AULA 18

Busca de palavras (string matching)

CLRS 32.1

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.

O valor s é um descolamento válido.

Busca de palavras em um texto

Problema: Dados $P[1 \mathinner{..} m]$ e $T[1 \mathinner{..} n]$, encontrar todos os deslocamentos válidos.

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

Os deslocamentos válidos são 1 e 3.

Simulação

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

Naive-String-Matcher

Recebe $P[1\,.\,.\,m]$ e $T[1\,.\,.\,n]$, e devolve todos os deslocamentos válidos.

NAIVE-STRING-MATCHER (P, m, T, n)

1 **para** $i \leftarrow 1$ **até** $n - m$ **faça**
2 $\quad q \leftarrow 0$
3 $\quad$ **enquanto** $q < m$ **e** $P[q+1] = T[i+q]$ **faça**
4 $\quad\quad q \leftarrow q + 1$
5 $\quad$ **se** $q = m$
6 $\quad\quad$ **então** "P ocorre com deslocamento $i - 1$"

Relação invariante: na linha 3 vale que

(i0) $P[1\,.\,.\,q] = T[i\,.\,.\,i{+}q{-}1]$

Consumo de tempo

linha	consumo de todas as execuções da linha
1	$\Theta(n - m + 1)$
2	$\Theta(n - m)$
3	$O((n - m)(m + 1)) = O((n - m)m)$
4	$O((n - m)m)$
5	$\Theta(n - m)$
6	$O(n - m)$
total	$\Theta(3(n - m) + 1) + O(2(n - m)m + n - m)$
	$= O((n - m + 1)m)$

Naive-String-Matcher

NAIVE-STRING-MATCHER (P, m, T, n)

```
1   i ← 1   q ← 0
2   enquanto i ≤ n − m + 1 faça
3       se q = m então ▷ caso 1
4           "P ocorre com deslocamento i − 1"
5           q ← 0
6           i ← i + 1
7       senão se P[q+1] = T[i+q] então ▷ caso 2
8           q ← q + 1
9       senão se P[q+1] ≠ T[i+q] então ▷ caso 3
10          q ← 0
11          i ← i + 1
```

Relação invariante: na linha 2 vale que

(i0) $P[1\,..\,q] = T[i\,..\,i{+}q{-}1]$

Consumo de tempo

Cada linha do algoritmo consome tempo $\Theta(1)$.

Caso 1 e caso 3 ocorrem $n - m + 1$ vezes (total).

Para cada valor de i o número de ocorrências do caso 2
é $\leq m$.

Logo, o número total de iterações das linhas 2–13
é $\leq (n - m + 1)m$.

Portanto, o consumo de tempo algoritmo é $\mathrm{O}((n - m + 1)m)$.

Mas, isto já sabíamos...

Conclusões

O consumo de tempo do algoritmo
NAIVE-STRING-MATCHER é $\mathrm{O}((n-m+1)m)$.

No pior caso, o consumo de tempo do algoritmo
NAIVE-STRING-MATCHER é $\Theta((n-m+1)m)$.

Simulação

$P =$ a b a b b a b a b b a

T

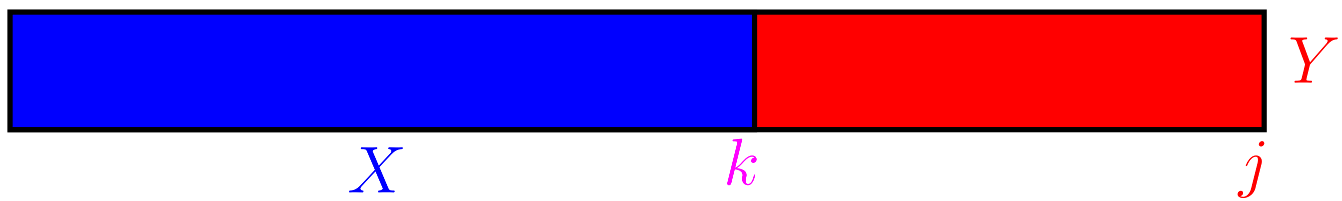
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1	a	b	a	b																			
2		a																					
3			a	b																			
4				a	b	a	b	b															
5					a																		
6						a	b	a	b	b	a	b	a	b	b								
7							a																
8								a	b	a													
9									a		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

