团体整体作战，不抛弃不放弃

第09讲 字符串

信息学院（智能应用研究院）
欧 新 宇

建议课时：2

第4章 字符串

考核要点

- **考核大纲**

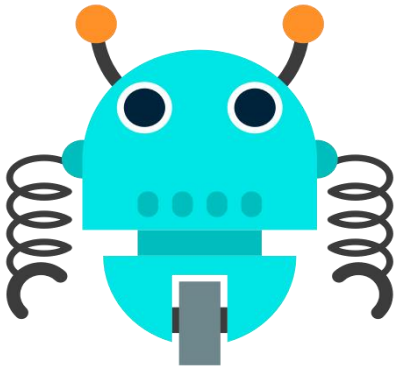
  字符串的模式匹配

- **复习要点**
  - 了解朴素模式匹配（BF暴力匹配）算法
  - 重点掌握KMP匹配算法的原理
  - 掌握next数组的推理过程，能够手工求next数组
  - 了解nextval数组的求解方法

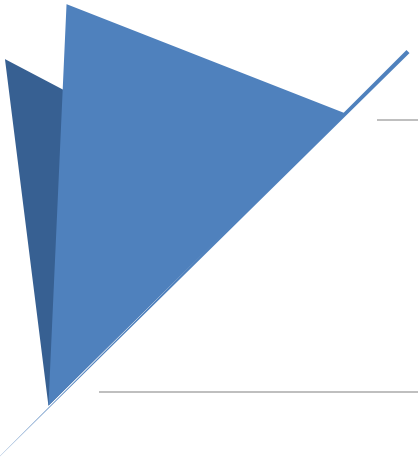
第09讲 字符串

本讲作业

- **必做题**
  - ✓ 【课堂互动9.1】字符串的基本概念和BF匹配算法
  - ✓ 【课堂互动9.2】KMP匹配算法
  - ✓ 【课后作业09】字符串
- **选做题**
  - ✓ 【课前自测09】字符串
  - ✓ 【扩展练习09】字符串



- 字符串的基本概念
- BF暴力匹配算法（朴素模式匹配）
- KMP匹配算法



字符串的基本概念

/ 字符串的定义

/ 字符串的抽象数据类型

/ 字符串的数据结构

/ 字符串的前缀、后缀和部分匹配

问题引入

模式匹配

查找功能：

给定一段文本，要求能够根据用户提供的特定关键词，找出关键词在文本中出现的位置。

用户给定的关键词或字符串被称为**模式串**，段落文本成为**目标串**，因此字符串匹配又称为模式匹配。

字符串的基本概念

什么是串

字符串（String），简称串，由零个或多个字符组成的有限序列，记作：S =‘$a_1a_2...a_n$’（n≥0）

- S是串名，单引号括起来的字符序列是串值，a_1可以是字母、数字或其他字符
- 串中字符的个数 n 称为串的长度，n = 0 时串称为空串，表示为 ∅
- 串中任意多个连续的字符组成的子序列称为串的子串，包含子串的串称为主串
- 字符在串中的序号称为字符的位置
- 子串在主串中的位置等于子串第一个字符在主串中的位置
- 一个或多个空格组成串称为空格串，空格串不是空串
- 两个串长度相等，且每个对应位置的字符都相等时，两个串相等

