Melhores momentos

AULA PASSADA

Busca de palavras em um texto

Problema: Dados $P[1 \,.\,.\, m]$ e $T[1 \,.\,.\, n]$, 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.

Idéia

$P =$ 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

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 $\qquad$ **então** "P ocorre com deslocamento $i - m$"

Relação invariante: no final da linha 2 vale que:

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

O segredo está em saber executar a linha 3 eficientemente.

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:

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

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:

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

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

Autômato

$P =$ a b a b a c a $\quad\quad \Sigma = \{$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	

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

Conclusões

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

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

AULA 20

Knuth-Morris-Pratt

CLRS 32.4

KMP e algoritmo do autômato

O algoritmo KMP-MATCHER pode ser visto como uma implementação do algoritmo

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

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

As iterações do algoritmo KMP-MATCHER em que o valor de i não é incrementado estão determinando o

"maior k tal que $P[1\,.\,.\,k]$ é sufixo de $P[1\,.\,.\,q]T[i]$"

KMP e algoritmo do autômato

O algoritmo KMP-MATCHER pode ser visto como uma implementação do algoritmo

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

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

O valor de i só é incrementado após a determinação desse

"maior k tal que $P[1\,.\,.\,k]$ é sufixo de $P[1\,.\,.\,q]T[i]$"

Para isto o algoritmo faz uso de uma certa função π.

Idéia

$P =$ 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

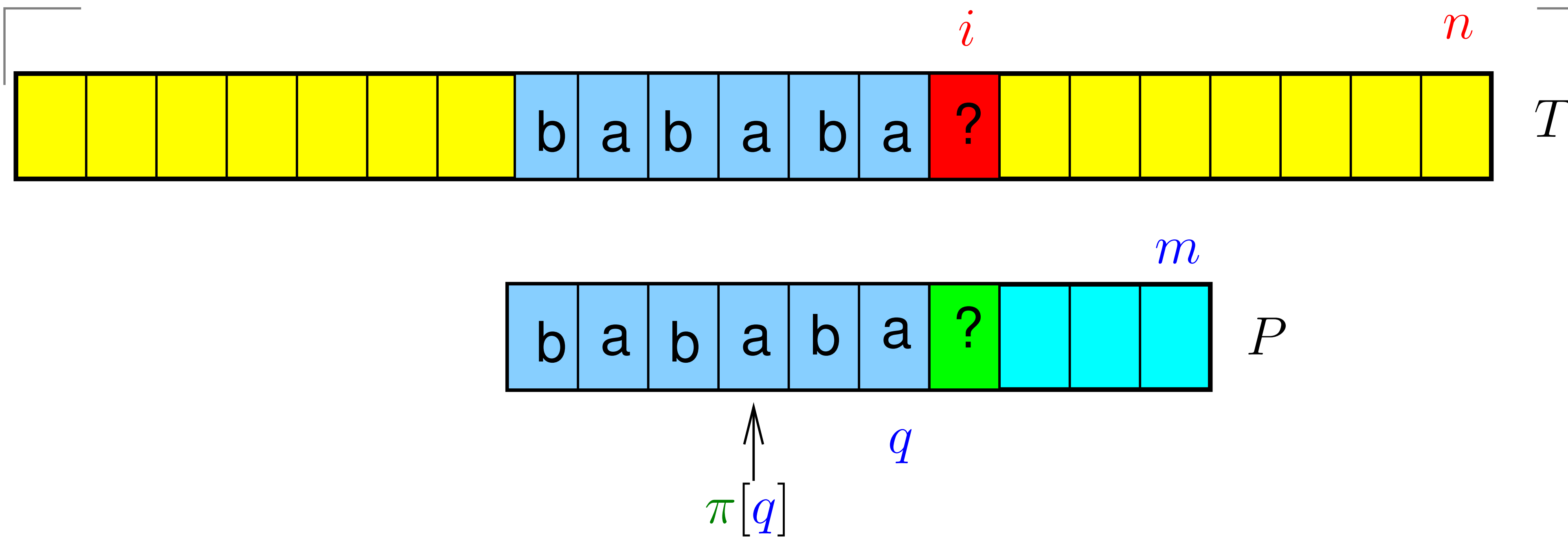
$\pi[q] :=$ maior comprimento de um prefixo próprio de $P[1\,..\,q]$ que é sufixo de $P[1\,..\,q]$