Prefixos e sufixos

$X[1\,..\,k]$ é prefixo de $Y[1\,..\,j]$ se

$$k \leq j \quad \text{e} \quad X[1\,..\,k] = Y[1\,..\,k].$$

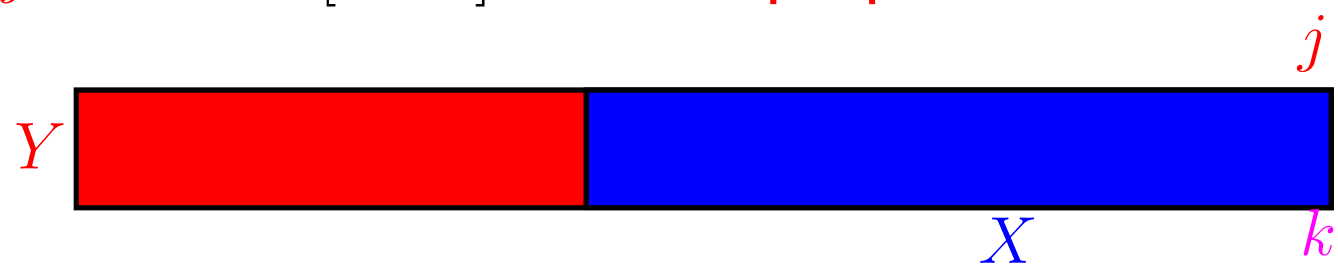
Se $k < j$, então $X[1\,..\,k]$ é prefixo próprio.



$X[1\,..\,k]$ é sufixo de $Y[1\,..\,j]$ se

$$k \leq j \quad \text{e} \quad X[1\,..\,k] = Y[j{-}k{+}1\,..\,j].$$

Se $k < j$, então $X[1\,..\,k]$ é sufixo próprio.



Exemplos

	1	2	3	4	5	6	7	8	9	10	11
P	a	b	a	b	b	a	b	a	b	b	a

$P[1\mathinner{.\,.}1]$ é prefixo próprio de $P[1\mathinner{.\,.}11]$

$P[1\mathinner{.\,.}3]$ é prefixo próprio de $P[1\mathinner{.\,.}7]$

$P[1\mathinner{.\,.}3]$ é sufixo próprio de $P[1\mathinner{.\,.}8]$

$P[1\mathinner{.\,.}5]$ é sufixo próprio de $P[1\mathinner{.\,.}10]$

$P[1\mathinner{.\,.}1]$ é sufixo próprio de $P[1\mathinner{.\,.}11]$

Algoritmo do autômato finito

CLRS 32.3

Algoritmo do autômato finito

Versão grosseira:

FINITE-AUTOMATON-MATCHER (P, m, T, n)

1 $q \leftarrow 0$
2 **para** $i \leftarrow 1$ **até** n **faça**
3 $\quad q \leftarrow$ maior k tal que $P[1\,..\,k]$ é sufixo de $T[1\,..\,i]$
4 $\quad$ **se** $q = m$
5 $\quad\quad$ **então** "P ocorre com deslocamento $i - m$"

Relação invariante: no final da linha 2 vale que:
q é o maior índice tal que

(i0) $P[1\,..\,q]$ é sufixo de $T[1\,..\,i{-}1]$

O segredo está em saber executar a linha 3 eficientemente.

Simulação

$P =$ a b a b c 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

? ? T

8 a b a b c

18 a b a b c a b a b b

Subestrutura ótima

Suponha que:

- q é o maior índice tal que $P[1 \,..\, q]$ é sufixo de $T[1 \,..\, i-1]$; e
- q' é o maior índice tal que $P[1 \,..\, q']$ é sufixo de $T[1 \,..\, i] = T[1 \,..\, i-1]a$.

Subestrutura ótima

Suponha que:

- q é o maior índice tal que $P[1 \mathinner{.\,.} q]$ é sufixo de $T[1 \mathinner{.\,.} i-1]$; e
- q' é o maior índice tal que $P[1 \mathinner{.\,.} q']$ é sufixo de $T[1 \mathinner{.\,.} i] = T[1 \mathinner{.\,.} i-1]a$.

Portanto,

- $P[q'] = a$ e
- $P[1 \mathinner{.\,.} q'-1]$ é prefixo de $P[1 \mathinner{.\,.} q]$.

