

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

前言

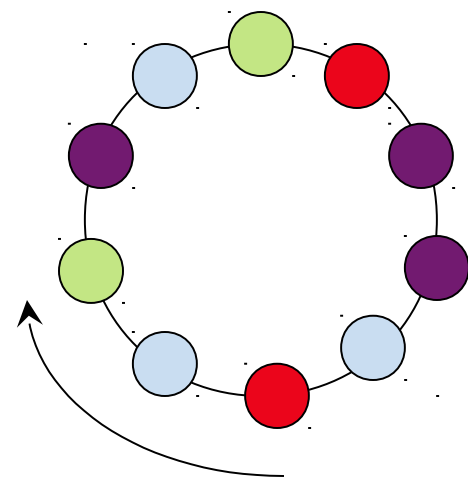
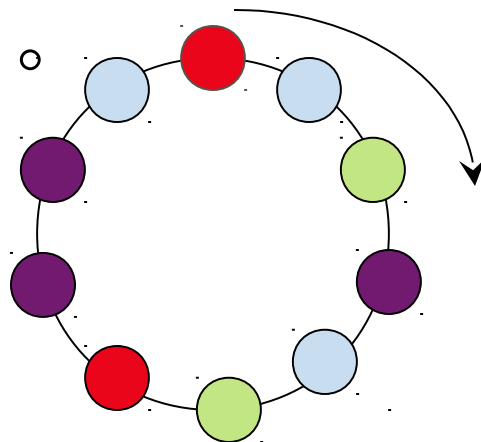
“最小表示法”比起动态规划、贪心等思想，在当今竞赛中似乎并不是很常见。但是在解决判断“同构”一类问题中却起着重要的作用。

本文即将讨论字符串中的同构问题，如何巧妙地运用最小表示法来解题呢，让我们继续一起思考吧。

问题引入

有两条环状的项链，每条项链上各有 N 个多种颜色的珍珠，相同颜色的珍珠，被视为相同问题：判断两条项链是否相同。

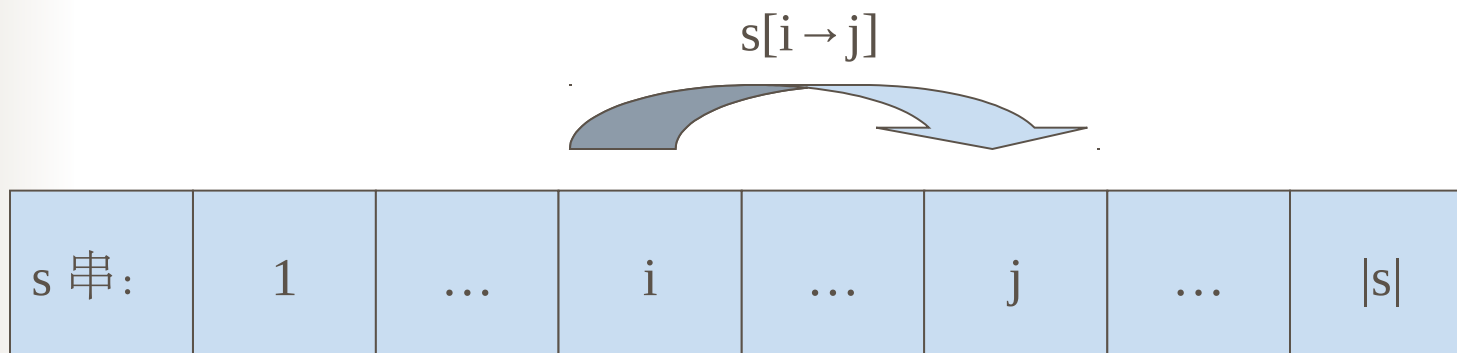
简单分析：由于项链是环状的，因此循环以后的项链被视为相同的，如图的两条项链就是一样的。