字符串的基本概念

字符串的抽象数据类型

ADT String{

数据对象： $D = \{a_i \mid a_i \in CharacterSet, i = 1,2,3,...,n, n \geq 0\}$

数据关系： $R = \{<a_{i-1}, a_i> \mid a_{i-1}, a_i \in D, i = 1,2,3,...,n\}$

基本操作：

StrAssign (&T, chars)：赋值，将串T赋值为chars

StrCompare (S,T)：比较两个串S和T是否相等

StrLength (S)：求串T的长度，返回元素个数

Index(S, T, p)：求主串S中串T的位置

Concat(&T, S1, S2)：将串S1和S2合并为新串T

SubString(&Sub, S, pos, len)：返回串S中起始位置为pos，长度为len的子串

StrCopy(&T, S)：由串S复制得到串T

}ADT String

字符串的基本概念

字符串的存储结构

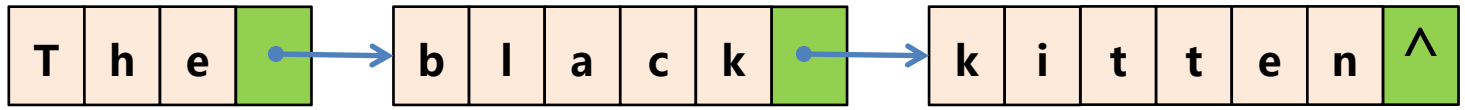
顺序存储结构 → 定长 固定空间

```
#define MaxLen 255          // 1. 定义字符串的最大长度
typedef struct SString {    // 2. 定义顺序串
    char ch[MaxLen];        // 2.1 定义字符串序列
    int length;             // 2.2 定义串内字符长度变量
}SString;
```

堆(顺序存储) → 动态 分配空间

```
#typedef struct HString {   // 1. 定义动态存储的堆字符串
    char *ch;               // 2. 按串长分配存储区
    int length;             // 3. 定义串内字符长度变量
}HString;
```

块(链式储存) → 以多字符存串结点



字符串的基本概念

串的前缀、后缀和部分匹配值

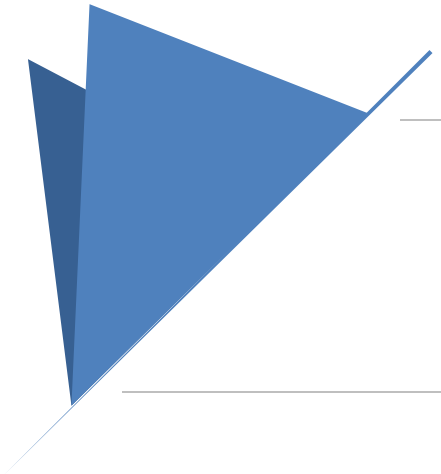
前缀：除最后一个字符以外，字符串的所有头部子串

后缀：除第一个字符以外，字符串的所有尾部子串

部分匹配：字符串的前缀和后缀的最长相等长度

部分匹配值：由每个子串构成的部分匹配值**序列**

例1：字符串 ababa			
子串	前缀	后缀	最长相等值
a	null	null	0
ab	a	b	0
aba	a,ab	a,ba	1
abab	a,ab,aba	b,ab,bab	2
ababa	a,ab,aba,abab	a,ba,aba,baba	3
部分匹配值：00123			



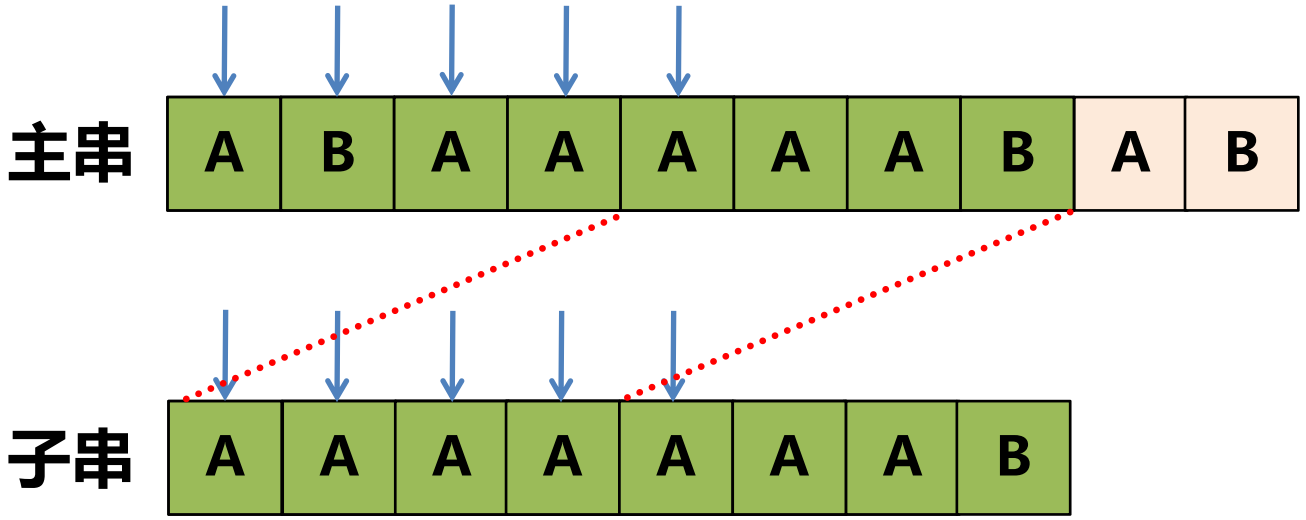
BF暴力匹配算法（朴素模式匹配）

/ BF算法的基本思想

BF暴力匹配算法（朴素模式匹配）

BF算法的基本思想

模式匹配是指从主串中查找与模式串完全相同的部分。其中最简单的方法是逐元素匹配，即**朴素模式匹配（Brute-Force，BF）算法。**



时间复杂度为O(m*n)