Logo, $P[1 \mathinner{.\,.} q']$ é sufixo de $P[1 \mathinner{.\,.} q]a$.

Subestrutura ótima

Suponha que:

- q é o maior índice tal que $P[1\,..\,q]$ é sufixo de $T[1\,..\,i-1]$; e
- q' é o maior índice tal que $P[1\,..\,q']$ é sufixo de $T[1\,..\,i] = T[1\,..\,i-1]a$.

Portanto,

- $P[q'] = a$ e
- $P[1\,..\,q'-1]$ é prefixo de $P[1\,..\,q]$.

Logo, $P[1\,..\,q']$ é sufixo de $P[1\,..\,q]a$.

Conclusão: q' é o maior k tal que $P[1\,..\,k]$ é sufixo de $P[1\,..\,q]a$

Subestrutura ótima

Suponha que:

- q é o maior índice tal que $P[1\,..\,q]$ é sufixo de $T[1\,..\,i-1]$; e
- q' é o maior índice tal que $P[1\,..\,q']$ é sufixo de $T[1\,..\,i] = T[1\,..\,i-1]a$.

Portanto,

- $P[q'] = a$ e
- $P[1\,..\,q'-1]$ é prefixo de $P[1\,..\,q]$.

Logo, $P[1\,..\,q']$ é sufixo de $P[1\,..\,q]a$.

Conclusão: q' é o maior k tal que $P[1\,..\,k]$ é sufixo de $P[1\,..\,q]a$

Outra conclusão: o valor de k depende apenas de $P[1\,..\,q]$ e do símbolo a.

Algoritmo do autômato finito

Versão melhorzinha:

FINITE-AUTOMATON-MATCHER (P, m, T, n)

1 $q \leftarrow 0$
2 **para** $i \leftarrow 1$ **até** n **faça**
3 $\quad q' \leftarrow$ maior k tq $P[1\,.\,.\,k]$ é sufixo de $P[1\,.\,.\,q]T[i]$
3 $\quad q \leftarrow q'$
4 $\quad$ **se** $q = m$
5 $\quad\quad$ **então** "P ocorre com deslocamento $i - m$"

Relação invariante: no final da linha 2 vale que:
q é o maior índice tal que

(i0) $P[1\,.\,.\,q]$ é sufixo de $T[1\,.\,.\,i{-}1]$

Pré-processamento

O segredo está em saber executar a linha 3 eficiente.

O valor de q' na linha 3 depende apenas de $P[1\,..\,q]$ e do alfabeto!

Alfabeto $\Sigma =$ conjunto dos possíveis símbolos de P e T.

Um pré-processamento de $P[1\,..\,m]$ produz uma tabela δ (conhecida com autômato) definida da seguinte maneira: para cada q em $0\,..\,m$ e a em Σ

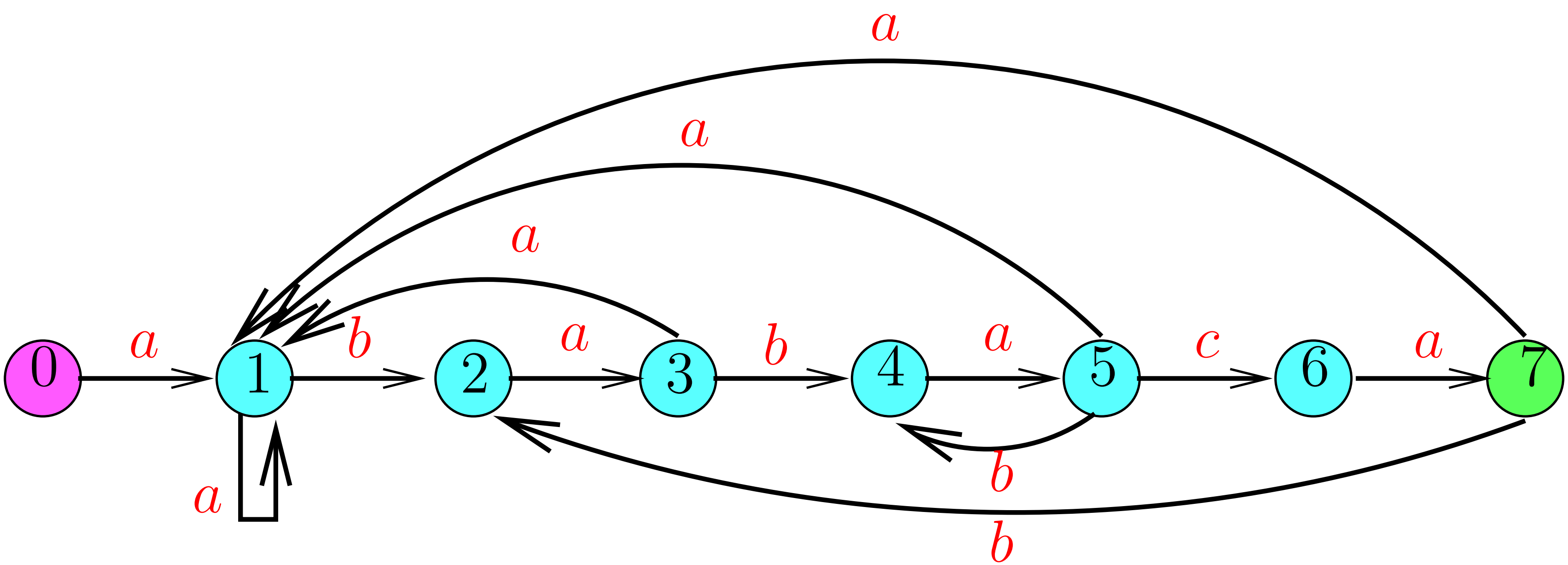
$$\delta[q, a] = \max\{k : P[1\,..\,k] \text{ é sufixo de } P[1\,..\,q]a\}.$$

