AULA 22

Busca de palavras (string matching)

PF 13

http://www.ime.usp.br/~pf/algoritmos/aulas/strma.html

Busca de palavras em um texto

Dizemos que um vetor $p[1\,..\,m]$ **ocorre em** um vetor $t[1\,..\,n]$ se

$$p[1\,..\,m] = t[s+1\,..\,s+m]$$

para algum s em $[0\,..\,n-m]$.

Exemplo:

	1	2	3	4	5	6	7	8	9	10
t	x	c	b	a	b	b	c	b	a	x

	1	2	3	4
p	b	c	b	a

$p[1\,..\,4]$ ocorre em $t[1\,..\,10]$ com deslocamento 5.

Busca de palavras em um texto

Problema: Dados $\mathtt{p}[1\,..\,\mathtt{m}]$ e $\mathtt{t}[1\,..\,\mathtt{n}]$,
encontrar o número de ocorrências de $\mathtt{p}$ em $\mathtt{t}$.

Exemplo: Para $\mathtt{n} = 10$, $\mathtt{m} = 4$, e

	1	2	3	4	5	6	7	8	9	10
t	b	b	a	b	a	b	a	c	b	a

	1	2	3	4
p	b	a	b	a

p ocorre 2 vezes em t.

Algoritmo trivial

p = a b a b b a b a b b a

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	
	a	b	a	a	b	a	b	a	b	b	a	b	a	b	a	b	b	a	b	a	b	b	a	t
1	a	b	a	b	b	a	b	a	b	b	a													

Algoritmo trivial

p = a b a b b a b a b b a

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	
	a	b	a	a	b	a	b	a	b	b	a	b	a	b	a	b	b	a	b	a	b	b	a	t
1	a	b	a	b	b	a	b	a	b	b	a													
2		a	b	a	b	b	a	b	a	b	b	a												

Algoritmo trivial

p = a b a b b a b a b b a

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	
	a	b	a	a	b	a	b	a	b	b	a	b	a	b	a	b	b	a	b	a	b	b	a	t
1	a	b	a	b	b	a	b	a	b	b	a													
2		a	b	a	b	b	a	b	a	b	b	a												
3			a	b	a	b	b	a	b	a	b	b	a											

Algoritmo trivial

p = a b a b b a b a b b a

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	
	a	b	a	a	b	a	b	a	b	b	a	b	a	b	a	b	b	a	b	a	b	b	a	t
1	a	b	a	b	b	a	b	a	b	b	a													
2		a	b	a	b	b	a	b	a	b	b	a												
3			a	b	a	b	b	a	b	a	b	b	a											
4				a	b	a	b	b	a	b	a	b	b	a										

Algoritmo trivial

p = a b a b b a b a b b a

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	
	a	b	a	a	b	a	b	a	b	b	a	b	a	b	a	b	b	a	b	a	b	b	a	t
1	a	b	a	b	b	a	b	a	b	b	a													
2		a	b	a	b	b	a	b	a	b	b	a												
3			a	b	a	b	b	a	b	a	b	b	a											
4				a	b	a	b	b	a	b	a	b	b	a										
5					a	b	a	b	b	a	b	a	b	b	a									
6						a	b	a	b	b	a	b	a	b	b	a								
7							a	b	a	b	b	a	b	a	b	b	a							
8								a	b	a	b	b	a	b	a	b	b	a						
9									a	b	a	b	b	a	b	a	b	b	a					
10										a	b	a	b	b	a	b	a	b	b	a				
11											a	b	a	b	b	a	b	a	b	b	a			
12												a	b	a	b	b	a	b	a	b	b	a		
13													a	b	a	b	b	a	b	a	b	b	a	

Algoritmo trivial

Devolve o número de ocorrências de p em t.

```
int trivial (unsigned char p[], int m,
             unsigned char t[], int n) {
    int r, k, ocorrs = 0;
1   for (k = 1; k <= n-m+1; k++) {
2      r = 0;
3      while (r < m && p[1+r] == t[k+r])
4         r += 1;
5      if (r == m) ocorrs += 1;
    }
6   return ocorrs;
 }
```