平均时间复杂度近似于O(m+n)

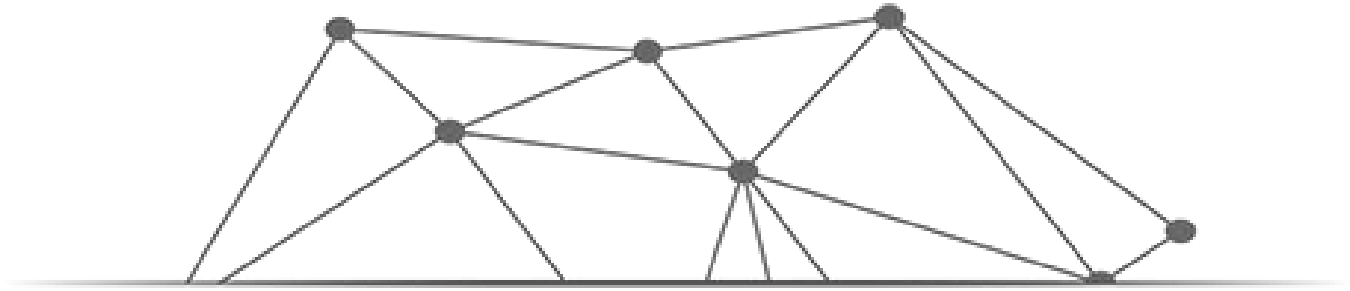
BF暴力匹配算法（朴素模式匹配）

BF算法的基本思想

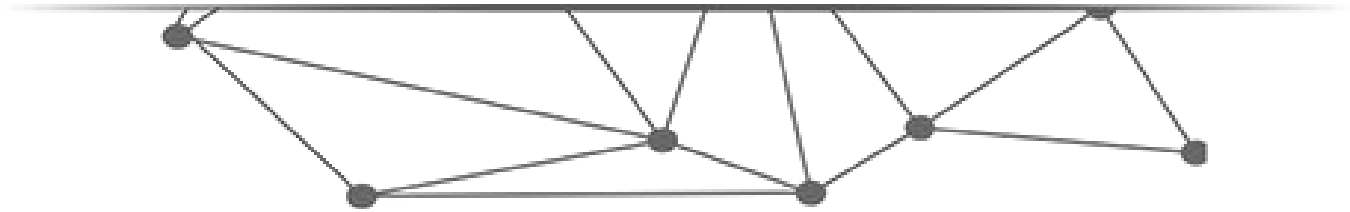
1. 执行循环，将主串的第i个字符与子串的第1个字符比较。
   - ✓ 若相等，则继续逐个比较后续字符
   - ✓ 若不相等，回溯到主串的下一个字符（i = i - j + 2），重新与子串第1个字符比较
2. 若S中存在与T匹配的子序列，则返回子序列的第1个字符序号；若不存在，则返回0

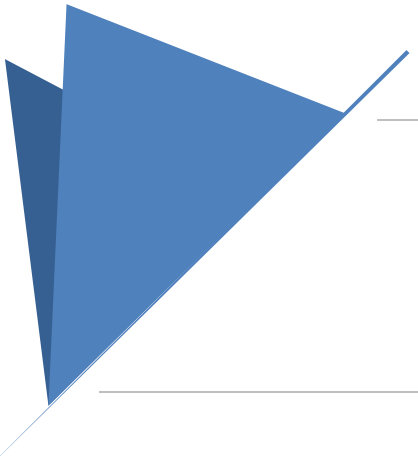
BF算法的基本流程

```
int Index (SString S, SString T) {   // S主串，T子串
    int i = 1, j = 1;                 // 初始化串的指针
    while (i<=S.length && j <=T.length) {
        if (S.ch[i] == T.ch[j]) {     // 相等，则继续比较
            i++;   j++;
        }
        else {                        // 不相等，则回溯
            i = i - j + 2;   j = 1;
        }
    }
    if (j > T.length)
        return i - T.length;
    else
        return 0;
}
```



