Busca por palavras em um texto - Algoritmos

Jeane Melo

Roteiro

- Introdução
- Algoritmo Naïve
- Algoritmo KMP
- Algoritmo Boyer-Moore
- Exemplos

Introdução

- Dados nem sempre se decompõem logicamente em registros independentes com pequenas partes identificáveis
- Este tipo de dados é caracterizado apenas pelo fato de que pode ser escrito como uma cadeia
- Uma cadeia é uma sequencia linear de caracteres, podendo ser muito longa

Introdução

- Cadeias podem ser centrais em sistemas de processamento de textos, recuperação de informação, estudo de sequências de DNA em biologia computacional, etc.
- Algoritmos eficientes são necessários para manipulá-los.

Introdução

- A maioria dos algoritmos para o problema de casamento de padrão pode ser facilmente estendida para encontrar todas as ocorrências do padrão no texto

- O problema de casamento de padrão pode ser visto também como um problema de busca com o padrão sendo a chave

O problema

- Dizemos que uma cadeia s *ocorre em* uma cadeia t se existe um índice k tal que
- s[0] == t[k], s[1] == t[k+1], . . . ,
  s[m-1] == t[k+m-1]
  - sendo m o comprimento de s e supondo que k+m-1 é menor que o comprimento de t.
- Dizemos também nesse caso que s *casa* (match) com t[k..k+m-1] ou que s casa com t a partir da posição k.

Exemplos

														a		e	n	s	e	a					
R	u	b	i	ã	o		f	i	t	a	v	a		a		e	n	s	e	a	d	a	,		-
														^											

														T	A	C	T	A							
G	T	A	G	T	A	T	A	T	A	T	A	T	A	T	A	C	T	A	C	T	A	G	T	A	G
														^											

O problema

- **PROBLEMA (STRING MATCHING):** Encontrar uma ocorrências de uma cadeia s em uma cadeia t.
- No contexto do problema, diz-se que s é uma *palavra* e t um *texto*. Assim, o problema é encontrar uma palavra num texto.
- O problema pode ter variações, como a de encontrar *todas* as ocorrências de uma palavra em um texto.

Algoritmo Naïve

Algoritmo Naïve

- O algoritmo mais óbvio de busca em cadeia, chamado algoritmo força-bruta ou algoritmo ingênuo (naïve)
- O pior caso de tempo de execução proporcional a MN
- N tamanho da sequencia
- M tamanho do padrão

Algoritmo Naïve

```
BruteForceMatch() {
   for (i=0; i<=n-m; i++) {
      j = 0;
      while (j<m && T[i+j]==P[j])
          j++;
      if (j == m) return i; // P encontrado em T[i]
   }
   return -1;  // P não foi encontrado
}
```

Exemplo

A B A C A A B A C C A B A C A B A A B B

1 2 3 4 5 6

A B A C A | B

7

A | B A C A B

8 9

A B | A C A B

10

A | B A C A B

11 Comparações

22 23 24 25 26 27

A B A C A B

Pior caso

- O pior caso ocorre quando, por exemplo, o padrão e o texto são os dois uma seqüência de zeros seguidos por um 1:
    - 00001 e 000000000000000001
- Ou seja, quando é preciso percorrer praticamente todo P várias vezes a cada posição de T, estando P no fim da cadeia

