

Projeto e Análise de Algoritmos

Aula 06 – Busca de Padrões em Texto

Edirlei Soares de Lima
<edirlei@iprj.uerj.br>

Processamento de Texto

- **Cadeia de caracteres:** sequência de elementos denominados caracteres.
- Os **caracteres** são escolhidos de um conjunto denominado alfabeto.
- **Casamento de cadeias de caracteres** (string matching) ou **casamento de padrão** (pattern matching): encontrar todas as ocorrências de um padrão em um texto.
- **Exemplos de aplicação:** edição de texto, recuperação de informação, estudo de sequências de DNA em biologia computacional, ...

Processamento de Texto

- **Texto:** arranjo $T[1..n]$ de tamanho n .
- **Padrão:** arranjo $P[1..m]$ de tamanho $m \leq n$.
- Os elementos de P e T são escolhidos de um alfabeto finito α de tamanho c .
 - Exemplo: $\alpha = \{0, 1\}$ ou $\alpha = \{a, b, \dots, z\}$
- **Casamento de cadeias ou casamento de padrão:** dados duas cadeias P (padrão) de comprimento m e T (texto) de comprimento n , onde $n \leq m$, deseja-se saber as ocorrências de P em T .

Cadeias de Caracteres

- “CGTAAACTGCTTTAATCAAACGC”
 - Cadeias de moléculas chamadas bases: adenina (A), guanina (G), citosina (C) e timina (T)
- “Projeto e Análise de Algoritmos”
- “<http://www.iprj.uerj.br>”
- “if ((x==0) && (y!=3))”

Encontrando Padrões em Textos

- Exemplo:
 - T = “kvjlixapejrbxeenpphkhtbkwyrwamnugzhppfxiyjyanhapfwbghxm
shrlyujfjhrsovkvveylnbxnawavgizyvmfohigeabgksfnbkmffxjffqbualetqr
phyrbjqdjqavctgxjifqgfydhoiwhrvwqbxgrixyzbpajnhopvlamhhfavoctd
fytvvggikngkwzixgjtlxkozjlefilbrboignbzsudssvqymnapbpqvlubdoyxkkw
hcoudvtkmikansgsutdjythzl”
 - P = “avoctdfytvv”

Encontrando Padrões em Textos (Força Bruta)

T =

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

P =

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

- **Objetivo:** descobrir se a string P (padrão) está em T
- **Força bruta:** caminha elemento a elemento da esquerda para a direita

Encontrando Padrões em Textos (Força Bruta)

T =

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

P =

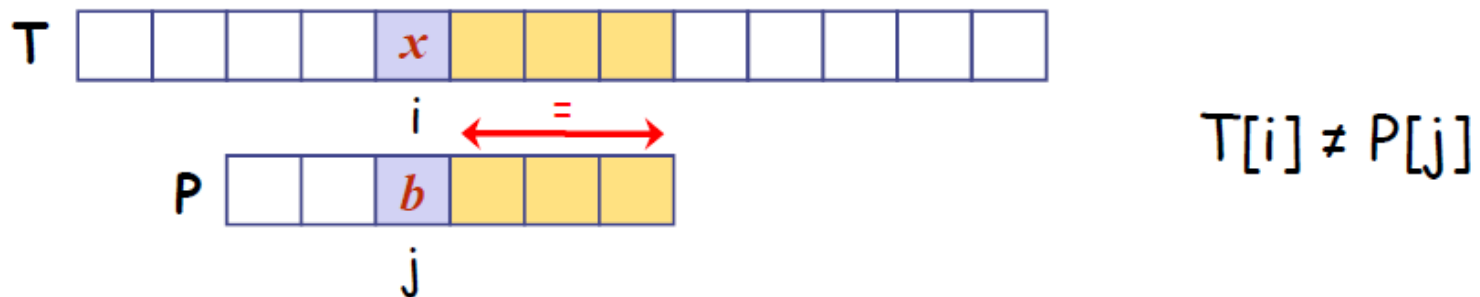
a	b	a	c	a	b
---	---	---	---	---	---

```
1. busca_padrao(t, n, p, m)
2.   para i ← 0 até n - m faça
3.     j ← 0
4.     enquanto (j < m) e t[i + j] == p[j] faça
5.       j ← j + 1
6.     se j == m então
7.       retorne i
8.   retorne "padrão p não encontrado em t"
```

