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】

【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。

⑴．$|s| = length(s)$，即s的长度。

⑵．$s[i]$为s的第i个字符。这里$1 \le i \le |s|$。

⑶．$s[i \to j] = copy(s, i, j-i+1)$，即截取出$s$的第$i$个字符到第$j$个字符的子串。这里$1 \le i \le j \le |s|$。特别的，$s[i \to i] = s[i]$。

⑷．定义s的一次循环$s^{(1)} = s[2 \to |s|] + s[1]$；而$s$的$k(k>1)$次循环$s^{(k)} = {s^{(k-1)}}^{(1)}$，$s$的零次循环$s^{(0)} = s$。

⑸．如果字符串$s1$可以经过有限次循环得到$s2$，即有$s2 = s1^{(k)} (k \in N)$，则称$s1$和$s2$是循环同构的。

⑹．设有两个映射$f_1, f_2 : A \to A$，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$s1^{(0)}, s1^{(1)}, \ldots, s1^{(|s1|-1)}$，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？

---

[1]这里 N=|s1|=|s2|。

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，$a_1, a_2, \cdots a_n$ 和 $b_1, b_2, \cdots b_n$，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 $T = \{t_1, t_2, \cdots, t_n\}$ 和映射集合 $F = \{f_1, f_2, \cdots, f_m\}$，其中 $f_i (1 \le i \le m)$ 是 T 到 T 的映射：$f_i : T \to T$。如果两个事物 $s, t \in T$，有一系列 F 映射的连接使 $f_{i_1} \bullet f_{i_2} \bullet \cdots \bullet f_{i_k}(t) = s$，则说 s 和 t 是 F 本质相同的。

这里F满足：

⑴．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。

⑵．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

$$f_{i_1}, f_{i_2}, \cdots, f_{i_k} \in F \text{，他们的连接 } f_{i_1} \bullet f_{i_2} \bullet \cdots \bullet f_{i_k}(t)=s \text{。}$$

由F的性质⑴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⑵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。

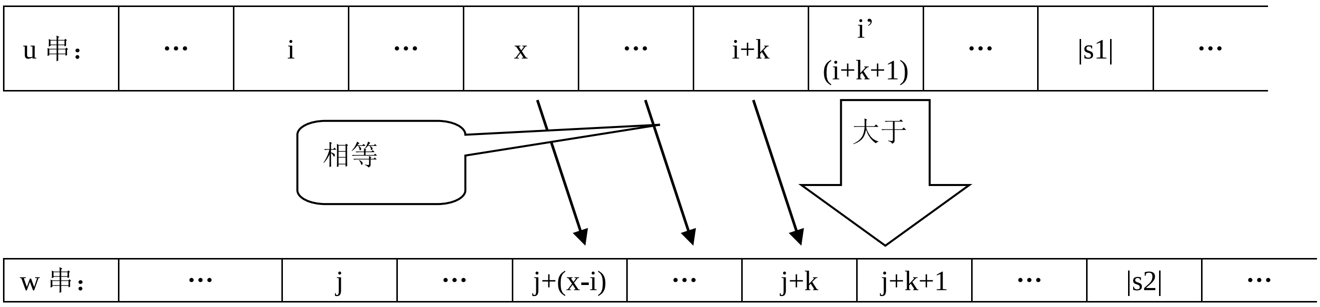
设$M(s)=\min\limits_{1 \le k \le |s|}\left\{\text{任意}0 \le i < |s|\text{，均有}s^{(k-1)} \ge s^{(i)}\right\}$，也就是说函数$M(s)$返回的是$s$最小表示串的第一个字符在原串中的位置[2]，如果有多个最小表示串，则取在原串中位置最小的一个。

[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。

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 $u[i \to i + |s1| - 1] = w[j \to j + |s2| - 1]$，迅速得到 $s1$ 和 $s2$ 是循环同构的。

当 $i \le M(s1)$ 且 $j \le M(s2)$ 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 $u[i+k] \ne w[j+k] (k \ge 0)$，如何向后滑动指针，使新指针仍然满足 i' 和 j' 仍然满足 $i' \le M(s1), j' \le M(s2)$。