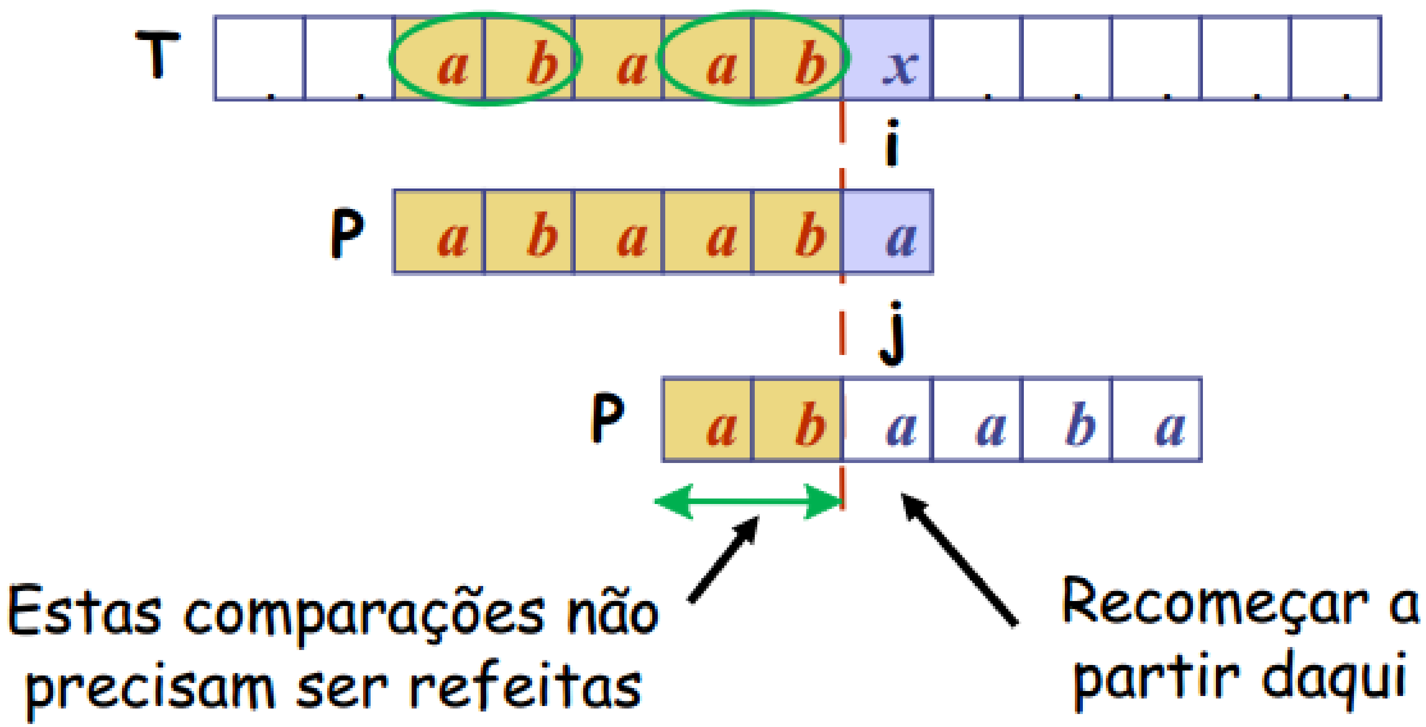
- Complexidade: m*n

Algoritmo KMP

Algoritmo KMP

- Knuth-Morris-Pratt
- 1970 – 1976
- Ideia
  - considerando o algoritmo “força bruta”, quando ocorre uma diferença entre T[i] e P[j], não seria possível fazer um deslocamento maior de P para a direita, evitando comparações redundantes?

Exemplo



T
a
b
a
a
b
x
i
P
a
b
a
a
b
a
j
P
a
b
a
a
b
a
Estas comparações não precisam ser refeitas
Recomeçar a partir daqui

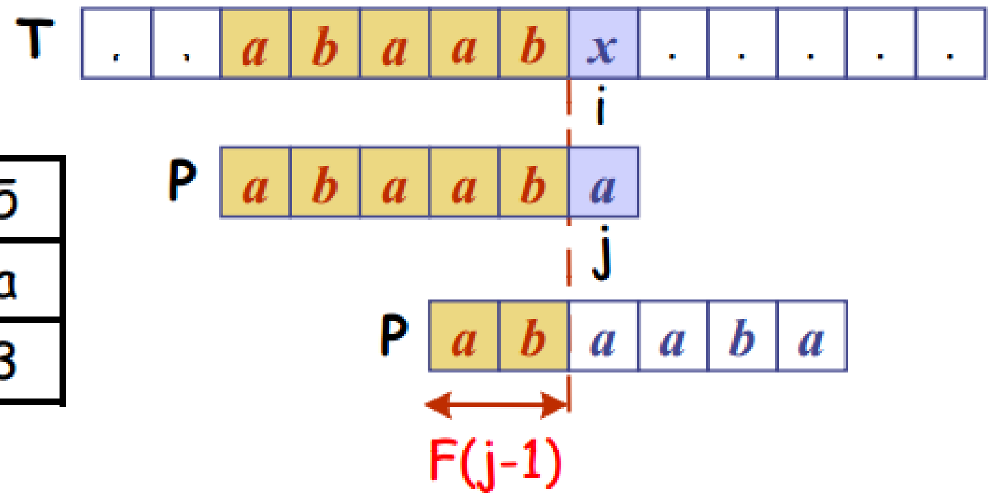
Função de falha

- Pré-processamento em P: determina se seus prefixos aparecem como subsequências dele mesmo.
- A função de falha F(k) será definida como o tamanho do maior prefixo de P[0 .. k] que é sufixo de P[1..k].

Função de falha

Exemplo:

k	0	1	2	3	4	5
P[k]	a	b	a	a	b	a
F(k)	0	0	1	1	2	3



Se P[j] ≠ T[i], então j receberá o valor F(j-1).