O(nm)

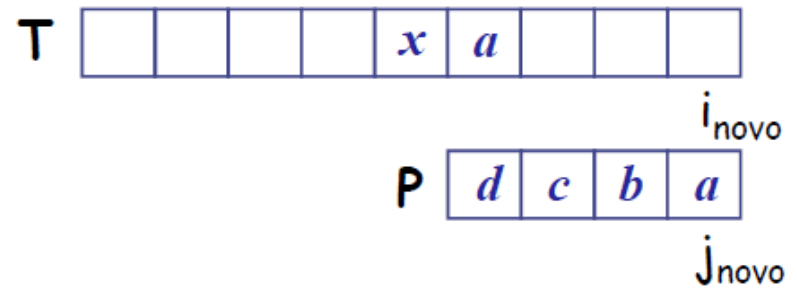
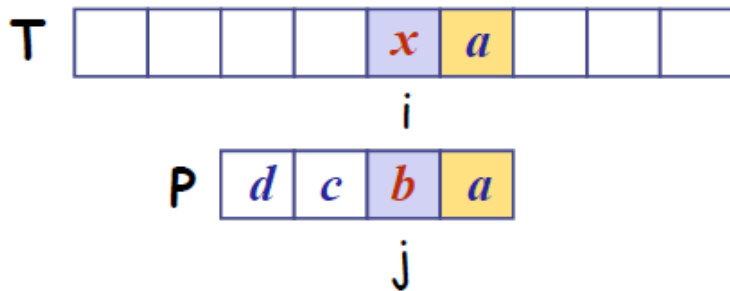
Encontrando Padrões em Textos

- É possível melhorar o algoritmo de força bruta?
- **Algoritmo de Boyer-Moore (1976)**
 - Baseia-se na alta probabilidade de encontrar diferenças em alfabetos grandes.
 - Por isso, P é comparado com T de trás para frente.
 - Quando se encontra uma diferença em $T[i]$, o padrão P dará um salto à frente, considerando-se as comparações já realizadas.



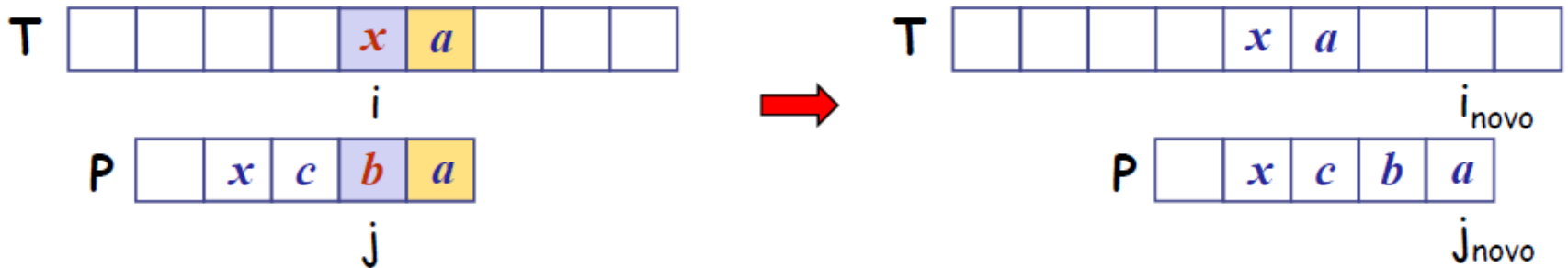
Algoritmo de Boyer-Moore

- No algoritmo é necessário considerar 3 casos:
 - **Caso 1:** P não contém x
 - Deslocar P para a direita, alinhando P[0] com T[i+1]