从不相等的 $u[i+k]$ 和 $w[j+k]$ 下手：如果 $u[i+k] > w[j+k]$，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



$u[x \to i'-1]$，当然，也可以表示为 $u[x \to i+k]$，对称的，可以在 w 串中找到一段字符 $w[i+(x-i) \to j+k]$，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 $u[x \to i+k-1] = w[i+(x-i) \to j+k-1]$。而 $u[i+k] > w[j+k]$，所以得到 $u[x \to i+k] > w[i+(x-i) \to j+k]$，即 $s1^{(x-1)}[1 \to i'-x+1] > w[i+(x-i) \to j+k]$。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 $w[i+(x-i) \to j+k]$。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 $u[i+k] < w[j+k]$，可将指针 j 向后滑动 $(k+1)$ 个字符：$j \leftarrow j+k+1$。仍满足 $i \le M(s1)$，$j \le M(s2)$。

于是设计出算法：

⑴．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。

⑵．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 $k \ge 0$ 使 $u[i+k] \ne w[j+k]$，如果这样的 k 不存在或 $k \ge |s1|$，

则说明 $u[i \to i+|s1|-1] = w[j \to j+|s1|-1]$，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⑶步。

⑶．如果 $u[i+k] > w[j+k]$，则将指针 i 向后滑动 $(k+1)$ 个字符：$i \leftarrow i+k+1$；

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

继续执行第⑵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 $u = s1 + s1 = 'babbababba'$，$w = s2 + s2 = 'bbababbaba'$，并设指针 $i = 1, j = 1$。

第一次执行⑵、⑶两步：

$$\begin{array}{c} i \\ \downarrow \\ u='babbababba' \\ w='bbababbaba' \\ \uparrow \\ j \end{array} \quad \overset{\text{找到最小的}k=1}{\Longrightarrow \cdots \Longrightarrow} \quad \begin{array}{c} i+k \\ \downarrow \\ u='babbababba' \\ w='bbababbaba' \\ \uparrow \\ j+k \end{array}$$

$$\overset{\text{由于}u[i+k] < w[j+k]}{\Longrightarrow \cdots \Longrightarrow} \quad \begin{array}{c} i \\ \downarrow \\ u='babbababba' \\ w='bbababbaba' \\ \uparrow \\ j \end{array}$$

第一次执行完毕后得到 $i = 1, j = 3$，下面第二次执行：

$$\begin{array}{c} i \\ \downarrow \\ u='babbababba' \\ w='bbababbaba' \\ \uparrow \\ j \end{array} \begin{array}{c} \text{找到最小的}k=0 \\ \Rightarrow\cdots\Rightarrow \end{array} \begin{array}{c} i+k \\ \downarrow \\ u='babbababba' \\ w='bbababbaba' \\ \uparrow \\ j+k \end{array}$$

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

现在结果是$i=2, j=3$，同理执行第三次：

$$\begin{array}{c} i \\ \downarrow \\ u='babbababba' \\ w='bbababbaba' \\ \uparrow \\ j \end{array} \begin{array}{c} \text{找到最小的}k=2 \\ \Rightarrow\cdots\Rightarrow \end{array} \begin{array}{c} i+k \\ \downarrow \\ u='babbababba' \\ w='bbababbaba' \\ \uparrow \\ j+k \end{array}$$

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

此时已经是$i=5, j=3$，执行第四次：

$$\begin{array}{c} i \\ \downarrow \\ u='babbababba' \\ w='bbababbaba' \\ \uparrow \\ j \end{array} \begin{array}{c} k<=\left|s1\right|=5\text{时未找}u[i+k]\neq w[j+k]! \\ \Rightarrow\cdots\Rightarrow \end{array} \begin{array}{c} i+k \\ \downarrow \\ u='babbababba' \\ w='bbababbaba' \\ \uparrow \\ j+k \end{array}$$

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。