Algoritmo trivial

```
int trivial (unsigned char p[], int m,
             unsigned char t[], int n) {
    int r, k, ocorrs = 0;
1   for (k = 1; k <= n-m+1; k++) {
2      r = 0;
3      while (r < m && p[1+r] == t[k+r])
4         r += 1;
5      if (r == m) ocorrs += 1;
    }
6   return ocorrs;
 }
```

Relação invariante: no início da linha 3 vale que

(i0) $\mathtt{p}[1\,..\,1+\mathtt{r}-1] = \mathtt{t}[\mathtt{k}\,..\,\mathtt{k}+\mathtt{r}-1]$

Algoritmo trivial

```
int trivial (unsigned char p[], int m,
             unsigned char t[], int n) {
    int r, k, ocorrs = 0;
1   for (k = 1; k <= n-m+1; k++) {
2      r = 0;
3      while (r < m && p[1+r] == t[k+r])
4         r += 1;
5      if (r == m) ocorrs += 1;
    }
6   return ocorrs;
 }
```

Consumo de tempo?

Consumo de tempo

Consumo de tempo da função `trivial`, versão direita para a esquerda.

linha	todas as execuções da linha
1	$= \mathtt{n} - \mathtt{m} + 2$
2	$= \mathtt{n} - \mathtt{m} + 1$
3	$\leq (\mathtt{n} - \mathtt{m} + 1)(\mathtt{m} + 1)$
4	$\leq (\mathtt{n} - \mathtt{m} + 1)\mathtt{m}$
5	$= \mathtt{n} - \mathtt{m} + 1$
6	$= 1$
total	$< 3\,(\mathtt{n} - \mathtt{m} + 2) + 2\,(\mathtt{n} - \mathtt{m} + 1)(\mathtt{m} + 1)$
	$= \mathrm{O}((\mathtt{n} - \mathtt{m} + 1)\mathtt{m})$

Pior caso

p = a a a a a a a a a a a

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	
	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	t
1	a	a	a	a	a	a	a	a	a	a	a													
2		a	a	a	a	a	a	a	a	a	a	a												
3			a	a	a	a	a	a	a	a	a	a	a											
4				a	a	a	a	a	a	a	a	a	a	a										
5					a	a	a	a	a	a	a	a	a	a	a									
6						a	a	a	a	a	a	a	a	a	a	a								
7							a	a	a	a	a	a	a	a	a	a	a							
8								a	a	a	a	a	a	a	a	a	a	a						
9									a	a	a	a	a	a	a	a	a	a	a					
10										a	a	a	a	a	a	a	a	a	a	a				
11											a	a	a	a	a	a	a	a	a	a	a			
12												a	a	a	a	a	a	a	a	a	a	a		
13													a	a	a	a	a	a	a	a	a	a	a	

Melhor caso

p = b a a a a a a a a a a

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	
	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	t
1	b	a	a	a	a	a	a	a	a	a	a													
2		b	a	a	a	a	a	a	a	a	a	a												
3			b	a	a	a	a	a	a	a	a	a	a											
4				b	a	a	a	a	a	a	a	a	a	a										
5					b	a	a	a	a	a	a	a	a	a	a									
6						b	a	a	a	a	a	a	a	a	a	a								
7							b	a	a	a	a	a	a	a	a	a	a							
8								b	a	a	a	a	a	a	a	a	a	a						
9									b	a	a	a	a	a	a	a	a	a	a					
10										b	a	a	a	a	a	a	a	a	a	a				
11											b	a	a	a	a	a	a	a	a	a	a			
12												b	a	a	a	a	a	a	a	a	a	a		
13													b	a	a	a	a	a	a	a	a	a	a	a

Conclusões

O consumo de tempo da função `trivial` no pior caso é $\mathrm{O}((\mathtt{n}-\mathtt{m}+1)\mathtt{m})$.

O consumo de tempo da função `trivial` no melhor caso é $\mathrm{O}(\mathtt{n}-\mathtt{m}+1)$.

Isto significa que no pior caso o consumo de tempo é essencialmente proporcional a $\mathtt{mn}$.

