

# Busca de palavras (string matching)

## AULA 24

PF 13

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

Navigation icons

Navigation icons

### 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.

Navigation icons

### 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	b	a	b	a	b	b	a	b	b	a
1	a	b	a	b	b	a	b	a	b	b	a													

Navigation icons

### Busca de palavras em um texto

**Problema:** Dados  $P[1..m]$  e  $T[1..n]$ , encontrar o número de ocorrências de  $P$  em  $T$ .

**Exemplo:** Para  $n = 10$ ,  $m = 4$ , 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  $T$ .

Navigation icons

### 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	b	a	b	a	b	b	a	b	b	a
1	a	b	a	b	b	a	b	a	b	b	a													
2	a	b	a	b	b	a	b	a	b	b	a													

Navigation icons

### 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, ocorrencias = 0;
    for (k = 1; k <= n-m+1; k++) {
        r = 0;
        while (r < m && P[1+r] == T[k+r])
            r += 1;
        if (r == m) ocorrencias += 1;
    }
    return ocorrencias;
}
    
```

### Algoritmo trivial

Relação invariante: no início da linha 3 vale que  
 $(i0) P[1..1+r-1] = T[k..k+r-1]$

### Consumo de tempo

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

linha todas as execuções da linha

```

1 = n - m + 2
2 = n - m + 1
3 ≤ (n - m + 1)(m + 1)
4 ≤ (n - m + 1)m
5 = n - m + 1
6 = 1
    
```

$$total < 3(n - m + 2) + 2(n - m + 1)(m + 1) = O((n - m + 1)m)$$

### Pior caso

```

P = 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 T
1 a a a a a a a a a a
2  a a a a a a a a a a
3   a a a a a a a a a a
4    a a a a a a a a a a
5     a a a a a a a a a a
6      a a a a a a a a a a
7       a a a a a a a a a a
8        a a a a a a a a a a
9         a a a a a a a a a a
10          a a a a a a a a a a
11           a a a a a a a a a a
12            a a a a a a a a a a
13             a a a a a a a a a a
    
```

### Melhor caso

```

P = b 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 T
1 b a a a a a a a a a
2  b a a a a a a a a a
3   b a a a a a a a a a
4    b a a a a a a a a a
5     b a a a a a a a a a
6      b a a a a a a a a a
7       b a a a a a a a a a
8        b a a a a a a a a a
9         b a a a a a a a a a
10          b a a a a a a a a a
11           b a a a a a a a a a
12            b a a a a a a a a a
13             b a a a a a a a a a
    
```

### Conclusões

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

O consumo de tempo da função trivial no melhor caso é  $O(n - m + 1)$ .

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

### Algoritmo trivial: direita para esquerda

```

P = a b a b 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 b a T
1 a b a b b a b b a
    
```

### Algoritmo trivial: direita para esquerda

```

P = a b a b 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 T
1 a b a b b a b b a
2  a b a b b a b b a
    
```

### Algoritmo trivial: direita para esquerda

```

P = a b a b 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 T
1 a b a b b a b b a
2  a b a b b a b b a
3   a b a b 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

```

Navigation icons

### 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

```

Navigation icons

### 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

```

Navigation icons

### 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

```

Navigation icons

### 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

```

Navigation icons

### 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, ocorrencias = 0;
    for (k = m; k <= n; k++) {
        r = 0;
        while (r < m && P[m-r] == T[k-r])
            r += 1;
        if (r == m) ocorrencias += 1;
    }
    return ocorrencias;
}

```

Navigation icons