Autômato

$P =$ a b a b a c a $\Sigma = \{$a, b, c$\}$

q	a	b	c	P
0	1	0	0	a
1	1	2	0	b
2	3	0	0	a
3	1	4	0	b
4	5	0	0	a
5	1	4	6	c
6	7	0	0	a
7	1	2	0	

Diagrama



$0..7 =$ conjunto de estados

$\Sigma = \{$a, b, c$\} =$ alfabeto

δ = função de transição

0 é estado incial e 7 é estado final

Algoritmo do autômato finito

FINITE-AUTOMATON-MATCHER (P, m, T, n)

0 $\delta \leftarrow$ COMPUTE-TRANSITION-FUNCTION(P, m, Σ)
1 $q \leftarrow 0$
2 **para** $i \leftarrow 1$ **até** n **faça**
3 $\quad q \leftarrow \delta[q, T[i]]$
4 $\quad$ **se** $q = m$
5 $\quad\quad$ **então** "P ocorre com deslocamento $i - m$"

Relação invariante: no final da linha 2 vale que:
q é o maior índice tal que

(i0) $P[1\,..\,q]$ é sufixo de $T[1\,..\,i{-}1]$

Consumo de tempo das linhas 1-5 é $\Theta(n)$.

Simulação

$P =$ a b a b a c a

i		1	2	3	4	5	6	7	8	9	10	11
T		a	b	a	b	a	b	a	c	a	b	a
q	0	1	2	3	4	5	4	5	6	7	2	3

Pré-processamento

COMPUTE-TRANSITION-FUNCTION(P, m, Σ)

para $q \leftarrow 0$ **até** m
    para cada a em Σ **faça**
        $k \leftarrow \min\{m+1, q+2\}$
        repita
            $k \leftarrow k - 1$
        até até $P[1\,.\,.\,k]$ seja sufixo de $P[1\,.\,.\,q]a$
        $\delta[q, a] \leftarrow k$
devolva δ

Consumo de tempo

linha	consumo de todas as execuções da linha
1	$\Theta(m)$
2	$\Theta(m\lvert\Sigma\rvert)$
3	$\Theta(m\lvert\Sigma\rvert)$
4-5	$\Theta(m^2\lvert\Sigma\rvert)$
6	$\Theta(m^3\lvert\Sigma\rvert)$
7	$\Theta(m\lvert\Sigma\rvert)$
8	$\Theta(m\lvert\Sigma\rvert)$

total $\Theta(m + 3m|\Sigma| + m^2|\Sigma| + m^3|\Sigma|)$

$$= \Theta(m^3|\Sigma|)$$

Conclusões

O consumo de tempo do algoritmo
COMPUTE-TRANSITION-FUNCTION é $\Theta(m^3|\Sigma|)$.