Algoritmo trivial: direita para esquerda

p = a b a b b a b a b b a

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	
	a	b	a	a	b	a	b	a	b	b	a	b	a	b	a	b	b	a	b	a	b	b	a	t
1	a	b	a	b	b	a	b	a	b	b	a													

Algoritmo trivial: direita para esquerda

p = a b a b b a b a b b a

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	
	a	b	a	a	b	a	b	a	b	b	a	b	a	b	a	b	b	a	b	a	b	b	a	t
1	a	b	a	b	b	a	b	a	b	b	a													
2		a	b	a	b	b	a	b	a	b	b	a												

Algoritmo trivial: direita para esquerda

p = a b a b b a b a b b a

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	
	a	b	a	a	b	a	b	a	b	b	a	b	a	b	a	b	b	a	b	a	b	b	a	t
1	a	b	a	b	b	a	b	a	b	b	a													
2		a	b	a	b	b	a	b	a	b	b	a												
3			a	b	a	b	b	a	b	a	b	b	a											

Algoritmo trivial: direita para esquerda

p = a b a b b a b a b b a

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	
	a	b	a	a	b	a	b	a	b	b	a	b	a	b	a	b	b	a	b	a	b	b	a	t
1	a	b	a	b	b	a	b	a	b	b	a													
2		a	b	a	b	b	a	b	a	b	b	a												
3			a	b	a	b	b	a	b	a	b	b	a											
4				a	b	a	b	b	a	b	a	b	b	a										

Algoritmo trivial: direita para esquerda

p = a b a b b a b a b b a

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	
	a	b	a	a	b	a	b	a	b	b	a	b	a	b	a	b	b	a	b	a	b	b	a	t
1	a	b	a	b	b	a	b	a	b	b	a													
2		a	b	a	b	b	a	b	a	b	b	a												
3			a	b	a	b	b	a	b	a	b	b	a											
4				a	b	a	b	b	a	b	a	b	b	a										
5					a	b	a	b	b	a	b	a	b	b	a									

Algoritmo trivial: direita para esquerda

p = a b a b b a b a b b a

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	
	a	b	a	a	b	a	b	a	b	b	a	b	a	b	a	b	b	a	b	a	b	b	a	t
1	a	b	a	b	b	a	b	a	b	b	a													
2		a	b	a	b	b	a	b	a	b	b	a												
3			a	b	a	b	b	a	b	a	b	b	a											
4				a	b	a	b	b	a	b	a	b	b	a										
5					a	b	a	b	b	a	b	a	b	b	a									
6						a	b	a	b	b	a	b	a	b	b	a								

Algoritmo trivial: direita para esquerda

p = a b a b b a b a b b a

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	
	a	b	a	a	b	a	b	a	b	b	a	b	a	b	a	b	b	a	b	a	b	b	a	t
1	a	b	a	b	b	a	b	a	b	b	a													
2		a	b	a	b	b	a	b	a	b	b	a												
3			a	b	a	b	b	a	b	a	b	b	a											
4				a	b	a	b	b	a	b	a	b	b	a										
5					a	b	a	b	b	a	b	a	b	b	a									
6						a	b	a	b	b	a	b	a	b	b	a								
7							a	b	a	b	b	a	b	a	b	b	a							

Algoritmo trivial: direita para esquerda

p = a b a b b a b a b b a

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	
	a	b	a	a	b	a	b	a	b	b	a	b	a	b	a	b	b	a	b	a	b	b	a	t
1	a	b	a	b	b	a	b	a	b	b	a													
2		a	b	a	b	b	a	b	a	b	b	a												
3			a	b	a	b	b	a	b	a	b	b	a											
4				a	b	a	b	b	a	b	a	b	b	a										
5					a	b	a	b	b	a	b	a	b	b	a									
6						a	b	a	b	b	a	b	a	b	b	a								
7							a	b	a	b	b	a	b	a	b	b	a							
8								a	b	a	b	b	a	b	a	b	b	a						
9									a	b	a	b	b	a	b	a	b	b	a					
10										a	b	a	b	b	a	b	a	b	b	a				
11											a	b	a	b	b	a	b	a	b	b	a			
12												a	b	a	b	b	a	b	a	b	b	a		
13													a	b	a	b	b	a	b	a	b	b	a	