明确几个记号和概念

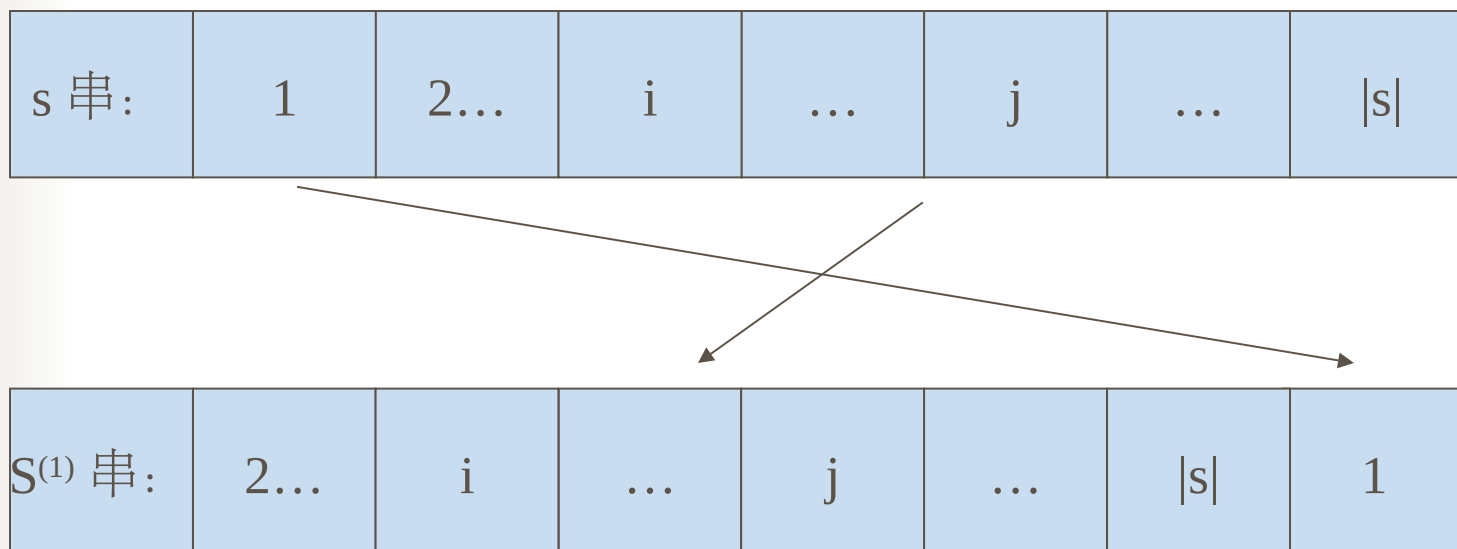
- (1) . $|s| = \text{length}(s)$, 即 s 的长度。
- (2) . $s[i]$ 为 s 的第 i 个字符。
- (3) . $s[i \rightarrow j] = \text{copy}(s, i, j-i+1)$ 。

这里 $1 \leq i \leq j \leq |s|$ 。



明确几个记号和概念

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



明确几个记号和概念

- (5) . 如果字符串 s_1 可以经过有限次循环得到 s_2 ,
则称 s_1 和 s_2 是循环同构的。例如:

$s_1 = 'a \quad b \quad c \quad d'$

一次循环

$'b \quad c \quad d \quad a'$

一次循环

$s_2 = 'c \quad d \quad a \quad b'$

s_1 和
 s_2
是循环
同构的
!

明确几个记号和概念

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

定义 f_1 和 f_2 的连接 $f_1 \bullet f_2(x) = f_1(f_2(x))$ 。

问题的数学语言表达形式

给定两个长度相等的字符串， $|s1|=|s2|$ ，
判断它们是否是循环同构的。

枚举算法

易知， $s1$ 的不同的循环串最多只有 $|s1|$ 个，
即 $s1, s1^{(1)}, s1^{(2)}, \dots, s1^{(|s1|-1)}$ ，
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

Time Limit Exceeded!

优点：思维简单，易于实现。

时间复杂度是 $O(N^2)$ 级（ $N=|s1|=|s2|$ ）。



如果 N 大一些，几十万，几百万.....