课堂互动 9.1





KMP匹配算法

/ KMP算法的动机

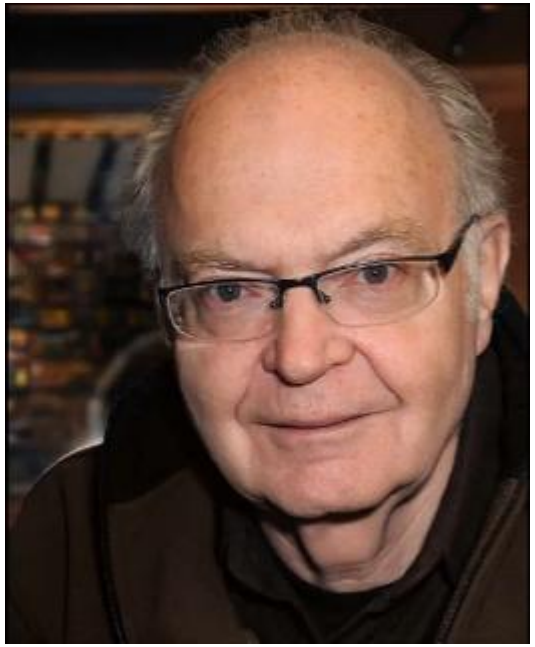
/ KMP算法的基本思想

/ KMP算法的代码实现

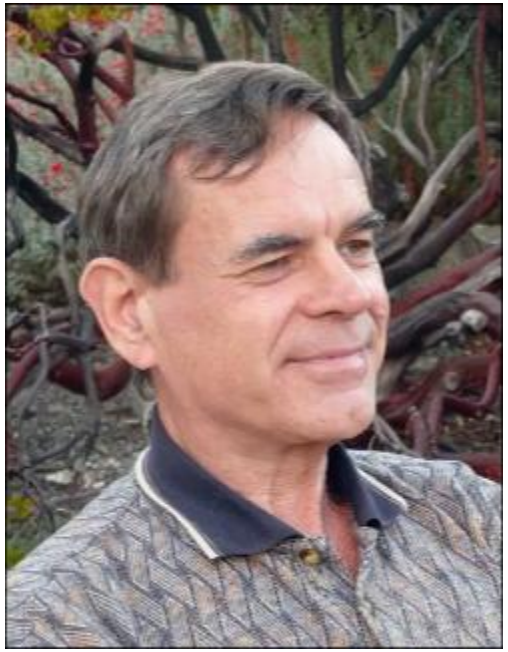
/ next[] 数组的计算方法

/ nextval[] 数组的计算方法

KMP匹配算法



