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. 计算机软件和硬件的关系
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

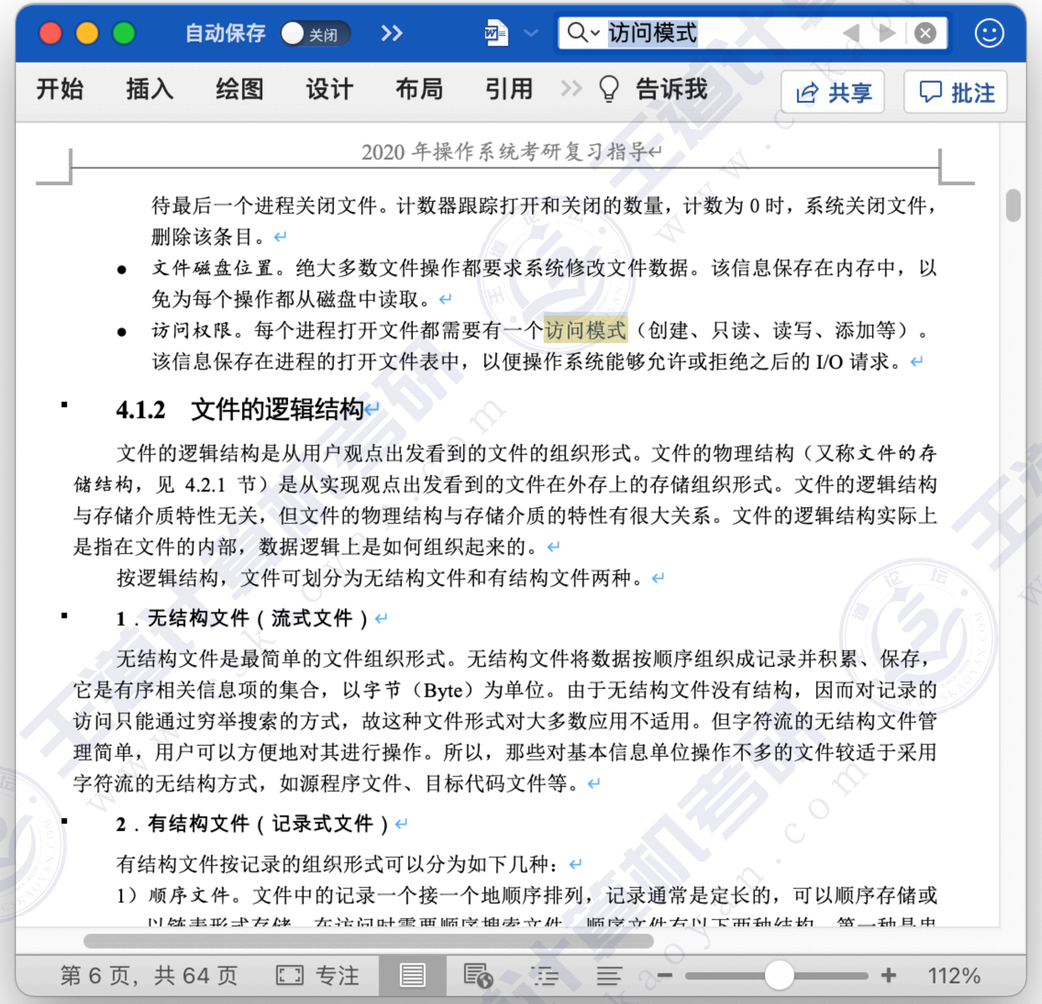
（一）数制与编码
1. 进位计数制及其相互转换

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



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嗨森哈哈哈哈哈哈嗝’

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嗨森哈哈哈哈哈哈嗝’

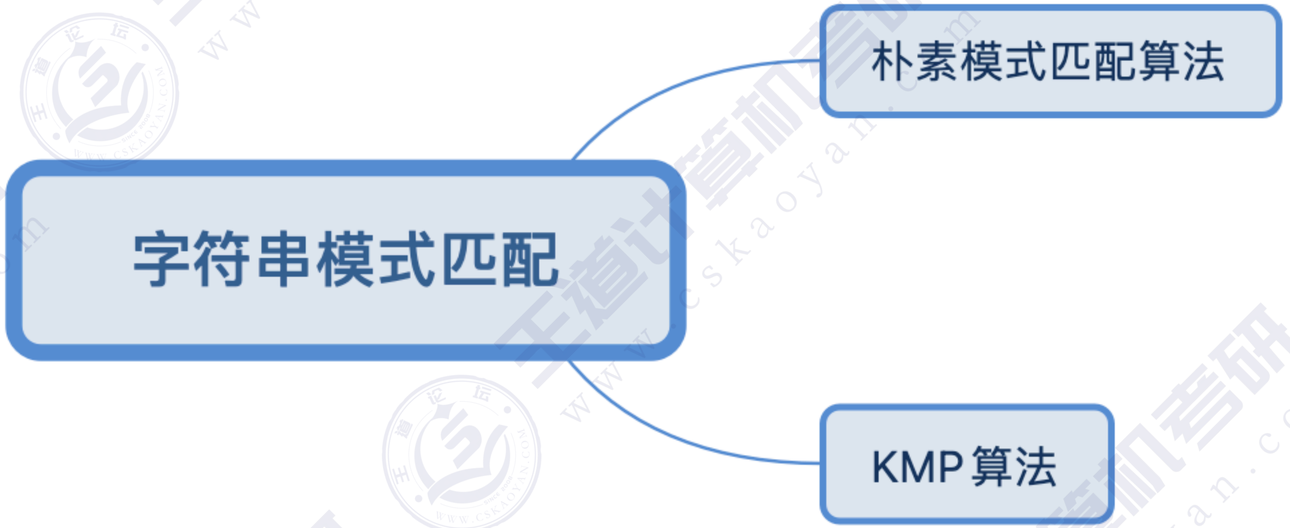
模式串 ‘笑出猪叫’

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



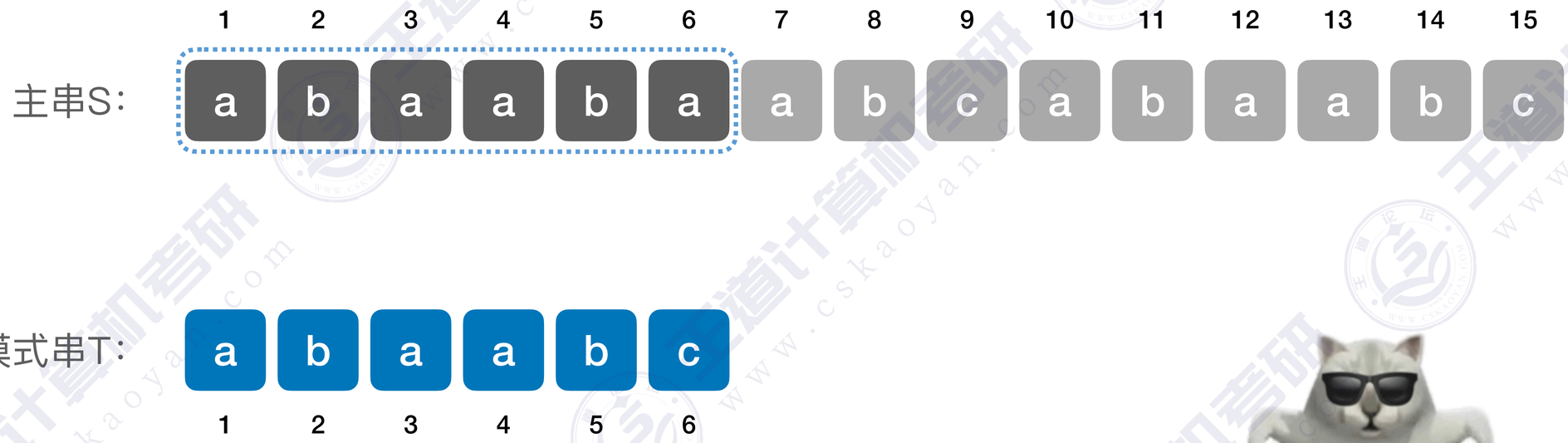
朴素模式匹配算法

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

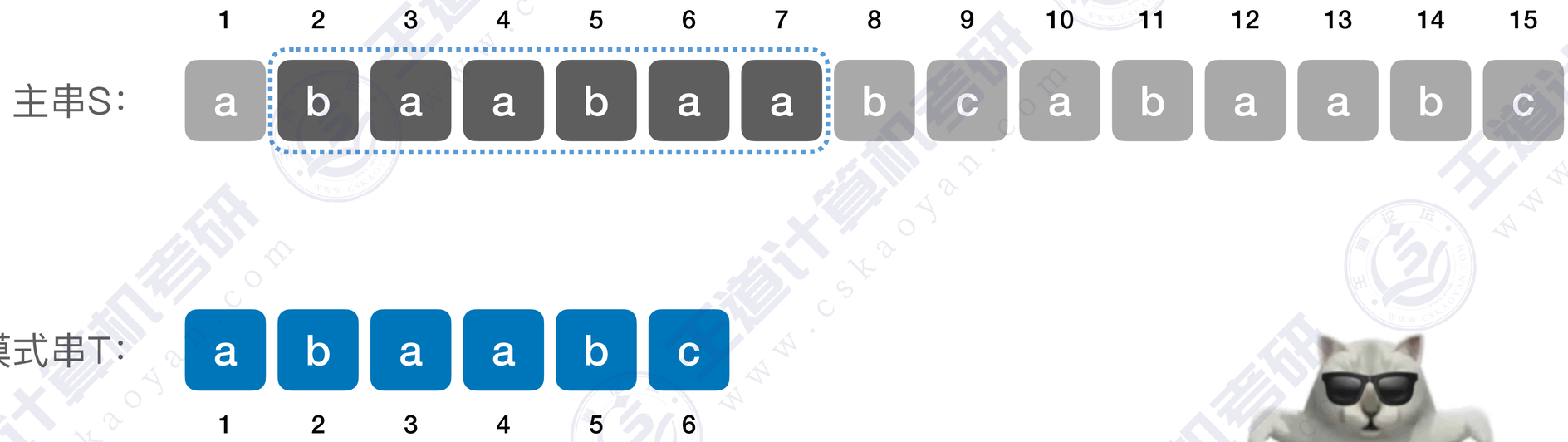
	1	2	3	4	5	6
模式串T:	a	b	a	a	b	c

开始暴力解决问题了

朴素模式匹配算法



朴素模式匹配算法



朴素模式匹配算法

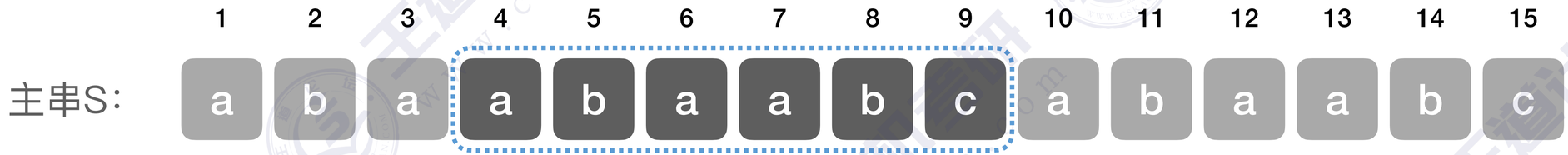
主串S:

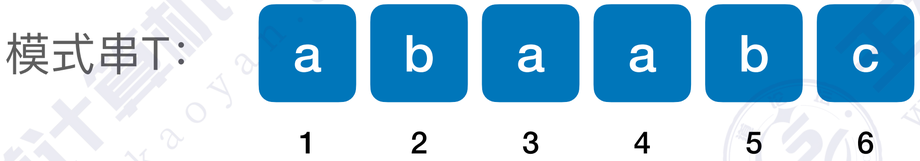
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
a	b	a	a	b	a	a	b	c	a	b	a	a	b	c

模式串T:

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

朴素模式匹配算法





朴素模式匹配算法

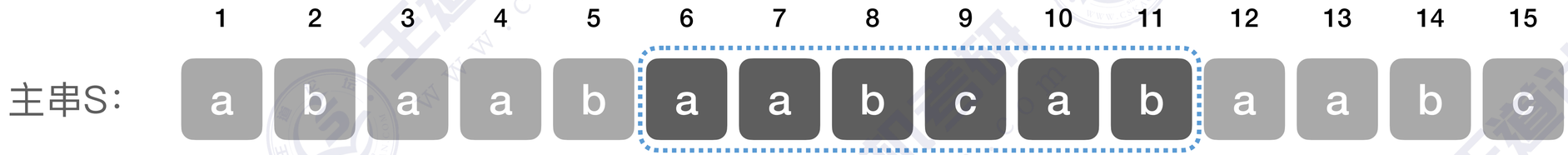
主串S:

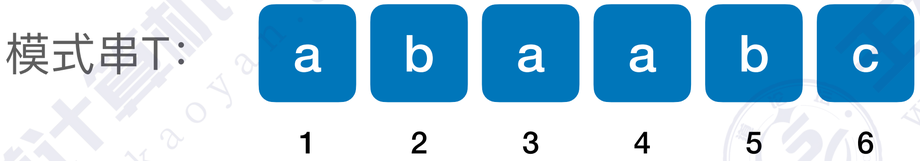
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
a	b	a	a	b	a	a	b	c	a	b	a	a	b	c