“maior prefixo de P[0 .. k] que é sufixo de P[1..k]”

Algoritmo KMP

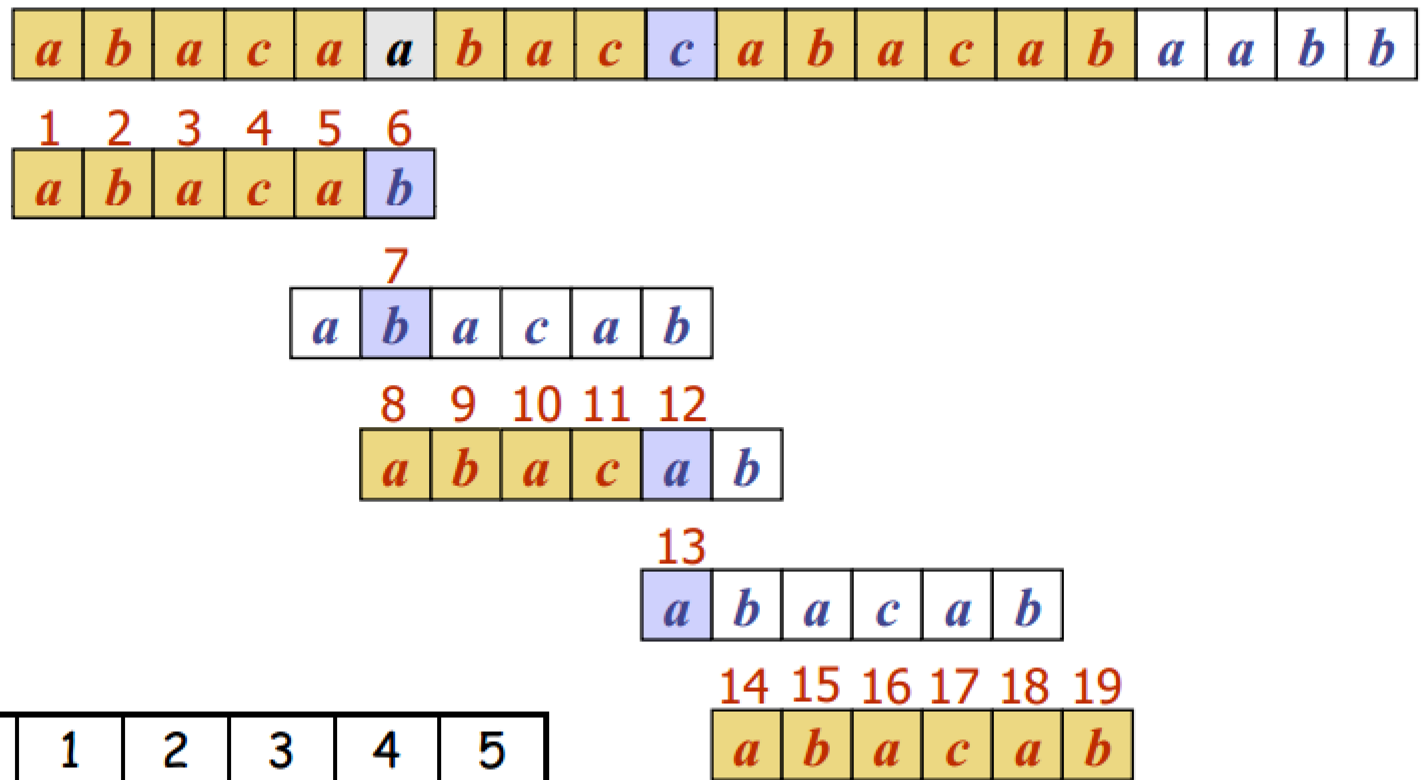
```
KMPMatch() {
    FailureFunction();      // Veremos que gasta tempo Θ(m)
    i = 0;
    j = 0;
    while (i < n)
        if (T[i] == P[j])
            if (j == m-1)
                return i-j;
            else
                { i++; j++; }
        else
            if (j != 0)
                j = F[j-1];
            else
                i++;
    return -1;
}
```