克鲁特Knuth
《计算机程序设计艺术》

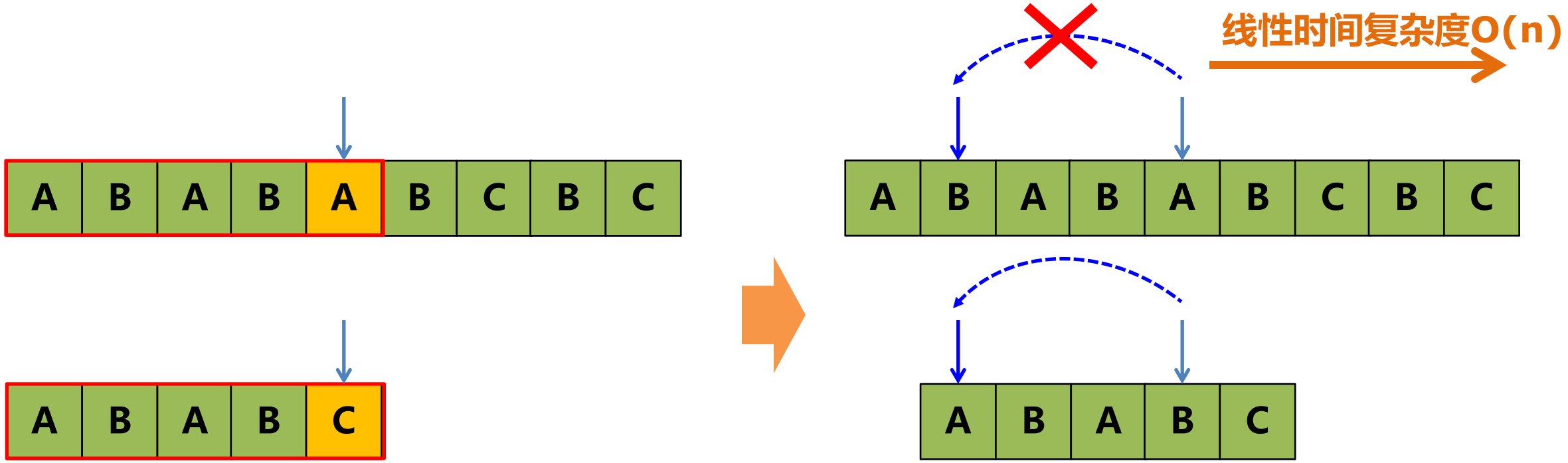


莫里斯Morris

普拉特Pratt

KMP匹配算法

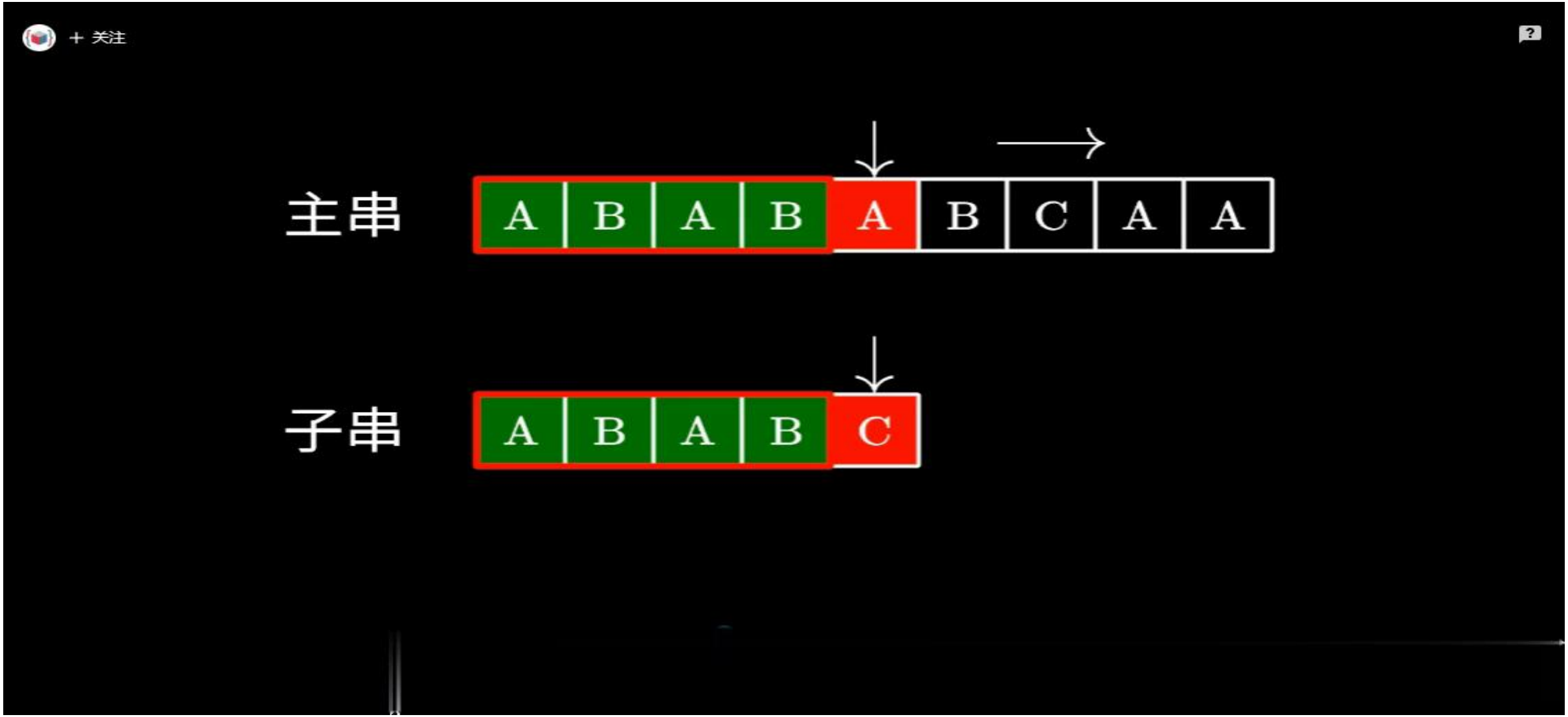
KMP算法的动机



是否能利用已获得的信息来避免暴力算法中"回退（backup）"的步骤?