Algoritmo trivial: direita para esquerda

Devolve o número de ocorrências de p em t.

```
int trivial (unsigned char p[], int m,
             unsigned char t[], int n) {
    int r, k, ocorrs = 0;
1   for (k = m; k <= n; k++) {
2      r = 0;
3      while (r < m && p[m-r] == t[k-r])
4         r += 1;
5      if (r == m) ocorrs += 1;
    }
6   return ocorrs;
 }
```

Algoritmo trivial: direita para esquerda

```
int trivial (unsigned char p[], int m,
             unsigned char t[], int n) {
    int r, k, ocorrs = 0;
1   for (k = m; k <= n; k++) {
2      r = 0;
3      while (r < m && p[m-r] == t[k-r])
4         r += 1;
5      if (r == m) ocorrs += 1;
    }
6   return ocorrs;
 }
```

Relação invariante: no início da linha 3 vale que

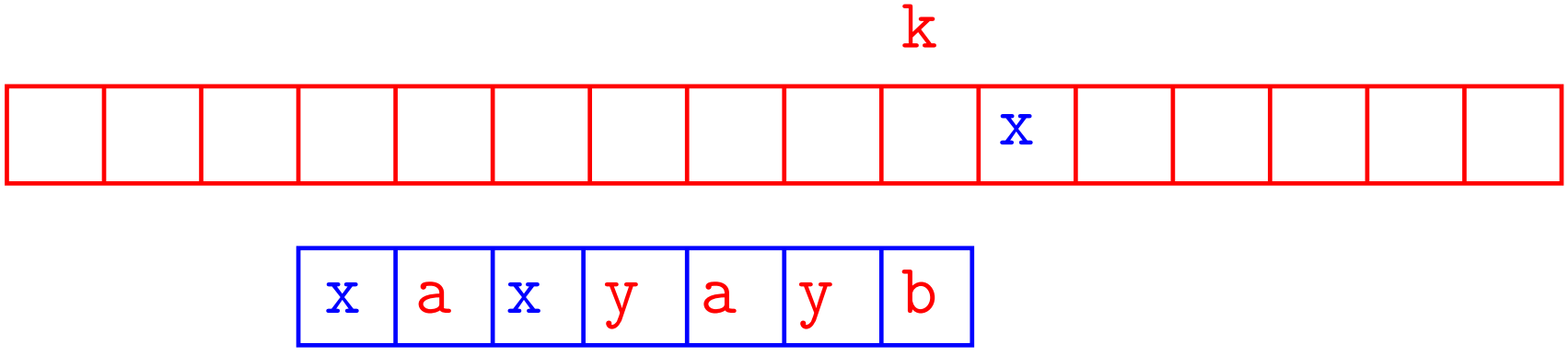
(i0) $p[m-r+1\,..\,m] = t[k-r+1\,..\,k]$

Algoritmo trivial: direita para esquerda

```
int trivial (unsigned char p[], int m,
             unsigned char t[], int n) {
    int r, k, ocorrs;
 3  ocorrs = 0; k = m;
 4  while (k <= n) {
 5     r = 0;
 6     while (r < m && p[m-r] == t[k-r])
 7        r += 1;
 8     if (r == m) ocorrs += 1;
 9     k += 1;                    <<< atenção aqui!
    }
11  return ocorrs;
  }
```

