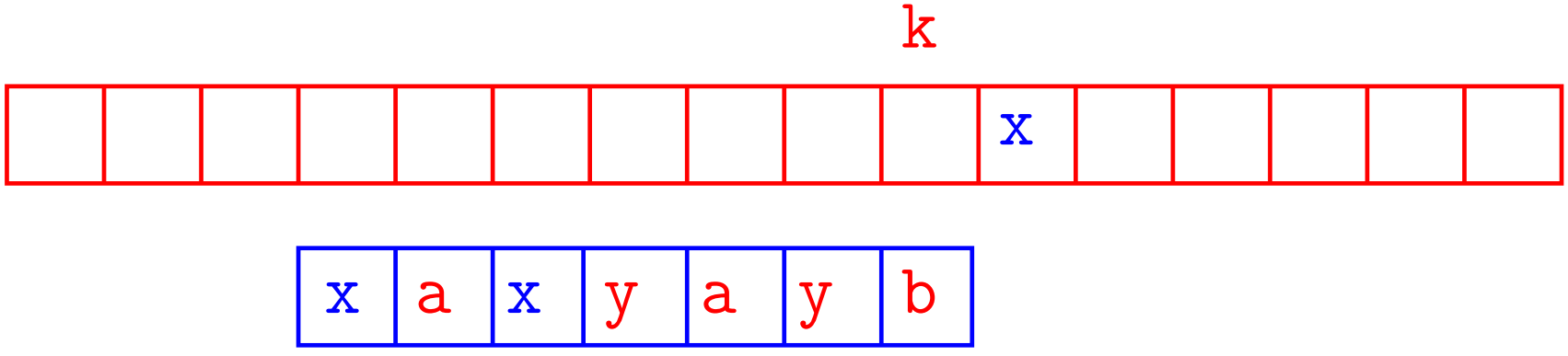
Melhores momentos

AULA 23

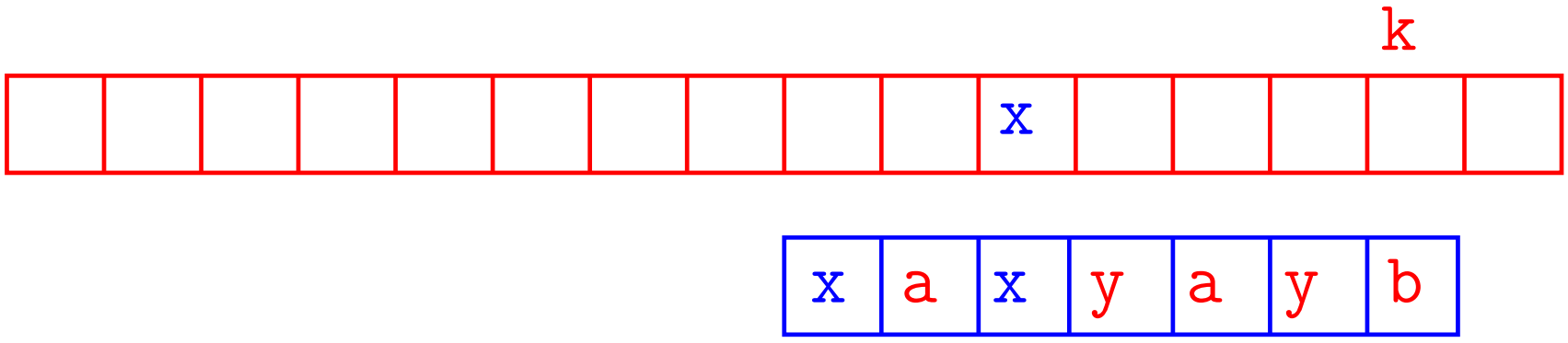
Primeiro algoritmo de Boyer-Moore

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
```

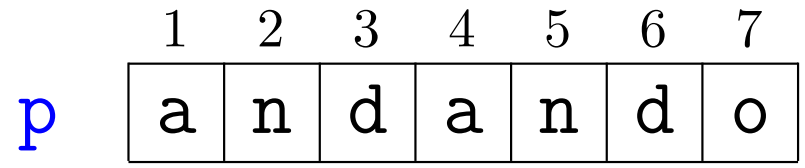
Boyer-Moore

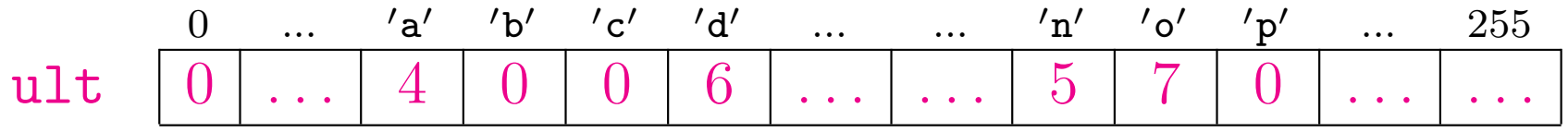
p = a n d a n d o

```
  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 . . .
```

