真。

5. 小结

经过一番努力，我们终于得到了一个与匹配算法本质不同的线性算法。在这个问题中，“最小表示法”思想引导我们从问题的另一方面分析，进而构造出一个全新的算法。比起匹配算法，它容易理解，便于记忆，实现起来，也不过是短短的几重循环，非常方便，便于应用于扩展问题。

四、 总结

“最小表示法”是判断两种事物本质是否相同的一种常见思想，它的通用性也是被人们认可的——无论是搜索中判重技术，还是判断图的同构之类复杂的问题，它都有着无可替代的作用。深入分析不难得出，该思想的精华在于引入了“序”这个概念，将待比较两个事物

化为规定顺序中最小的（也可能是最大的）再加以比较。

然而值得注意的是，在如今的信息学竞赛中，试题纷繁复杂，使用的算法也不再拘泥于几个经典的算法，改造经典算法或是将多种算法组合是常用的方法之一。正如本文讨论的问题，单纯的寻求字符串的最小表示显得得不偿失，但利用“最小表示法”的思想，和字符串的最小表示这个客观存在的事物，我们却找到了一个简单、优秀的算法。

因此，在解决实际问题时，只有深入分析，敢于创新，才能将问题

化纷繁为简洁，
化无序为有序。