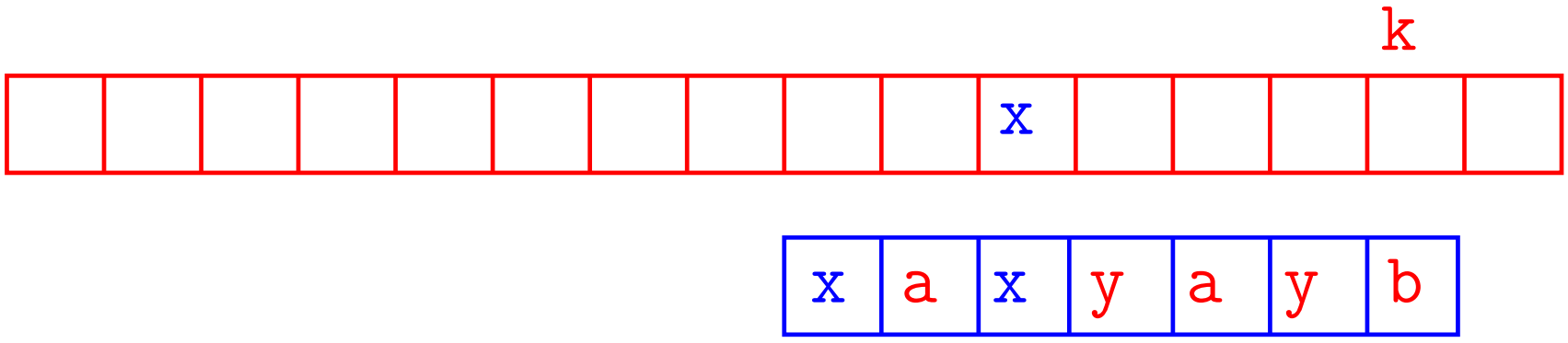
Primeiro algoritmo de Boyer-Moore

O primeiro algoritmo de R.S. Boyer e J.S. Moore (1977) é baseado na seguinte heurística.



Primeiro algoritmo de Boyer-Moore

O primeiro algoritmo de R.S. Boyer e J.S. Moore (1977) é baseado na seguinte heurística.



Boyer-Moore

p = a n d a n d o

```
  1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
  a s   a n d o r i n h a s   a n d a m   a n d a n d o   a l t o t
1 a n d a n d o
```

Boyer-Moore

p = a n d a n d o

```
  1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
  a s   a n d o r i n h a s   a n d a m   a n d a n d o   a l t o  t
1 a n d a n d o
2           a n d a n d o
```

Boyer-Moore

p = a n d a n d o

```
  1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
  a s   a n d o r i n h a s   a n d a m   a n d a n d o   a l t o  t
1 a n d a n d o
2             a n d a n d o
3                 a n d a n d o
```

Boyer-Moore

p = a n d a n d o

```
  1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
  a s   a n d o r i n h a s   a n d a m   a n d a n d o   a l t o  t
1 a n d a n d o
2             a n d a n d o
3                 a n d a n d o
4                                     a n d a n d o
```

Boyer-Moore

p = a n d a n d o

```
  1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
  a s   a n d o r i n h a s   a n d a m   a n d a n d o   a l t o  t
1 a n d a n d o
2             a n d a n d o
3                 a n d a n d o
4                                     a n d a n d o
5                                       a n d a n d o
```

Boyer-Moore

p = a n d a n d o

```
  1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
  a s   a n d o r i n h a s   a n d a m   a n d a n d o   a l t o  t
1 a n d a n d o
2             a n d a n d o
3                 a n d a n d o
4                                     a n d a n d o
5                                       a n d a n d o
6                                                     a n d a . . .
```

Boyer-Moore

p = a b a b b a b a b b a

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	
	a	b	a	a	b	a	b	a	b	b	a	b	a	b	a	b	b	a	b	a	b	b	a	t
1	a	b	a	b	b	a	b	a	b	b	a													

Boyer-Moore

p = a b a b b a b a b b a

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	
	a	b	a	a	b	a	b	a	b	b	a	b	a	b	a	b	b	a	b	a	b	b	a	t
1	a	b	a	b	b	a	b	a	b	b	a													
2			a	b	a	b	b	a	b	a	b	b	a											

Boyer-Moore

p = a b a b b a b a b b a

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	
	a	b	a	a	b	a	b	a	b	b	a	b	a	b	a	b	b	a	b	a	b	b	a	t
1	a	b	a	b	b	a	b	a	b	b	a													
2			a	b	a	b	b	a	b	a	b	b	a											
3					a	b	a	b	b	a	b	a	b	b	a									

Boyer-Moore

p = a b a b b a b a b b a

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	
	a	b	a	a	b	a	b	a	b	b	a	b	a	b	a	b	b	a	b	a	b	b	a	t
1	a	b	a	b	b	a	b	a	b	b	a													
2			a	b	a	b	b	a	b	a	b	b	a											
3					a	b	a	b	b	a	b	a	b	b	a									
4							a	b	a	b	b	a	b	a	b	b	a							