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 $\geq$ 1 e n $\geq$ 0, e devolve o número de ocorrências de p em t.

```
int BoyerMoore (char p[], int m,
                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;
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 $O((n - m + 1)m)$.

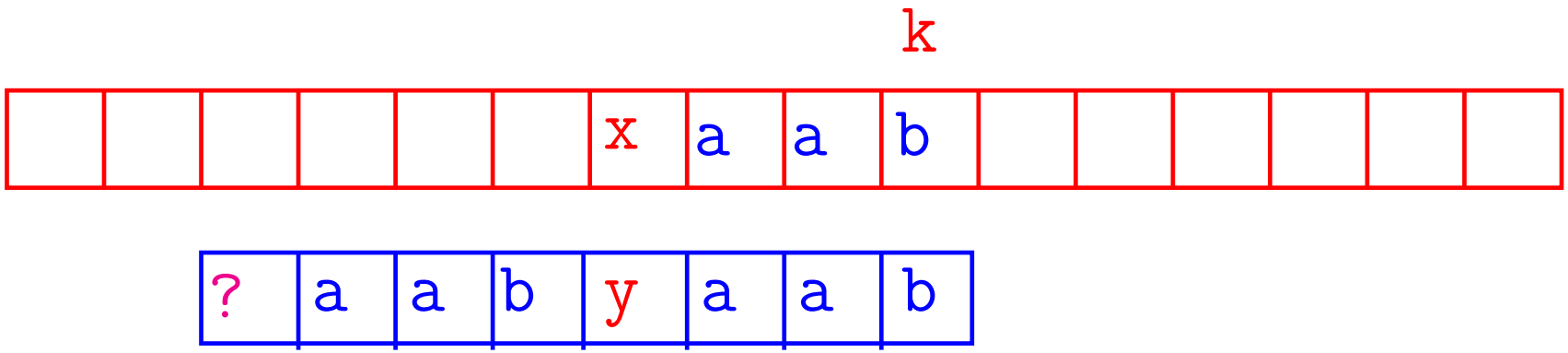
O consumo de tempo da função `BoyerMoore` no melhor caso é $O(n/m)$.