构造新的算法

首先构造新的模型：

$S=s1+s1$ 为主串， $s2$ 为模式串。

如果 $s1$ 和 $s2$ 是循环同构的，
那么 $s2$ 就一定可以在 S 中找到匹配！

匹配算法：理论的下界

在 S 中寻找 s_2 的匹配是有很多 $O(N)$ 级的算法的。

本题最优算法的时空复杂度均为 $O(N)$ 级。

这已经是理论的下界了。

小结：枚举和匹配算法

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 $O(N)$ 级的算法。

探索新的算法

但是问题是否已经完美解决了呢？

KMP 算法的缺点：

难理解，难记忆；

可扩展性不强。

[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。

相同的两列数

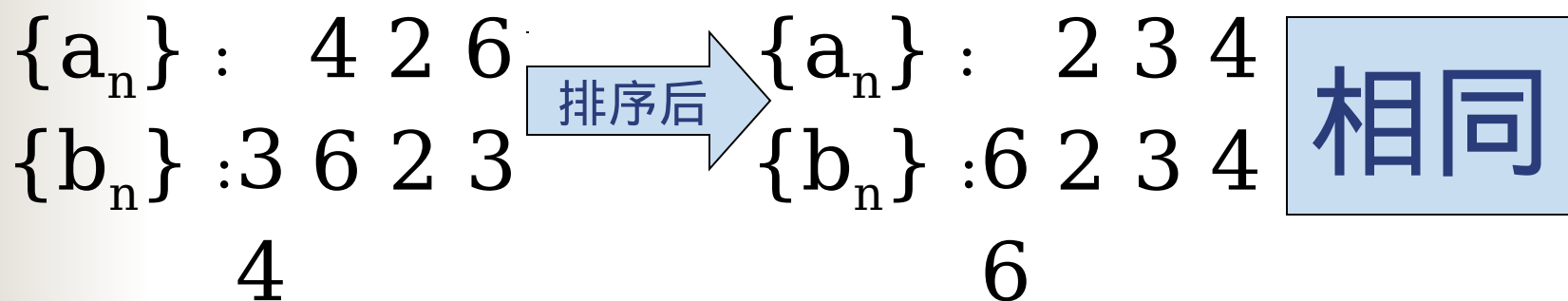
$\{a_n\}$:	4	2	6	3
$\{b_n\}$:	6	2	3	4

[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。

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。



小结：引例

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！

定义：“最小表示法”

设有事物集合 $T = \{t_1, t_2, \dots, t_n\}$,

映射集合 $F = \{f_1, f_2, \dots, f_m\}$ 。

任意 $f \in F$ 均为 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 的映射的连接 $f_{i1} \cdot f_{i2} \cdot \dots$

$\cdot f_{ik}(s) = t$,

则说 s 和 t 是 F 本质相同的。

定义：“最小表示法”

其中 F 满足两个条件：

(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 。即任意一个事物 $t \in T$, 它和自己是 F 本质相同的。

即“本质相同”这个概念具有自反性。

(2) . 任意 $s, t \in T$, 若在 F 的一系列映射作用下, s 和 t 是 F 本质相同的。那么一定有另一系列属于 F 的映射作用下, t 和 s 是 F 本质相同的。即“本质相同”这个概念具有对称性。

定义：“最小表示法”

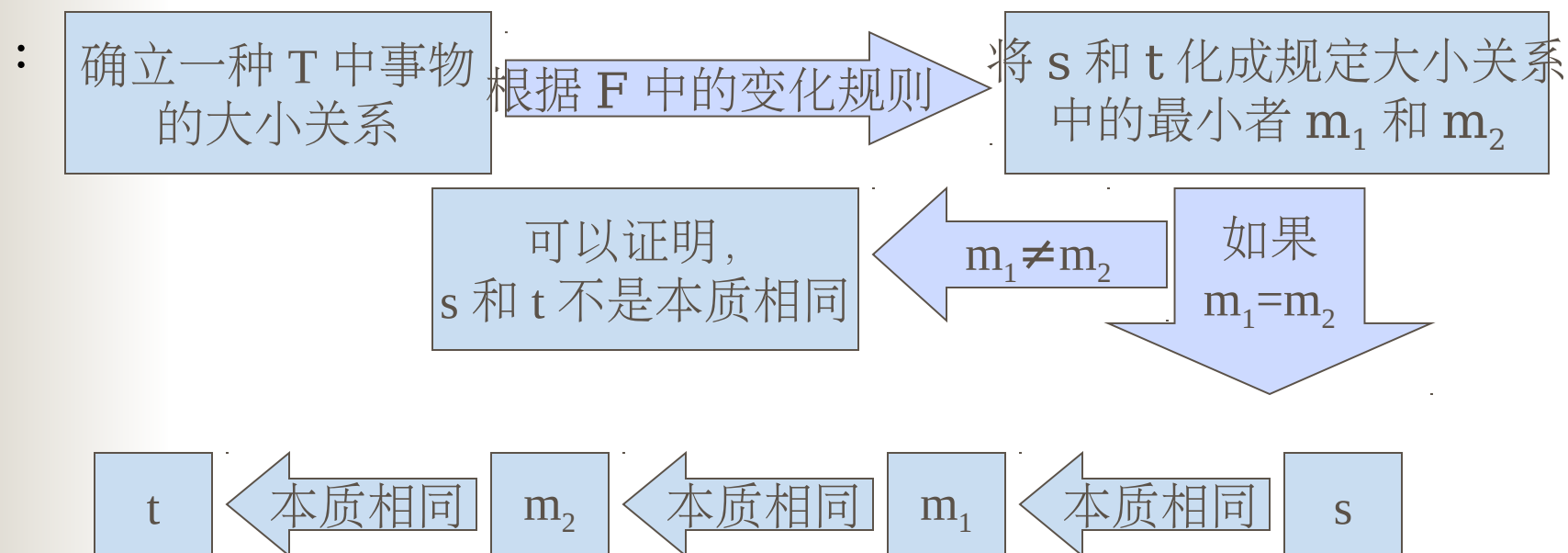
另外，根据“本质相同”概念的定义很容易知道，“本质相同”这个概念具有传递性。

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 是本质相同的。

定义：“最小表示法”