Boyer-Moore

p = a b a b b a b a b b a

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	
	a	b	a	a	b	a	b	a	b	b	a	b	a	b	a	b	b	a	b	a	b	b	a	t
1	a	b	a	b	b	a	b	a	b	b	a													
2			a	b	a	b	b	a	b	a	b	b	a											
3					a	b	a	b	b	a	b	a	b	b	a									
4							a	b	a	b	b	a	b	a	b	b	a							
5								a	b	a	b	b	a	b	a	b	b	a						

Boyer-Moore

p = a b a b b a b a b b a

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	
	a	b	a	a	b	a	b	a	b	b	a	b	a	b	a	b	b	a	b	a	b	b	a	t
1	a	b	a	b	b	a	b	a	b	b	a													
2			a	b	a	b	b	a	b	a	b	b	a											
3					a	b	a	b	b	a	b	a	b	b	a									
4							a	b	a	b	b	a	b	a	b	b	a							
5								a	b	a	b	b	a	b	a	b	b	a						
6										a	b	a	b	b	a	b	a	b	b	a				

Boyer-Moore

p = a b a b b a b a b b a

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	
	a	b	a	a	b	a	b	a	b	b	a	b	a	b	a	b	b	a	b	a	b	b	a	t
1	a	b	a	b	b	a	b	a	b	b	a													
2			a	b	a	b	b	a	b	a	b	b	a											
3					a	b	a	b	b	a	b	a	b	b	a									
4							a	b	a	b	b	a	b	a	b	b	a							
5								a	b	a	b	b	a	b	a	b	b	a						
6										a	b	a	b	b	a	b	a	b	b	a				
7												a	b	a	b	b	a	b	a	b	b	a		

Boyer-Moore

p = a b a b b a b a b b a

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	
	a	b	a	a	b	a	b	a	b	b	a	b	a	b	a	b	b	a	b	a	b	b	a	t
1	a	b	a	b	b	a	b	a	b	b	a													
2			a	b	a	b	b	a	b	a	b	b	a											
3					a	b	a	b	b	a	b	a	b	b	a									
4							a	b	a	b	b	a	b	a	b	b	a							
5								a	b	a	b	b	a	b	a	b	b	a						
6										a	b	a	b	b	a	b	a	b	b	a				
7												a	b	a	b	b	a	b	a	b	b	a		
8													a	b	a	b	b	a	b	a	b	b	a	

Primeiro algoritmo de Boyer-Moore

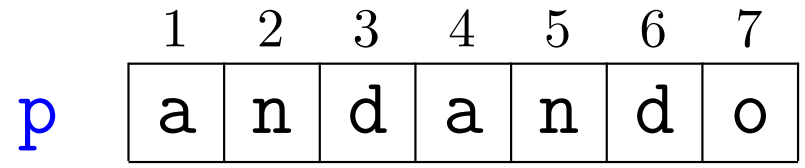
Ideia ("*bad-character heuristic*"): calcular um deslocamento de modo que `t[k+1]` fique emparelhado com a última ocorrência do caractere `t[k+1]` em `p`.

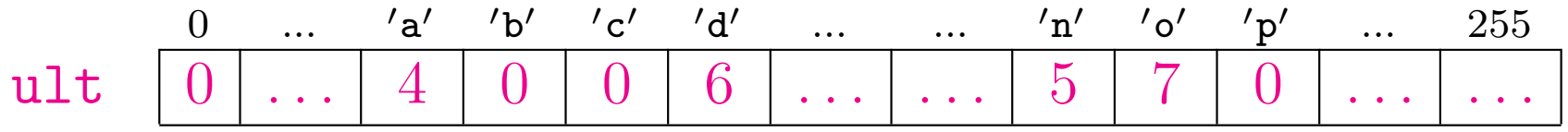
Suponha que o conjunto a que pertencem todos os elementos de `p` e de `t` é conhecido de antemão. Este conjunto é o **alfabeto** do problema.

Suponha que o alfabeto é o conjunto de todos os 256 caracteres.

