数据结构与算法（四）

张铭 主讲

采用教材：张铭，王腾蛟，赵海燕 编写
高等教育出版社，2008. 6（“十一五”国家级规划教材）

https://pkumooc.coursera.org/bdsalgo-001

主要内容

- 字符串基本概念
- 字符串的存储结构
- 字符串运算的算法实现
- 字符串的模式匹配
  - 朴素算法
  - KMP 快速模式匹配

4.3 字符串的模式匹配

- 模式匹配 (pattern matching)
  - 一个目标对象 T（字符串）
  - （pattern）P（字符串）

  在目标 T 中寻找一个给定的模式P的过程
- 应用
  - 文本编辑时的特定词、句的查找
    - UNIX/Linux: sed, awk, grep
  - DNA 信息的提取
  - 确认是否具有某种结构
  - …
- 模式集合

字符串的模式匹配

- 用给定的模式 P，在目标字符串 T 中搜索与模式 P 全同的一个子串，并求出 T 中第一个和 P 全同匹配的子串（简称为"配串"）,返回其首字符位置

$$T \quad t_0\ t_1 \ldots \ t_i t_{i+1} \quad t_{i+2} \quad \ldots \quad t_{i+m-2} \quad t_{i+m-1} \ldots t_{n-1}$$

$$\qquad\qquad\qquad\quad \| \quad \| \quad \| \qquad\qquad \| \qquad \|$$

$$P \qquad\qquad\quad p_0\ p_1 \quad p_2 \quad \ldots \quad p_{m-2} \quad p_{m-1}$$