给定 T 和 F ，如何判断 T 中两个事物 s 和 t 是否互为 F 本质相同呢？

“最小表示法”就是可以应用于此类题目的一种思想



“最小表示法”在本题的应用

在本题中，
事物集合表示的是不同的字符串，
映射集合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。

最直接最简单的方法：

~~分别求出 s_1 和 s_2 的最小表示比较它们是否相同~~


“最小表示法”在本题的应用

现在换一种思路：

设函数 $M(s)$ 返回值意义为：

从 s 的第 $M(s)$ 个字符引起的 s 的一个
循环表示是 s 的最小表示。

若有多多个值，则返回最小的一个。

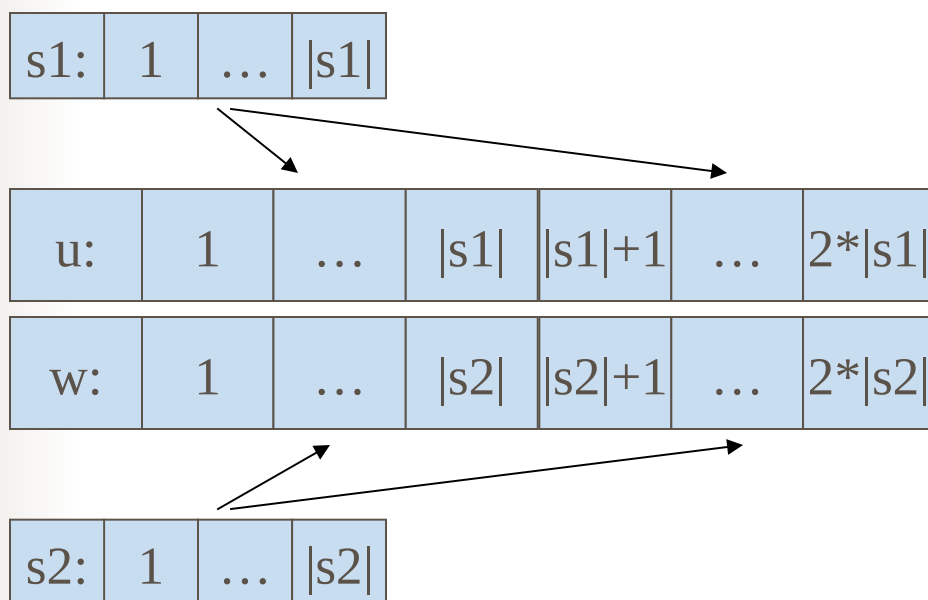

 $s = \text{'b b b a a b'}$

如 $M(\text{'bbbaab'}) = 4$

“最小表示法”在本题的应用

现在换一种思路：

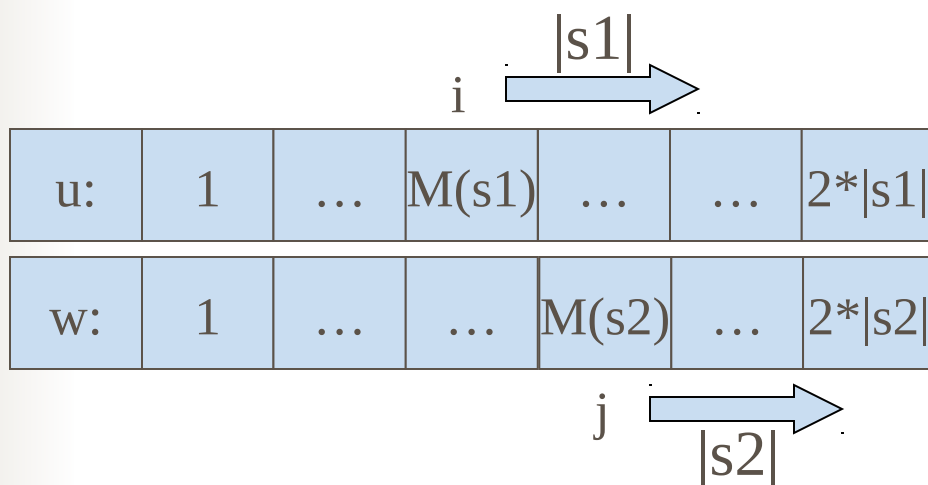
设 $u = s1 + s1$, $w = s2 + s2$ 并设指针 i, j 指向 u, w 第一个字符