模式串T:

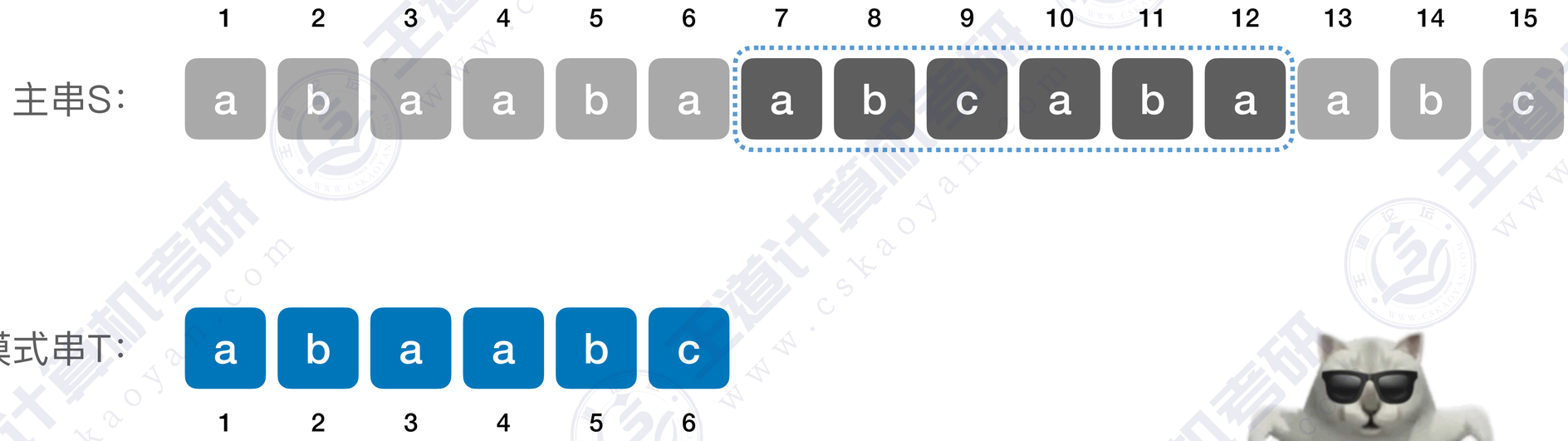
1	2	3	4	5	6
a	b	a	a	b	c

朴素模式匹配算法

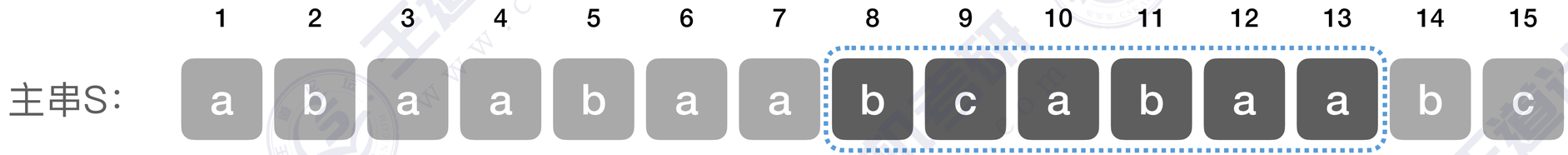


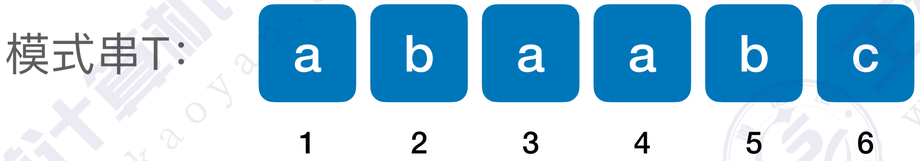


朴素模式匹配算法

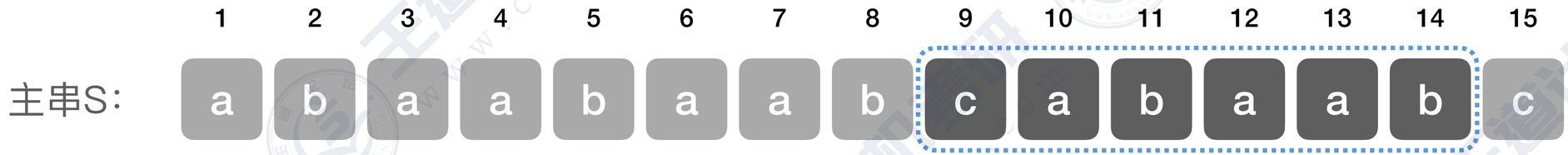


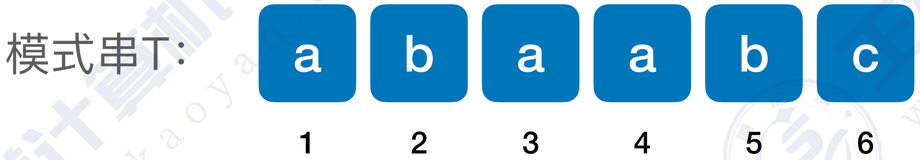
朴素模式匹配算法



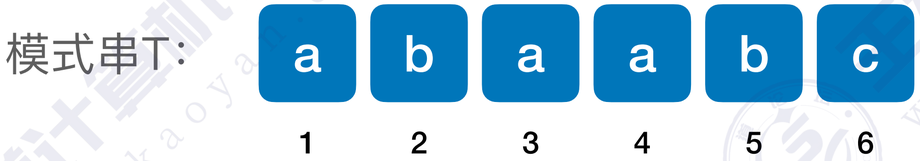


朴素模式匹配算法



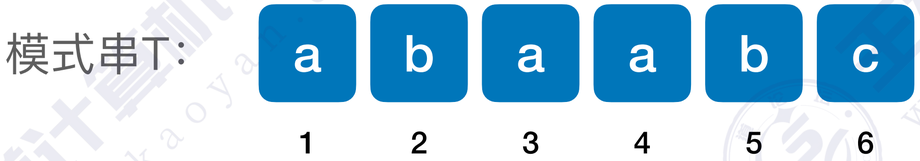


朴素模式匹配算法



开始暴力解决问题了

朴素模式匹配算法



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，则返回它在主串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