为使模式 P 与目标 T 匹配，必须满足

$$p_0\ p_1\ p_2 \ldots p_{m-1} = t_i\ t_{i+1}\ t_{i+2} \ldots t_{i+m-1}$$

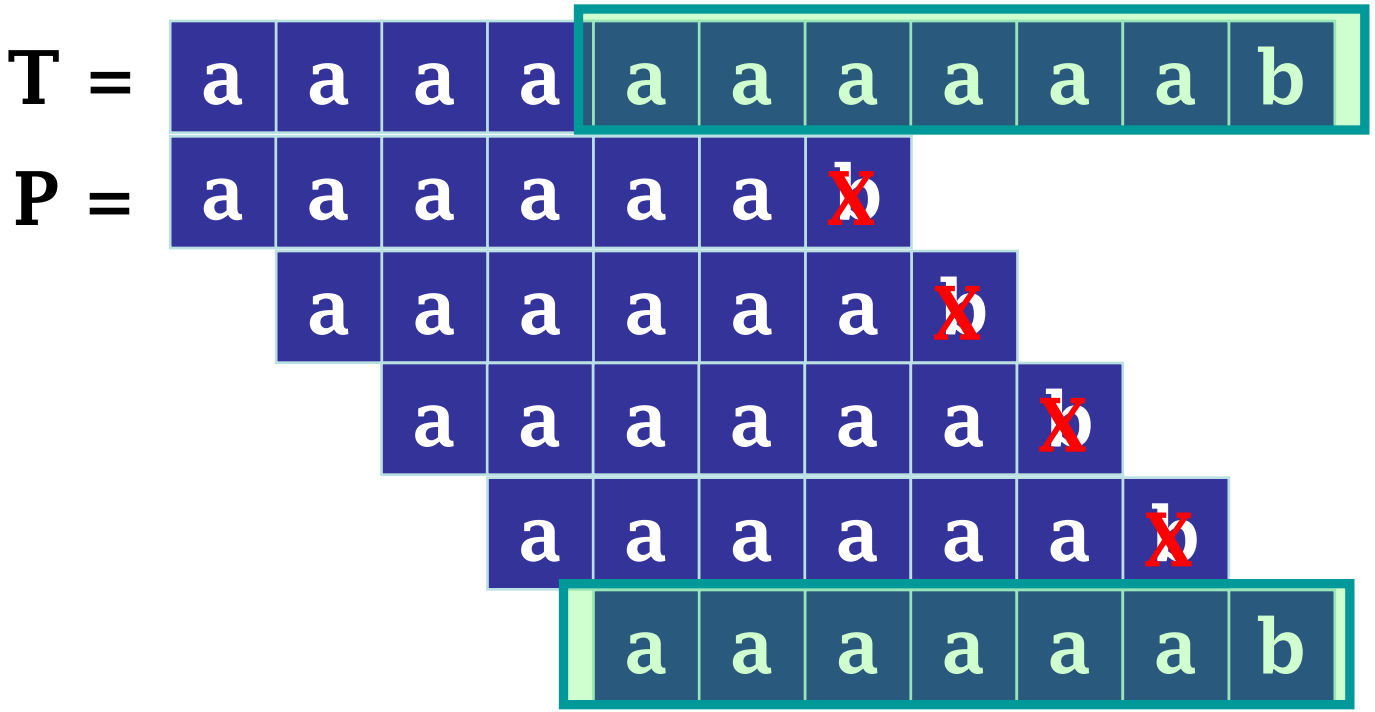
模式匹配的目标和算法

- 目标：在大文本（诸如，句子、段落，或书本）中定位（查找）特定的模式
- 解决模式匹配问题的算法
  - 朴素（称为“Brute Force”，也称“Naive”）
  - Knuth-Morrit-Pratt（KMP 算法）
  - ……

