

本章在408中的地位

小秘密：王道书每章开头有考纲



- (四) 图的基本应用
1. 最小(代价)生成树
 2. 最短路径
 3. 拓扑排序
 4. 关键路径

五、查找

- (一) 查找的基本概念
- (二) 顺序查找法
- (三) 分块查找法
- (四) 折半查找法
- (五) B 树及其基本操作、B⁺树的基本概念
- (六) 散列 (Hash) 表
- (七) 字符串模式匹配
- (八) 查找算法的分析及应用

六、排序

- (一) 排序的基本概念
- (二) 插入排序
1. 直接插入排序
 2. 折半插入排序
- (三) 起泡排序 (Bubble Sort)
- (四) 简单选择排序
- (五) 希尔排序 (Shell Sort)
- (六) 快速排序
- (七) 堆排序
- (八) 二路归并排序 (Merge Sort)
- (九) 基数排序
- (十) 外部排序
- (十一) 各种排序算法的比较



你好像没有任何牌面吧?

(十二) 排序算法的应用

计算机组成原理

[考查目标]

1. 理解单处理器计算机系统中各部件的内部工作原理、组成结构以及相互连接方式,具有完整的计算机系统的整机概念。
2. 理解计算机系统层次化结构概念,熟悉硬件与软件之间的界面,掌握指令集体系结构的基本知识和基本实现方法。
3. 能够综合运用计算机组成的基本原理和基本方法,对有关计算机硬件系统中的理论和实际问题进行计算、分析,对一些基本部件进行简单设计;并能对高级程序设计语言(如 C 语言)中的相关问题进行分析。

一、计算机系统概述

(一) 计算机系统层次结构

1. 计算机系统的基本组成
2. 计算机硬件的基本组成
3. 计算机软件 and 硬件的关系
4. 计算机系统的工作过程

(二) 计算机性能指标

吞吐量、响应时间, CPU 时钟周期、主频、CPI、CPU 执行时间, MIPS、MFLOPS、GFLOPS、TFLOPS、PFLOPS、EFLOPS、ZFLOPS。

二、数据的表示和运算

(一) 数制与编码

1. 进位计数制及其相互转换



本节内容

字符串

朴素模式匹配算法

什么是字符串的模式匹配

自动保存 关闭 >> 访问模式

开始 插入 绘图 设计 布局 引用 >> 告诉我 共享 批注

2020 年操作系统考研复习指导

待最后一个进程关闭文件。计数器跟踪打开和关闭的数量，计数为 0 时，系统关闭文件，删除该条目。

- 文件磁盘位置。绝大多数文件操作都要求系统修改文件数据。该信息保存在内存中，以免为每个操作都从磁盘中读取。
- 访问权限。每个进程打开文件都需要有一个访问模式（创建、只读、读写、添加等）。该信息保存在进程的打开文件表中，以便操作系统能够允许或拒绝之后的 I/O 请求。

4.1.2 文件的逻辑结构

文件的逻辑结构是从用户观点出发看到的文件的组织形式。文件的物理结构（又称文件的存储结构，见 4.2.1 节）是从实现观点出发看到的文件在外存上的存储组织形式。文件的逻辑结构与存储介质特性无关，但文件的物理结构与存储介质的特性有很大关系。文件的逻辑结构实际上是指在文件的内部，数据逻辑上是如何组织起来的。

按逻辑结构，文件可划分为无结构文件和有结构文件两种。

1. 无结构文件（流式文件）

无结构文件是最简单的文件组织形式。无结构文件将数据按顺序组织成记录并积累、保存，它是有序相关信息项的集合，以字节（Byte）为单位。由于无结构文件没有结构，因而对记录的访问只能通过穷举搜索的方式，故这种文件形式对大多数应用不适用。但字符流的无结构文件管理简单，用户可以方便地对其进行操作。所以，那些对基本信息单位操作不多的文件较适于采用字符流的无结构方式，如源程序文件、目标代码文件等。

2. 有结构文件（记录式文件）

有结构文件按记录的组织形式可以分为如下几种：

- 1) 顺序文件。文件中的记录一个接一个地顺序排列，记录通常是定长的，可以顺序存储或以链表形式存储。在访问时需要顺序地查找文件。顺序文件有以下两种结构：第一种是

第 6 页，共 64 页 专注 112%

JOJO 的奇妙冒险

Baidu 百度 百度一下

网页 资讯 视频 图片 知道 文库 贴吧 地图 采购 更多

百度为您找到相关结果约 30,700,000 个 搜索工具

JOJO 的奇妙冒险(荒木飞吕彦创作的漫画) - 百度百科

类型：漫画作品
作者：荒木飞吕彦
简介：《JOJO 的奇妙冒险》是由日本漫画家荒木飞吕彦所著漫画。漫画于 1987 年至 2004 年在集英社的少年漫画杂志少年 JUMP 上连载（1987 年..

作品简介 故事内容 各卷目录 作品内容 角色介绍 更多 >

baike.baidu.com/

JOJO 的奇妙冒险:黄金之风 - 高清视频在线观看

已完结 全 39 集
2018 日本
类型：动画 热血 奇幻
百科：公元 2001 年，意大利那不勒斯的少年乔鲁诺·乔巴拿（DIO 的儿子），为了达成梦想、成为一名黑道巨星，而加入了黑手党组织“热情”，并成为布加拉提小队的成员，故事由此开始.....更多 >

第一集 第二集 第三集 第四集 第五集 第六集 第七集 第八集 第九集 第十集 第十一集 第十二集 第十三集 第十四集 第十五集 第十六集 第十七集 第十八集 第十九集 第二十集 第二十一集 第二十二集 第二十三集 第二十四集 第二十五集 第二十六集 第二十七集 第二十八集 第二十九集 第三十集 第三十一集 第三十二集 第三十三集 第三十四集 第三十五集 第三十六集 第三十七集 第三十八集 第三十九集

什么是字符串的模式匹配



主串

‘嘿嘿嘿红红火火恍恍惚惚嗨皮开森猴开森笑出猪叫哈哈哈哈哈嗨森哈哈哈哈哈嗝’

模式串

‘笑出猪叫’

字符串模式匹配：在主串中找到与模式串相同的子串，并返回其所在位置。

什么是字符串的模式匹配

与模式串匹配的子串

主串

‘嘿嘿嘿红红火火恍恍惚惚嗨皮开森猴开森笑出猪叫哈哈哈哈哈嗨森哈哈哈哈哈嗝’

模式串

‘笑出猪叫’

模式串

‘笑出喵叫’

子串——主串的一部分，一定存在
模式串——不一定能在主串中找到

字符串模式匹配：在主串中找到与模式串相同的子串，并返回其所在位置。

两种模式匹配算法

字符串模式匹配

朴素模式匹配算法

KMP 算法

朴素模式匹配算法

主串S: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

a	b	a	a	b	a	a	b	c	a	b	a	a	b	c
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

模式串T: 1 2 3 4 5 6

a	b	a	a	b	c
---	---	---	---	---	---



开始暴力解决问题了

朴素模式匹配算法

主串S:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
a	b	a	a	b	a	a	b	c	a	b	a	a	b	c

模式串T:

1	2	3	4	5	6
a	b	a	a	b	c



开始暴力解决问题了

朴素模式匹配算法

主串S: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
a b a a b a a b c a b a a b c

模式串T: a b a a b c
1 2 3 4 5 6



开始暴力解决问题了

朴素模式匹配算法

主串S:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
a	b	a	a	b	a	a	b	c	a	b	a	a	b	c

模式串T:

a	b	a	a	b	c
1	2	3	4	5	6



开始暴力解决问题了

朴素模式匹配算法

主串S: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
a b a a b a a b c a b a a b c

模式串T: a b a a b c
1 2 3 4 5 6



开始暴力解决问题了

朴素模式匹配算法

主串S: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

a	b	a	a	b	a	a	b	c	a	b	a	a	b	c
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

模式串T:

a	b	a	a	b	c
1	2	3	4	5	6



开始暴力解决问题了

朴素模式匹配算法

主串S: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
a b a a b a a b c a b a a b c

模式串T: a b a a b c
1 2 3 4 5 6



开始暴力解决问题了

朴素模式匹配算法

主串S: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
a b a a b a a b c a b a a b c

模式串T: a b a a b c
1 2 3 4 5 6



开始暴力解决问题了

朴素模式匹配算法

主串S:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
a	b	a	a	b	a	a	b	c	a	b	a	a	b	c

模式串T:

a	b	a	a	b	c
1	2	3	4	5	6



开始暴力解决问题了

朴素模式匹配算法

主串S:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
a	b	a	a	b	a	a	b	c	a	b	a	a	b	c

模式串T:

a	b	a	a	b	c
1	2	3	4	5	6



开始暴力解决问题了

朴素模式匹配算法

主串S: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
a b a a b a a b c a b a a b c

模式串T: a b a a b c
1 2 3 4 5 6



开始暴力解决问题了

朴素模式匹配算法

主串S: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
a b a a b a a b c a b a a b c

模式串T: a b a a b c
1 2 3 4 5 6



开始暴力解决问题了

主串长度为 n ，模式串长度为 m

最多对比 $n-m+1$ 个子串

朴素模式匹配算法：将主串中所有长度为 m 的子串依次与模式串对比，直到找到一个完全匹配的子串，或所有的子串都不匹配为止。

朴素模式匹配算法

Index(S,T): 定位操作。若主串S中存在与串T值相同的子串，则返回它的主串S中第一次出现的位置；否则函数值为0。

```
int Index(SString S, SString T){
    int i=1, n=StrLength(S), m=StrLength(T);
    SString sub;    //用于暂存子串
    while(i<=n-m+1){
        SubString(sub, S, i, m);
        if(StrCompare(sub, T)!=0) ++i;
        else return i;    //返回子串在主串中的位置
    }
    return 0;    //S中不存在与T相等的子串
}
```

最多对比 $n-m+1$ 个子串

子串和模式串对比，若不匹配，则匹配下一个子串

取出从位置 i 开始，长度为 m 的子串

原来是这样啊



接下来：不使用字符串的基本操作，直接通过数组下标实现朴素模式匹配算法

朴素模式匹配算法

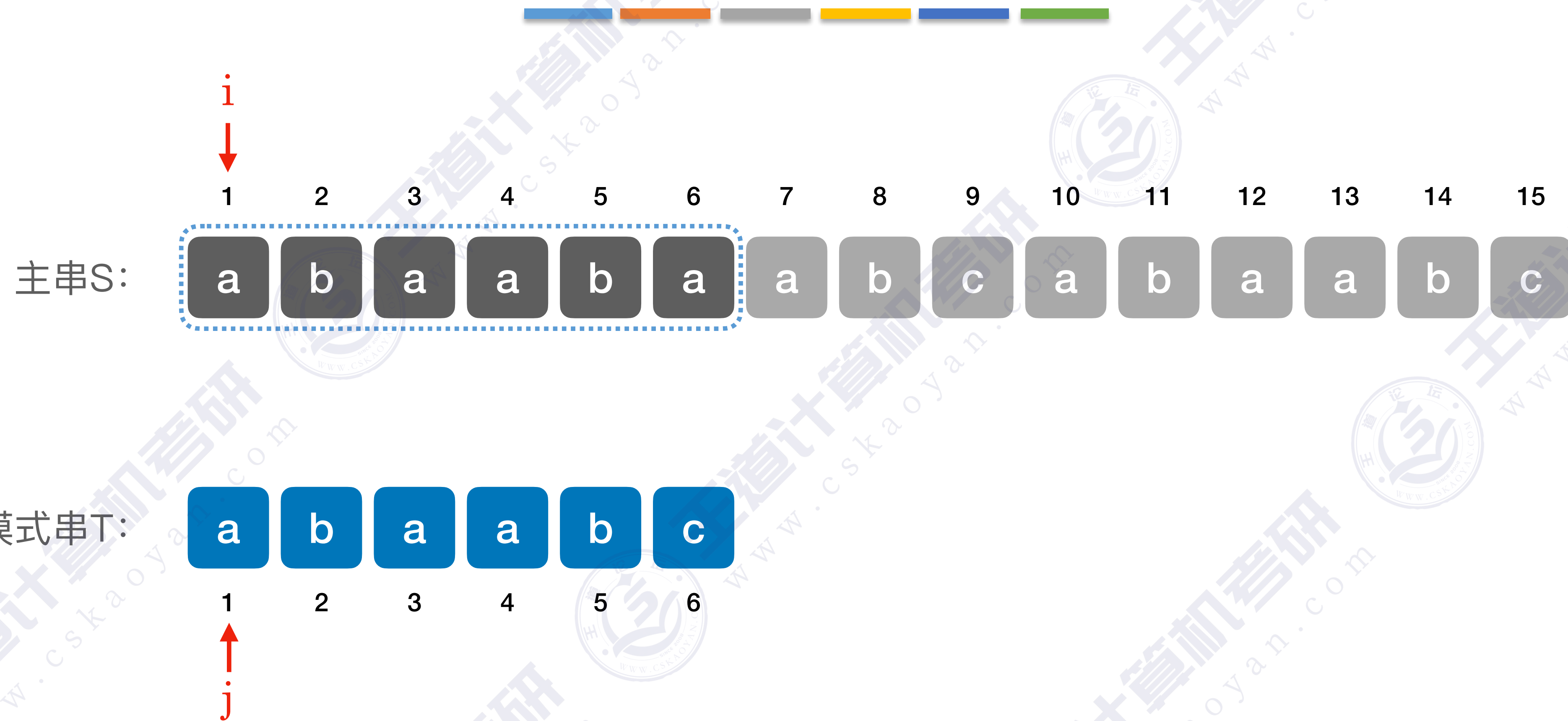
主串S:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
a	b	a	a	b	a	a	b	c	a	b	a	a	b	c