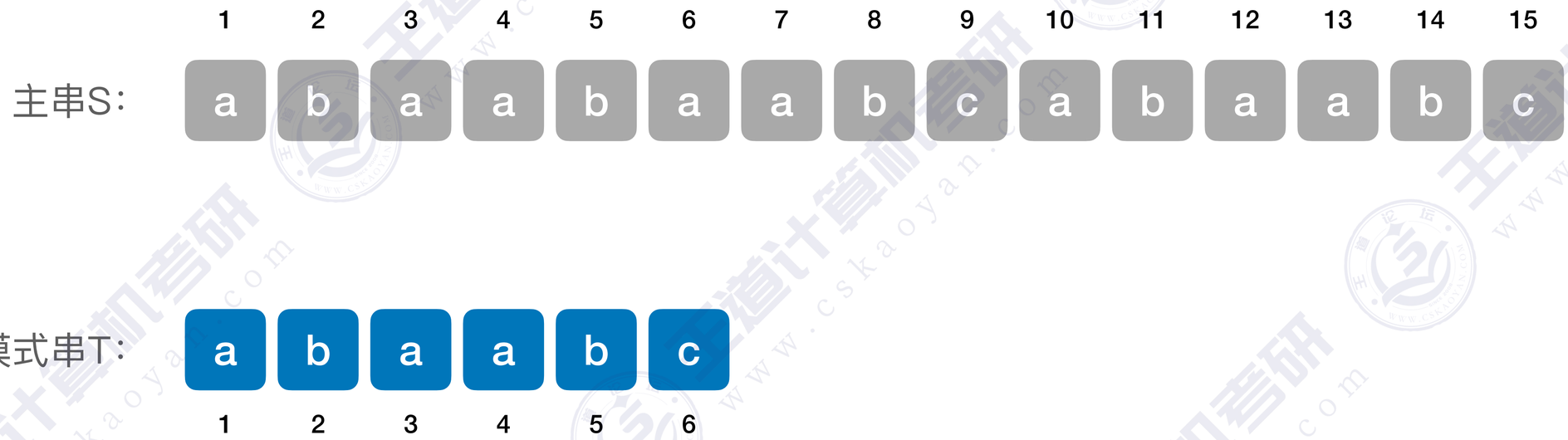
取出从位置 i 开始，长度为m的子串

子串和模式串对比，若不匹配，则匹配下一个子串

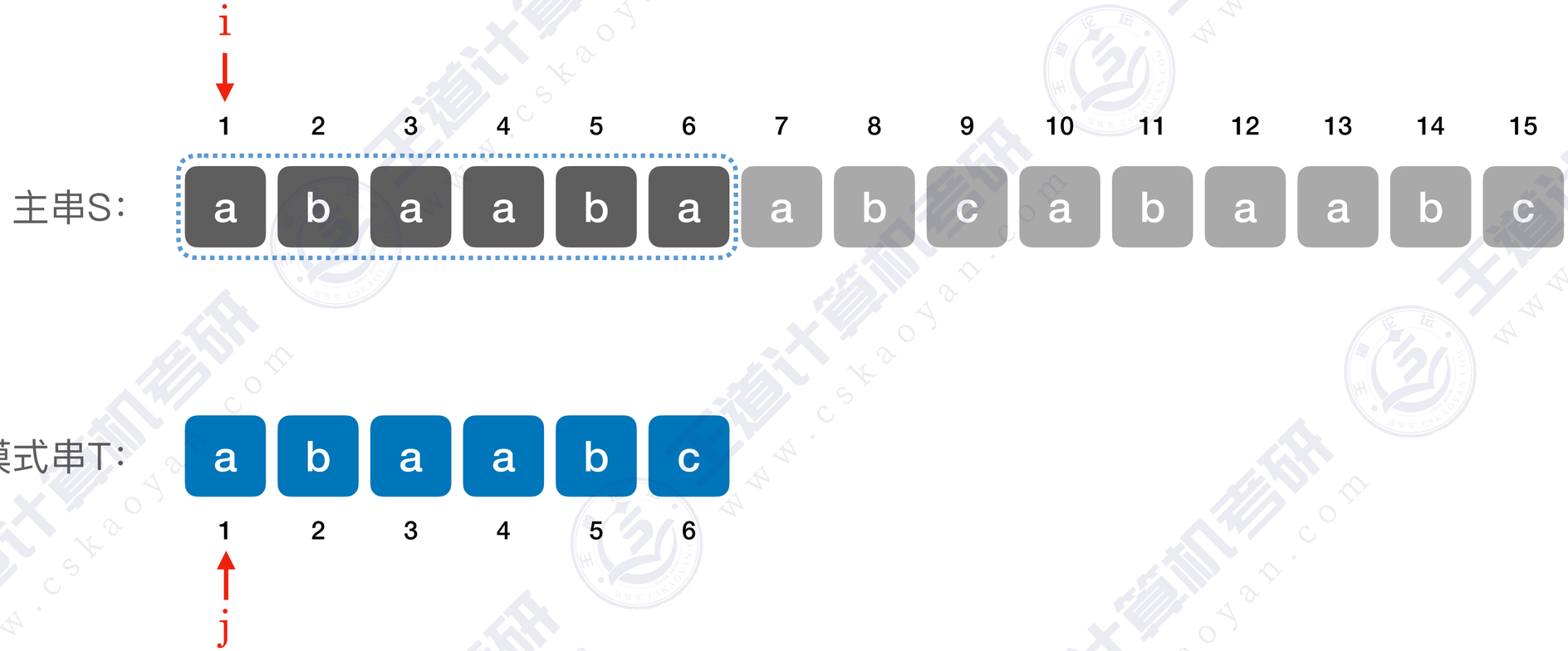
原来是这样啊

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

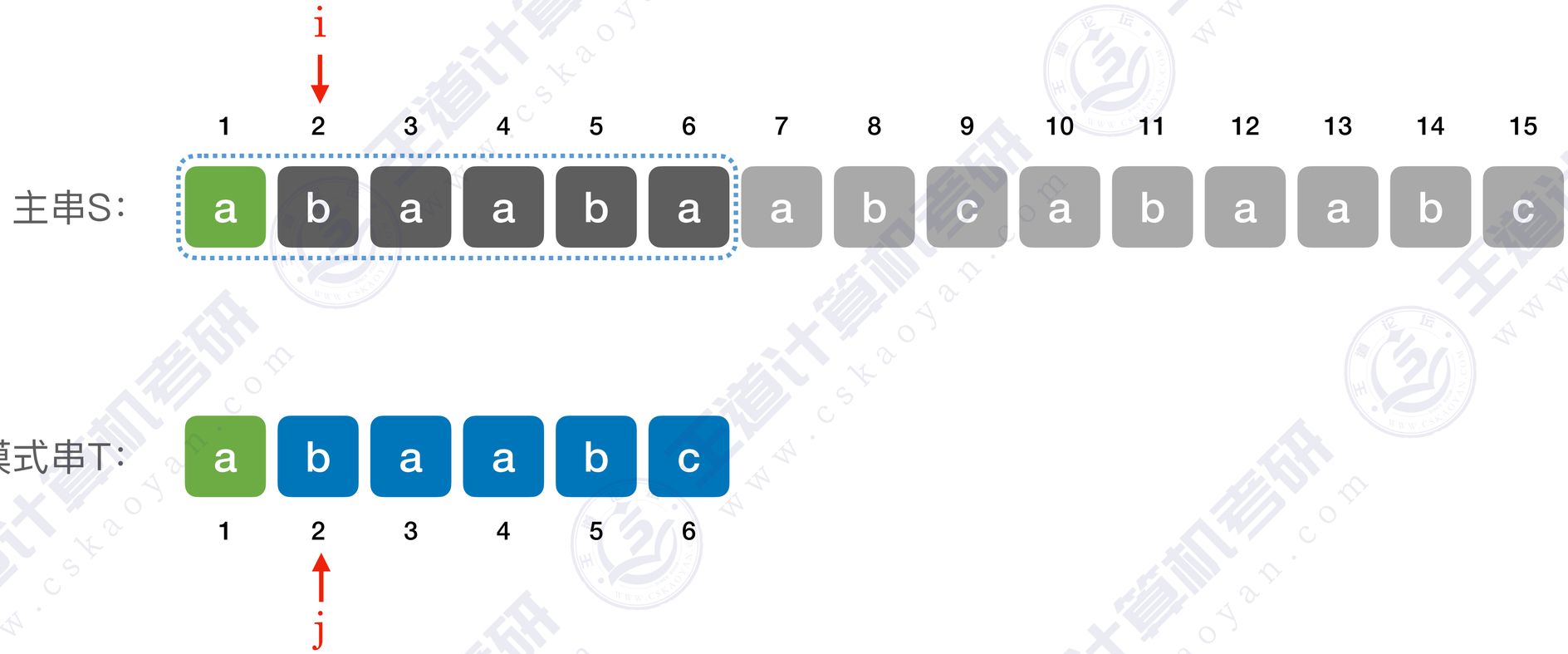
朴素模式匹配算法



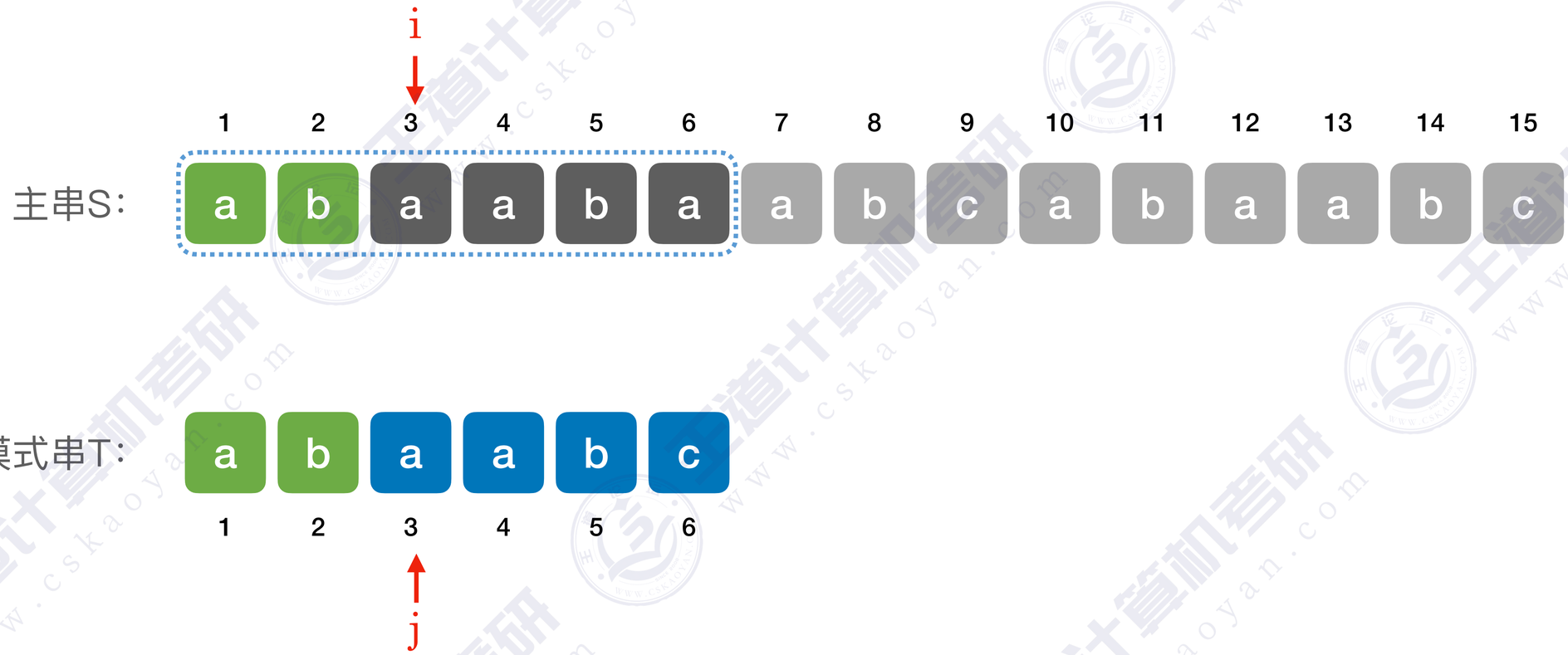
朴素模式匹配算法



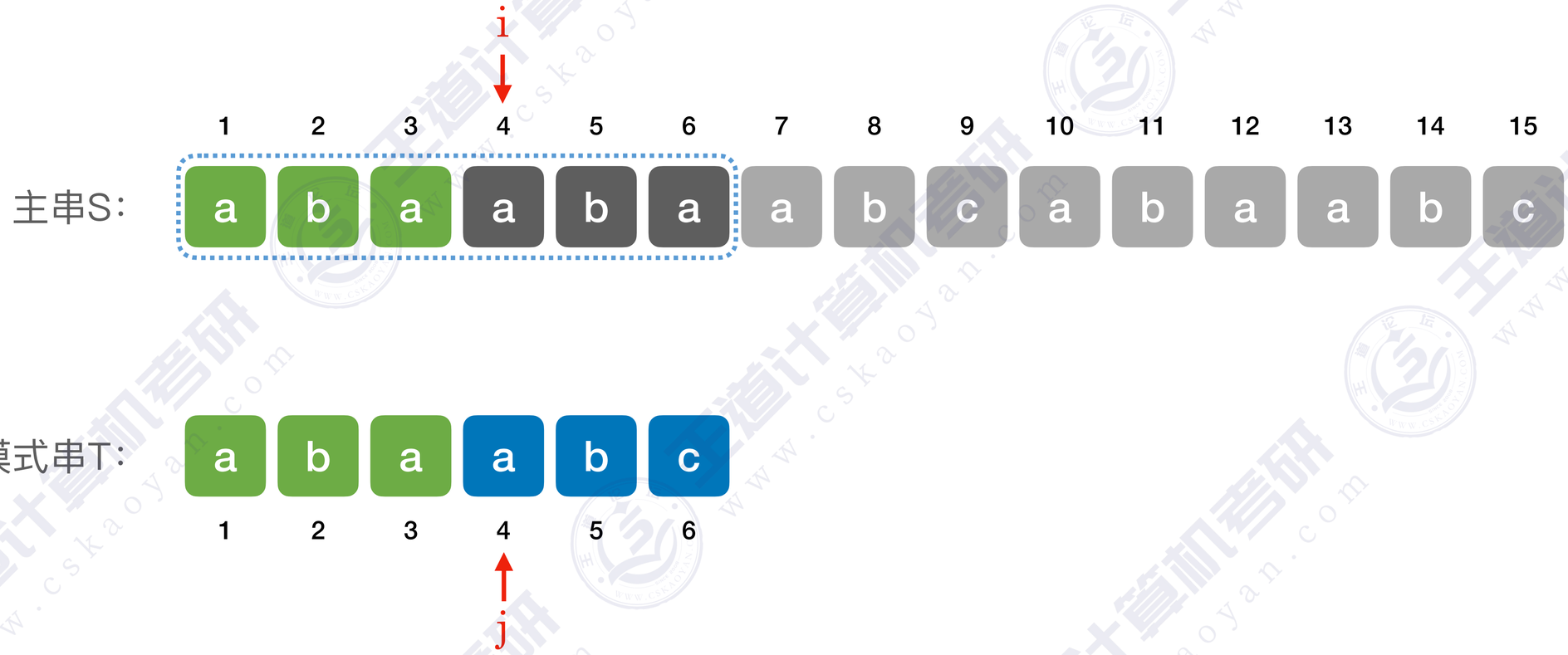
朴素模式匹配算法



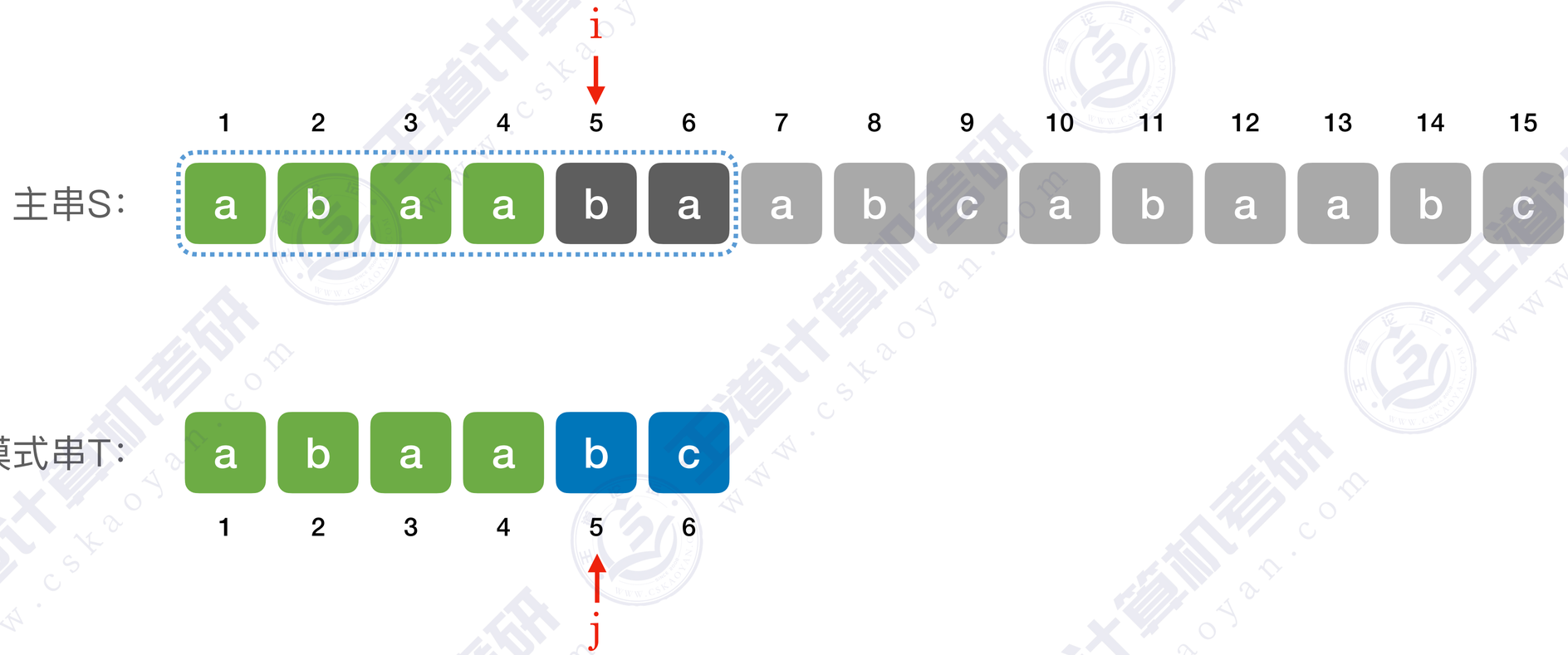