j é incrementado n vezes no máximo

Como é um decremento, será executado até n vezes

Tempo do laço `while`: $O(n)$

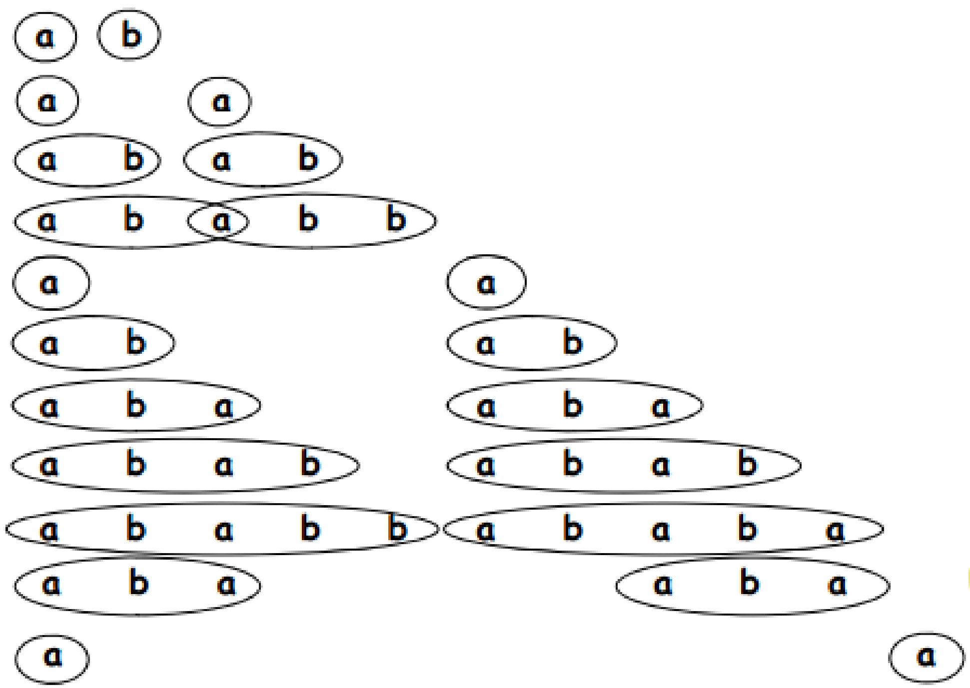
Exemplo



j	0	1	2	3	4	5
P[j]	a	b	a	c	a	b
F(j)	0	0	1	0	1	2

Cálculo da função de falha

j	0	1	2	3	4	5	6	7	8	9	10
P[j]	a	b	a	b	b	a	b	a	b	a	a
F[j]	0	0	1	2	0	1	2	3	4	3	1



Uso “recursivo” de F

```
FailureFunction() {
   F[0] = 0;
   j = 0;  // índice que percorre os prefixos
   i = 1;  // índice que percorre os sufixos
   while (i < m)
        if (P[i] == P[j])  // já combinaram j+1 caracteres
             F[i++] = ++j;
        else if (j == 0)
                  F[i++] = 0;
              else
                  j = F[j-1]; // uso "recursivo" de F
}
```

j é incrementado m-1 vezes no máximo

Como é um decremento, será executado até m-1 vezes

Tempo: $\Theta(m)$

Algoritmo Boyer-Moore

Algoritmo Boyer-Moore

- Pré-processamento da palavra
- Boyer e Moore (1976) descobriram uma maneira de acelerar o algoritmo trivial.
- O algoritmo de Boyer-Moore depende do conhecimento prévio do "alfabeto" do problema, ou seja, do conjunto de caracteres usado na palavra e no texto.
- Essa hipótese está automaticamente satisfeita no nosso caso: o alfabeto é o conjunto de todos os 256 caracteres.

Algoritmo Boyer-Moore

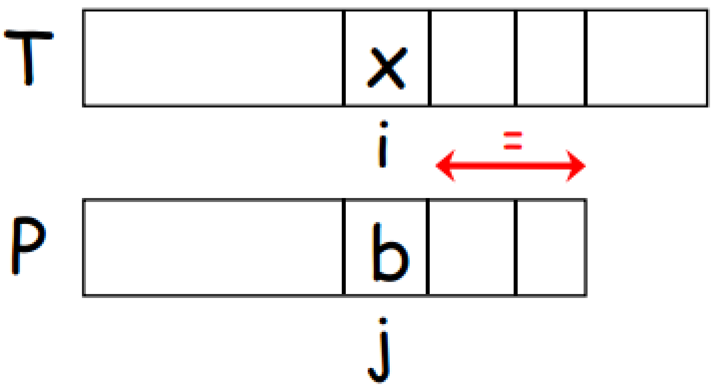
- Idéia:
  - Suponha que acabamos de comparar a cadeia s com a cadeia t+k.
  - A próxima comparação não precisa acontecer necessariamente entre p e t+k+1: podemos passar a tratar imediatamente de t+k+d
  - onde d é calculado *de modo que* t[k+1] *fique emparelhado com a última ocorrência do caractere* t[k+1] *em* p.

Algoritmo Boyer-Moore

- O algoritmo faz a varredura dos símbolos do padrão da direita para à esquerda (rightmost).
- O algoritmo utiliza duas funções pré-processadas para deslocar o padrão à direita.