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

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

Iteração



Relações invariantes:

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

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

KMP-Matcher

KMP-MATCHER (P, m, T, n)

$\pi \leftarrow$ COMPUTE-PREFIX-FUNCTION(P, m)
$i \leftarrow 1 \quad q \leftarrow 0$
enquanto $i \leq n + 1$ **faça**
    se $q = m$ **então** ▷ caso 1
        "P ocorre com deslocamento $i - 1 - m$"
        $q \leftarrow \pi[q]$ ▷ volta padrão
    senão se $P[q{+}1] = T[i]$ **então** ▷ caso 2
        $q \leftarrow q + 1$ ▷ avança padrão
        $i \leftarrow i + 1$ ▷ avança texto
    senão se $P[q{+}1] \neq T[i]$ **e** $q > 0$ **então** ▷ caso 3
        $q \leftarrow \pi[q]$ ▷ volta padrão
    senão se $P[q{+}1] \neq T[i]$ **e** $q = 0$ **então** ▷ caso 4
        $i \leftarrow i + 1$ ▷ avança texto

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

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 $\Theta(n)$.

KMP-Matcher

KMP-MATCHER (P, m, T, n)

$\pi \leftarrow$ COMPUTE-PREFIX-FUNCTION(P, m)
$q \leftarrow 0$
para $i \leftarrow 1$ **até** n **faça**
    enquanto $q > 0$ **e** $P[q{+}1] \neq T[i]$ **faça**
        $q \leftarrow \pi[q]$
    se $P[q{+}1] = T[i]$ **então**
        $q \leftarrow q + 1$
    se $q = m$ **então**
        “P ocorre com deslocamento $i - m$”
        $q \leftarrow \pi[q]$

Compute-Prefix-Function (grosseiro)

COMPUTE-PREFIX-FUNCTION (P, m)

```
π[1] ← 0
q ← 2
enquanto q ≤ m faça
    k ← q − 1
    repita
        i ← k
        j ← q
        enquanto i > 0 e P[i] = P[j] faça
            i ← i − 1    j ← j − 1
        se i > 0
            então k ← k − 1
    até que i = 0
    π[q] ← k
devolva π
```

Consumo de tempo

linha	consumo de todas as execuções da linha
0-1	$\Theta(1)$
2	$\Theta(m)$
3	$\Theta(m-1)$
4-6	$O(m^2)$
7-8	$O(m^3)$
9-11	$O(m^2)$
12	$\Theta(m-1)$
13	$O(m)$
total	$\Theta(3m-1) + O(m^3 + 2m^2 + m)$
	$= O(m^3)$

Função prefixo e algoritmo do autômato

O algoritmo COMPUTE-PREFIX-FUNCTION pode ser semelhante ao algoritmo

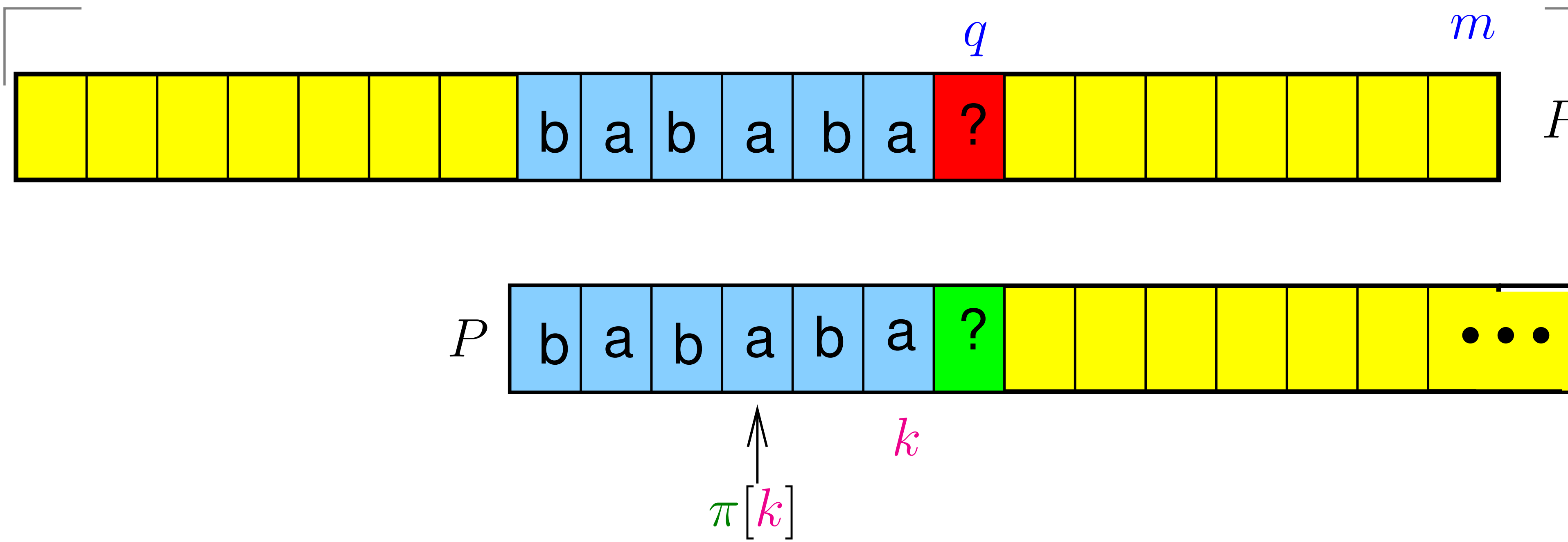
COMPUTE-PREFIX-FUNCTION (P, m)