朴素模式匹配算法



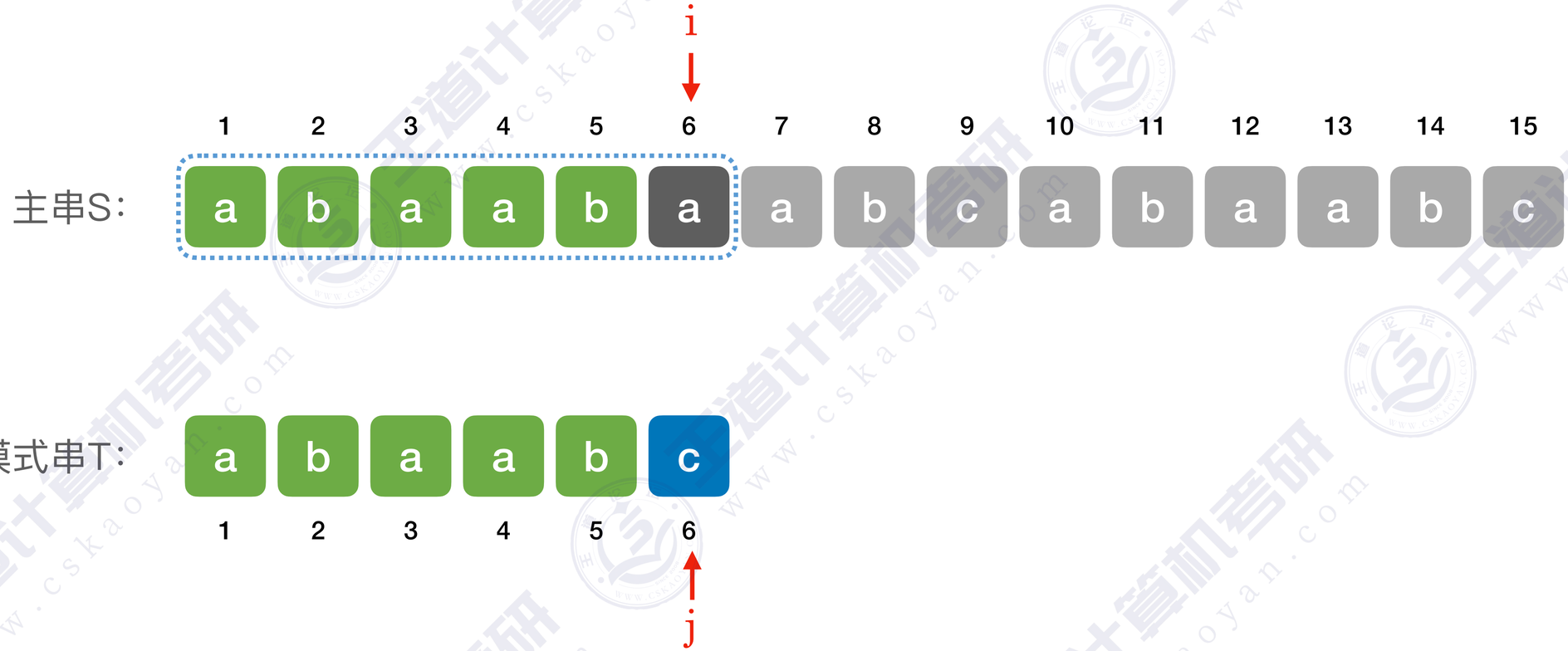
朴素模式匹配算法



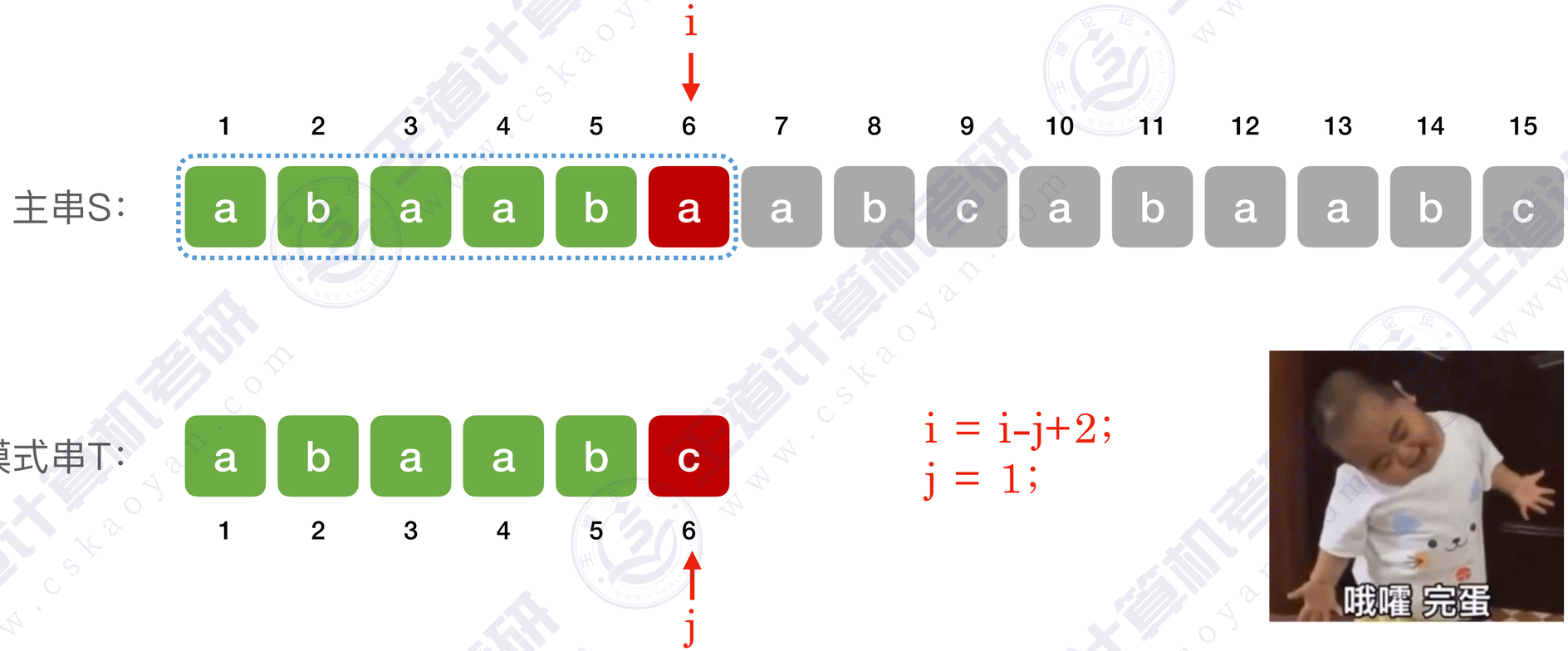
朴素模式匹配算法



朴素模式匹配算法



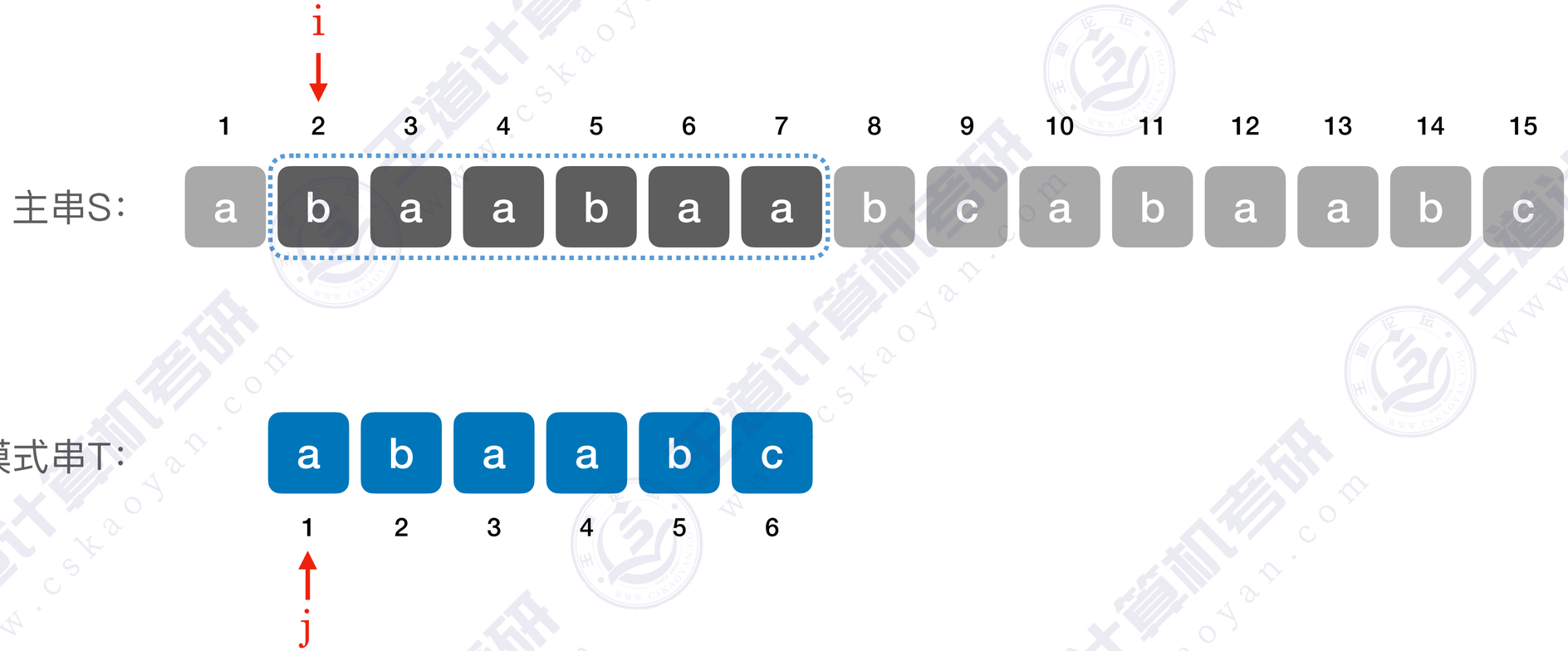
朴素模式匹配算法



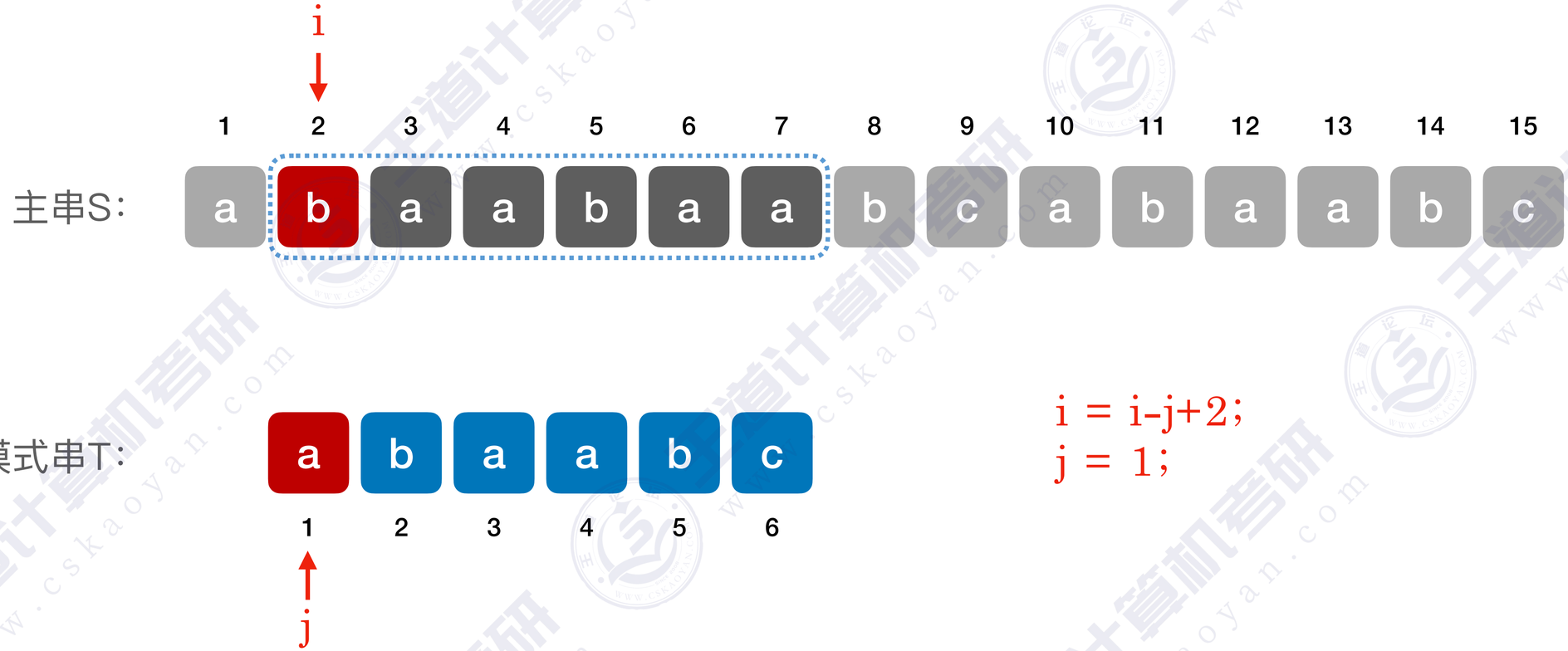

```
i = i-j+2;
j = 1;
```


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



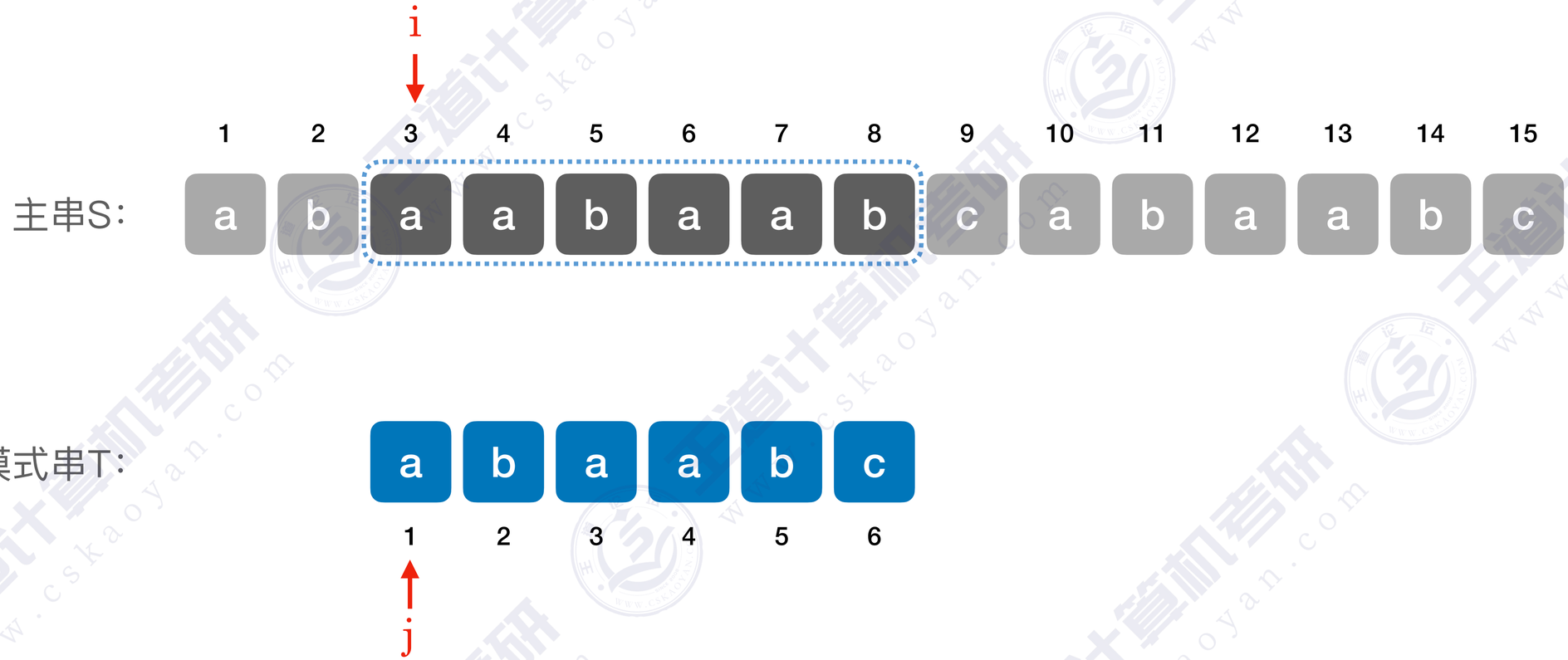
朴素模式匹配算法