KMP匹配算法

KMP算法的基本思想



KMP匹配算法

KMP算法的代码实现

1. 定义两个指针，分别指向主串和子串的工作元素
2. 若主串和子串未完，则持续对比
   - ✓ 若单个字符匹配，则指针后移1位
   - ✓ 若单个字符失配，则子串根据next数组跳过前面的一些字符
   - ✓ 若子串第一个字符失配，则将主串指针向前移动1位
3. 若匹配成功，返回子串的起始位置

KMP算法主程序

```
Status KMP_Search (SString string, SString patt, int next[])
{
    int i =1, j = 1;
    while (i <= string.length && j <= patt.length) {
        if (string.ch[i] == patt.ch[j]) {
            i++;   j++;
        }
        else if (j >1)
            j = next[j];
        else
            i++;
    }
    if (j > patt.length)
        return i - patt.length;
}
```

主串中的
指针 i 永不回退
时间复杂度O(n)

KMP匹配算法

模式串 next[j] 的计算方式

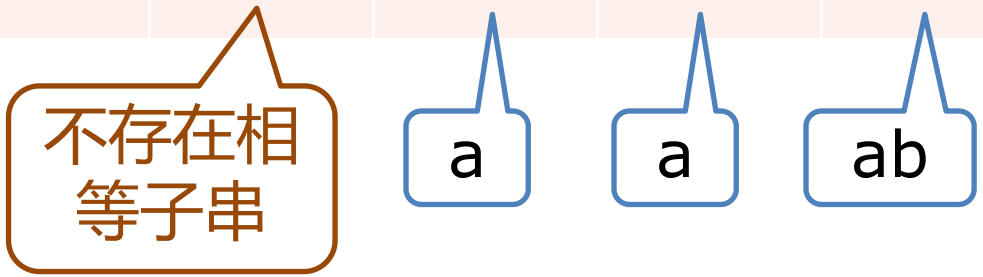
❶ next[1] = **0**，next[2] = **1**

❷ 若存在n个前缀 = n个后缀，则next[j] = n+1

❸ 不存在 前缀 = 后缀，则 next[j] = 1

例2:

j	1	2	3	4	5	6	7
模式串	a	b	c	a	a	b	b
Next[j]	**0**	**1**	1	1	2	2	3



KMP匹配算法

模式串 next[j] 的计算方式

例3: 已知字符串S为 ‘abaabaabcabaabc’ ，模式串T为 ‘abaabc’ 。采用KMP算法进行匹配，第一次出现“失配”（S[i] ≠ T[j]）时，i=j=5，则下次开始匹配时，i 和 j 的值分别是（ **C** ）：

A. i=1, j=0　　B. i=5, j=0　　C. i=5, j=2　　D. i=6, j=2

【解析】

1. 在KMP算法中，主串指针 i 永不回退，因此 i = 5
2. 当 j=5 时，新的 j 值等于 next[5]

j	1	2	3	4	5	6
模式串	a	b	a	a	b	c
部分匹配字符	-	-	-	a	a	ab
next[j]	0	1	1	2	2	3

KMP匹配算法

next[] 数组的优化

当模式串中有多个连续相同的值，则可能出现多次重复匹配。

例如，主串 s=aaabaaaab，模式串 p=aaaab

主串	a	a	a	b	a	a	a	a	b
j	1	2	3	4	5				
模式串	a	a	a	**a**	b				
Next[j]	**0**	**1**	2	3	4				

③ ② ①

当发生失配时，需要多次连续回退。因为 $p_{next[4]=3}=p_4=a$，$p_{next[3]=2}=p_3=a$，$p_{next[2]=1}=p_2=a$。

解决办法：

对next[j]进行再次递归，将next[j]修正为next[next[j]]