O consumo de tempo do algoritmo
FINITE-AUTOMATON-MATCHER é $\Theta(n + m^3|\Sigma|)$.

Knuth-Morris-Pratt

CLRS 32.4

Nova simulação

$P =$ 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																				
4			a	b																				
4				a	b	a	b	b																
8						a	b	a	b	b	a	b	a	b	b									
15											a	b	a	b	b									
15													a	b	a	b	b	a	b	a	b	b	a	
i																								
	1	1	1	3	1	1	1	2	1	1	1	1	1	1	3	1	1	1	1	1	1	1	1	

Função prefixo

$\pi[q] :=$ maior comprimento de um prefixo próprio de $P[1\,..\,q]$ que é sufixo de $P[1\,..\,q]$

$:= \max\{k : k < q$ e $P[1\,..\,k]$ é sufixo de $P[1\,..\,q]\}$

q	1	2	3	4	5	6	7	8	9	10	11
P	a	b	a	b	b	a	b	a	b	b	a
π	0	0	1	2	0	1	2	3	4	5	6

q	1	2	3	4	5	6	7	8	9	10
P	a	b	a	b	a	b	a	b	c	a
π	0	0	1	2	3	4	5	6	0	1

KMP-Matcher

```
KMP-MATCHER (P, m, T, n)
π ← COMPUTE-PREFIX-FUNCTION(P, m)
i ← 1   q ← 0
enquanto i ≤ n + 1 faça
    se q = m então ▷ caso 1
        "P ocorre com deslocamento i − m"
        q ← π[q]
    senão se P[q+1] = T[i] então ▷ caso 2
        q ← q + 1
        i ← i + 1
    senão se P[q+1] ≠ T[i] e q > 0 então ▷ caso 3
        q ← π[q]
    senão se P[q+1] ≠ T[i] e q = 0 então ▷ caso 4
        i ← i + 1
```

Suponha $T[n+1]$ = "símbolo que não ocorre em $P[1\,..\,m]$"

Correção

Relação invariante: na linha 2 vale que

(i0) $P[1\,..\,q]$ é sufixo de $T[1\,..\,i{-}1]$

Na linha 2 vale ainda que

(i1) $P[1\,..\,k]$ não é sufixo de $T[1\,..\,i]$, para $k \geq q+2$.

Invariante (i1) implica que $i-m, i-m+1, \ldots, i-q-1$ não são deslocamentos válidos, ou seja

$$P[1\,..\,m] \text{ não é sufixo de } T[1\,..\,i+1]$$

$$P[1\,..\,m] \text{ não é sufixo de } T[1\,..\,i+2]$$

$$\ldots$$

$$P[1\,..\,m] \text{ não é sufixo de } T[1\,..\,i+m-q-1].$$

Consumo de tempo

Exceto a linha 0, cada linha do algoritmo consome tempo $\Theta(1)$.

Caso 2 e caso 4 ocorrem $n + 1$ vezes (total).

Para cada valor de i, número de ocorrências do caso 1 e caso 3 é $\leq m$.

Logo, o número total de iterações das linhas 2–12
é $\leq n + 1 + (n + 1)m = (n + 1)(m + 1)$.

Portanto, o consumo de tempo total das linhas 1–12
é $\mathrm{O}(nm)$.

Consumo de tempo

Exceto a linha 0, cada linha do algoritmo consome tempo $\Theta(1)$.

Caso 2 e caso 4 ocorrem $n+1$ vezes (total).

Para cada valor de i, número de ocorrências do caso 1 e caso 3 é $\leq m$.

Logo, o número total de iterações das linhas 2–12

é $\leq n+1+(n+1)m = (n+1)(m+1)$.

Portanto, o consumo de tempo total das linhas 1–12

é $\mathrm{O}(nm)$.

EXAGERO!

Consumo de tempo

Exceto a linha 0, cada linha do algoritmo consome tempo $\Theta(1)$.

Caso 2 e caso 4 ocorrem $n+1$ vezes (total).

O número de ocorrências do caso 1 e caso 3 é $\leq$ número de ocorrências do caso 2, pois

- q nunca é negativo;
- q é incrementado no caso 2;
- q é decrementado no caso 1 e no caso 3.

Logo, o número total de iterações é $\leq n+1+n+1 = 2n+2$.

Portanto, o consumo de tempo das linhas 1–12 é $\Theta(n)$.