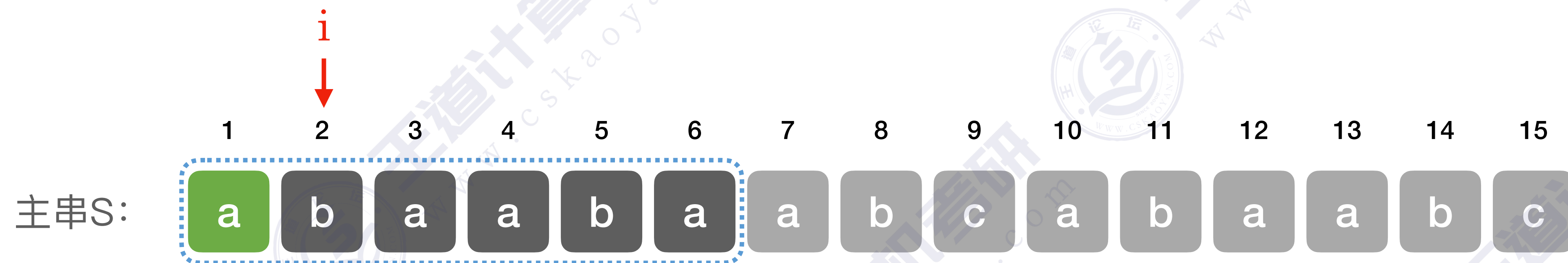
模式串T:

a	b	a	a	b	c
1	2	3	4	5	6

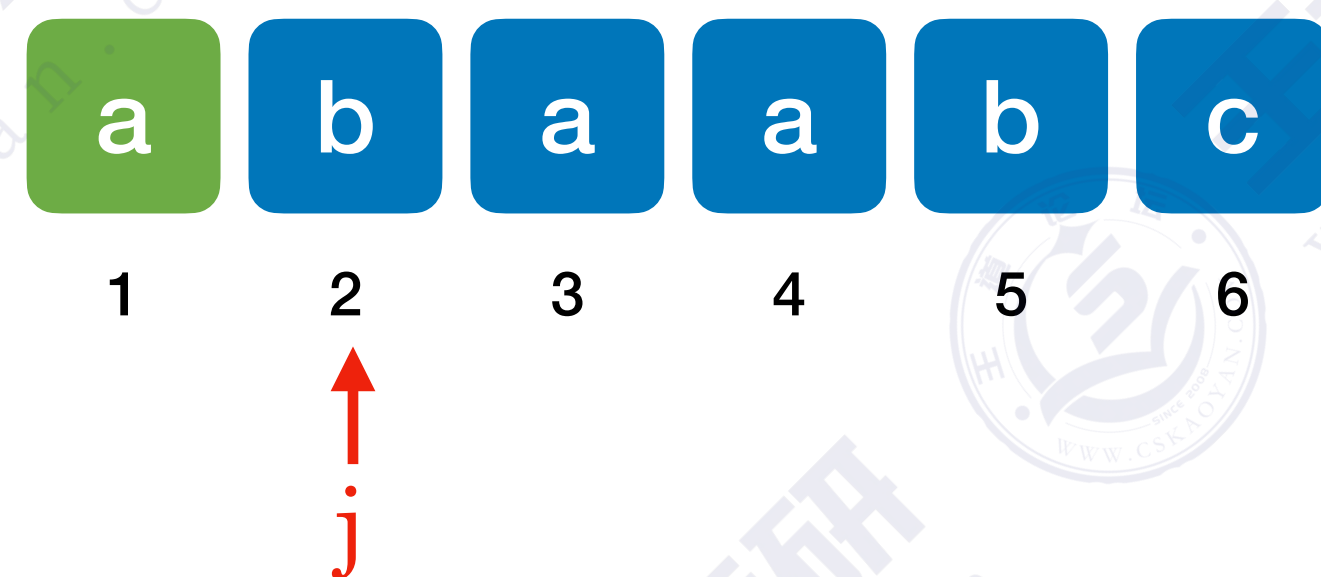
朴素模式匹配算法



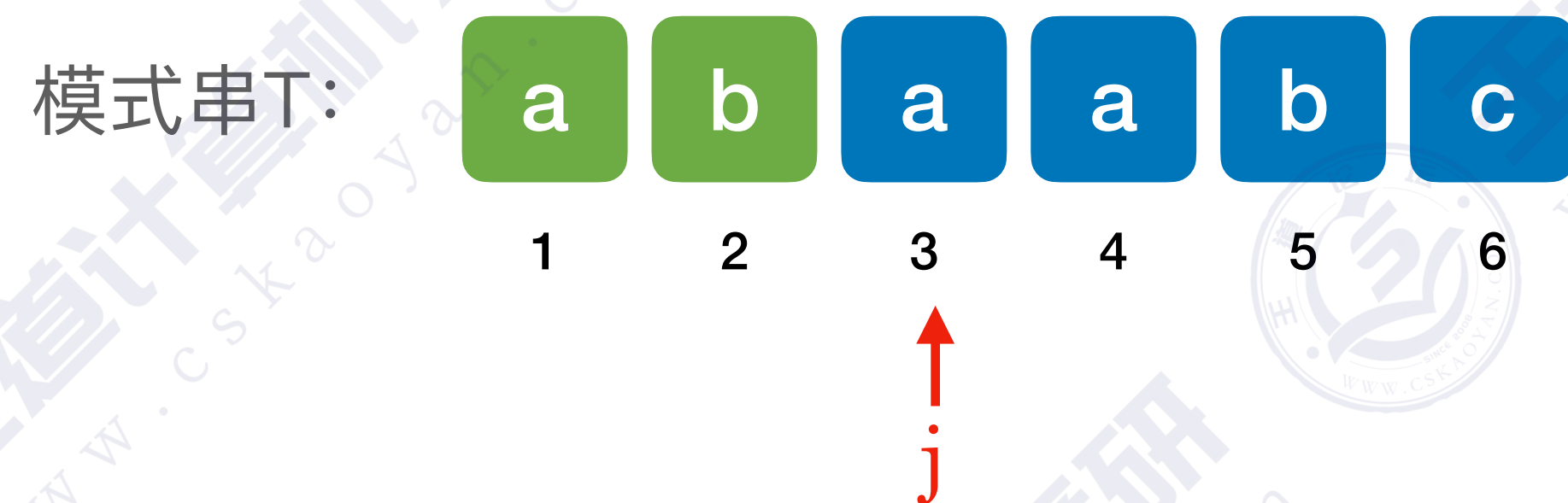
朴素模式匹配算法



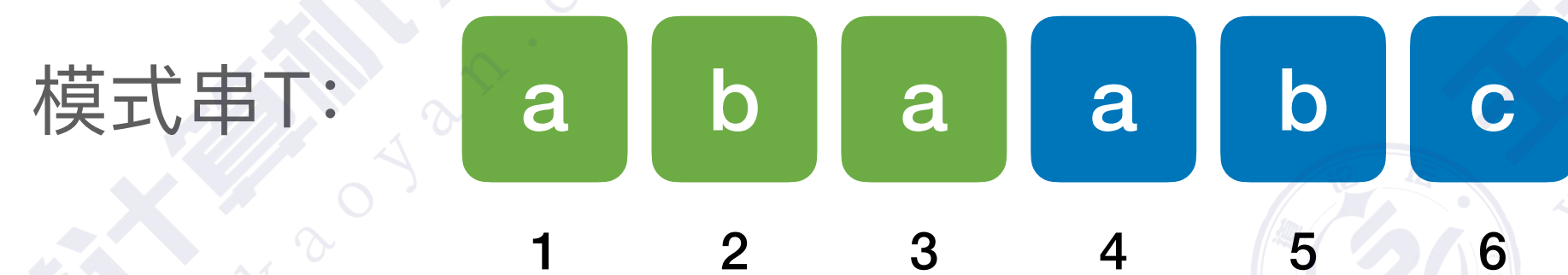
模式串T:



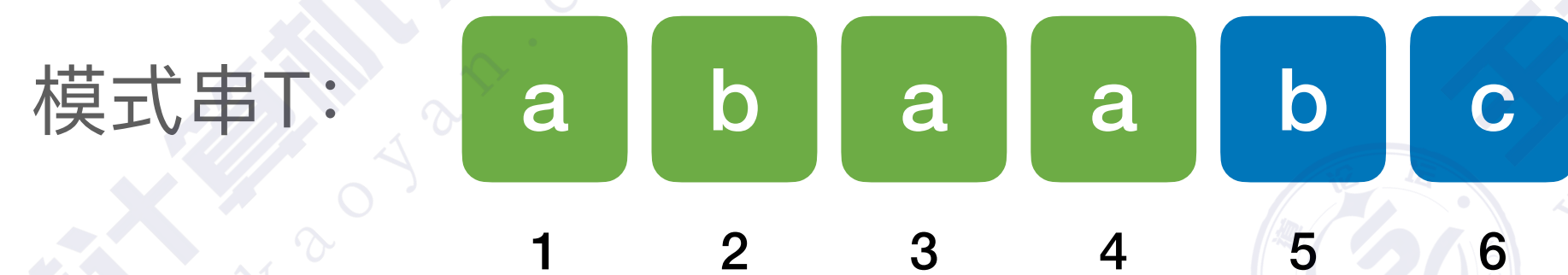
朴素模式匹配算法



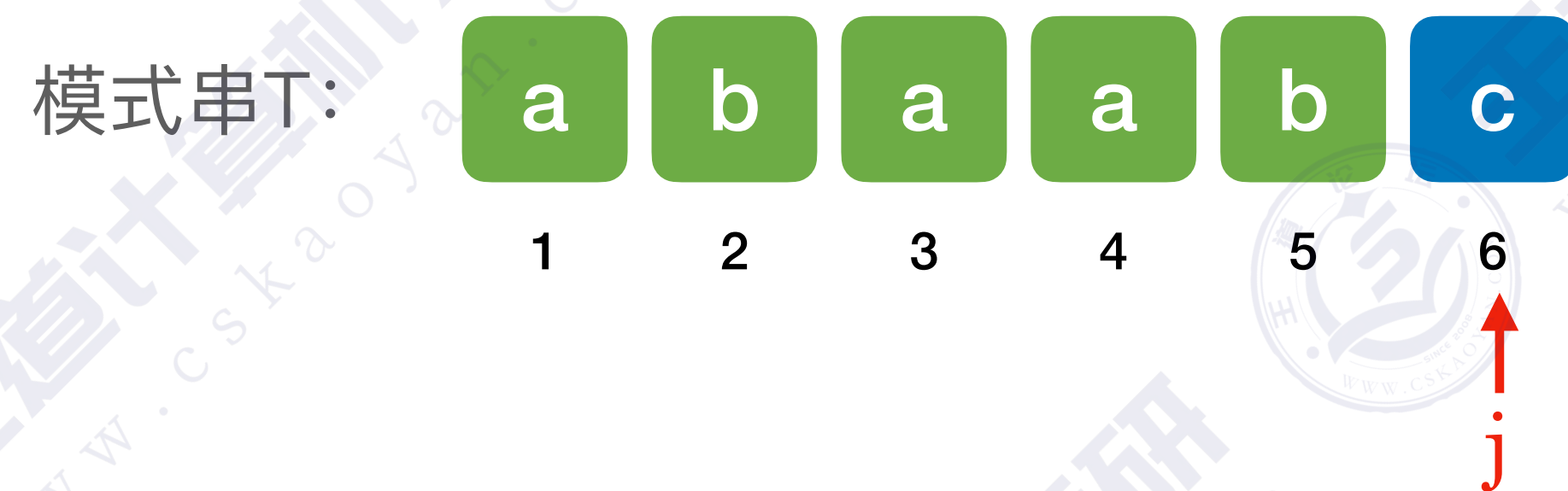
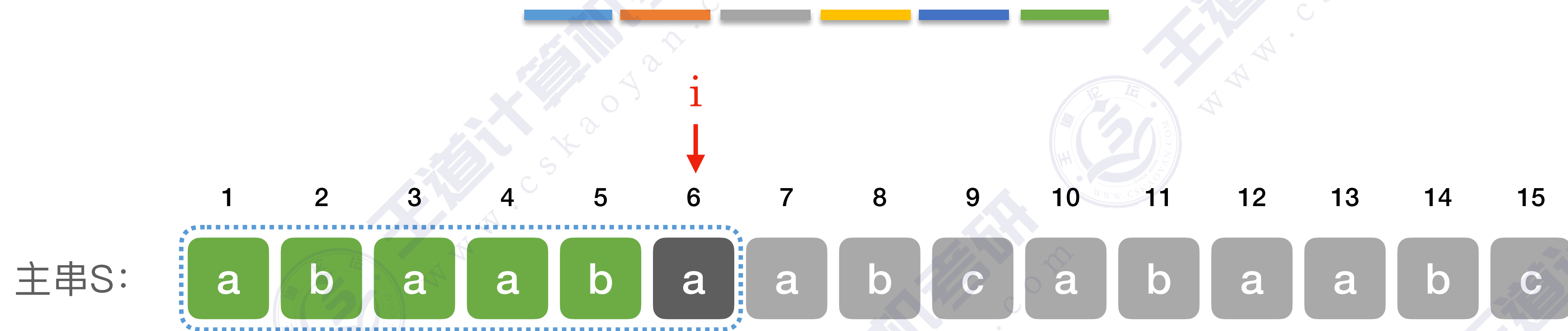
朴素模式匹配算法



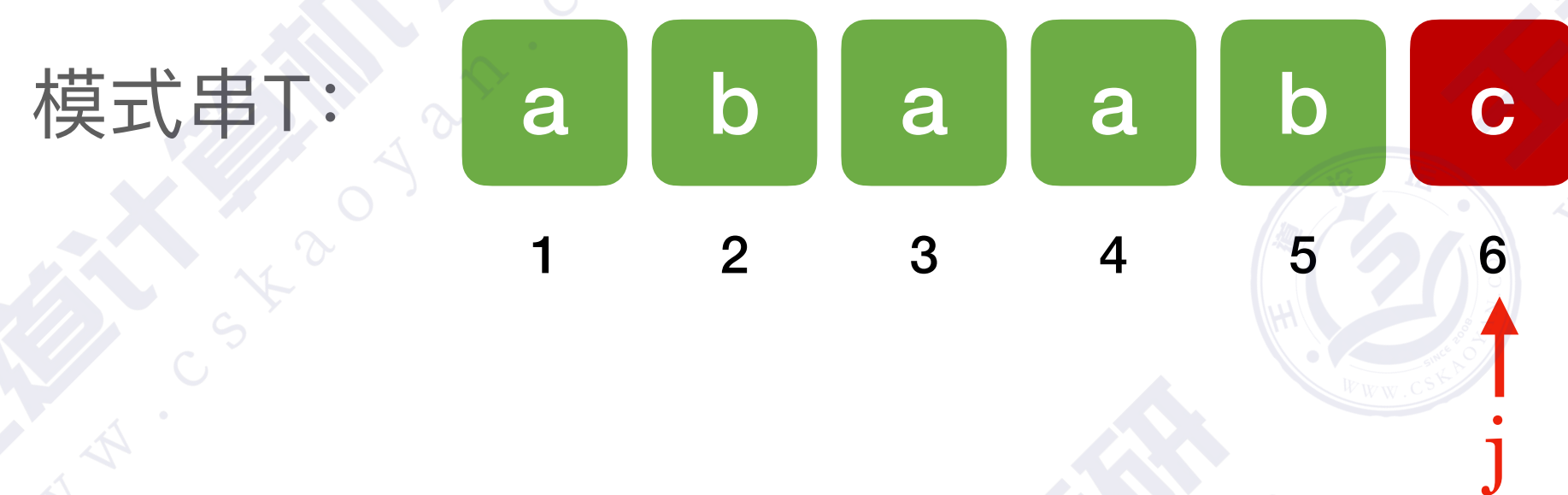
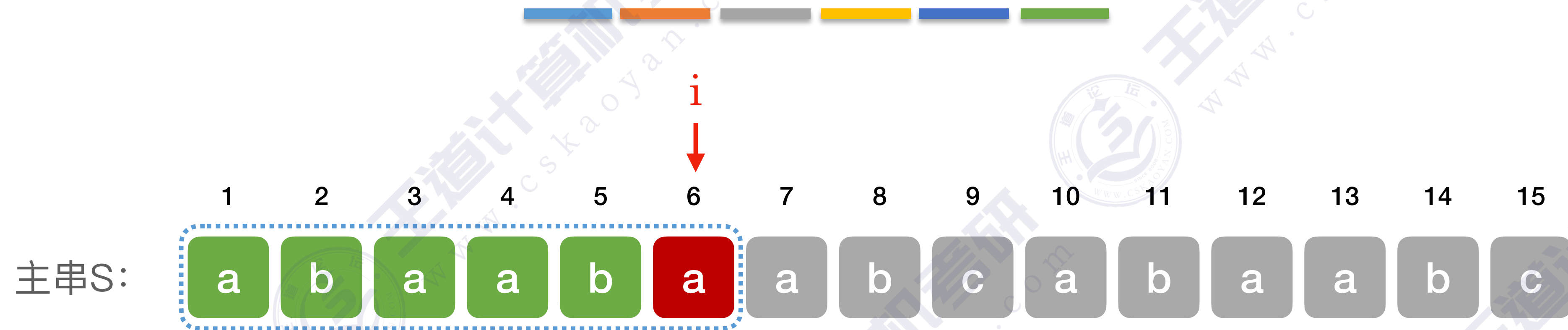
朴素模式匹配算法