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

AULA 24

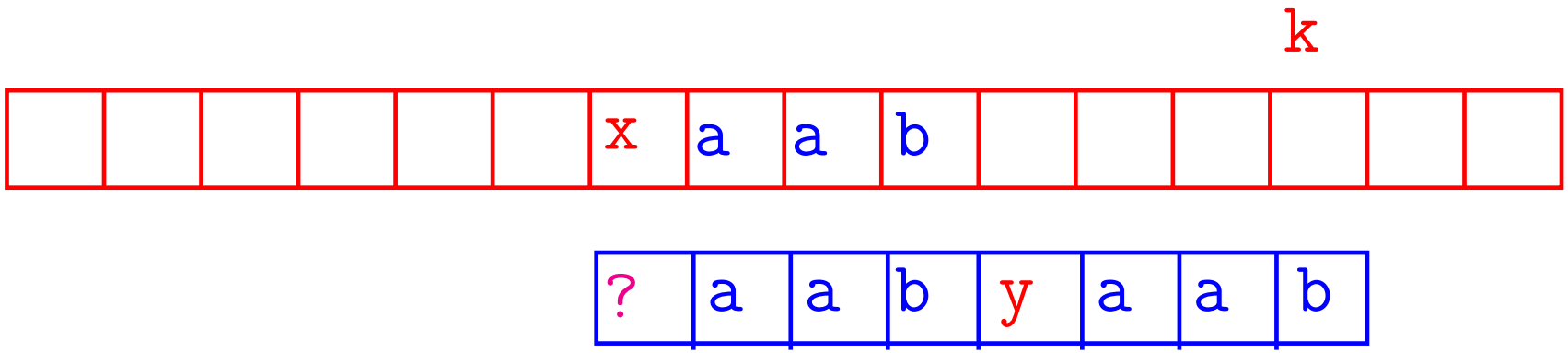
Segundo algoritmo de Boyer-Moore

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



Segundo algoritmo de Boyer-Moore

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



Boyer-Moore 2

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 2

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 2

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 2

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 2

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 2

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 n d o
```

Boyer-Moore 2

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 n d o
7                      a n d a n d o
```

Boyer-Moore 2

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 n d o
7                      a n d a n d o
8                        a n d a n d o
```

Boyer-Moore 2

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 n d o
7                     a n d a n d o
8                       a n d a n d o
9                         a n d a n d o
```

Boyer-Moore 2

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 n d o
 7                     a n d a n d o
 8                       a n d a n d o
 9                         a n d a n d o
10                           a n d a n d o
```

Boyer-Moore 2

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 n d o
7                      a n d a n d o
8                        a n d a n d o
9                          a n d a n d o
10                           a n d a n d o
11                             a n d a n d o
```

Boyer-Moore 2

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 n d o
7                     a n d a n d o
8                       a n d a n d o
9                         a n d a n d o
10                          a n d a n d o
11                            a n d a n d o
15                                  a n d a n d o
```

Boyer-Moore 2

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 n d o
7                     a n d a n d o
8                       a n d a n d o
9                         a n d a n d o
10                          a n d a n d o
11                            a n d a n d o
15                                  a n d a n d o
16                                        a n d a . . .
```

Boyer-Moore 2

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 2

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 2

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 2

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 2

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 2

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	...

Segundo algoritmo de Boyer-Moore

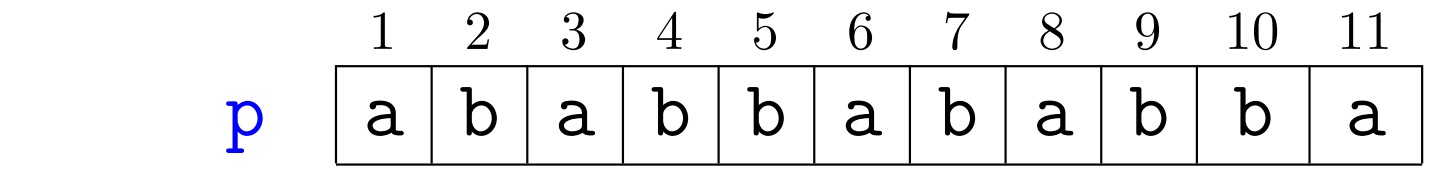
Ideia (*good-suffix heuristic*): para cada índice `i` calcular o maior `j` em 1 . . `m`-1 tal que

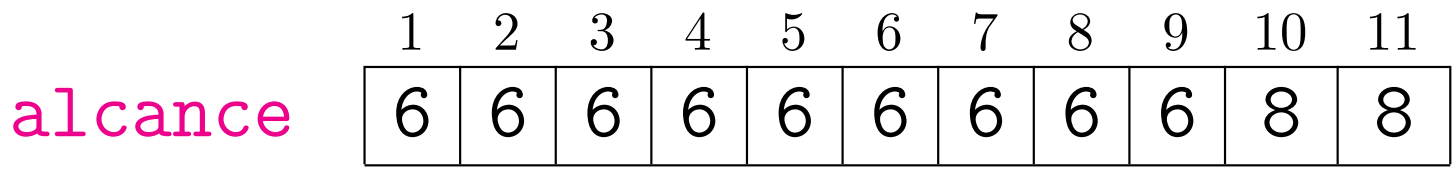
- `p`[1 . . `j`] é um sufixo de `p`[`i` . . `m`] ou
- `p`[`i` . . `m`] é um sufixo de `p`[1 . . `j`].

