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[0 : m] **ocorre em** um vetor t[0 : n] se

$$p[0:m] = t[s:s+m]$$

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

Exemplo:

	0	1	2	3	4	5	6	7	8	9
t	x	c	b	a	b	b	c	b	a	x

	0	1	2	3
p	b	c	b	a

p[0 : 4] ocorre em t[0 : 10] com deslocamento 4.

Busca de palavras em um texto

Problema: Dados p[0 : m] e t[0 : n], encontrar o número de ocorrências de p em t.

Exemplo: Para $n = 10$, $m = 4$, e

	0	1	2	3	4	5	6	7	8	9
t	b	b	a	b	a	b	a	c	b	a

	0	1	2	3
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

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
	a	b	a	a	b	a	b	a	b	b	a	b	a	b	a	b	b	a	b	a	b	b	a	t
0	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

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
	a	b	a	a	b	a	b	a	b	b	a	b	a	b	a	b	b	a	b	a	b	b	a	t
0	a	b	a	b	b	a	b	a	b	b	a												
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

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
	a	b	a	a	b	a	b	a	b	b	a	b	a	b	a	b	b	a	b	a	b	b	a	t
0	a	b	a	b	b	a	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										

Algoritmo trivial

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

```
   0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
   a b a a b a b a b b a  b  a  b  a  b  b  a  b  a  b  b  a  t
0  a b a b b a 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
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

```
    0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
    a b a a b a b a b b a b a b a b b a b a b b a t
0   a b a b b a 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
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
```

Algoritmo trivial

Devolve o número de ocorrências de p em t.

```
def trivial(p, m, t, n):
0    ocorrs = 0
1    for k in range(0,n-m+1,1):
2       r = 0
3       while r < m and p[r] == t[k+r]:
4          r += 1
5       if r == m:  ocorrs += 1
6    return ocorrs
```

Algoritmo trivial

Relação invariante: no início da linha 3 vale que

(i0) $p[0:1+r-1] = t[k:k+r-1]$

Consumo de tempo

Consumo de tempo da função trivial

linha	todas as execuções da linha
0	$= 1$
1	$= n-m+2$
2	$= n-m+1$
3	$\leq (n-m+1)(m+1)$
4	$\leq (n-m+1)m$
5	$= n-m+1$
6	$= 1$
total	$< 3\,(n-m+2) + 3\,(n-m+1)(m+1)$ $= O((n-m+1)m)$

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	

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 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.

```
def trivial(p, m, t, n):
0    ocorrs = 0
1    for k in range(m, n, 1):
2        r = 1
3        while r <= m and p[m-r] == t[k-r])
4            r += 1
5        if r == m+1:  ocorrs += 1
6    return ocorrs
```

Algoritmo trivial: direita para esquerda

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

```
def trivial(p, m, t, n):
    ocorrs = 0
    k = m
    while k <= n:
       r = 1
       while r <= m and p[m-r] == t[k-r]:
          r += 1
       if r == m+1:  ocorrs += 1
       k += 1
    return ocorrs
```