Algoritmo Boyer-Moore

- Comparação feita de trás pra frente
- Quando encontra uma diferença, o algoritmo dá um salto, considerando as comparações já realizadas

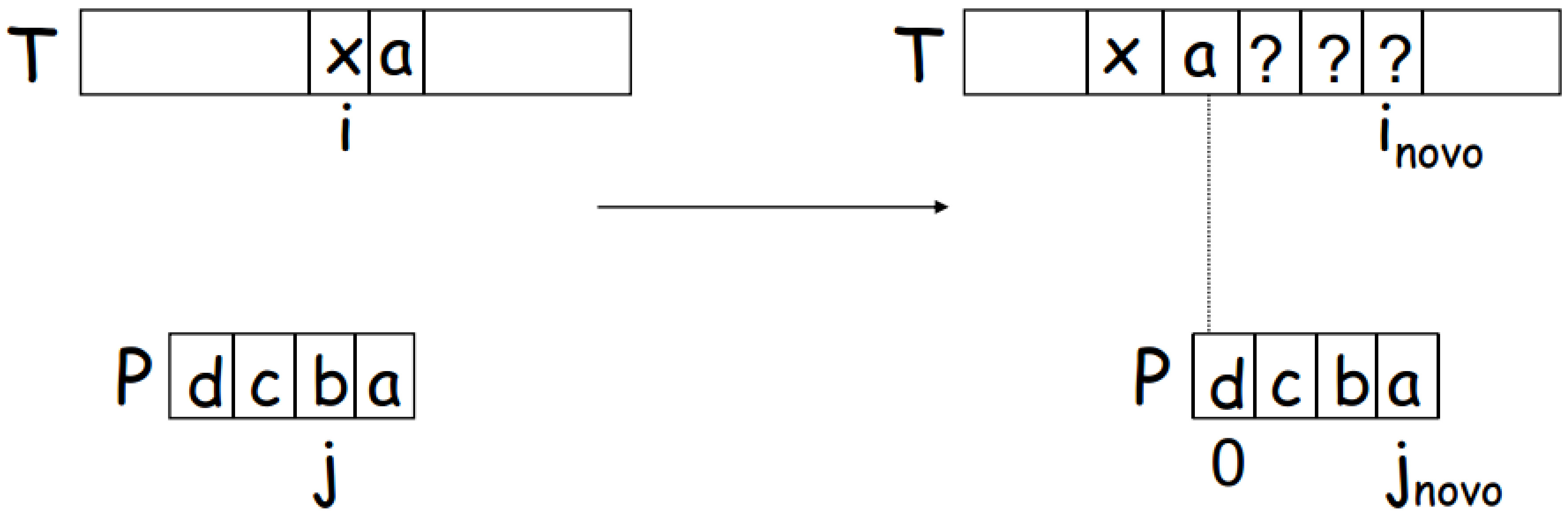


$T[i] \neq P[j]$

- Consideramos então três casos

Algoritmo Boyer-Moore – Caso 1

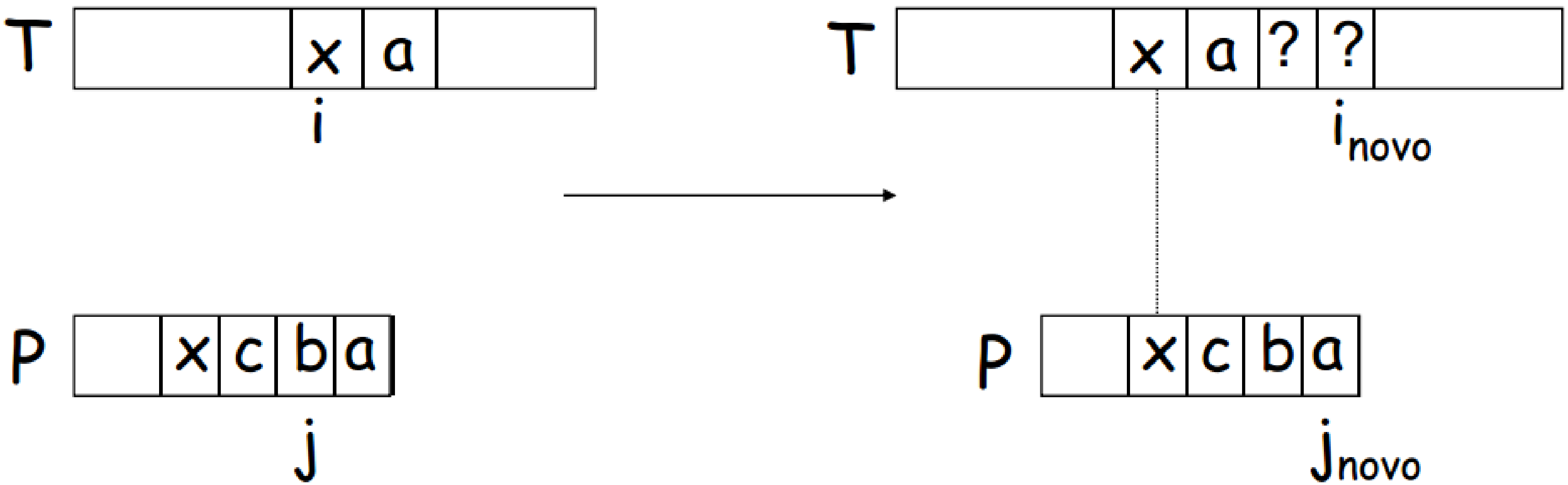
- P não contém x.