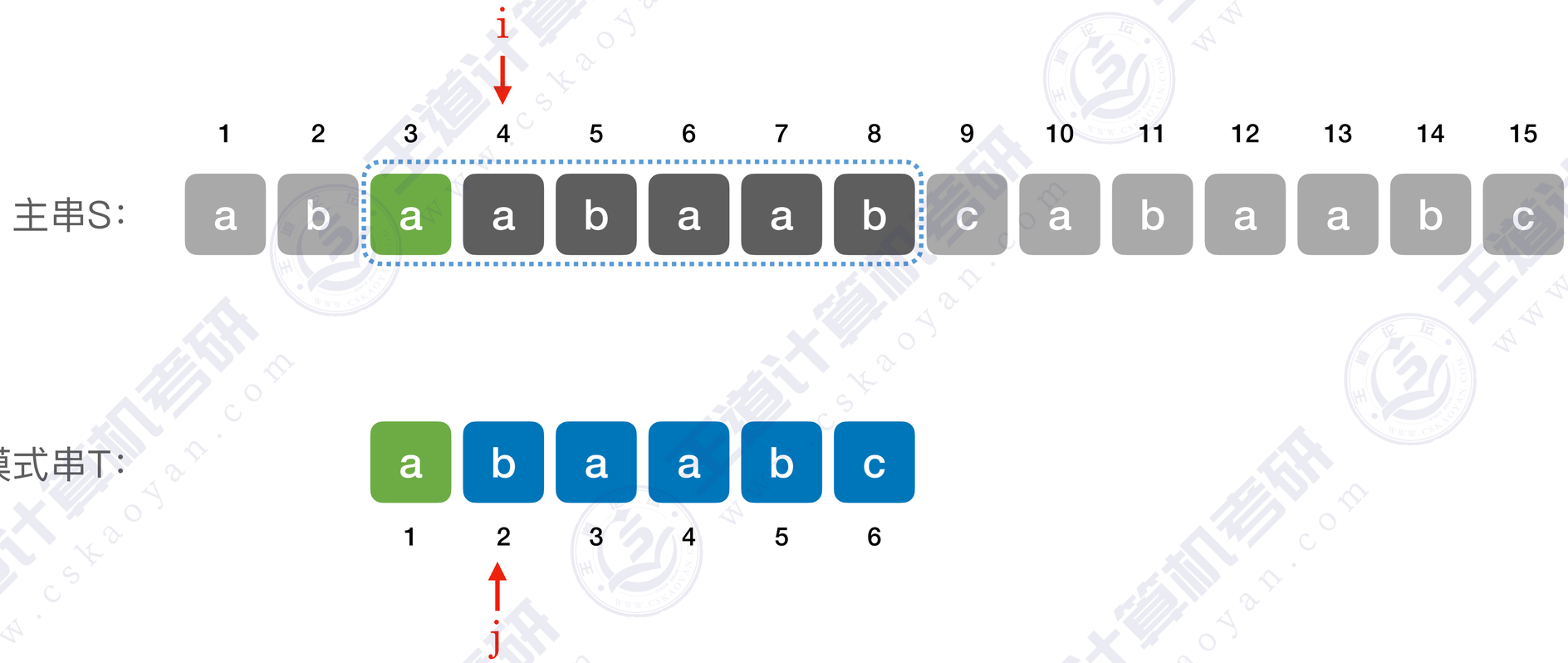

```
i = i-j+2;
j = 1;
```

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

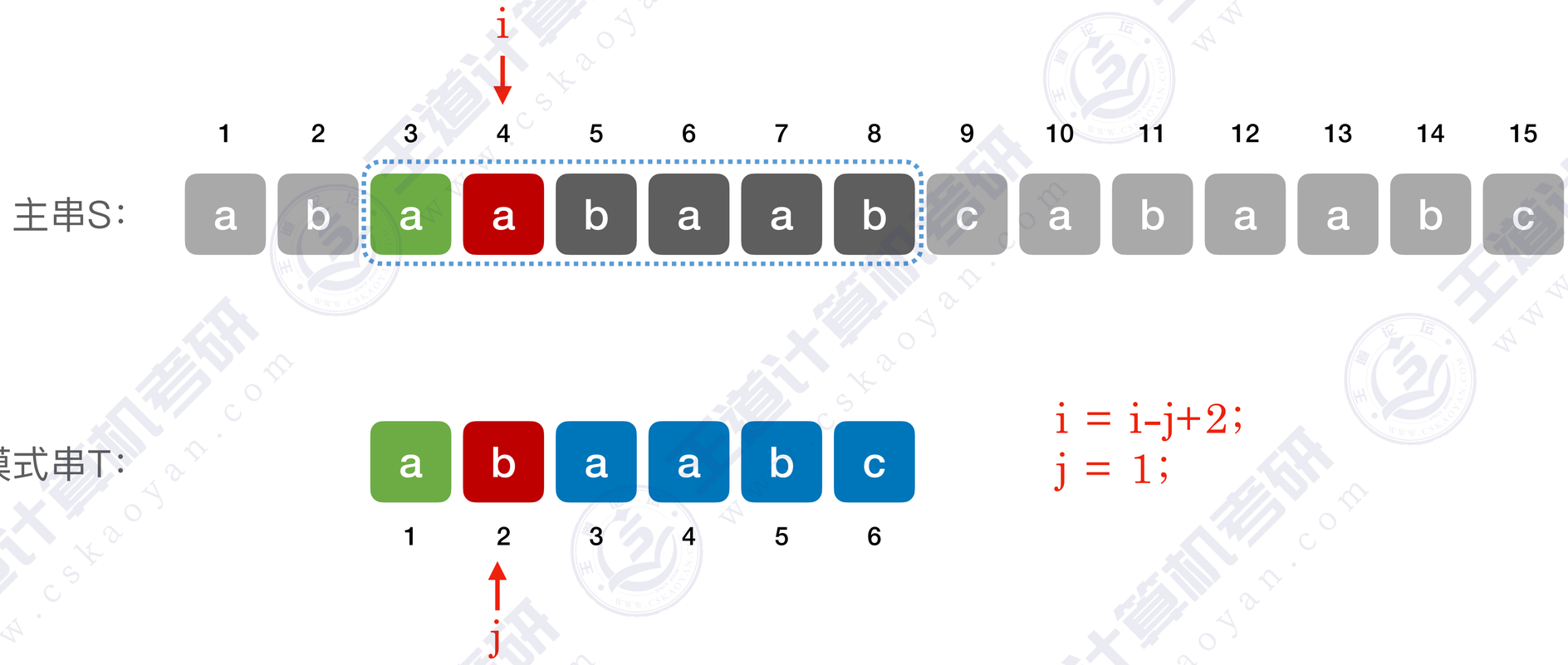
朴素模式匹配算法



朴素模式匹配算法

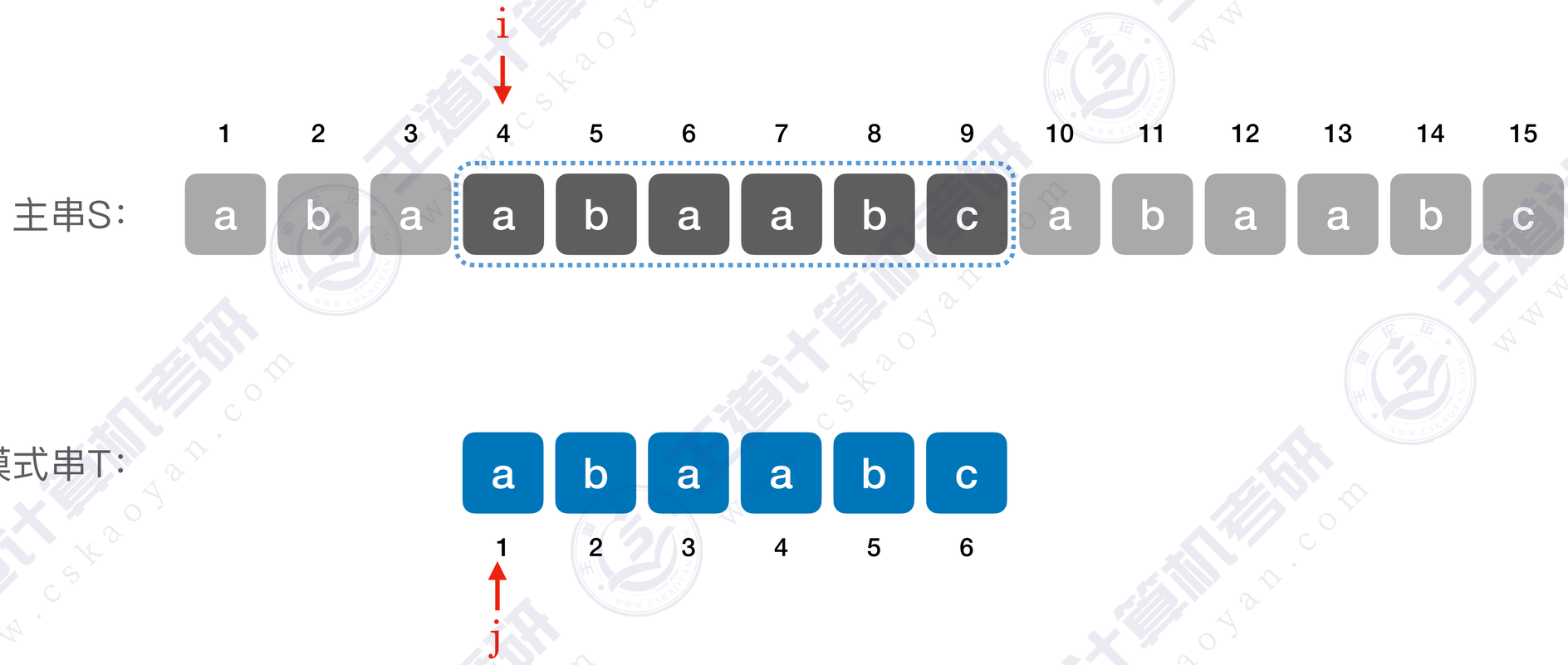


朴素模式匹配算法

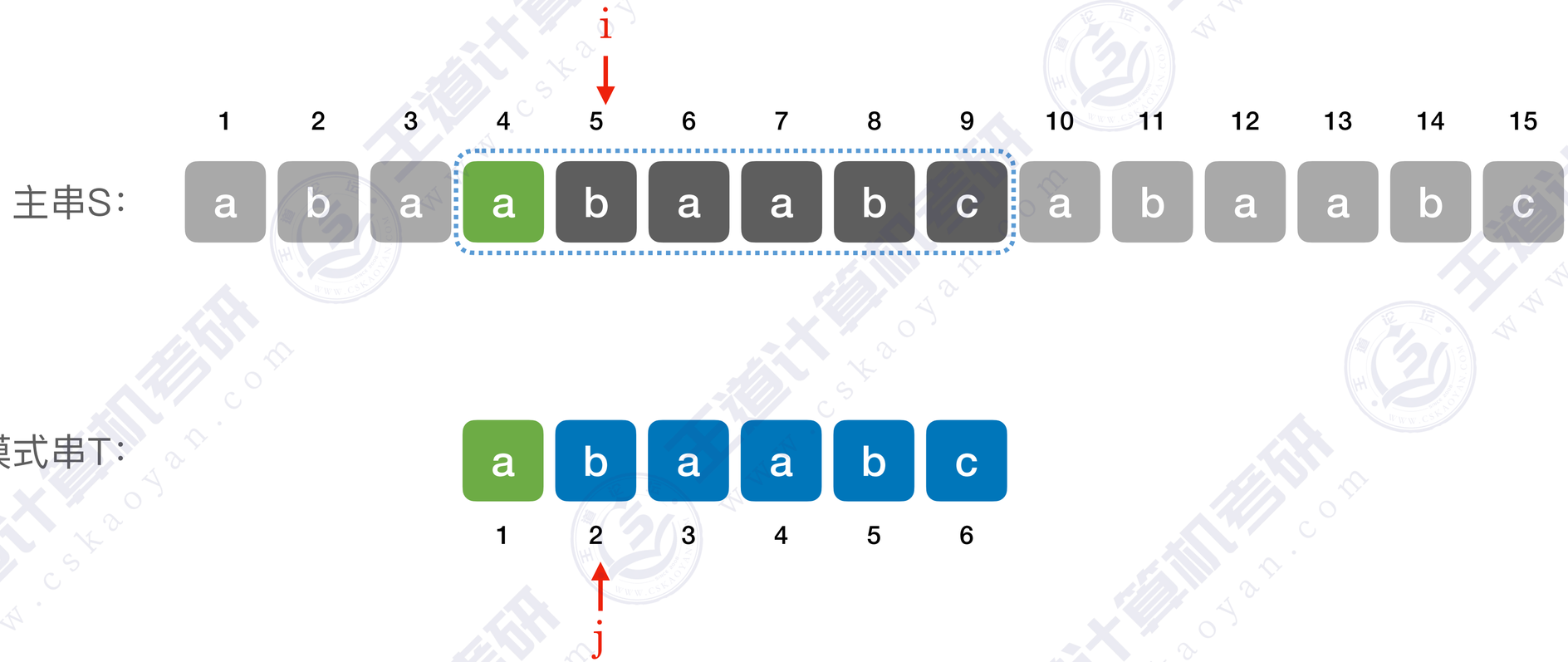


若当前子串匹配失败，则主串指针 i 指向下一个子串的第一个位置，模式串指针 j 回到模式串的第一个位置

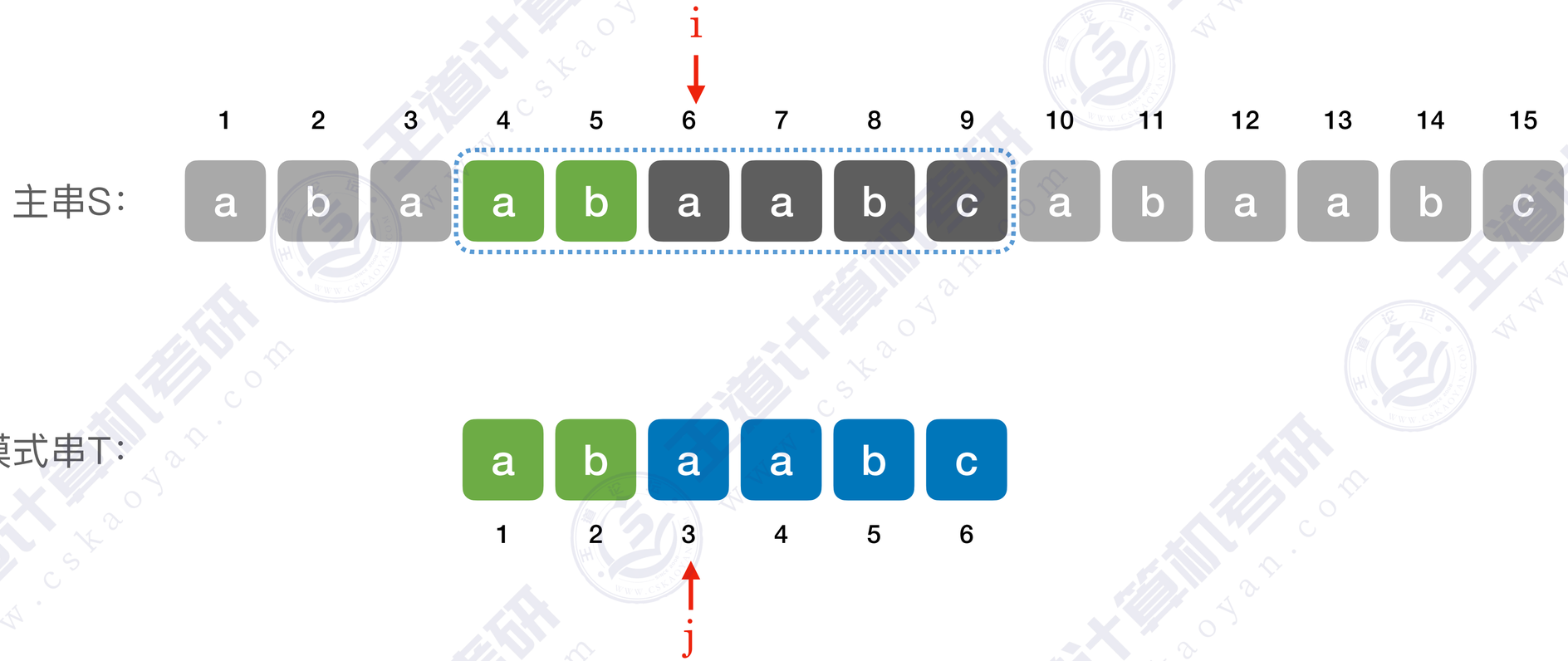
朴素模式匹配算法