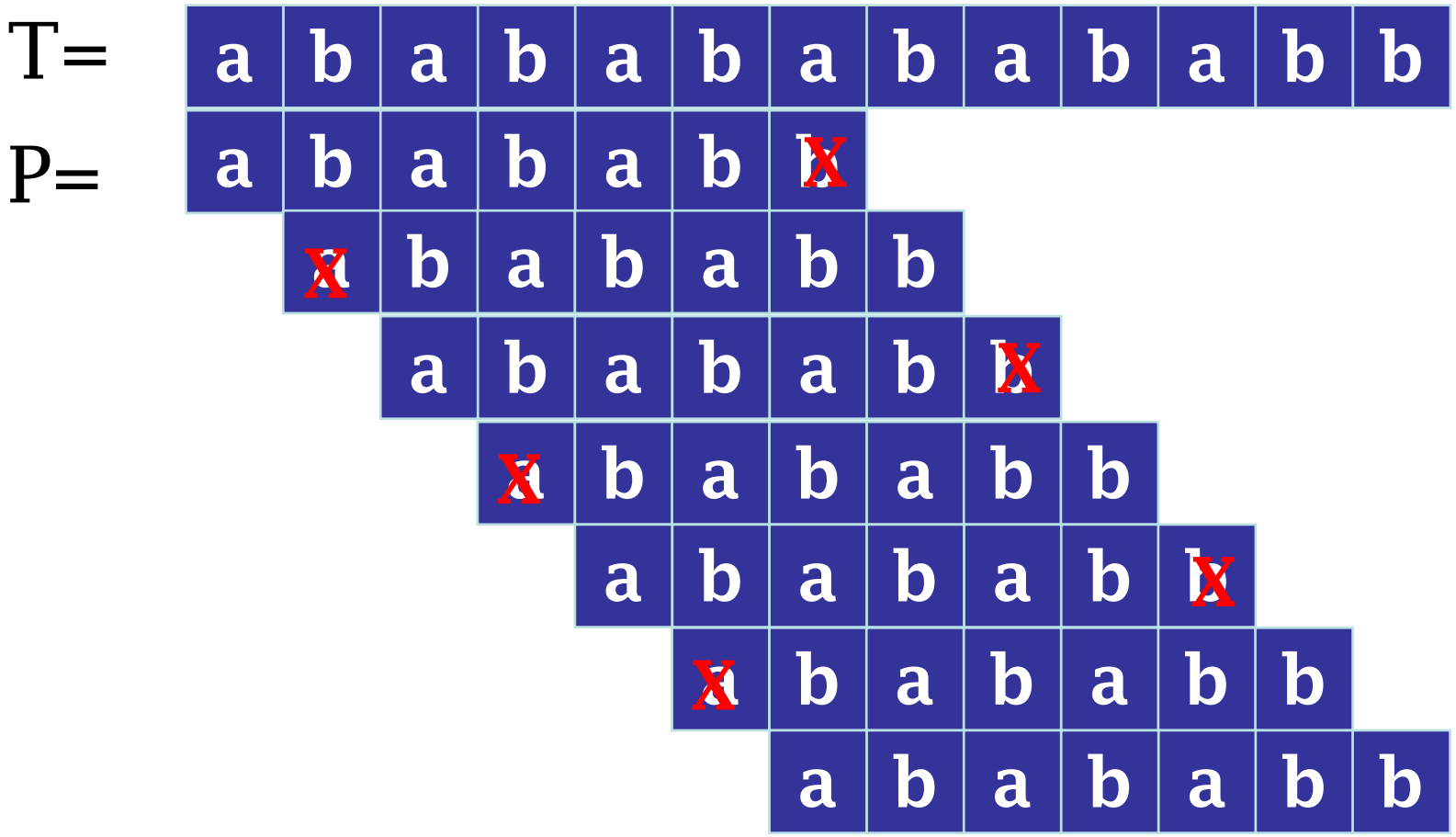
朴素模式匹配（穷举法）

- 设 $T = t_0t_1, t_2, \ldots,t_{n-1}$ ，$P = p_0, p_1, \ldots, p_{m-1}$
  - i 为 T 中字符的下标，j 为 P 中字符的下标
  - 匹配成功 ($p_0 = t_i$, $p_1 = t_{i+1}$, …， $p_{m-1} = t_{i+m-1}$)
    - 即，T.substr(i, m) == P.substr(0, m)
  - 匹配失败 ($p_j \neq t_i$) 时，
    - 将 P 右移再行比较
  - 尝试所有的可能情况

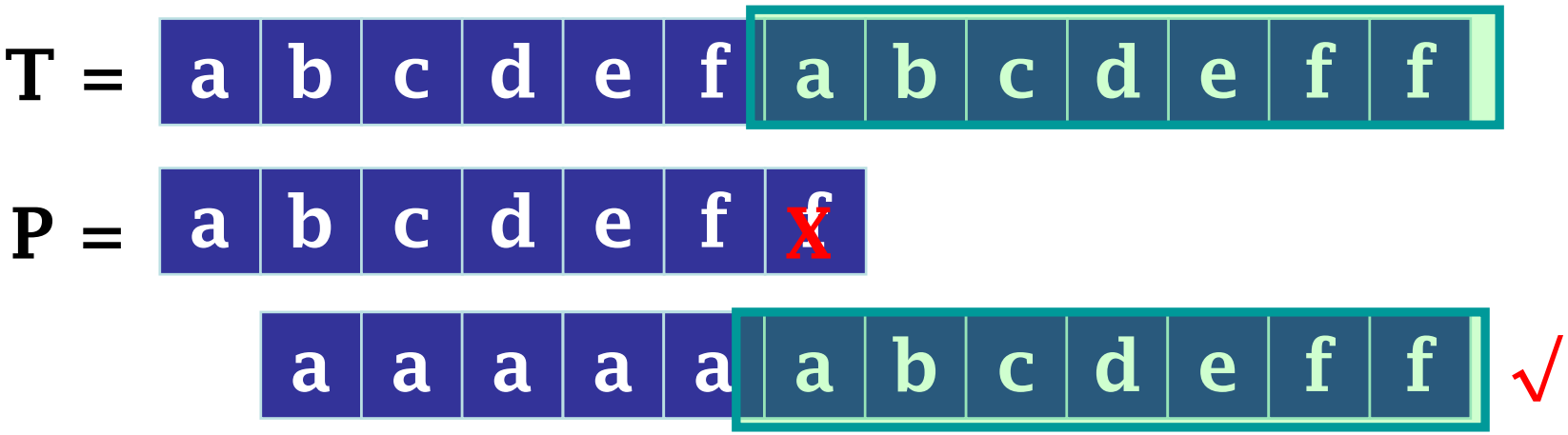
朴素匹配例1



朴素模式匹配例2


T=
a b a b a b a b a b a b b
P=
a b a b a b b
X
a b a b a b b
X
a b a b a b b
X
a b a b a b b
X
a b a b a b b
X
a b a b a b b
X
a b a b a b b