- Deslocar P para a direita, alinhando P[0] com T[i+1].

Algoritmo Boyer-Moore – Caso 2

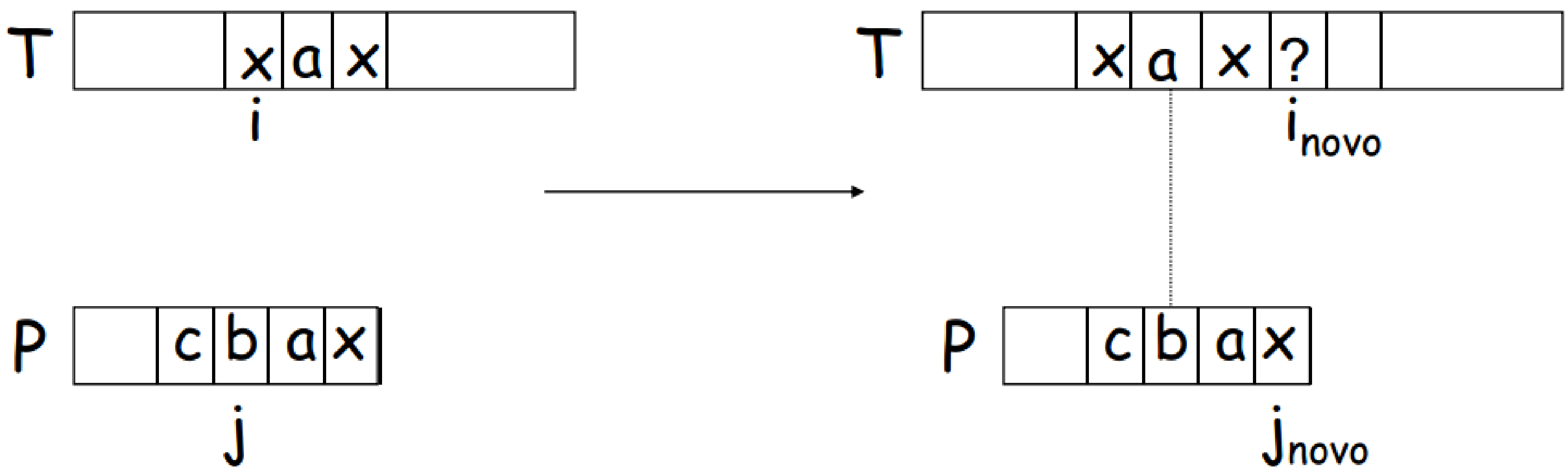
- A última ocorrência de x em P está algum índice menor do que j.



- Deslocar P para a direita, até que a última ocorrência de x fique alinhada com T[i].