“最小表示法”在本题的应用

现在换一种思路：

如果 $s1$ 和 $s2$ 是循环同构的，那么当 i, j 分别指向 $M(s1), M(s2)$ 时，一定可以得到 $u[i \rightarrow i+|s1|-1] = w[j \rightarrow j+|s2|-1]$ ，迅速输出正确解。



“最小表示法”在本题的应用

现在换一种思路：

同样 $s1$ 和 $s2$ 循环同构时，当 i, j 分别满足
 $i \leq M(s1), j \leq M(s2)$ 时，
两指针仍有机会达到 $i=M(s1), j=M(s2)$ 这个状态。

问题转化成，两指针分别向后滑动比较，如果比较失败，如何正确的滑动指针，新指针 i', j' 仍然满足
 $i' \leq M(s1), j' \leq M(s2)$

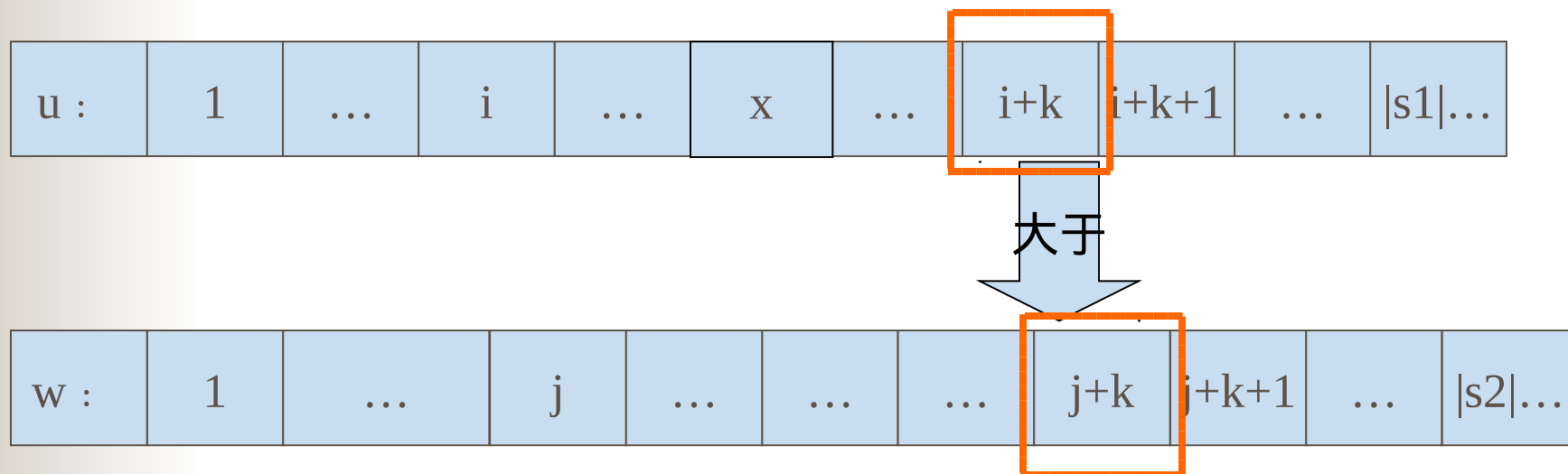
“最小表示法”在本题的应用

设指针 i, j 分别向后滑动 k 个位置后比较失败 ($k \geq 0$)，即有

$$u[i+k] \neq w[j+k]$$

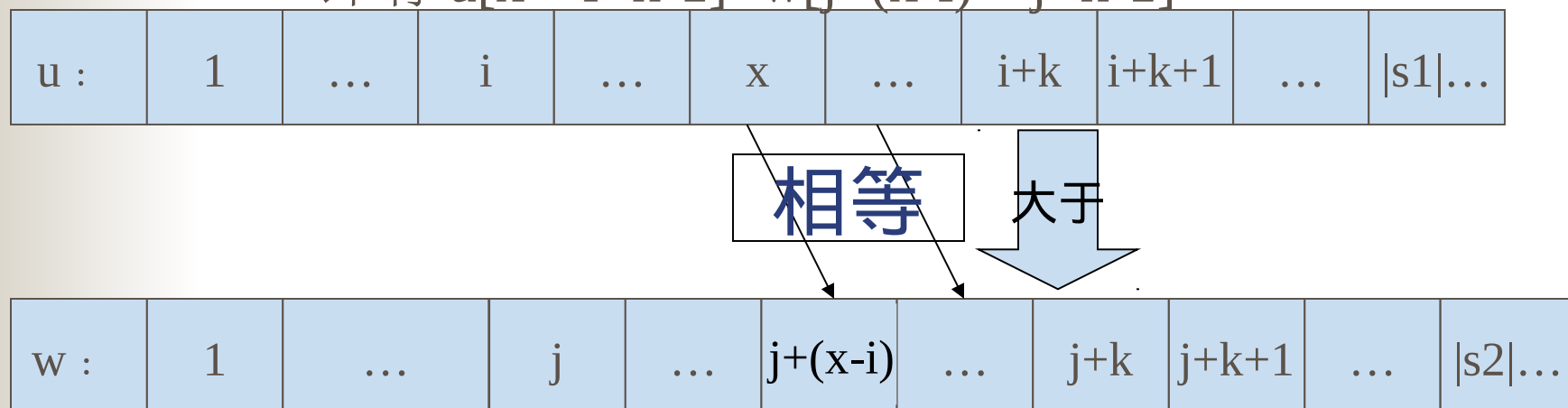
设 $u[i+k] > w[j+k]$ ，同理可以讨论 $u[i+k] < w[j+k]$ 的情况。