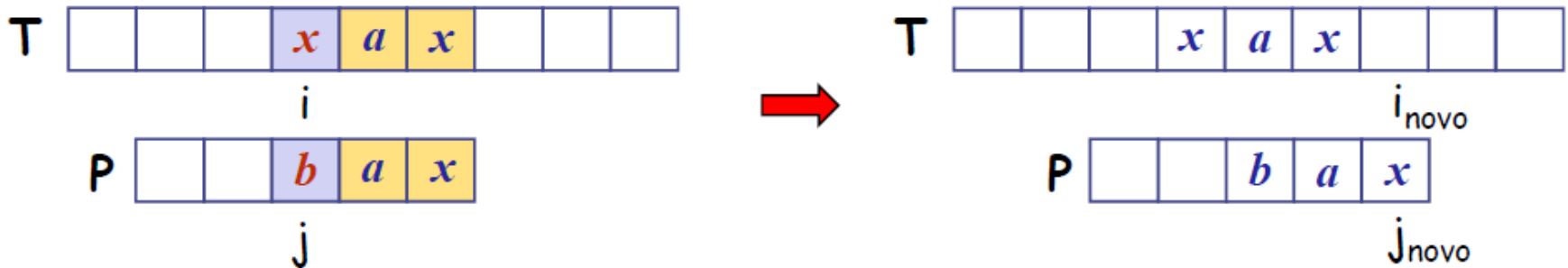
Algoritmo de Boyer-Moore

- No algoritmo é necessário considerar 3 casos:
 - **Caso 2:** A última ocorrência de x em P está algum índice menor do que j .
 - Deslocar P para a direita, até que a última ocorrência de x fique alinhada com $T[i]$.