KMP匹配算法

next[] 数组的优化 -> nextval[] 数组

1. nextval[1] = **0**
2. 若第2位与第1位相同，则为0；若不相同，则为1
3. 从第3位开始，根据next值指路。
   若元素值相同，则等于目标元素nextval值；不相同，则等于当前next值

j	1	2	3	4	5
模式串	a	a	a	**a**	b
next[j]	**0**	**1**	2	3	4
nextval[j]	**0**	0	0	0	4

直接退回到起点

KMP匹配算法

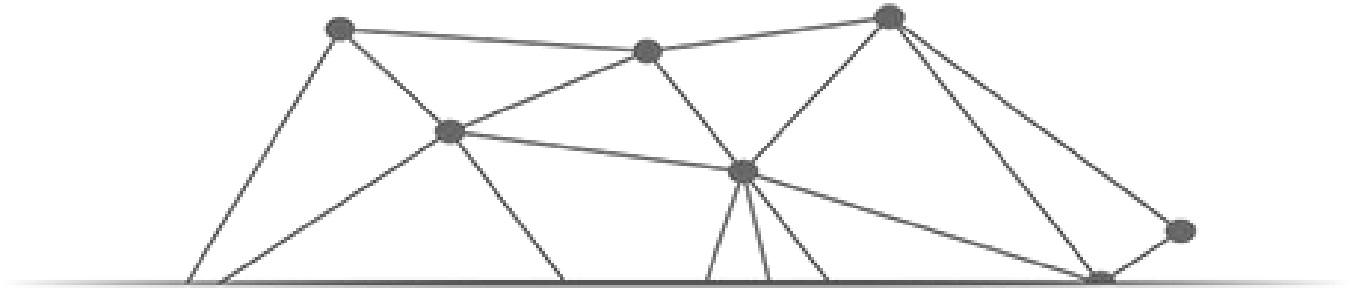
next[] 数组的优化 -> nextval[] 数组

例4: 串 ‘ababaaababaa’ 的nextval数组为（ C ）：

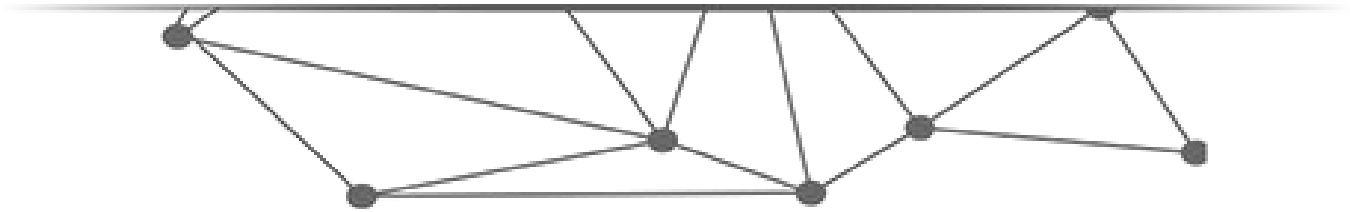
A. 010112010102　　B. 010114110102　　C. 010104210104　　D. 011102110104

【解析】

j	1	2	3	4	5	6	7	8	9	10	11	12
模式串	a	b	a	b	a	a	a	b	a	b	a	a
next[j]	**0**	**1**	1	2	3	4	2	2	3	4	5	6
nextval[j]	**0**	**1**	0	1	0	4	2	1	0	1	0	4



课堂互动 9.2



第09讲 串

小　结

- **串**是内容受限的线性表，它限定表中的元素为字符。
- 常见的模式匹配，包括BF匹配算法和KMP匹配算法
- BF暴力模式匹配算法**时间复杂度O(m*n)**，KMP匹配算法**时间复杂度O(m+n)**
- BF暴力模式匹配算法失配时，主串 i 回溯到 **i-j+2**，模式串 j 回溯到 **j=1**
- KMP匹配算法失配时，主串 i 不需要回溯，模式串 j 回溯与模式串有关，使用next[j]/nextval[j]表示
- nextval[] 数组时对next[] 数组的优化，可以有效解决连续字符重复对比的问题

欧老师的联系方式

读万卷书 行万里路 只为最好的修炼

QQ：14777591 (宇宙骑士)

Email：ouxinyu@alumni.hust.edu.cn

Website: http://ouxinyu.cn

Tel：18687840023

地址：安宁校区 诚远楼201

南院 智能应用研究院A306-2