朴素匹配例3



朴素模式匹配算法：其一

```
int FindPat_1(string S, string P, int startindex) {
    // 从S末尾倒数一个模式长度位置
    int LastIndex = S.length() - P.length();
    int count = P.length();
    // 开始匹配位置startindex的值过大，匹配无法成功
    if (LastIndex < startindex)
        return (-1);
    // g为S的游标，用模式P和S第g位置子串比较，若失败则继续循环
    for (int g = startindex; g <= LastIndex; g++)  {
        if ( P == S.substr(g, count))
            return  g;
    }
    // 若for循环结束，则整个匹配失败，返回值为负，
    return  (-1);
}
```

朴素模式匹配算法：其二

```
int  FindPat_2(string T, string P, int  startindex) {
   // 从T末尾倒数一个模板长度位置
   int LastIndex = T.length() - P.length();
   // 开始匹配位置startindex的值过大，匹配无法成功
   if  (LastIndex < startindex) return (-1);
   // i 是指向T内部字符的游标，j 是指向P内部字符的游标
   int i = startindex,   j = 0;
   while (i < T.length()  &&  j < P.length())     // “<=”呢？
        if (P[j] == T[i])
            {  i++;    j++;   }
        else
            {  i = i - j + 1;   j = 0; }
   if ( j  >=  P.length())   // “>” 可以吗？
        return (i - j);          // 若匹配成功，则返回该T子串的开始位置
   else    return -1;          // 若失败，函数返回值为负
}
```

朴素模式匹配代码（简洁）

```
int FindPat_3(string T, string P, int startindex) {
    //g为T的游标，用模板P和T第g位置子串比较，
    //若失败则继续循环
    for (int g= startindex; g <= T.length() - P.length(); g++) {
        for (int j=0;  ((j<P.length()) && (T[g+j]==P[j])) ; j++) ;
        if (j == P.length())
            return  g;
    }
    return(-1);  // for结束，或startindex值过大,则匹配失败
}
```