当 $i \leq x \leq i+k$ 时，我们来研究 $s1^{(x-1)}$ 。



“最小表示法”在本题的应用

因为 $u[x]$ 在 $u[i]$ 后 $(x-i)$ 个位置，
 对应的可以找到在 $w[j]$ 后 $(x-i)$ 个位置的 $w[j+(x-i)]$ ，
 同样对应的有 $u[x+1]$ 和 $w[j+(x+1-i)]$ ， $u[x+2]$ 和 $w[j+(x+2-i)]$ ，
 直到它们都是相等的 $k-1$ 。
 即有 $u[x \rightarrow i+k-1] = w[j+(x-i) \rightarrow j+k-1]$ 。



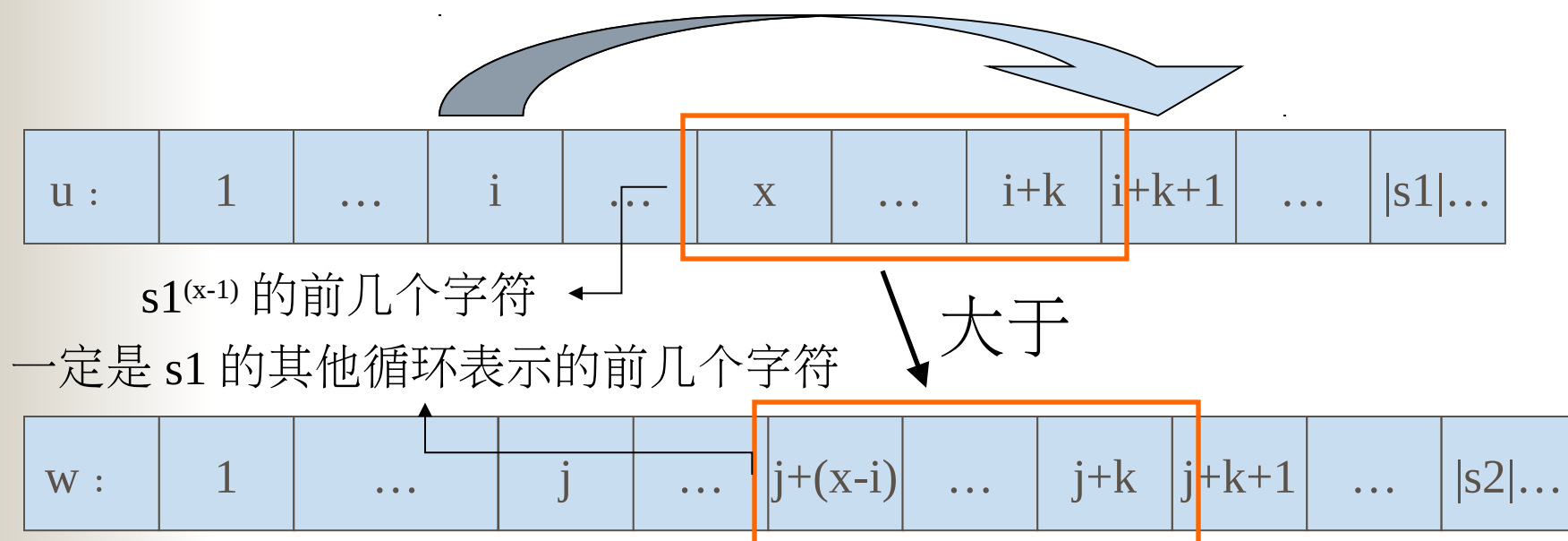
“最小表示法”在本题的应用

很容易就得到 $u[x \rightarrow i+k] > w[j+(x-i) \rightarrow j+k]$ 。

所以 $s1^{(x-1)}$ 不可能是 $s1$ 的最小表示！

因此 $M(s1) > i+k$ ，

指针 i 滑到 $u[i+k+1]$ 处仍可以保证小于等于 $M(s1)$ ！



“最小表示法”在本题的应用

同理，当 $u[i+k] < w[j+k]$ 的时候，可以将指针 j 滑到 $w[j+k+1]$ 处！

也就是说，两指针向后滑动比较失败以后，
指向较大字符的指针向后滑动 $k+1$ 个位置。

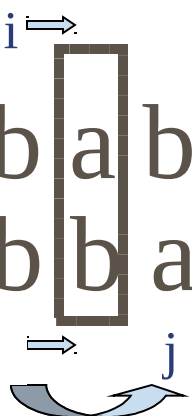
下面让我们将这种方法应用于一个实例。

“最小表示法”在本题的应用

设 $s1='babba'$, $s2='bbaba'$ 。

比较失败时 $k=1$

$u='b \boxed{a} b b a b a b b a'$
 $w='b \boxed{b} a b a b b a b a'$




因为
 $u[i+k] < w[j+k]$
所以移动指针 j

不相等

“最小表示法”在本题的应用

设 $s1='babba'$, $s2='bbaba'$ 。

比较失败时 $k=0$


 i
 $u='b a b b a b a b b a'$
 $w='b b a b a b b a b a'$
 j

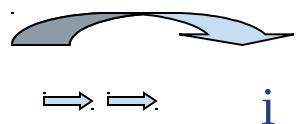
因为
 $u[i+k] > w[j+k]$
 所以移动指针 i

不相等

“最小表示法”在本题的应用

设 $s1='babba'$, $s2='bbaba'$ 。

比较失败时 $k=2$



$u='b\ a\ b\ \boxed{b}\ a\ b\ a\ b\ b\ a'$
 $w='b\ b\ a\ b\ \boxed{a}\ b\ b\ a\ b\ a'$
 $j \Rightarrow \Rightarrow$