Para implementar essa ideia, basta fazermos um pré-processamento que **só depende** de `p`.

Segundo algoritmo de Boyer-Moore

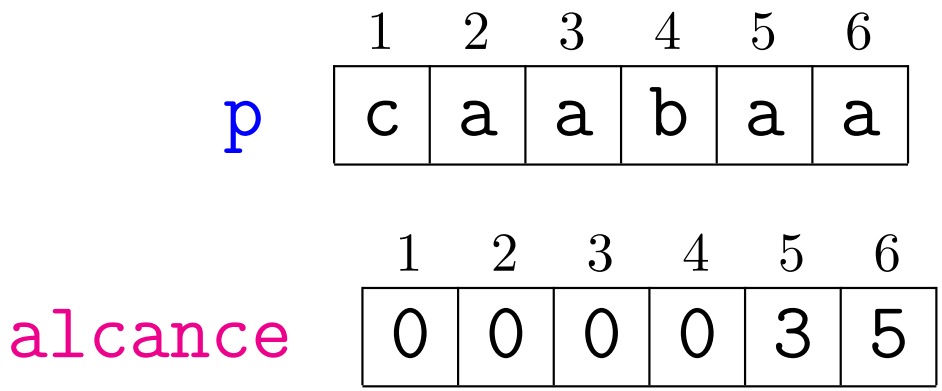
Para implementar essa ideia, fazemos um pré-processamento de `p`, determinando para cada índice `i` o índice `alcance[i]` de um "sufixo bom".





Segundo algoritmo de Boyer-Moore

Para implementar essa ideia fazemos um pré-processamento de `p`, determinando para cada índice `i` o índice `alcance[i]` de um "sufixo bom".



Segundo algoritmo de Boyer-Moore

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

```
int BoyerMoore2 (char p[], int m,
                 char t[], int n) {
   int *alcance;
   int i, r, k, ocorrs;
   /* pre-processamento da palavra p */
1   alcance = preProcessamento(p, m);
2   /* em branco */
```

Segundo 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 (r == 0) k += 1;
10      else k += m - alcance[m-r+1];
     }
11   free(alcance);
12   return ocorrs;
   }
```

Pré-processamento