Primeiro algoritmo de Boyer-Moore

Para implementar essa ideia, fazemos um pré-processamento de p, determinando para cada símbolo x do alfabeto a posição de sua última ocorrência em p.





Primeiro algoritmo de Boyer-Moore

Recebe vetores p[1 . . m] e t[1 . . n] de caracteres, com m ≥ 1 e n ≥ 0, e devolve o número de ocorrências de p em t.

```
int BoyerMoore (unsigned char p[], int m,
                unsigned char t[], int n) {
    int ult[256];
    int i, r, k, ocorrs;

    /* pre-processamento da palavra p */
1   for (i=0; i < 256; i++) ult[i] = 0;
```

Primeiro algoritmo de Boyer-Moore

Recebe vetores p[1 . . m] e t[1 . . n] de caracteres, com m ≥ 1 e n ≥ 0, e devolve o número de ocorrências de p em t.

```
int BoyerMoore (unsigned char p[], int m,
                unsigned char t[], int n) {
    int ult[256];
    int i, r, k, ocorrs;

    /* pre-processamento da palavra p */
1   for (i=0; i < 256; i++) ult[i] = 0;
2   for (i=1; i <= m; i++) ult[p[i]] = i;
```

Primeiro algoritmo de Boyer-Moore

```
   /* busca da palavra p no texto t */
 3   ocorrs = 0; k = m;
 4   while (k <= n) {
 5      r = 0;
 6      while (r < m && p[m-r] == t[k-r])
 7         r += 1;
 8      if (r == m) ocorrs += 1;
 9      if (k == n) k += 1;          /* cai fora */
10      else k += m - ult[t[k+1]] + 1;
    }
11   return ocorrs;
  }
```

Pior caso

p = a a a a a a a a a a a

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	
	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	t
1	a	a	a	a	a	a	a	a	a	a	a													
2		a	a	a	a	a	a	a	a	a	a	a												
3			a	a	a	a	a	a	a	a	a	a	a											
4				a	a	a	a	a	a	a	a	a	a	a										
5					a	a	a	a	a	a	a	a	a	a	a									
6						a	a	a	a	a	a	a	a	a	a	a								
7							a	a	a	a	a	a	a	a	a	a	a							
8								a	a	a	a	a	a	a	a	a	a	a						
9									a	a	a	a	a	a	a	a	a	a	a					
10										a	a	a	a	a	a	a	a	a	a	a				
11											a	a	a	a	a	a	a	a	a	a	a			
12												a	a	a	a	a	a	a	a	a	a	a		
13													a	a	a	a	a	a	a	a	a	a	a	

Melhor caso

p = a a a a b

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	
?	?	?	?	a	c	?	?	?	?	a	c	?	?	?	?	a	c	?	?	?	?	a	t
1	a	a	a	a	b																		

Melhor caso

p = a a a a b

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
	?	?	?	?	a	c	?	?	?	?	a	c	?	?	?	?	a	c	?	?	?	?	a	t
1	a	a	a	a	b																		
2							a	a	a	a	b												

Melhor caso

p = a a a a b

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
	?	?	?	?	a	c	?	?	?	?	a	c	?	?	?	?	a	c	?	?	?	?	a	t
1	a	a	a	a	b																		
2							a	a	a	a	b												
3													a	a	a	a	b						

Melhor caso

p = a a a a b

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
	?	?	?	?	a	c	?	?	?	?	a	c	?	?	?	?	a	c	?	?	?	?	a	t
1	a	a	a	a	b																		
2							a	a	a	a	b												
3													a	a	a	a	b						
4																			a	a	a	a	b

Conclusões

O consumo de tempo da função `BoyerMoore` no pior caso é $O((n - m + 1)m)$.

O consumo de tempo da função `BoyerMoore` no melhor caso é $O(n/m)$.

Isto significa que no pior caso o consumo de tempo é essencialmente proporcional a mn e no melhor caso o algoritmo é **sublinear**.

Aula que vem

Na aula que vem,
veremos uma segunda heurística de Boyer-Moore
e a versão mais conhecida do algoritmo.

Podemos conversar sobre o EP5.