因为
 $u[i+k] > w[j+k]$
 所以移动指针 i

不相等

“最小表示法”在本题的应用

设 $s1 = \text{'babba'}$, $s2 = \text{'bbaba'}$ 。

$i \Rightarrow \Rightarrow \Rightarrow \Rightarrow$

$u = \text{'b a b b } \boxed{\text{a b a b b}} \text{a'}$

$w = \text{'b b } \boxed{\text{a b a b b}} \text{a b a'}$

$j \Rightarrow \Rightarrow \Rightarrow \Rightarrow$

$$u[5 \rightarrow 9] = w[3 \rightarrow 7] !$$

所以 $s1$ 和 $s2$ 是循环同构的！

“最小表示法”在本题的应用

在这个例子中，算法正确出解。



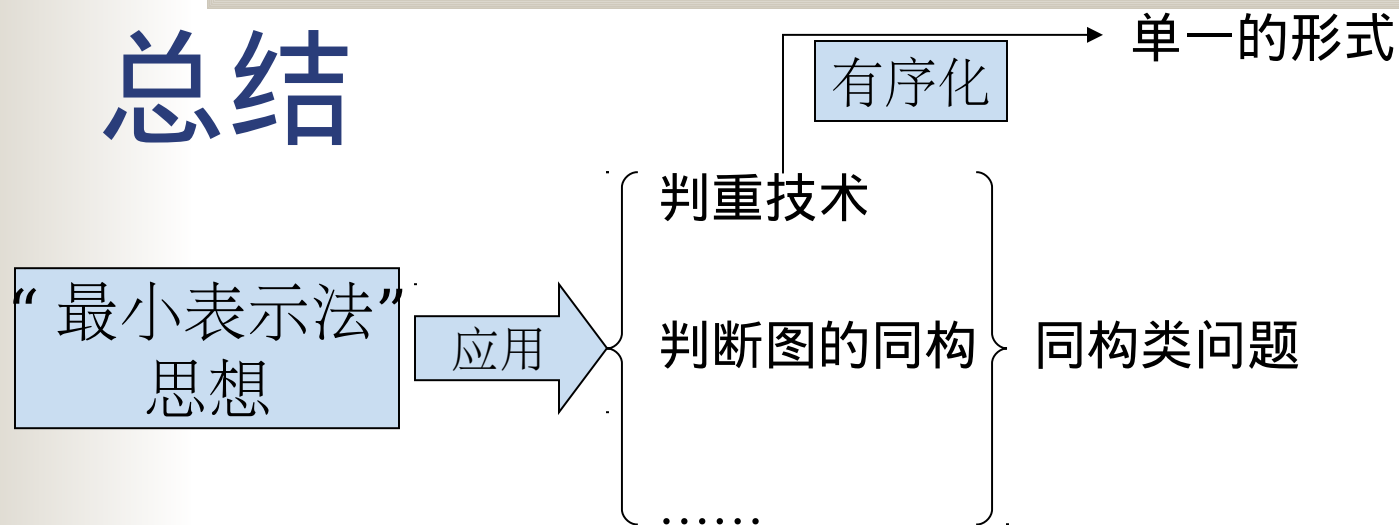