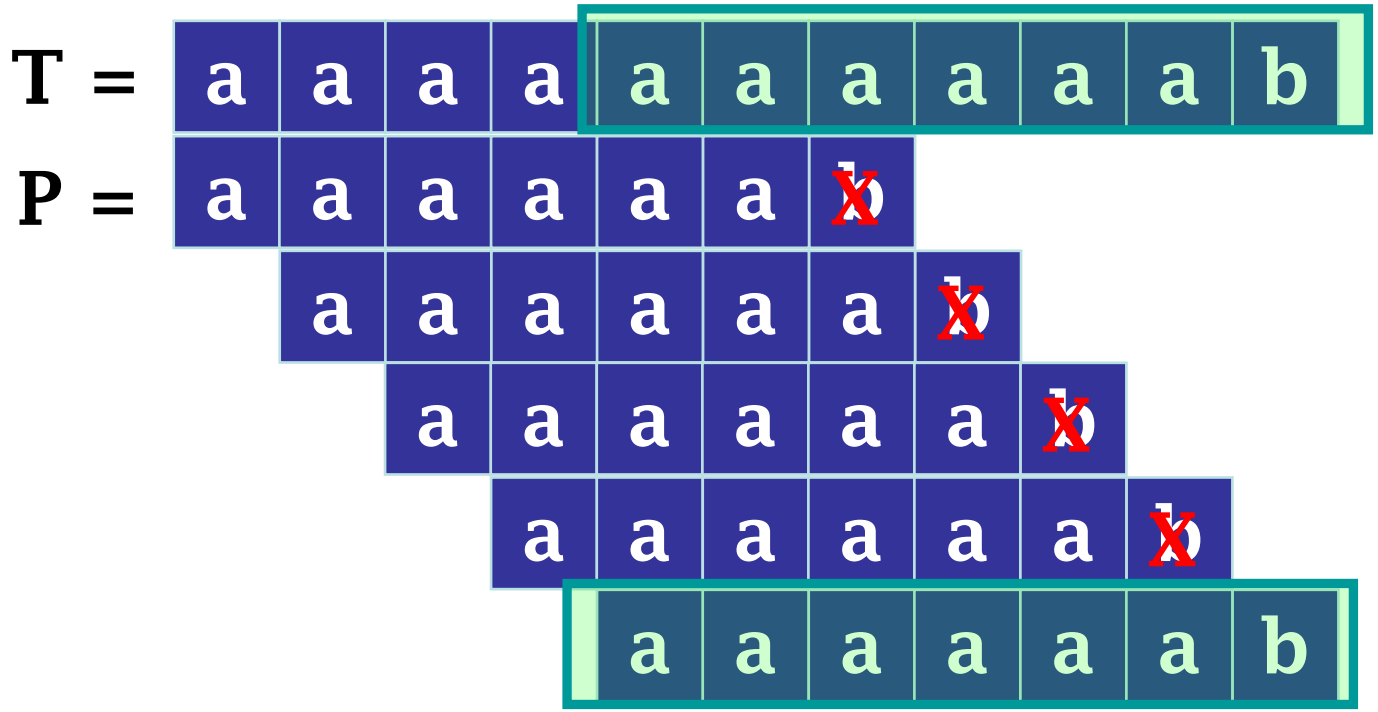
模式匹配原始算法：效率分析

- 假定目标 T 的长度为 n，模式 P 长度为 m，m≤n
  - 在最坏的情况下，每一次循环都不成功，则一共要进行比较（n-m+1）次
  - 每一次“相同匹配”比较所耗费的时间，是 P 和 T 逐个字符比较的时间，最坏情况下，共 m 次
  - 因此，整个算法的最坏时间开销估计为

$$O(m \bullet n)$$

朴素模式匹配算法：最差情况

- 模式与目标的每一个长度为 m 的子串进行比较
  - 目标形如 $a^{n-1}X$
  - 模式形如 $a^{m-1}b$



- 总比较次数：
  - $m(n - m + 1)$
- 时间复杂度：
  - $O(mn)$

朴素模式匹配算法：最佳情况-找到模式

- 在目标的前 m 个位置上找到模式，设 m = 5

AAAAAAAAAAAAAAAAAAAAH

AAAAA

5次比较

- 总比较次数：m
- 时间复杂度：O(m)

朴素模式匹配算法：

最佳情况-没找到模式

- 总是在第一个字符上不匹配
- 总比较次数：
  - n - m + 1
- 时间复杂度：
  - O(n)

```
AAAAAAAAAAAAAAAAAAAAH
OOOOH                    1次比较
AAAAAAAAAAAAAAAAAAAAH
 OOOOH                   1次比较
AAAAAAAAAAAAAAAAAAAAH
  OOOOH                  1次比较
AAAAAAAAAAAAAAAAAAAAH
   OOOOH                 1次比较
.....
AAAAAAAAAAAAAAAAAAAAH
                 OOOOH 1次比较
```

思考：朴素算法的冗余运算

- 朴素算法之所以较慢的原因是有冗余运算
- e.g.,
  - 由1）可知： $p_5 \neq t_5$ ， $p_0=t_0$, $p_1 = t_1$,同时由$p_0 \neq p_1$可得知$p_0 \neq t_2$ 故将 P 右移一位后第2）趟比较一定不等；比较冗余
  - 那么把 P 右移几位可以消除冗余的比较而不丢失配串呢？

```
T  a b a c a a b a c c a b a c a b a a
P  a b a c a b
```

1）$p_5 \neq T_5$ P右移一位

```
T  a b a c a a b a c c a b a c a b a a
P        a b a c a b
```

2）$p_0 \neq T_1$ P右移一位

```
T a b a c a a b a c c a b a c a b a a
P          a b a c a b
```

3）$p_1 \neq T_3$ P右移一位

```
T a b a c a a b a c c a b a c a b a a
P      a b a c a b
……
```

数据结构与算法

谢谢聆听

国家精品课“数据结构与算法”

https://pkumooc.coursera.org/bdsalgo-001

张铭，王腾蛟，赵海燕

高等教育出版社，2008. 6。“十一五”国家级规划教材