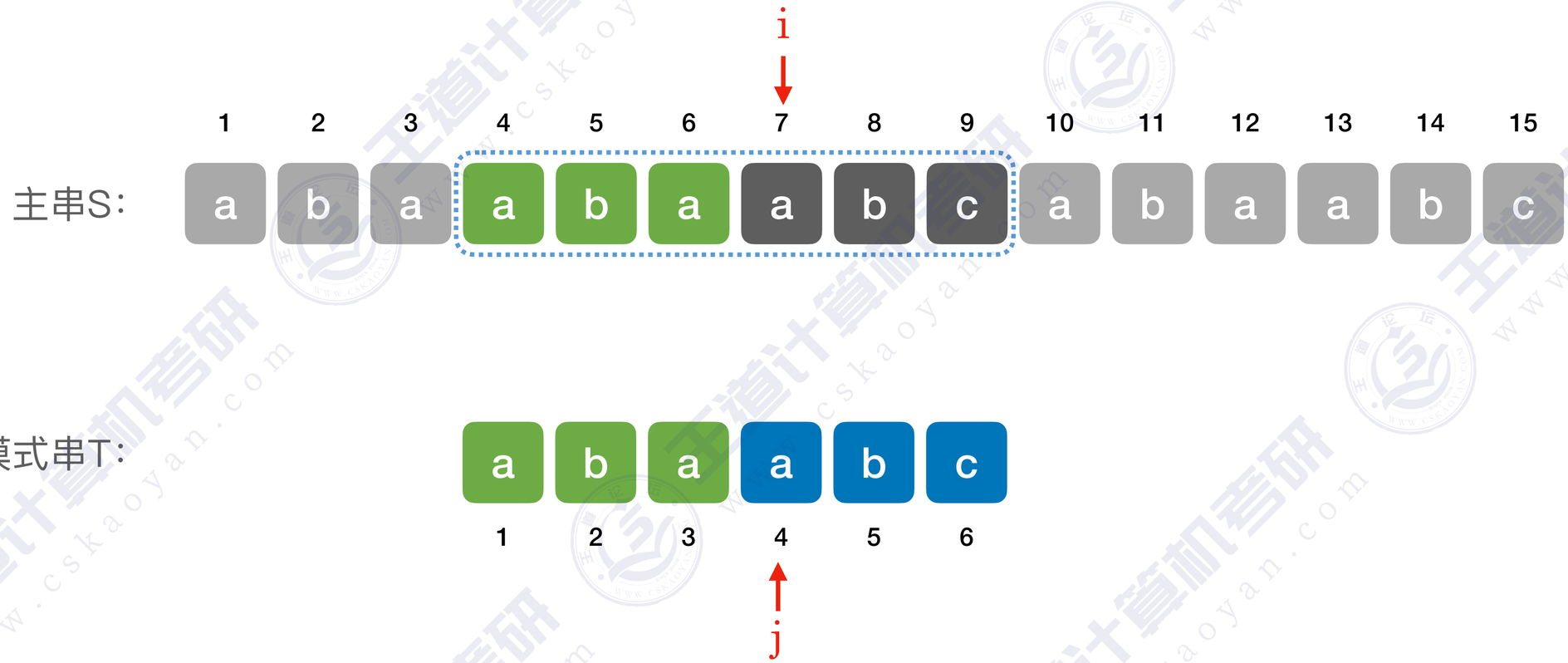
朴素模式匹配算法



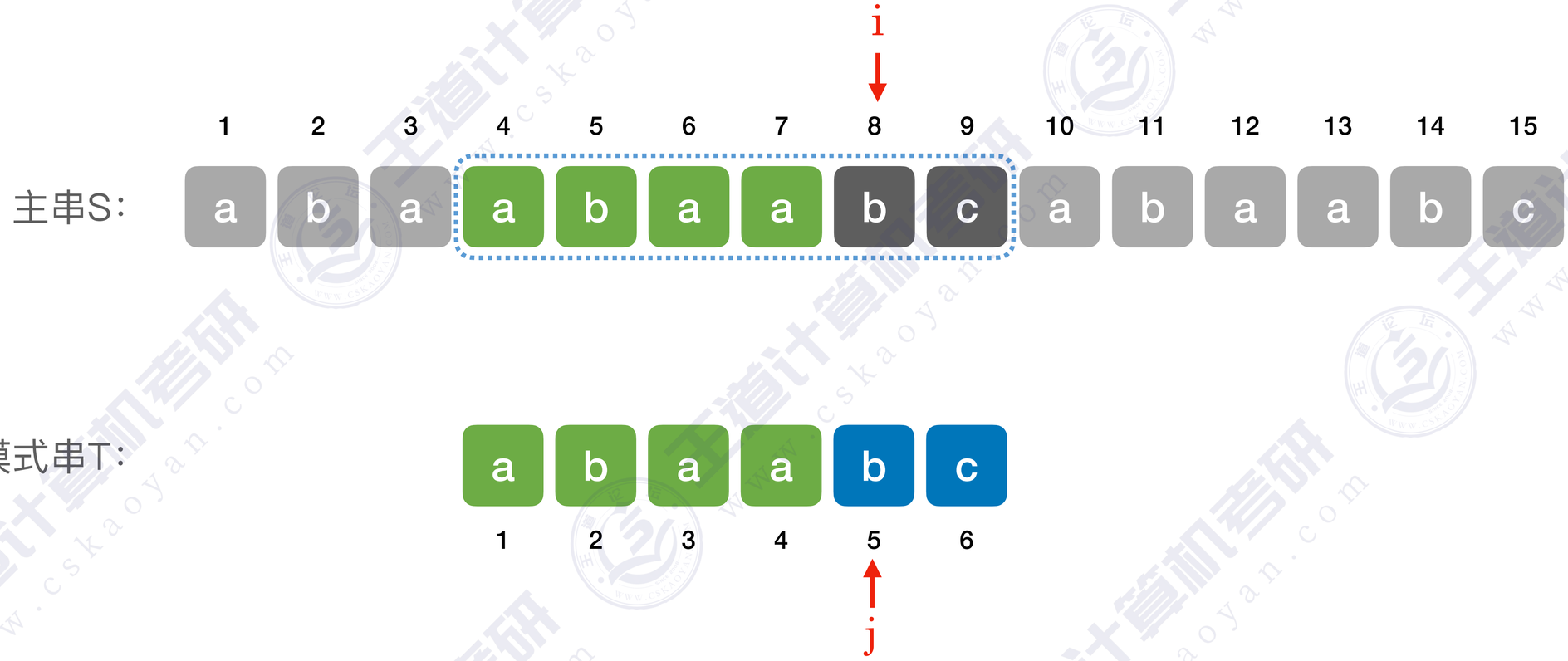
朴素模式匹配算法



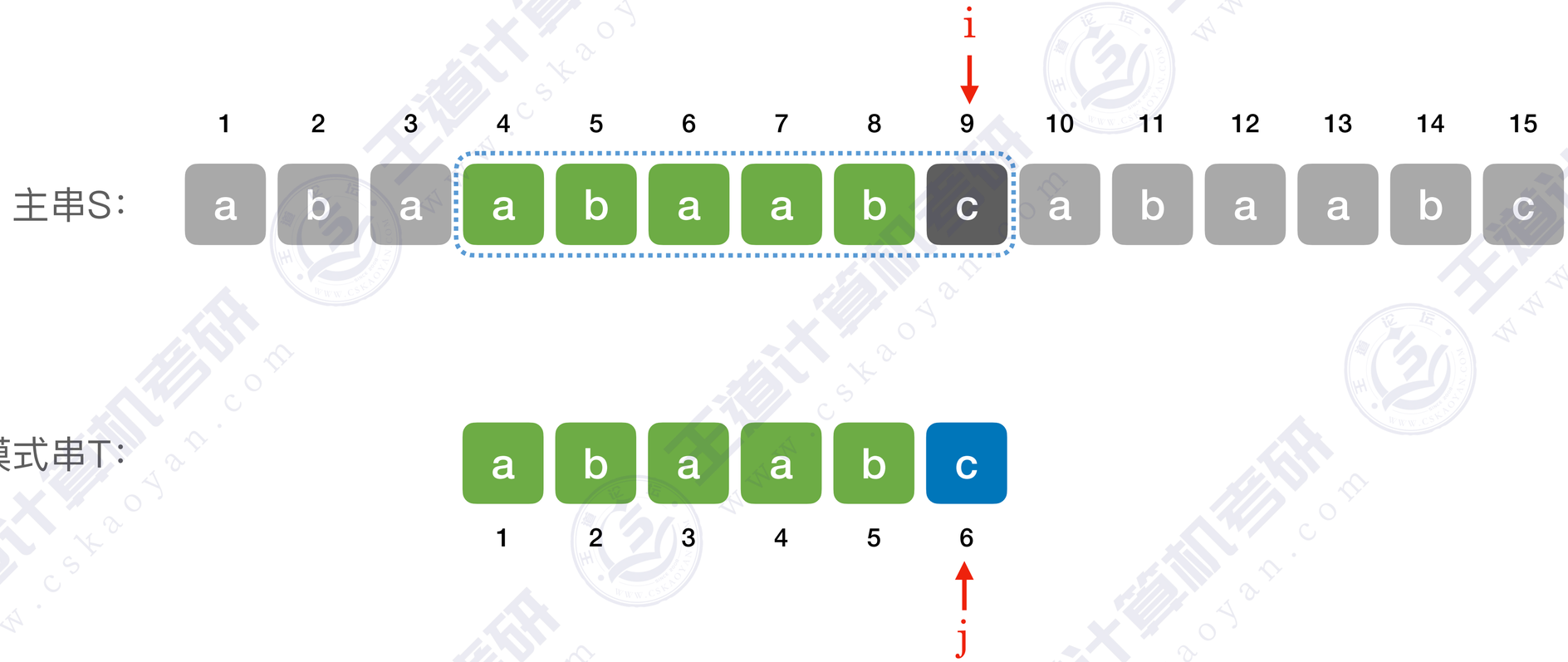
朴素模式匹配算法



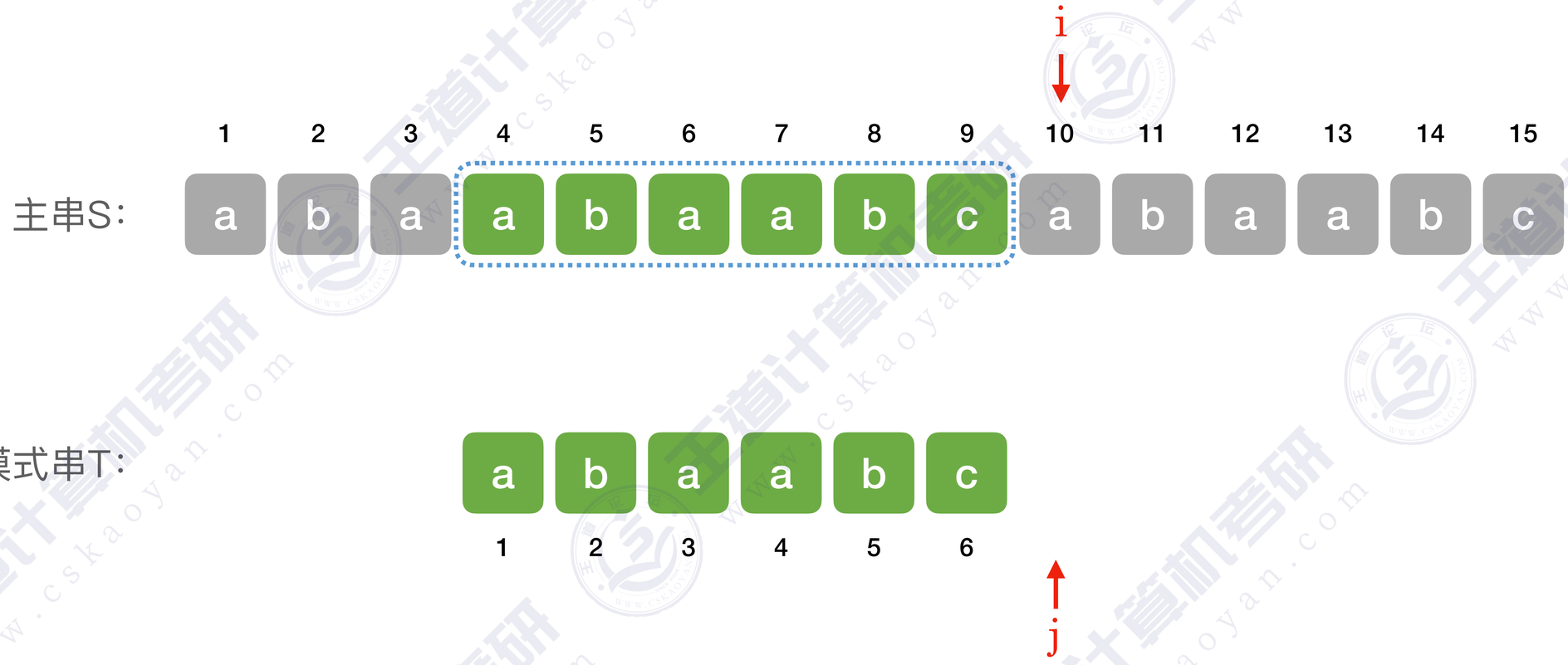
朴素模式匹配算法



朴素模式匹配算法



朴素模式匹配算法



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

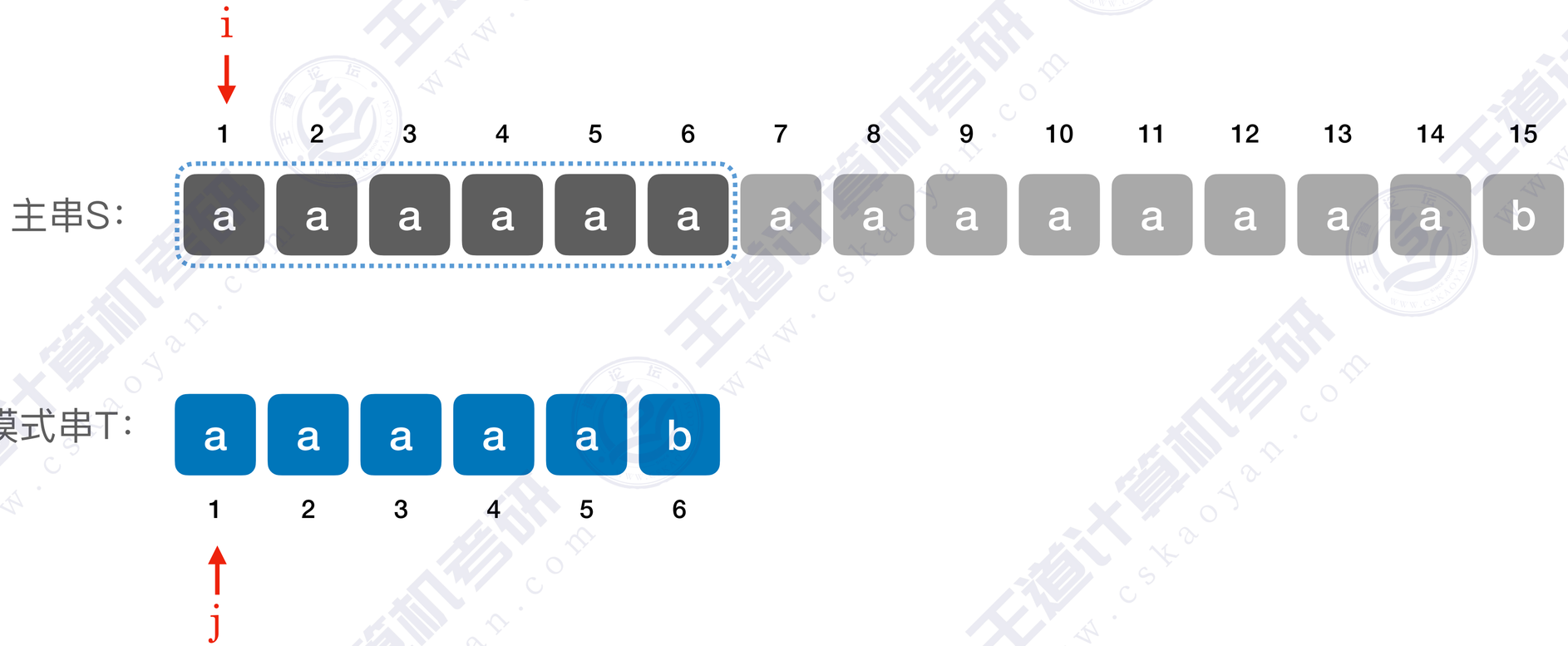
```
int Index(SString S,SString T){
    int i=1,j=1;