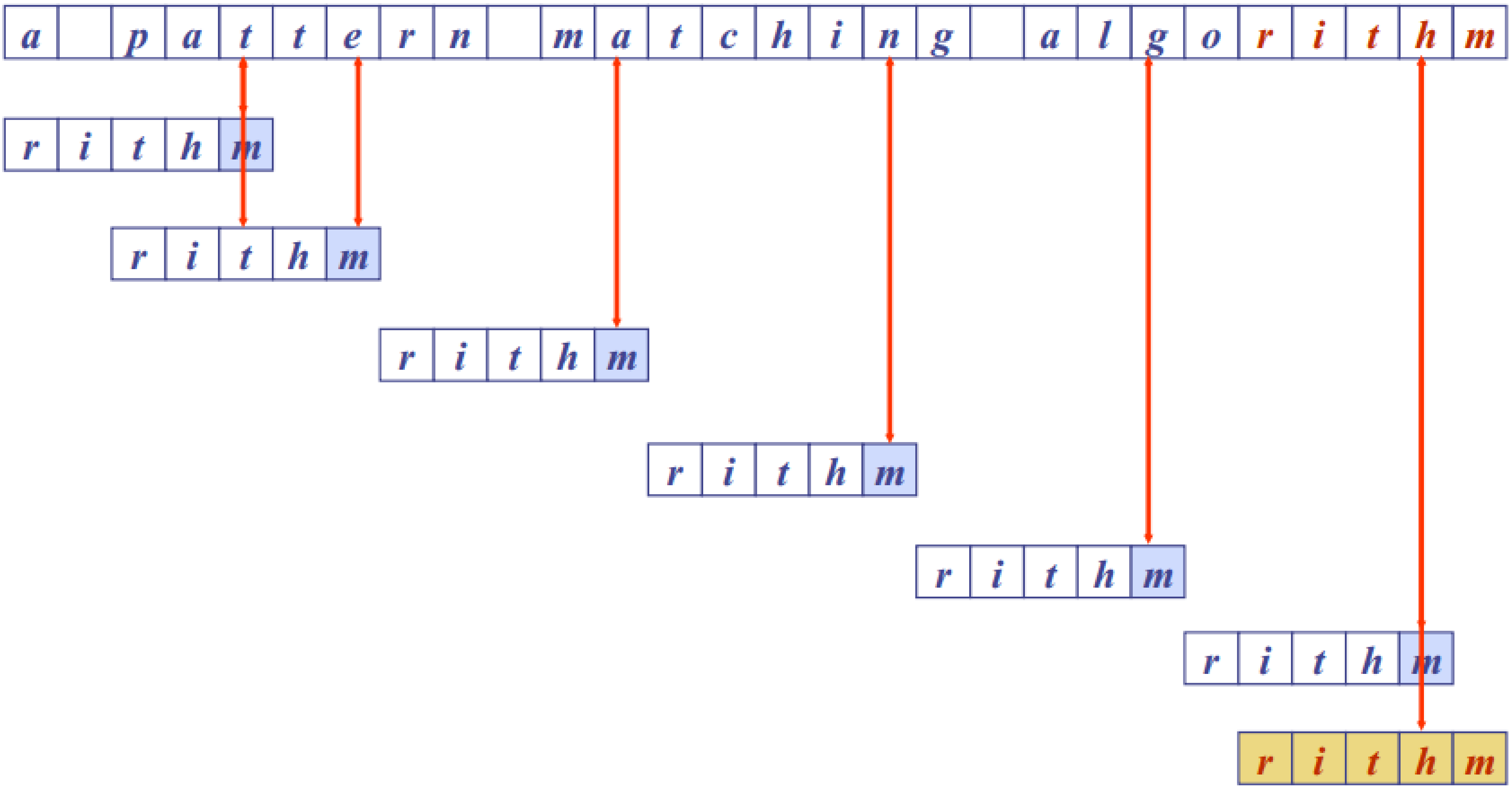
Algoritmo Boyer-Moore – Caso 3

- A última ocorrência de x em P está em algum índice maior do que j.