算法的具体描述和证明请同学们自行完成，这里不再赘述。

小结：“最小表示法”思想

经过努力，我们终于找到了一个与匹配算法本质不同的线性算法。

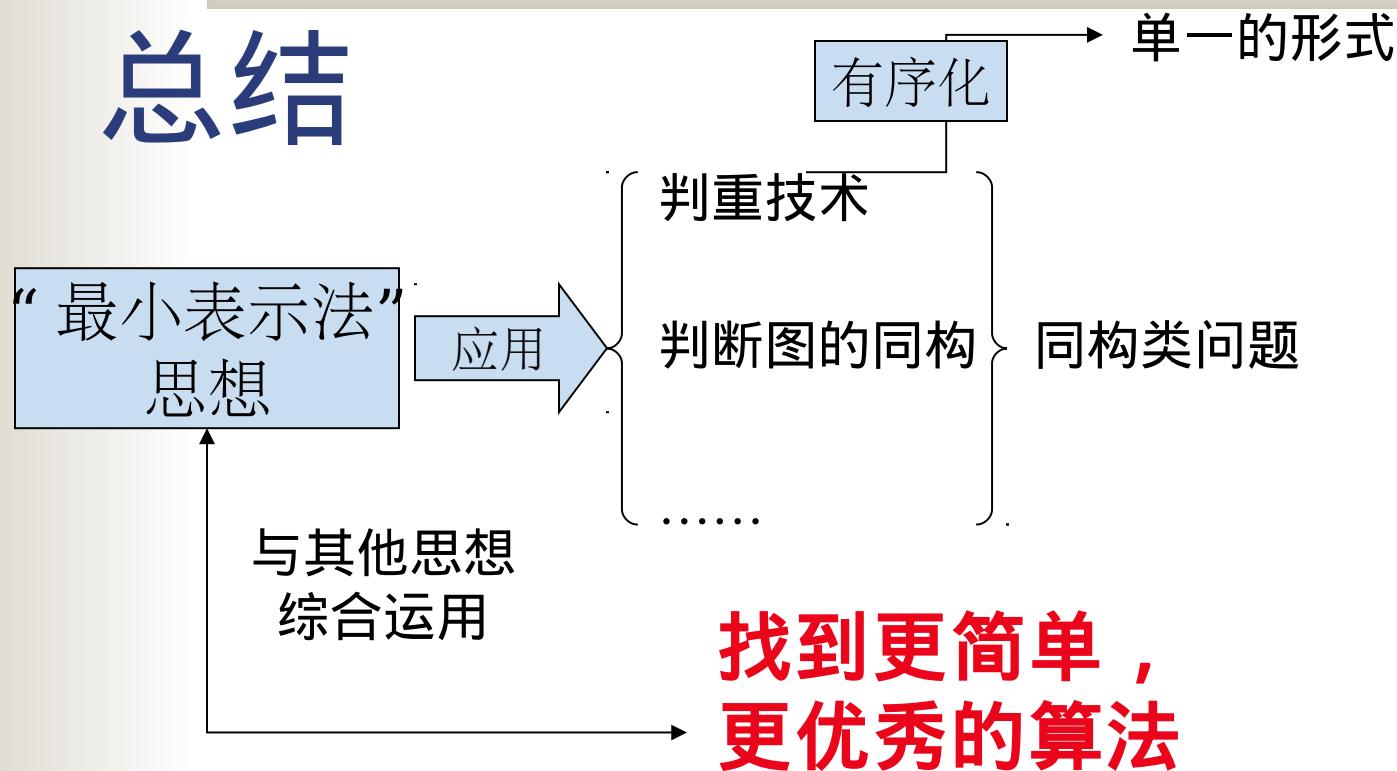
比较点	匹配算法	“最小表示法”思想
时间复杂度	$O(N)$ 级	同样优秀的线性算法
辅助空间	记录 next 数组 $O(N)$ 级	只需要记录两个指针 常数级别
算法实现	难懂，难记忆	简洁，便于记忆
可扩展性	受 next 数组严重制约	很强

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。仔细分析可以得出，其思想精华在于引入了“序”这个概念，从而将纷繁的待处理对象化为单一的形式，便于比较。

总结



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。

总结

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

谢谢大家

Thank you!