    while(i<=S.length && j<=T.length){
        if(S.ch[i]==T.ch[j]){
            ++i; ++j;    //继续比较后继字符
        }
        else{
            i=i-j+2;
            j=1;         //指针后退重新开始匹配
        }
    }
    if(j>T.length)
        return i-T.length;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

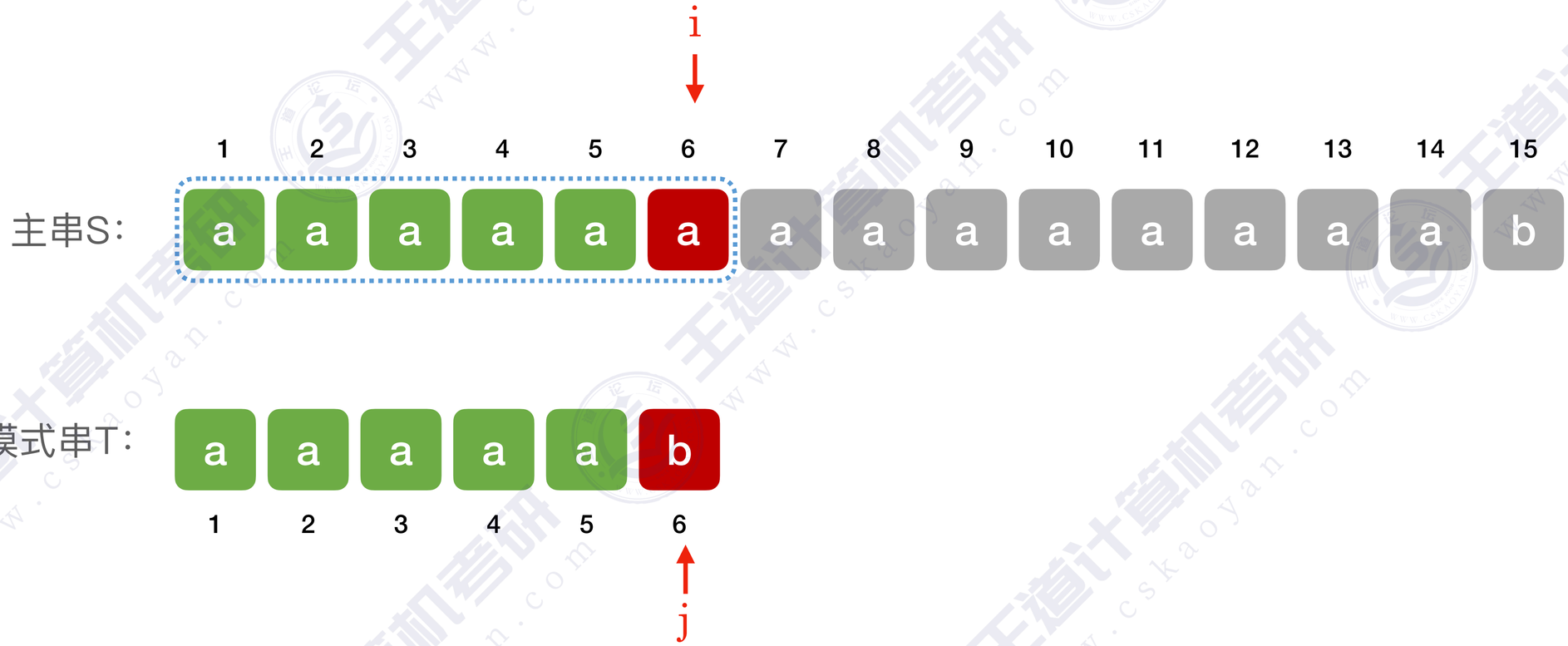
朴素模式匹配算法

最坏时间复杂度 = O(nm)