```
int *
preProcessamento(char p[], int m) {
    int i, r, j, *alcance;
    alcance = malloc((m+1)*sizeof(int));
1   for (i = m; i >= 1; i--) {
2      j = m-1; r = 0
3      while (m-r >= i && j-r >= 1)
4         if (p[m-r] == p[j-r]) r += 1;
5         else j -= 1, r = 0;
6      alcance[i] = j;
    }
7   return alcance;
}
```

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	
	?	?	?	c	b	?	?	?	c	b	?	?	?	c	b	?	?	?	c	b	?	?	?	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	
	?	?	?	c	b	?	?	?	c	b	?	?	?	c	b	?	?	?	c	b	?	?	?	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	
	?	?	?	c	b	?	?	?	c	b	?	?	?	c	b	?	?	?	c	b	?	?	?	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	
	?	?	?	c	b	?	?	?	c	b	?	?	?	c	b	?	?	?	c	b	?	?	?	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				

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	
	?	?	?	c	b	?	?	?	c	b	?	?	?	c	b	?	?	?	c	b	?	?	?	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				
5																					a	a	a	...

Conclusões

O consumo de tempo da função `BoyerMoore2` no pior caso é $\mathrm{O}((\mathtt{n}-\mathtt{m}+1)\mathtt{m})$.

O consumo de tempo da função `BoyerMoore2` no melhor caso é $\mathrm{O}(\mathtt{n}/\mathtt{m})$.

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

Terceiro algoritmo de Boyer-Moore

O algoritmo de Boyer-Moore propriamente dito é uma fusão dos dois anteriores:

> *a cada passo, o algoritmo escolhe o maior dos deslocamentos ditados pelas tabelas* `ult` *e* `alcance`*.*

Comentários finais

Fonte: http://www.quickmeme.com/

MAC0122 – Edição 2020

Livros

Nossa referência básica **foi** o livro

PF = *Paulo Feofiloff*,
Algoritmos em linguagem C,

Este livro é baseado no material do site

Projeto de Algoritmos em C.

Outro livro **foi**

S = *Robert Sedgewick*,
Algorithms in C, vol. 1

MAC0122

MAC0122 **foi** uma disciplina introdutória em:

Algoritmos:

- recursão: torres de Hanoi, EP1, EP4, ...
- divisão-e-conquista: Mergesort, Quicksort, EP4
- pré-processamento: Heapsort, Boyer-Moore, EP2, EP4, EP5
- heurísticas: Boyer-Moore, EP5
- algoritmos de enumeração: `nrainhas`, `cavalo`, EP1, EP5
- algoritmos de busca: busca binária, busca em listas, em árvores de busca, em hashing, busca em largura (`distancias`)
- programação dinâmica ("recursão com tabela"): números binomiais, ...

MAC0122

MAC0122 **foi** uma disciplina introdutória em:

Estruturas de dados:

- listas lineares encadeadas, listas encadeadas circulares, listas com e sem cabeça: EP2
- pilhas: EP3
- filas: distâncias
- heaps: EP4
- tabelas de símbolos: listas ligadas, hash, árvores binárias de busca, EP3

MAC0122

MAC0122 **foi** uma disciplina introdutória em:

Correção de algoritmos:

- relações invariantes: vários problemas nas aulas

MAC0122

MAC0122 **foi** uma disciplina introdutória em:

Correção de algoritmos:

- relações invariantes: vários problemas nas aulas

Eficiência de algoritmos:

- consumo de tempo: vários problemas nas aulas
- notação assintótica O: vários problemas nas aulas
- análise experimental: vários problemas nas aulas
- consumo de espaço:
  Mergesort usa espaço extra $\mathrm{O}(\mathtt{n})$,
  Quicksort usa espaço extra $\mathrm{O}(\lg \mathtt{n})$

MAC0122

MAC0122 **combinou** conceitos e recursos de programação:

- recursão: EP1, EP4 e EP5
- strings: todos os EPs?
- endereços e ponteiros: EP2, EP3, EP4 e EP5
- registros e structs: EP2, EP3, EP4 e EP5
- alocação dinâmica de memória: EP2, EP3, EP4 e EP5
- interfaces: EP2, EP3 e EP4

que nasceram de aplicações cotidianas em ciência da computação.

O que fazer depois de MAC0122?

MAC0121 - Algoritmos e Estruturas de Dados I
é a versão de MAC0122 ministrada para o BCC.

A continuação natural de MAC0121/MAC0122 é

- MAC0323 Algoritmos e Estruturas de Dados II

  `https://uspdigital.usp.br/jupiterweb/obterDisciplina?sgldis=MAC0323`

Outras disciplinas suuuper legais:

- MAC0338 Análise de Algoritmos
- MAC0328 Algoritmos em Grafos
- MAC0325 Otimização Combinatória
- MAC0331 Geometria Computacional
- MAC0385 Estruturas de Dados Avançadas

Pausa para nossos comerciais

- EP5: 25/NOV
- Prova 3: quarta-feira, 27/NOV
- Prova Sub: quarta-feira, 5/DEZ

Pausa para nossos comerciais

- `EP5`: 25/NOV
- Prova 3: quarta-feira, 27/NOV
- Prova Sub: quarta-feira, 5/DEZ

E agradecimentos!

A vocês, pela atençao!!!!

Pausa para nossos comerciais

- EP5: 25/NOV
- Prova 3: quarta-feira, 27/NOV
- Prova Sub: quarta-feira, 5/DEZ

E agradecimentos!

A vocês, pela atençao!!!!

E ao Professor José Coelho de Pina por todo o material!

Fonte: http://dawallpaperz.blogspot.com.br/