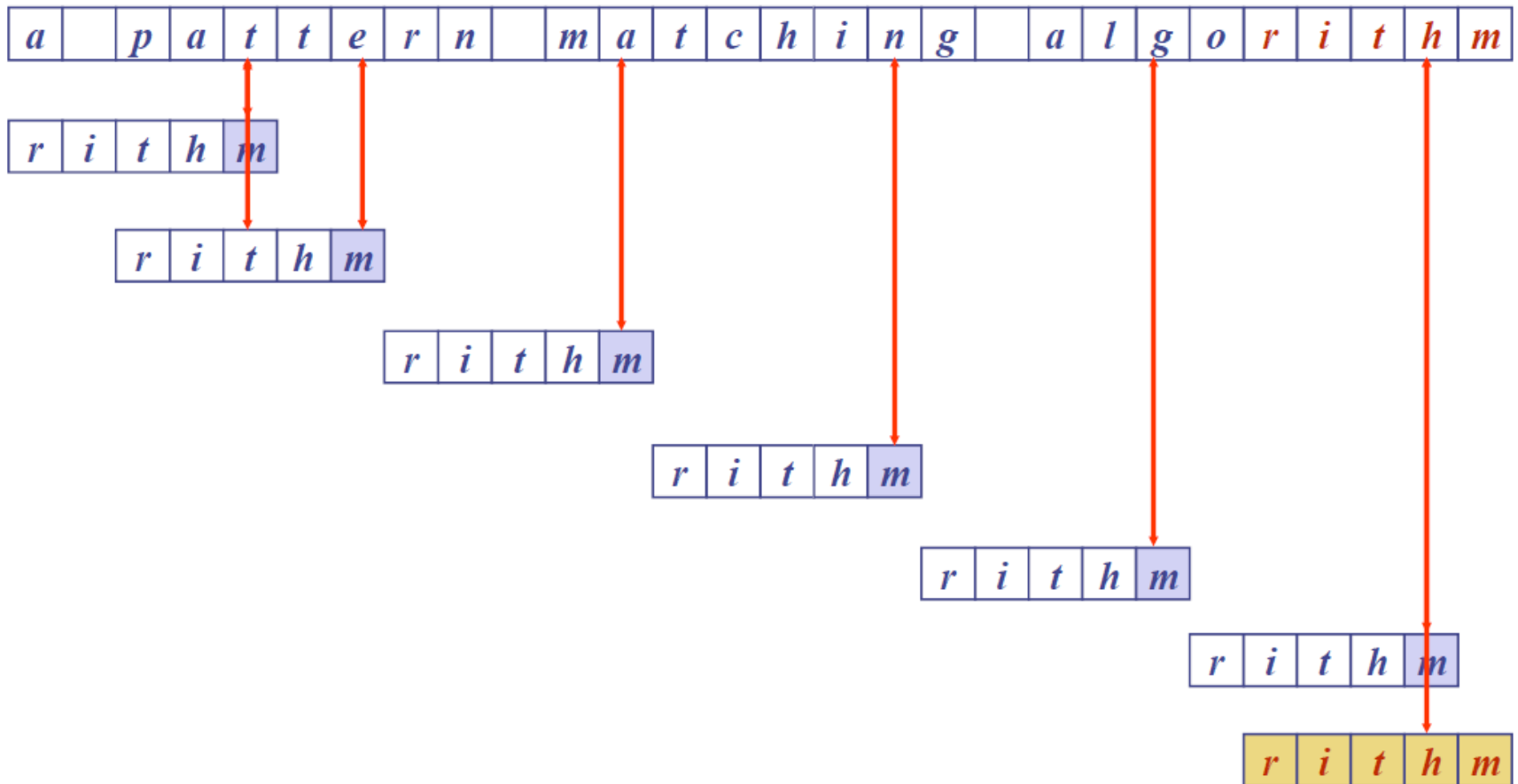


Algoritmo de Boyer-Moore

- No algoritmo é necessário considerar 3 casos:
 - **Caso 3:** A última ocorrência de x em P está em algum índice maior do que j .
 - Deslocar P apenas uma posição para a direita.



Algoritmo de Boyer-Moore – Exemplo



Algoritmo de Boyer-Moore – Exemplo

<i>a</i>	<i>b</i>	<i>a</i>	<i>c</i>	<i>a</i>	<i>a</i>	<i>b</i>	<i>a</i>	<i>d</i>	<i>c</i>	<i>a</i>	<i>b</i>	<i>a</i>	<i>c</i>	<i>a</i>	<i>b</i>	<i>a</i>	<i>a</i>	<i>b</i>	<i>b</i>
----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------

1

<i>a</i>	<i>b</i>	<i>a</i>	<i>c</i>	<i>a</i>	<i>b</i>
----------	----------	----------	----------	----------	----------

4 3 2

<i>a</i>	<i>b</i>	<i>a</i>	<i>c</i>	<i>a</i>	<i>b</i>
----------	----------	----------	----------	----------	----------

5

<i>a</i>	<i>b</i>	<i>a</i>	<i>c</i>	<i>a</i>	<i>b</i>
----------	----------	----------	----------	----------	----------

6

<i>a</i>	<i>b</i>	<i>a</i>	<i>c</i>	<i>a</i>	<i>b</i>
----------	----------	----------	----------	----------	----------

7

<i>a</i>	<i>b</i>	<i>a</i>	<i>c</i>	<i>a</i>	<i>b</i>
----------	----------	----------	----------	----------	----------

13 12 11 10 9 8

<i>a</i>	<i>b</i>	<i>a</i>	<i>c</i>	<i>a</i>	<i>b</i>
----------	----------	----------	----------	----------	----------

Algoritmo de Boyer-Moore

- Através de um pré-processamento, o algoritmo de Boyer-Moore calcula uma função L, onde L(x) é definida como:
 - o maior índice i tal que P[i] = x;
 - -1, caso este índice não exista;
- Exemplo: $\alpha = \{a, b, c, d\}$

P =

a	b	a	c	a	b
0	1	2	3	4	5

x	a	b	c	d
L(x)	4	5	3	-1

```
1. Boyer_Moore(T, n, P, m,  $\alpha$ , w)
2.   para k  $\leftarrow$  0 até w faça
3.     L[k]  $\leftarrow$  -1
4.   para k  $\leftarrow$  0 até m faça
5.     L[P[k]]  $\leftarrow$  k
6.   i  $\leftarrow$  m - 1
7.   j  $\leftarrow$  m - 1
8.   repita
9.     se T[i] == P[j] então
10.      se j == 0 então
11.        retorne i
12.      senão
13.        i  $\leftarrow$  i - 1
14.        j  $\leftarrow$  j - 1
15.      senão
16.        i  $\leftarrow$  i + m - min{j, 1 + L[T[i]]}
17.        j  $\leftarrow$  m - 1
18.   enquanto i > n - 1
19.   retorne -1
```

O(nm)

Exercício

1) Considerando a seguinte cadeia de DNA:

– T = “CTAATCGCTTAATCAAACGC”

Realize o passo a passo do algoritmo Boyer-Moore para verificar se o padrão P = “ATCA” ocorre em T.

– Construa o vetor L e ilustre todos os alinhamentos e comparações dos caracteres do padrão P em T.

Exercícios

Lista de Exercícios 06 – Busca de Padrões em Texto

<http://www.inf.puc-rio.br/~elima/paa/>



Leitura Complementar

- Halim e Halim. **Competitive Programming**, 3rd Edition, 2003.
- **Capítulo 6: String Processing**