朴素模式匹配算法



朴素模式匹配算法

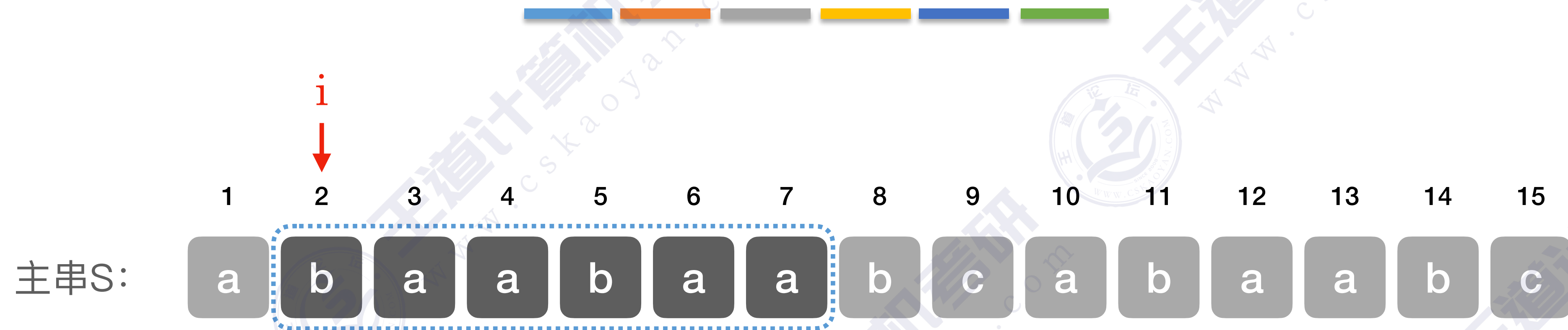


$$i = i - j + 2;$$
$$j = 1;$$

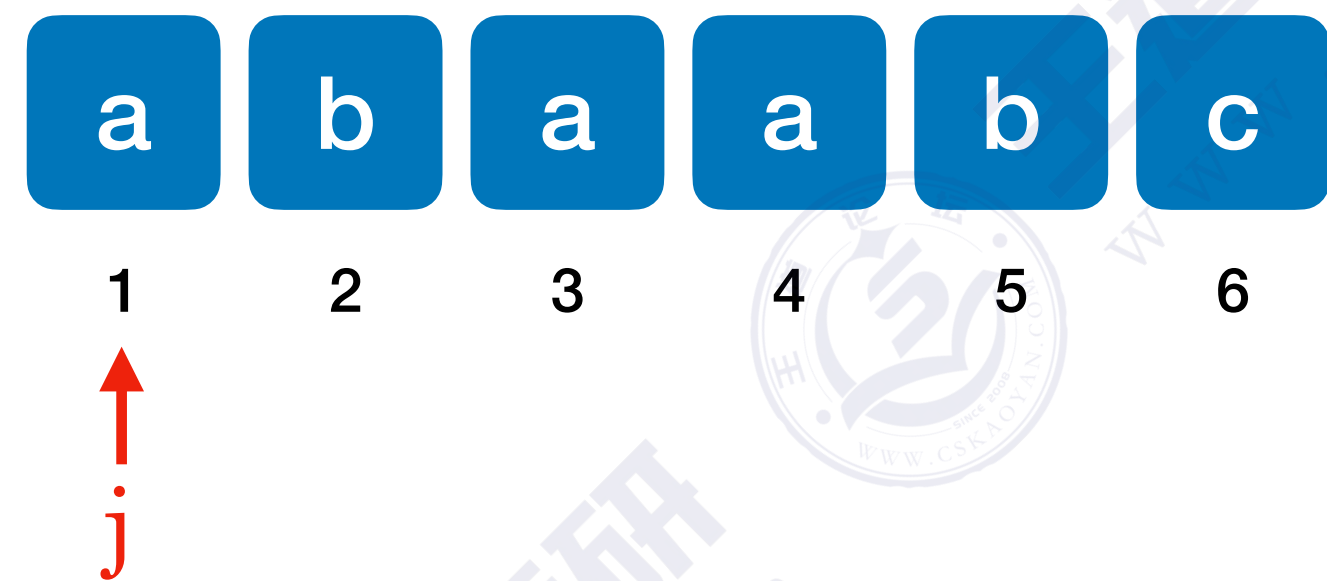


若当前子串匹配失败，则主串指针 i 指向下一个子串的第一个位置，模式串指针 j 回到模式串的第一个位置

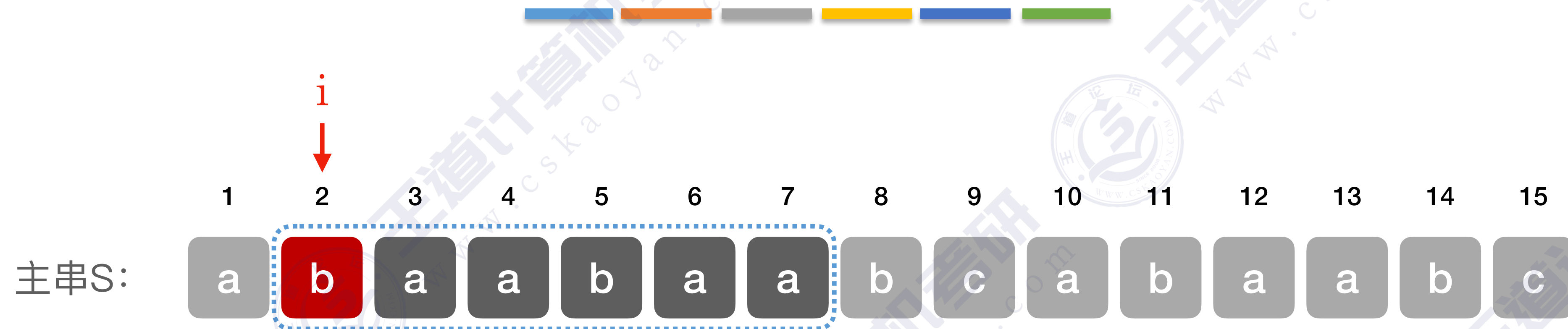
朴素模式匹配算法



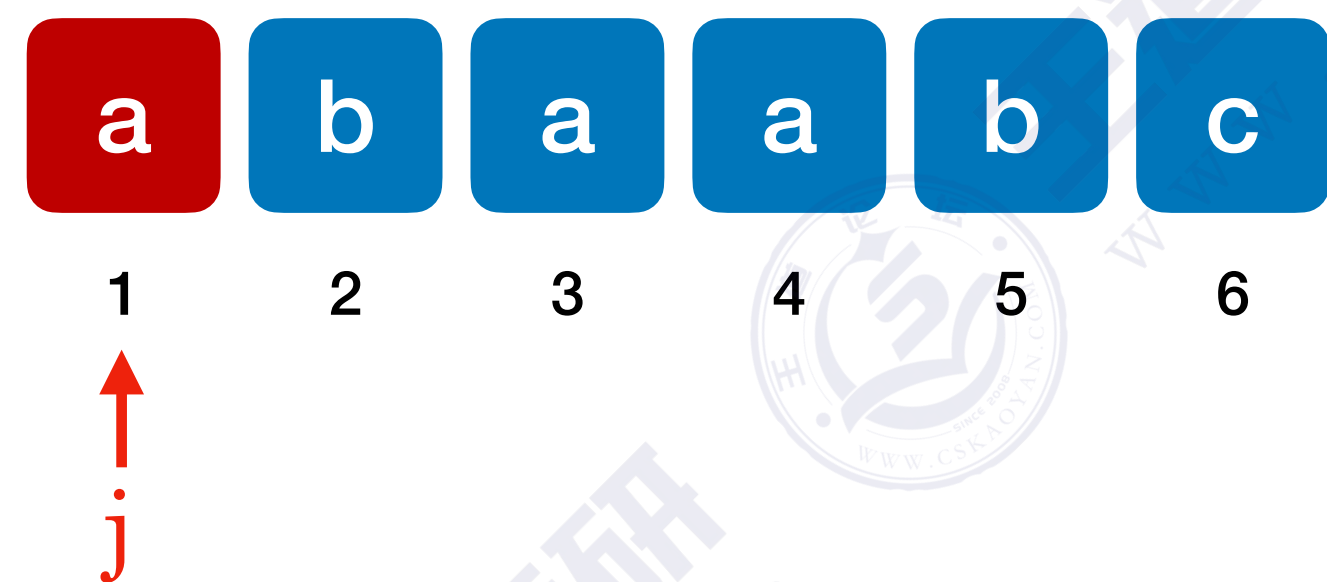
模式串T:



朴素模式匹配算法



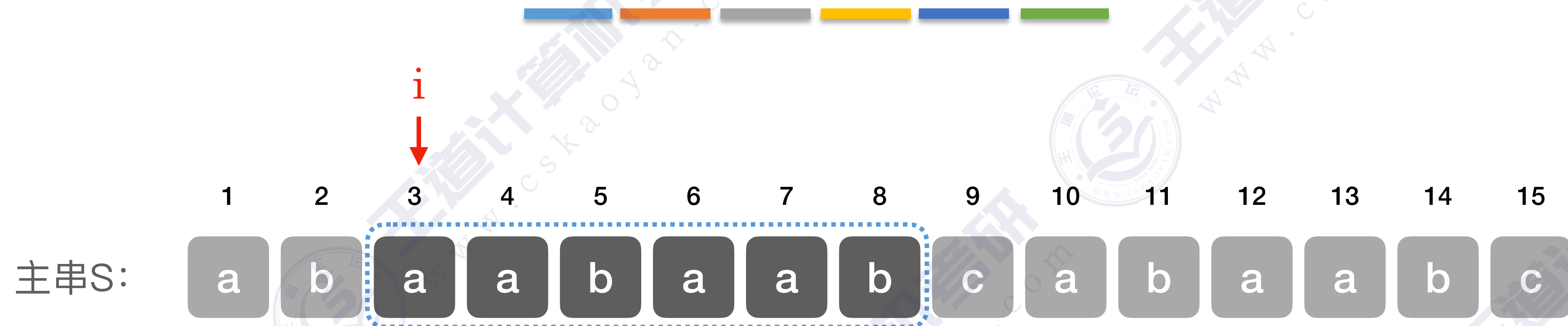
模式串T:



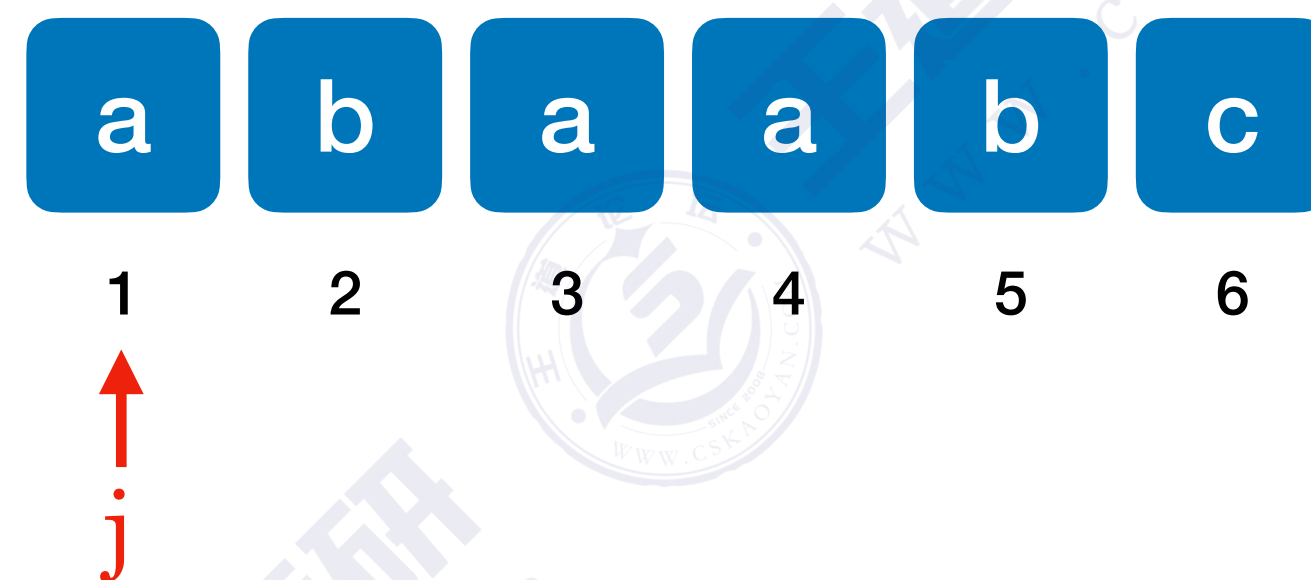
$i = i - j + 2;$
 $j = 1;$

若当前子串匹配失败，则主串指针 i 指向下一个子串的第一个位置，模式串指针 j 回到模式串的第一个位置

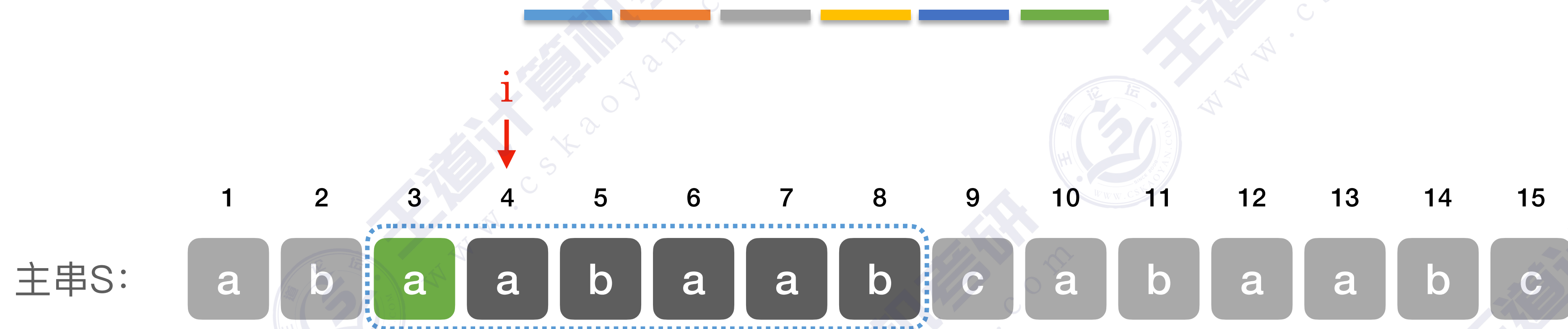
朴素模式匹配算法



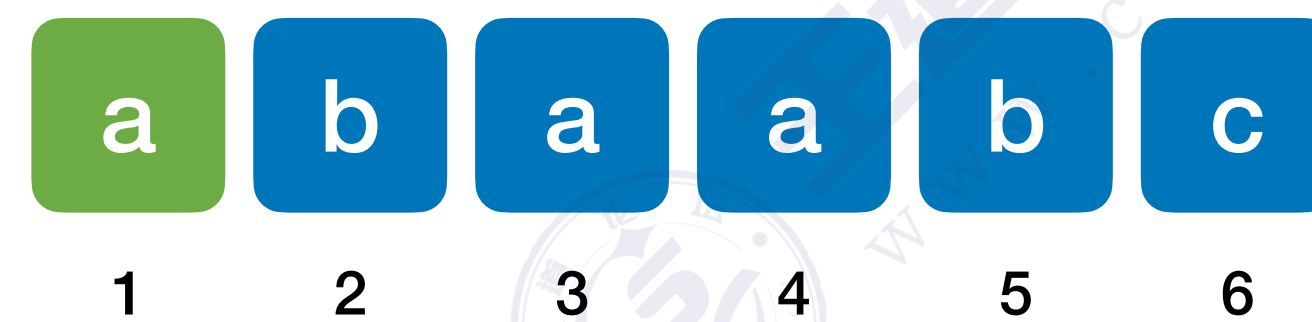
模式串T:



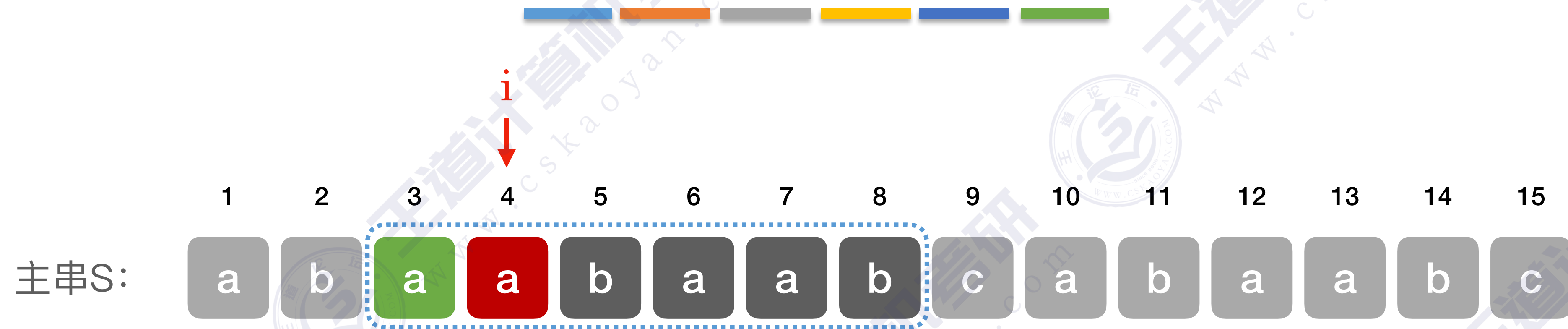
朴素模式匹配算法



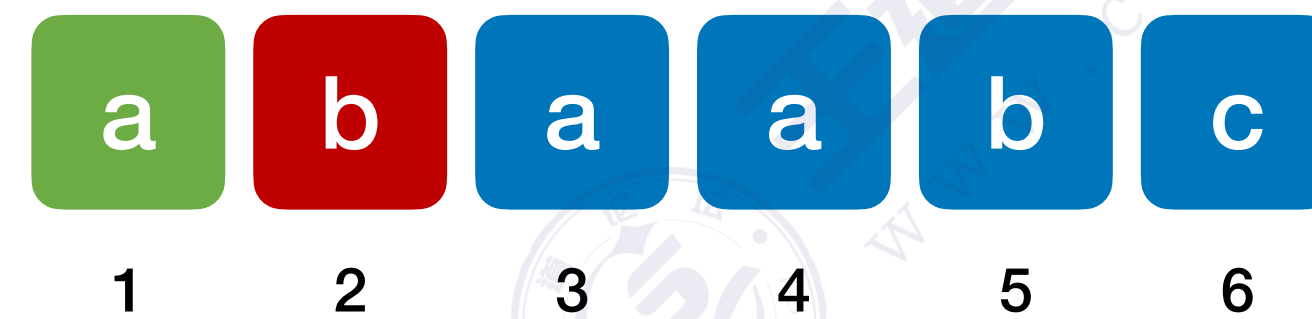
模式串T:



朴素模式匹配算法



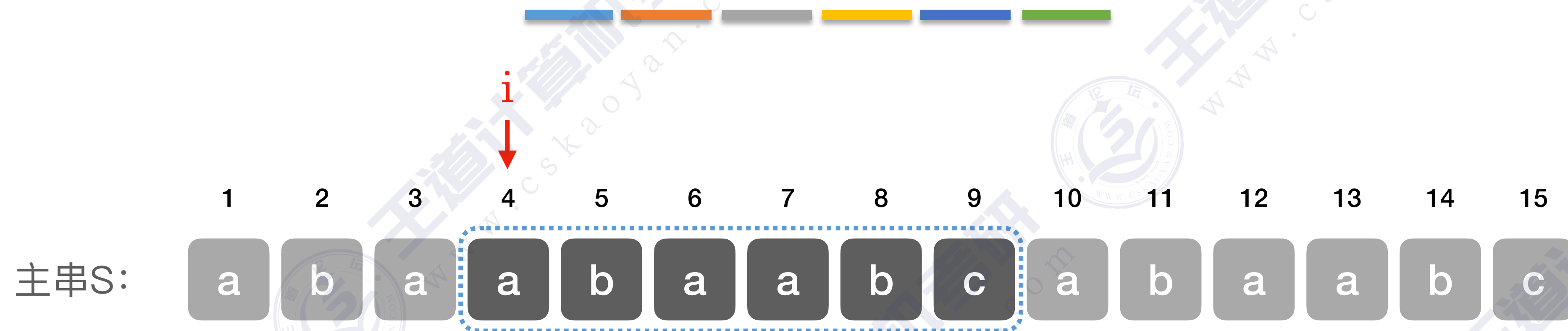
模式串T:



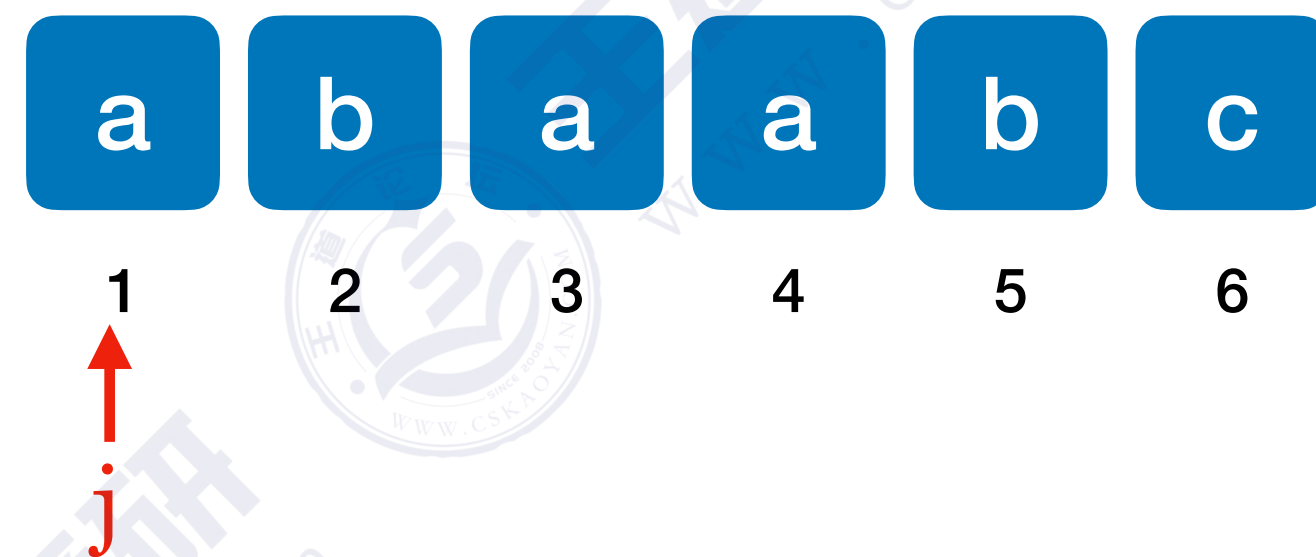
$i = i - j + 2;$
 $j = 1;$

若当前子串匹配失败，则主串指针 i 指向下一个子串的第一个位置，模式串指针 j 回到模式串的第一个位置

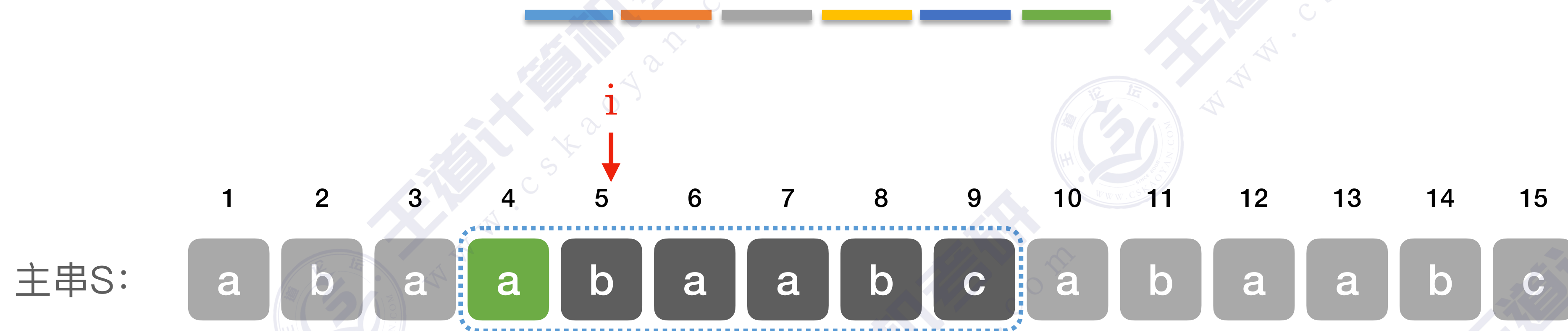
朴素模式匹配算法



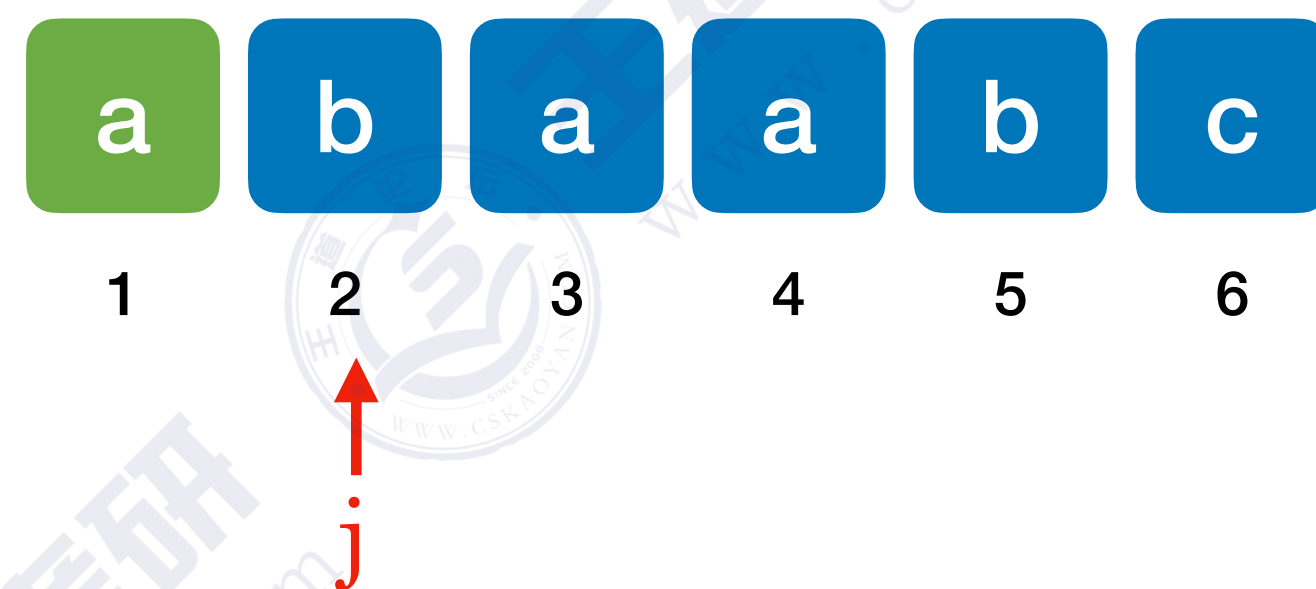
模式串T:



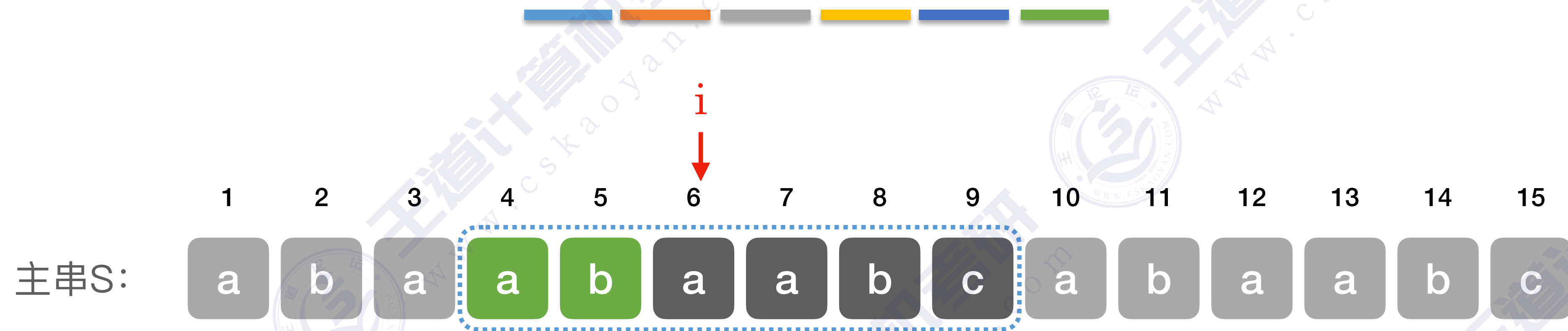
朴素模式匹配算法



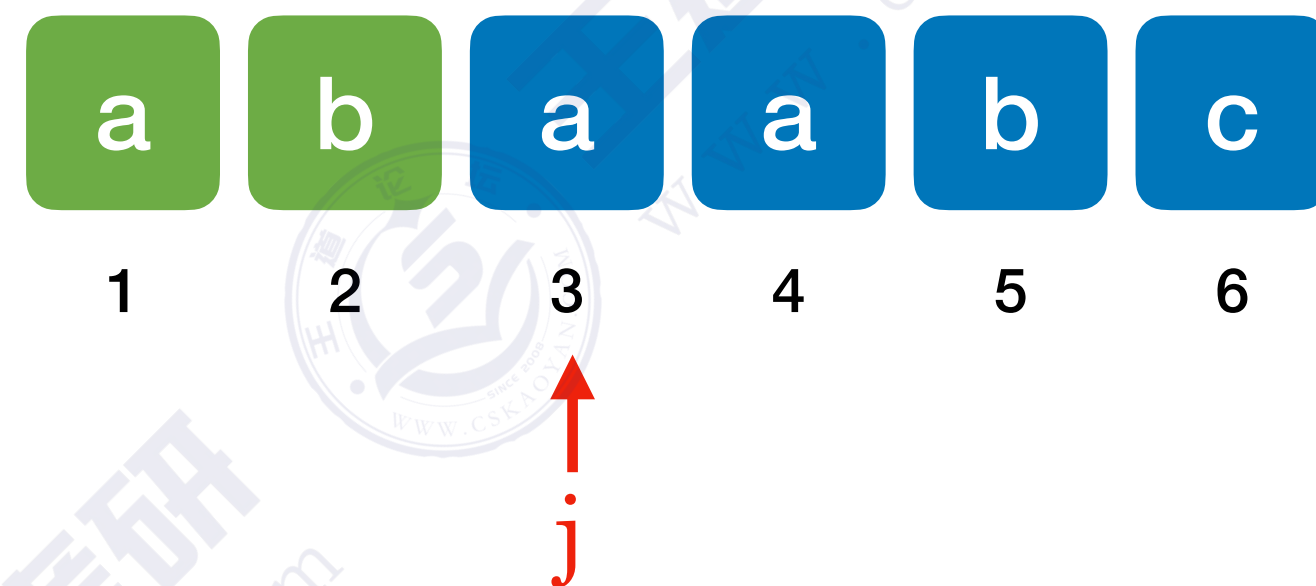
模式串T:



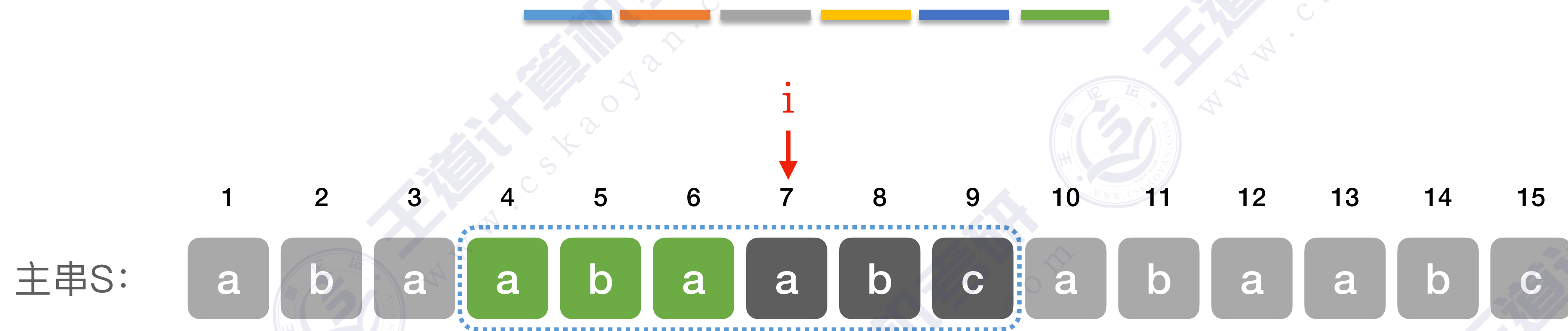
朴素模式匹配算法



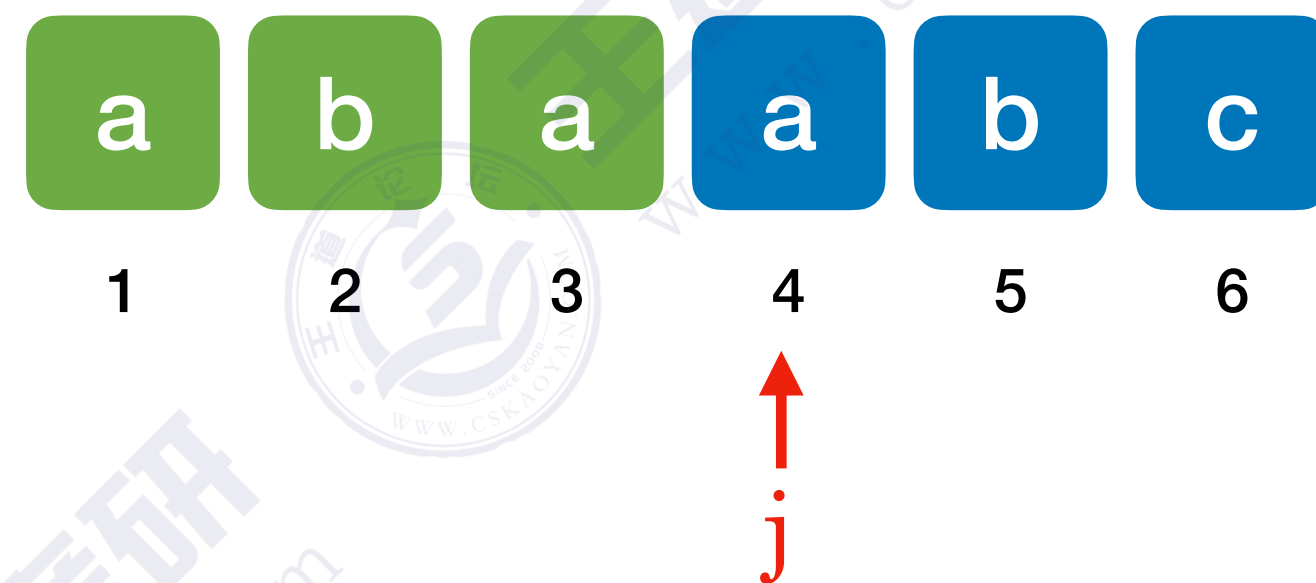
模式串T:



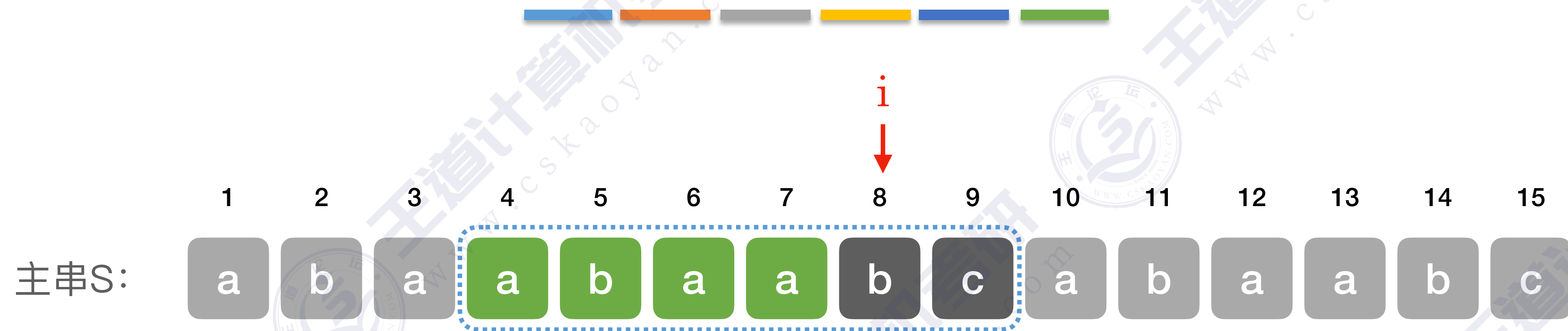
朴素模式匹配算法



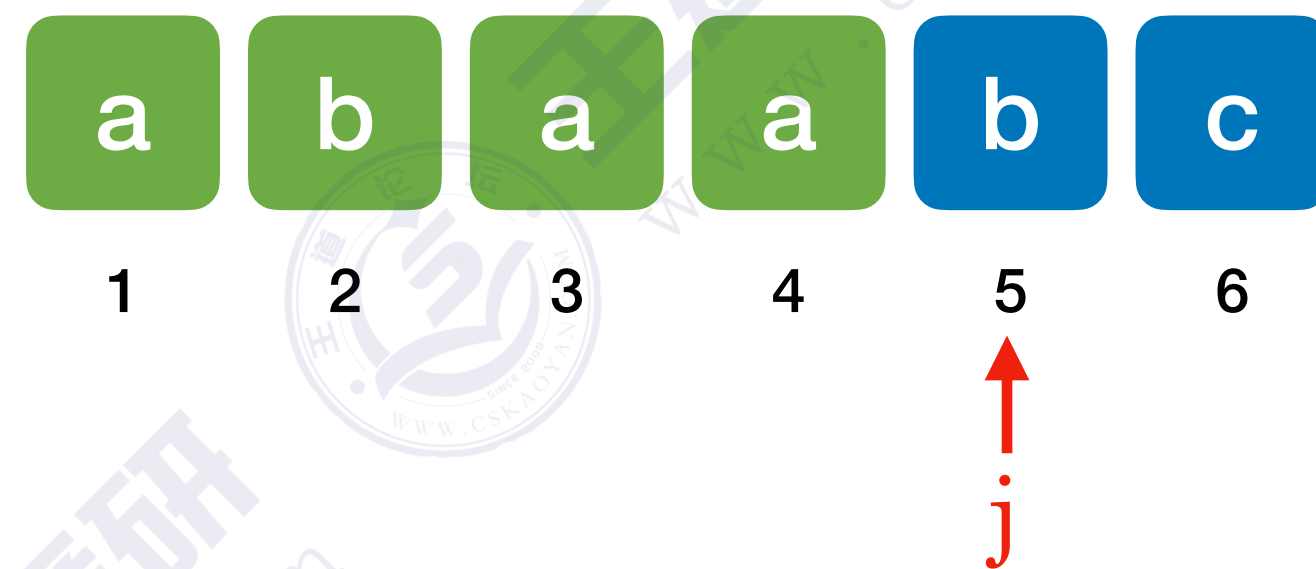
模式串T:



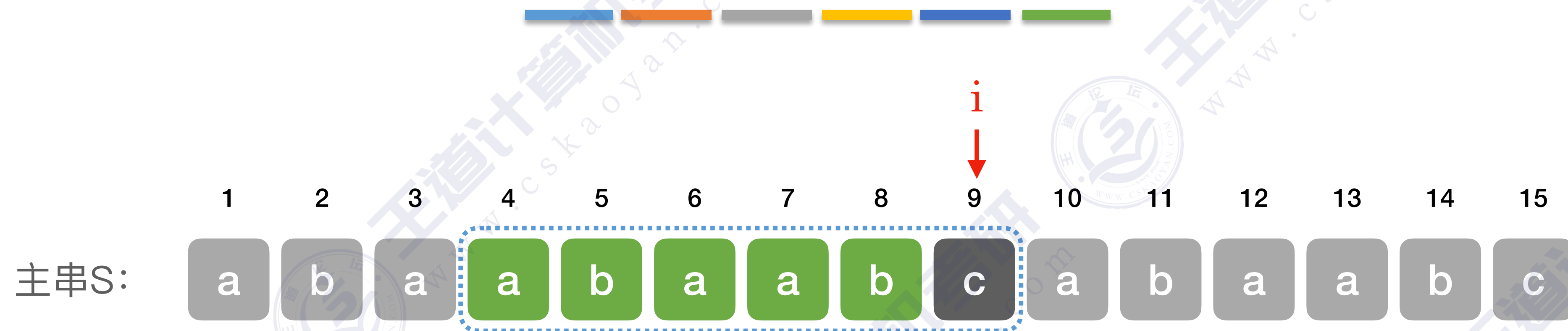
朴素模式匹配算法



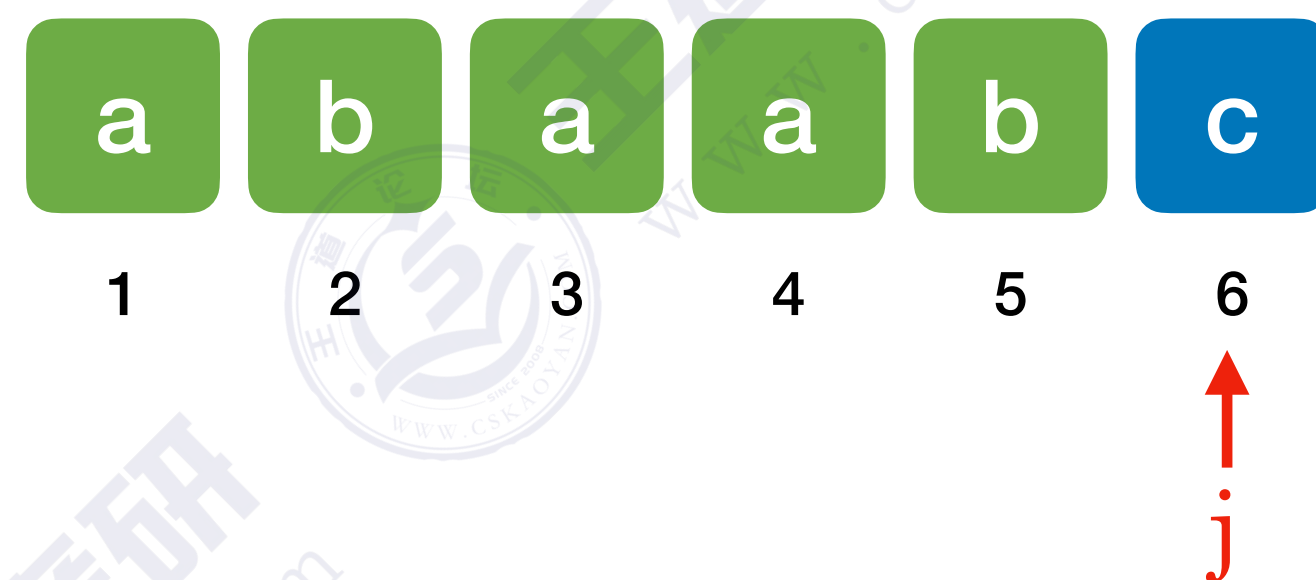
模式串T:



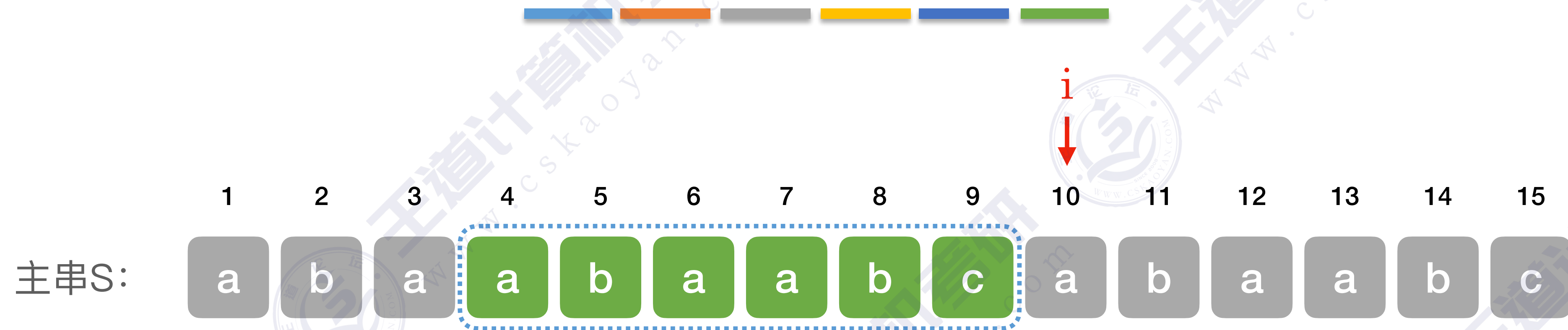
朴素模式匹配算法



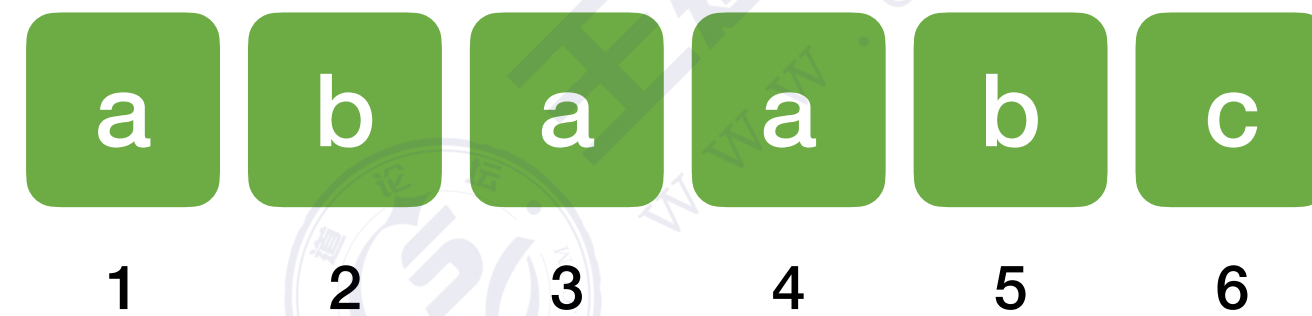
模式串T:



朴素模式匹配算法



模式串T:



若 $j > T.length$, 则当前子串匹配成功, 返回当前子串第一个字符的位置 —— $i - T.length$

朴素模式匹配算法

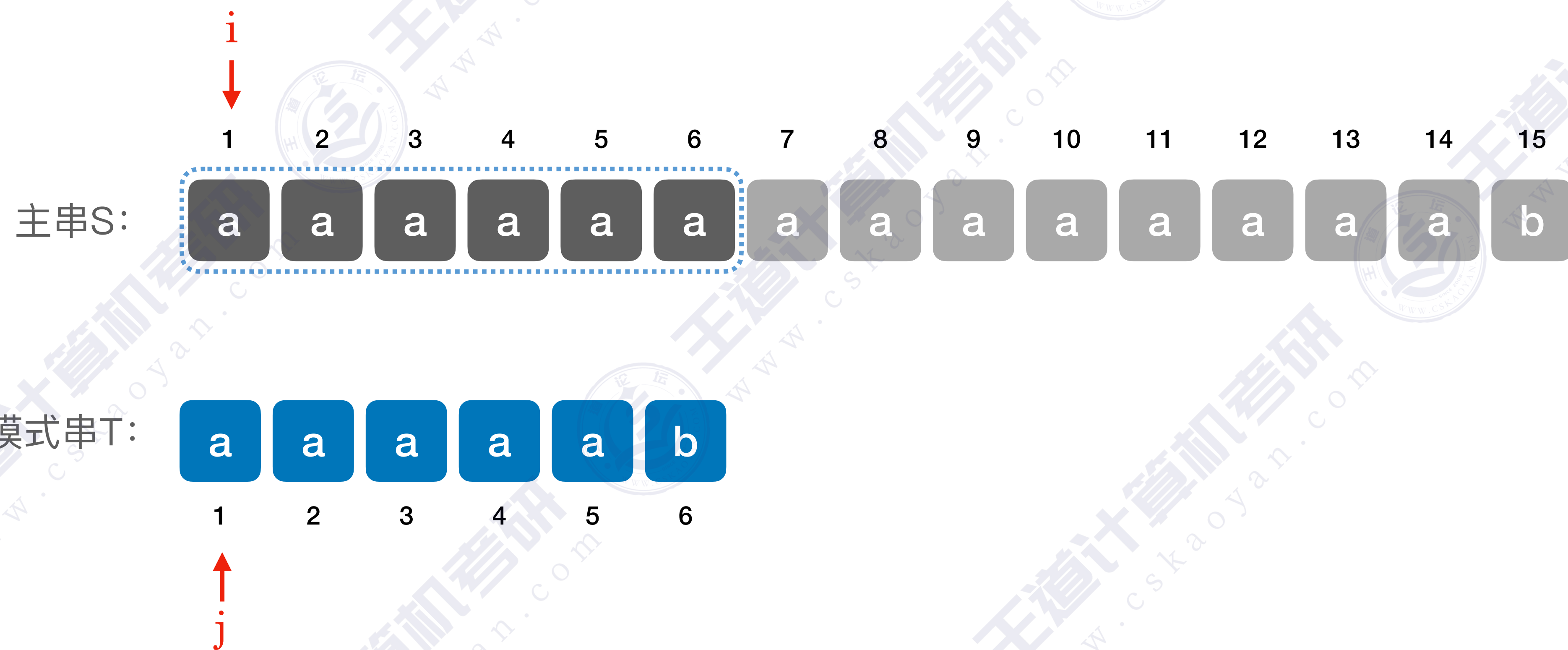
```
int Index(SString S, SString T){
    int i=1, j=1;
    while(i<=S.length && j<=T.length){
        if(S.ch[i]==T.ch[j]){
            ++i; ++j; //继续比较后继字符
        }
        else{
            i=i-j+2;
            j=1; //指针后退重新开始匹配
        }
    }
    if(j>T.length)
        return i-T.length;
    else
        return 0;
}
```

设主串长度为 n ，模式串长度为 m ，则

最坏时间复杂度 = $O(nm)$

朴素模式匹配算法

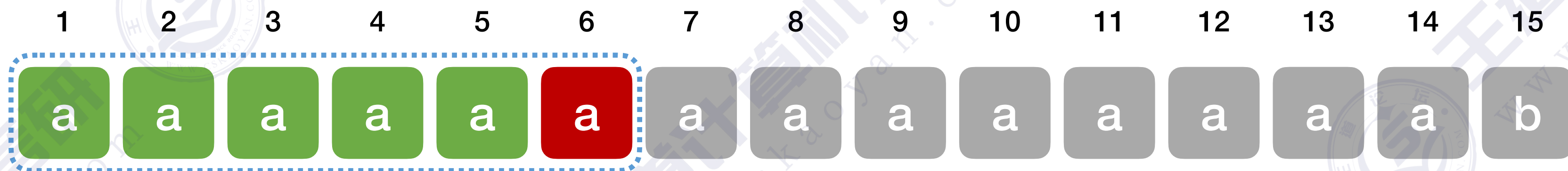
最坏时间复杂度 = $O(nm)$



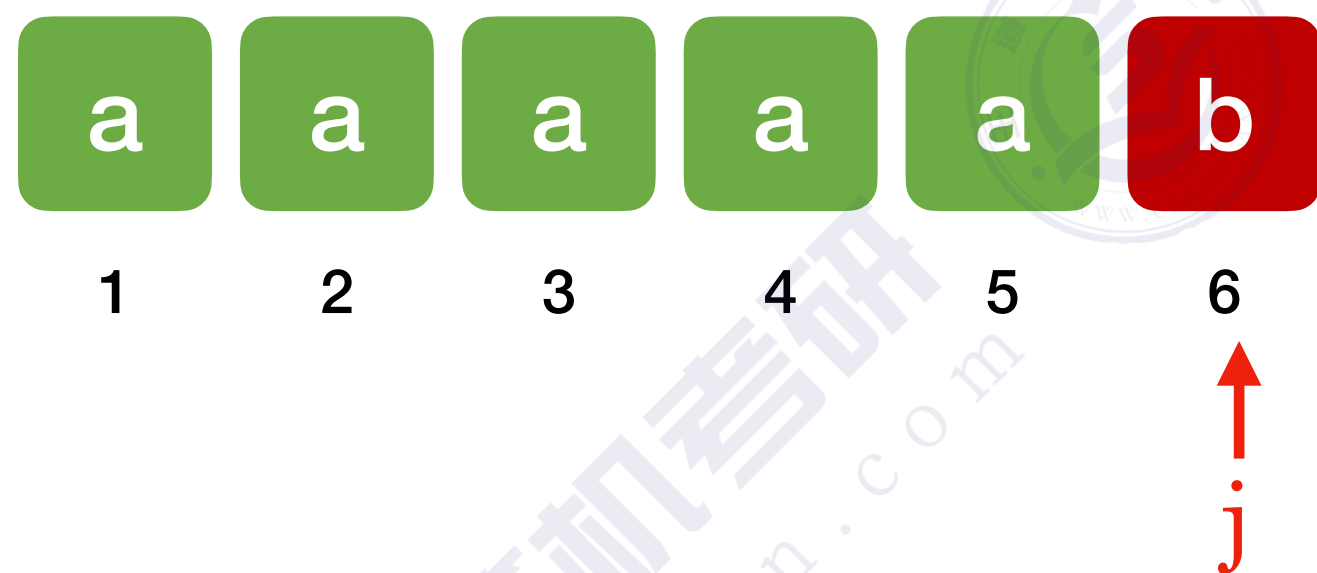
朴素模式匹配算法

最坏时间复杂度 = $O(nm)$

主串S:



模式串T:

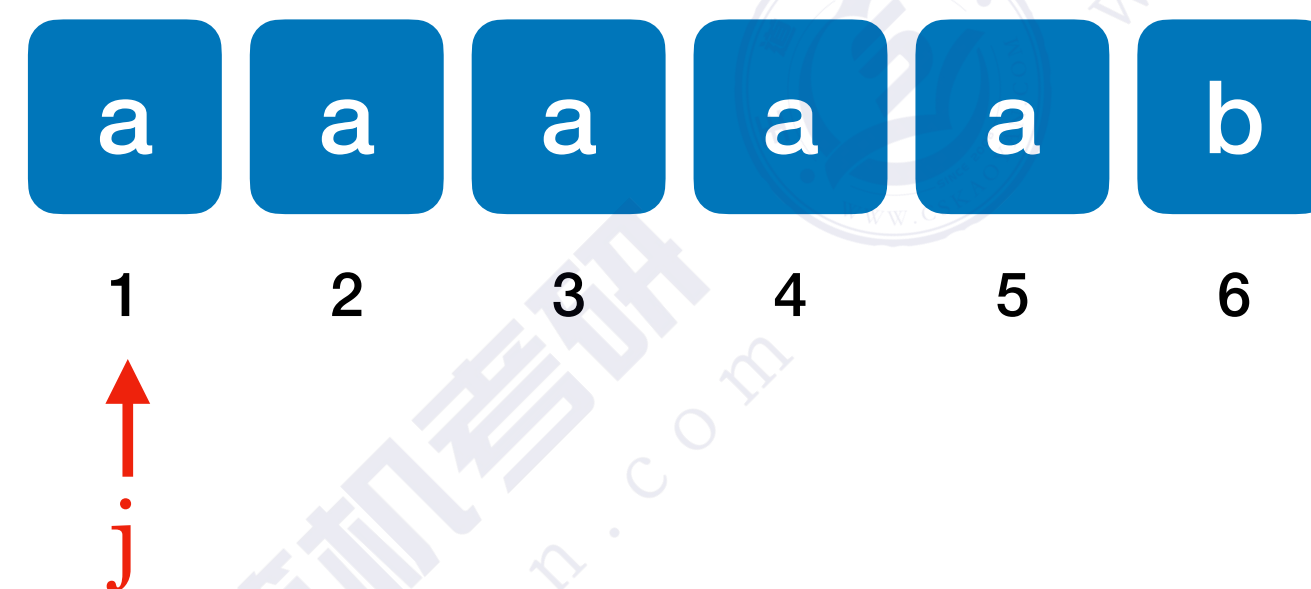


朴素模式匹配算法

最坏时间复杂度 = $O(nm)$

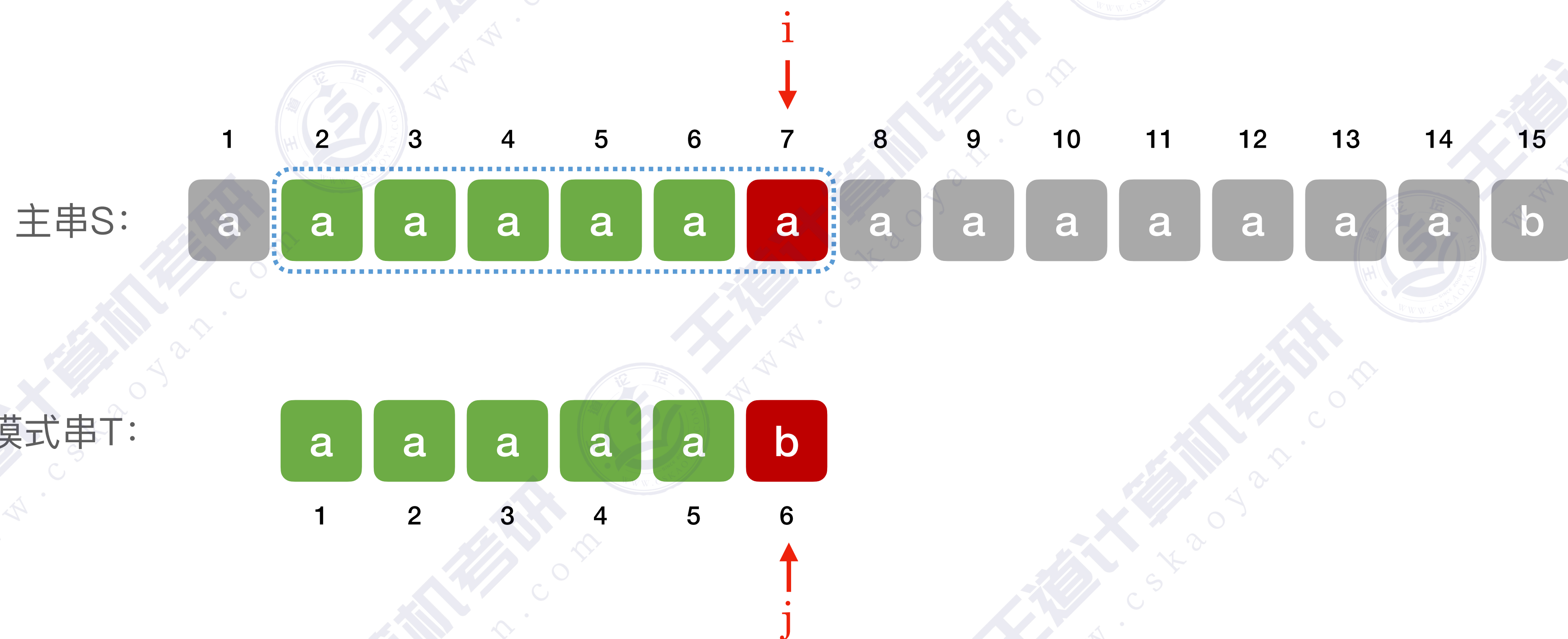


模式串T:



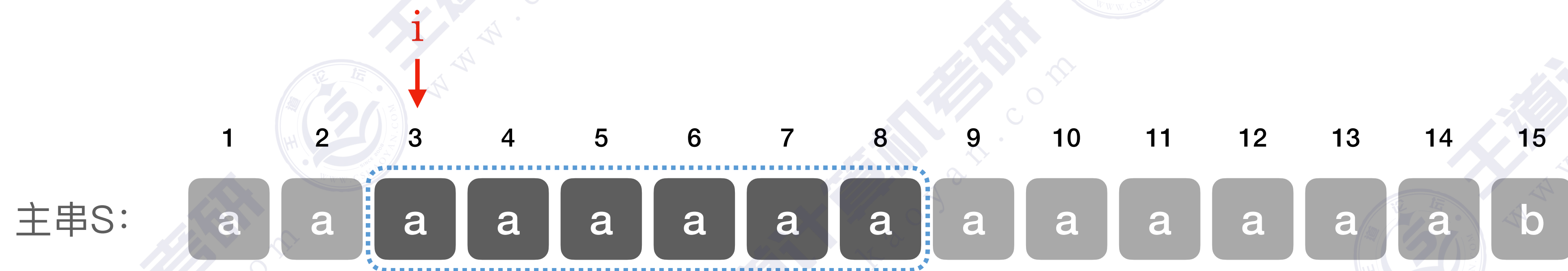
朴素模式匹配算法

最坏时间复杂度 = $O(nm)$

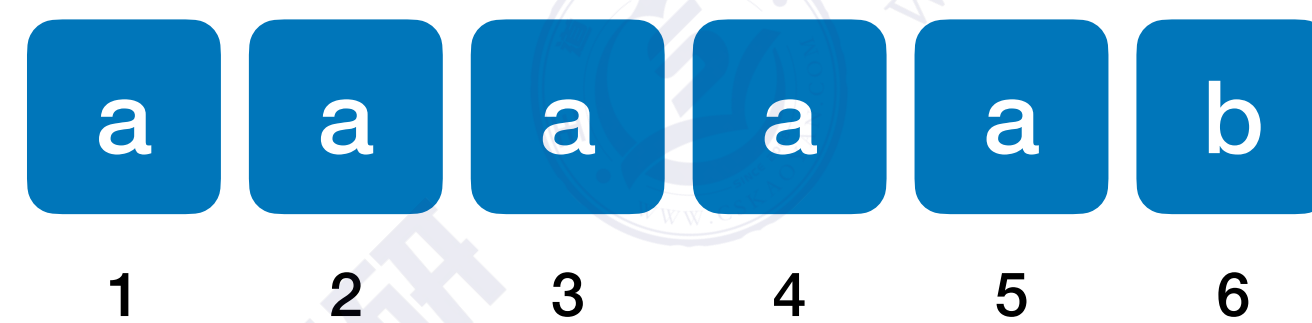


朴素模式匹配算法

最坏时间复杂度 = $O(nm)$

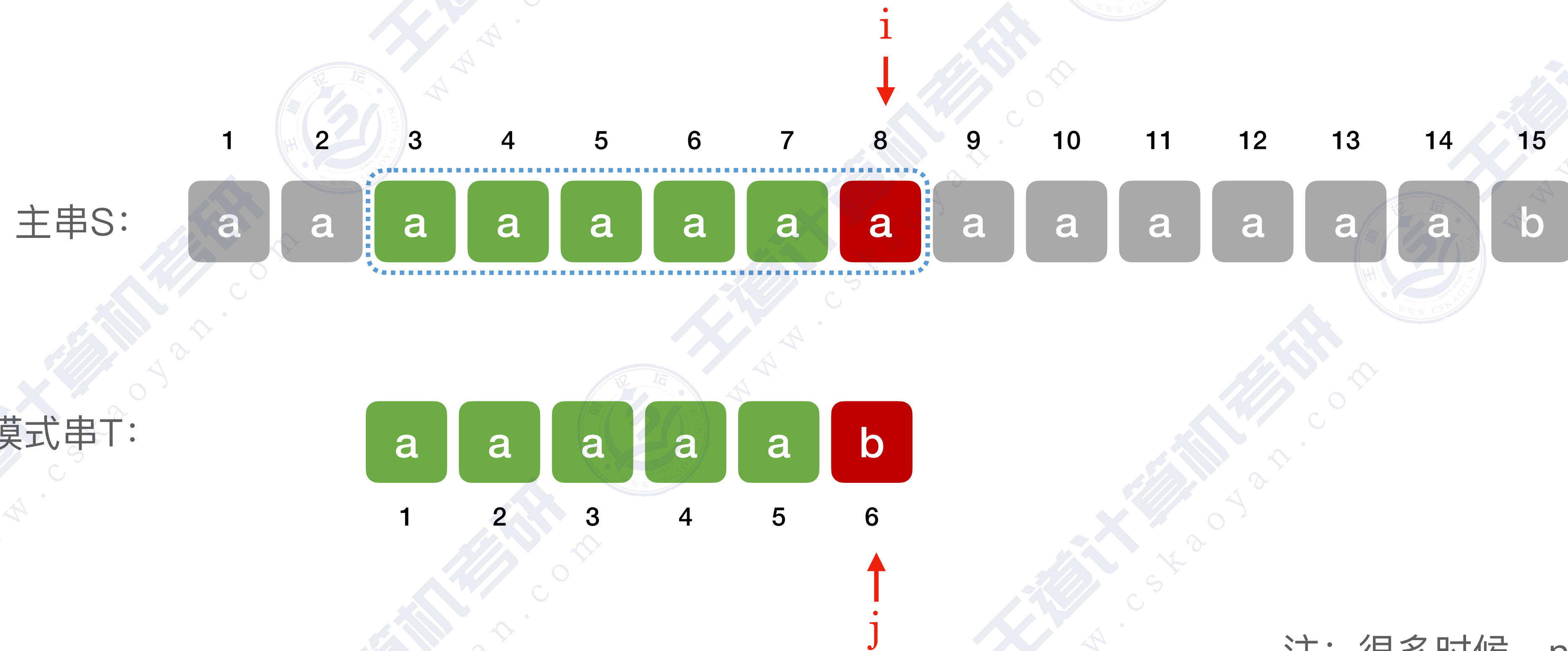


模式串T:



朴素模式匹配算法

最坏时间复杂度 = $O(nm)$



注：很多时候， $n \gg m$

最坏的情况，每个子串都要对比 m 个字符，共 $n-m+1$ 个子串，复杂度 = $O((n-m+1)m) = O(nm)$

知识回顾与重要考点

朴素模式匹配算法

算法思想

主串长度 n ，模式串长度 m

将主串中所有长度为 m 的子串与模式串对比

找到第一个与模式串匹配的子串，并返回子串起始位置

若所有子串都不匹配，则返回0

最坏时间复杂度= $O(nm)$