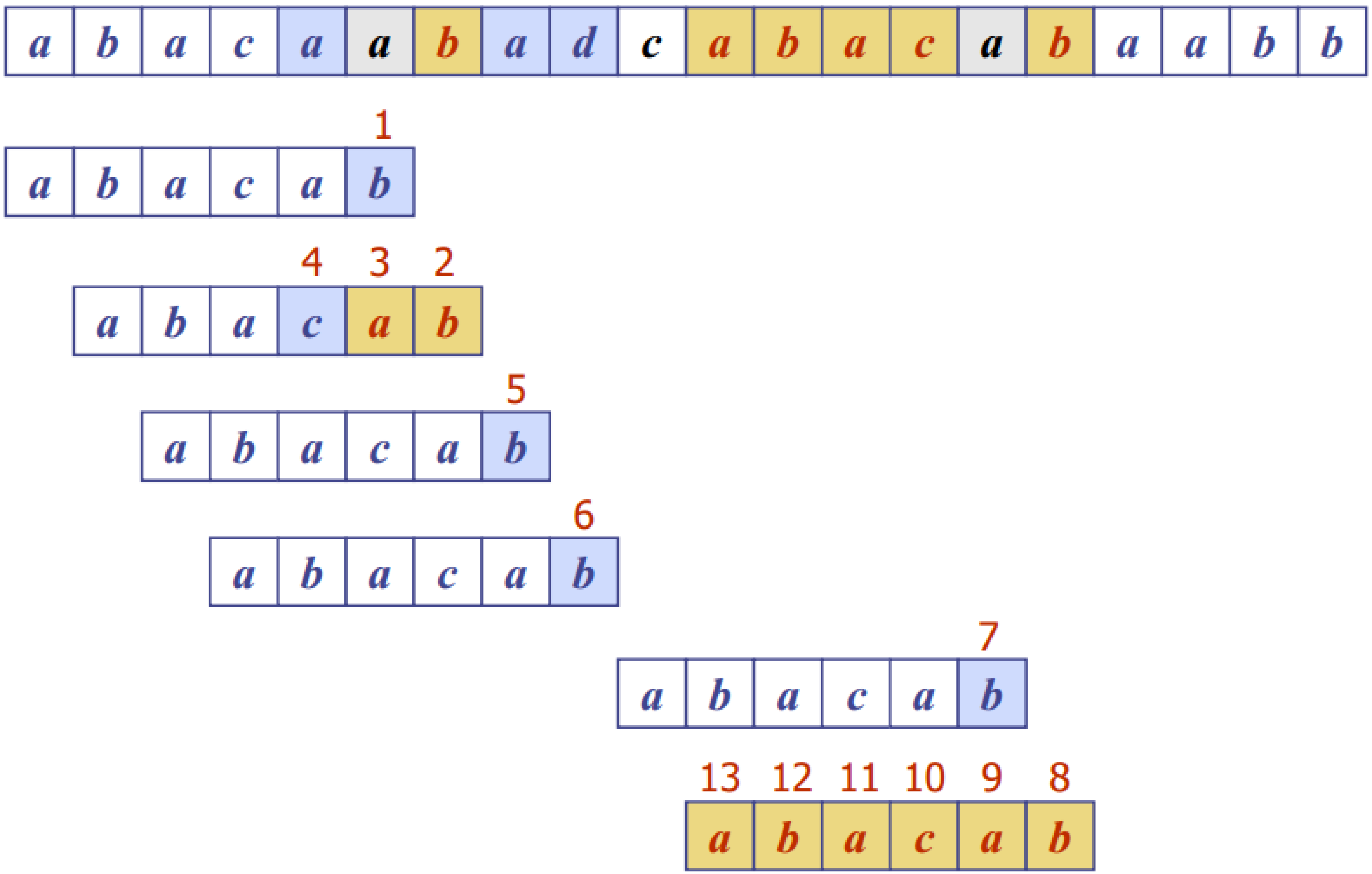


- Deslocar P apenas uma posição para a direita.

Exemplo 1


a pattern matching algorithm
rithm
rithm
rithm
rithm
rithm
rithm
rithm

Exemplo 2


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

Boyer-Moore – Pré-processamento

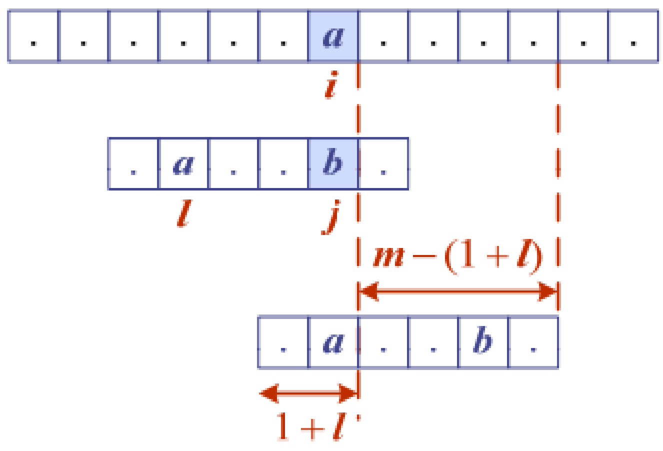
- Através de um pré-processamento, o algoritmo de Boyer-Moore calcula uma função L: $\Sigma \rightarrow \mathbf{I}$, onde L(x) é definida como:
  - o maior índice i tal que P[i] = x;
  - -1, caso este índice não exista.
- Exemplo: $\Sigma = \{a, b, c, d\}$

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

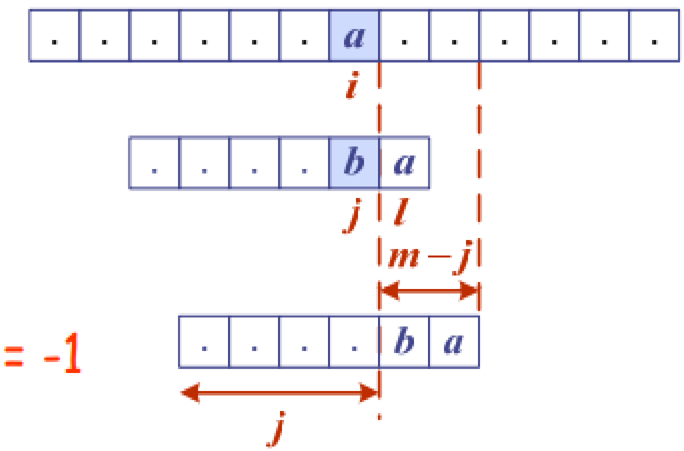
x	a	b	c	d
L(x)	4	5	3	-1

```
BoyerMooreMatch() {
   for (k=0; k<|Σ|; k++)
        L[k] = -1;
   for (k=0; k<m; k++)
        L[P[k]] = k;
   i = m-1;
   j = m-1;
   repeat
     if (T[i] == P[j])
        if (j == 0) return i;
        else { i--; j--; }
     else {
        l = L[T[i]];
        i += m - min{j, 1+l};
        j = m-1;
     }
   until (i > n-1);
   return -1;
}
```

Caso 2



Caso 3



Caso 1: `i += m;` pois *l* = -1

Referências

- Página Paulo Feofillof: http://www.ime.usp.br/~pf/algoritmos/aulas/strma.html
- http://www.ufjf.br/jairo_souza/files/2009/12/7-Strings-Casamento-de-padr%C3%B5es.pdf (figuras e textos)
- http://www.comp.ita.br/~alonso/ensino/CT234/CT234-Cap07.pdf (figuras e textos)
- Cormen – cap 32