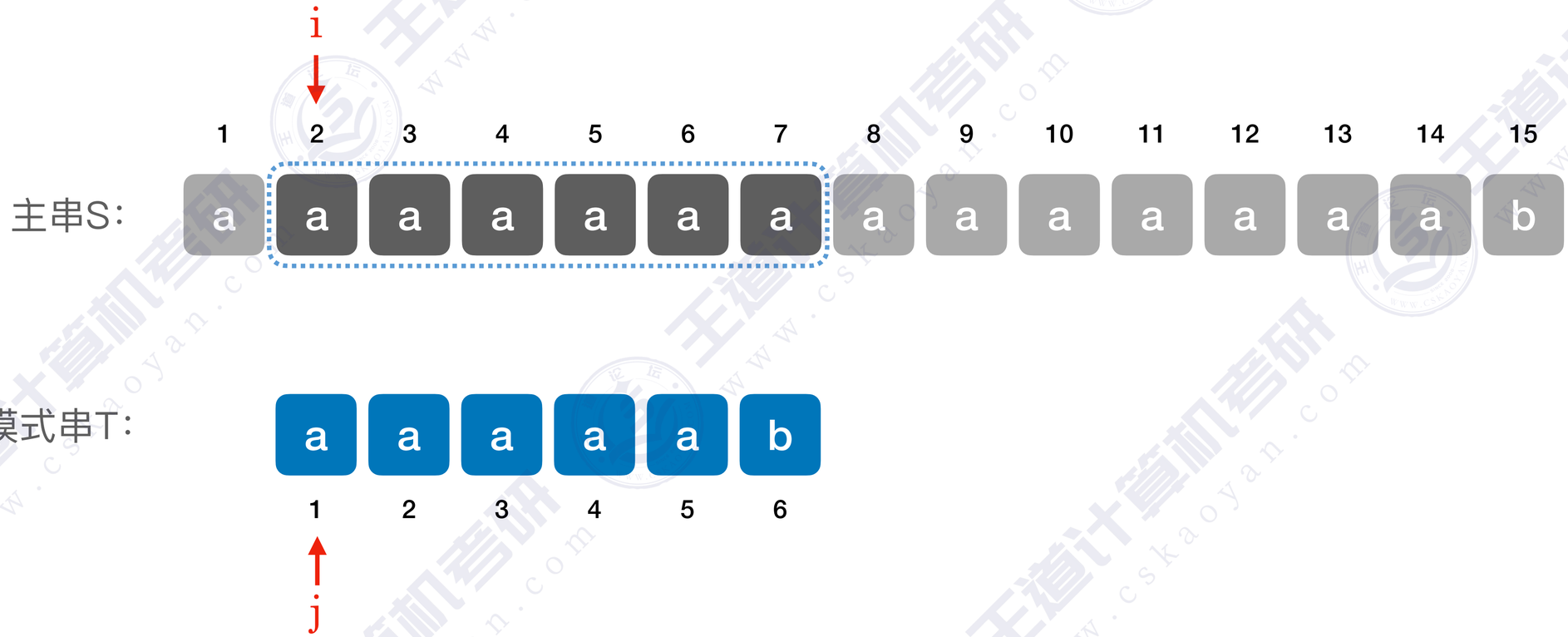
朴素模式匹配算法

最坏时间复杂度 = O(nm)



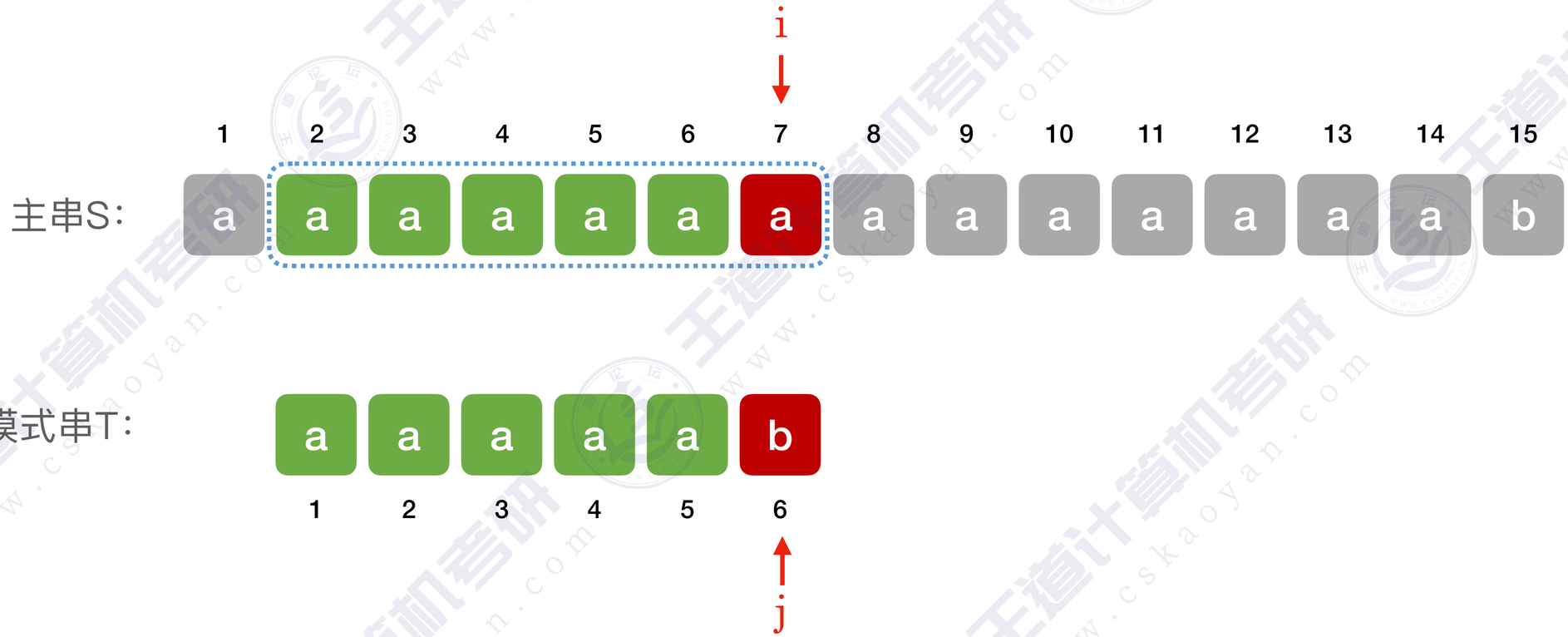
朴素模式匹配算法

最坏时间复杂度 = O(nm)



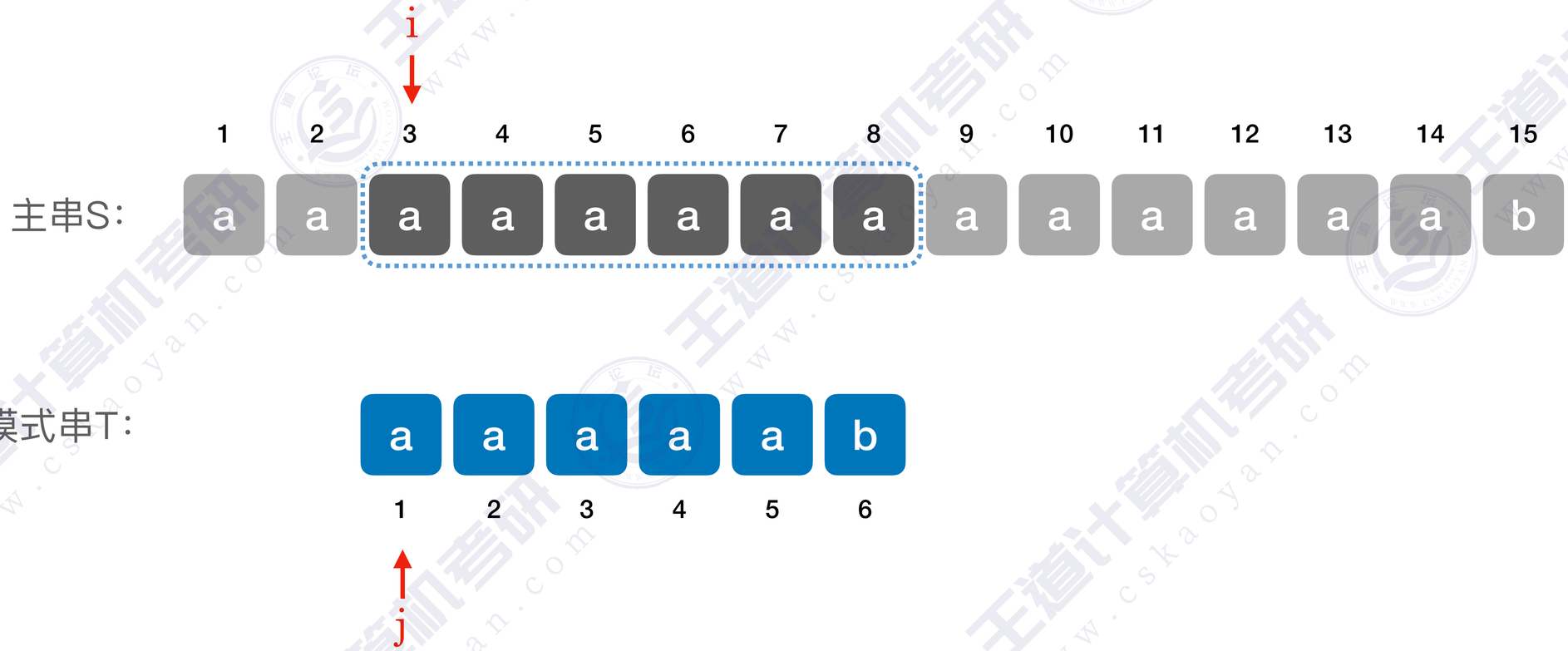
朴素模式匹配算法

最坏时间复杂度 = O(nm)



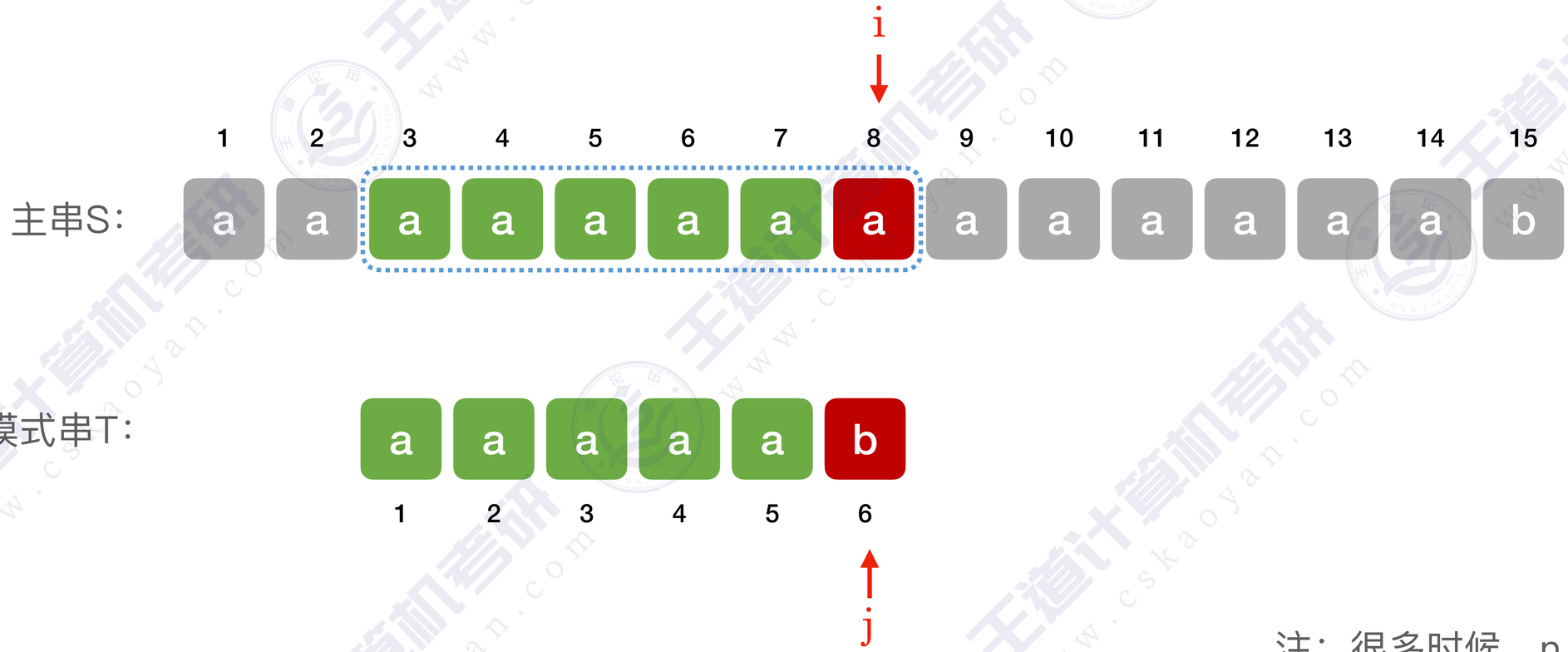
朴素模式匹配算法

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)



注：很多时候，n >> 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