para $q \leftarrow 1$ **até** m **faça**
    $\pi[q] \leftarrow$ maior k tal que $P[1\,..\,k]$ é sufixo próprio de $P[1\,..\,q]$

O valor de q só é incrementado após a determinação desse

"maior k tal que $P[1\,..\,k]$ é sufixo próprio de $P[1\,..\,q]$"

Iteração



Para determinar o

"maior k tal que $P[1\,..\,k]$ é sufixo próprio de $P[1\,..\,i]$"

podemos utilizar os valores de π já calculados.

Subestrutura ótima

Suponha que $P[1\,..\,k+1]$ é o maior prefixo próprio de $P[1\,..\,q]$ que é sufixo de $P[1\,..\,q]$ ($\pi[q] = k+1$).

Subestrutura ótima

Suponha que $P[1\,..\,k+1]$ é o maior prefixo próprio de $P[1\,..\,q]$ que é sufixo de $P[1\,..\,q]$ ($\pi[q] = k+1$).

Portanto,

- $P[k+1] = P[q]$ e
- $P[1\,..\,k]$ é prefixo próprio e sufixo de $P[1\,..\,q-1]$.

Logo, $P[1\,..\,k]$ é prefixo e sufixo de $P[1\,..\,\pi[q-1]]$.

Subestrutura ótima

Suponha que $P[1\,..\,k+1]$ é o maior prefixo próprio de $P[1\,..\,q]$ que é sufixo de $P[1\,..\,q]$ ($\pi[q] = k+1$).

Portanto,

- $P[k+1] = P[q]$ e
- $P[1\,..\,k]$ é prefixo próprio e sufixo de $P[1\,..\,q-1]$.

Logo, $P[1\,..\,k]$ é prefixo e sufixo de $P[1\,..\,\pi[q-1]]$.

Se $k \neq \pi[q-1]$, então

Subestrutura ótima

Suponha que $P[1\,..\,k+1]$ é o maior prefixo próprio de $P[1\,..\,q]$ que é sufixo de $P[1\,..\,q]$ ($\pi[q] = k+1$).

Portanto,

- $P[k+1] = P[q]$ e
- $P[1\,..\,k]$ é prefixo próprio e sufixo de $P[1\,..\,q-1]$.

Logo, $P[1\,..\,k]$ é prefixo e sufixo de $P[1\,..\,\pi[q-1]]$.

Se $k \neq \pi[q-1]$, então

$P[1\,..\,k]$ é prefixo próprio e sufixo de $P[1\,..\,\pi[q-1]]$.

Logo, $P[1\,..\,k]$ é prefixo e sufixo de

$$P[1\,..\,\pi[\pi[q-1]]] = P[1\,..\,\pi^2[q-1]].$$

Subestrutura ótima (cont.)

Se $k \neq \pi^2[q-1]$, então

Subestrutura ótima (cont.)

Se $k \neq \pi^2[q-1]$, então

$P[1\,..\,k]$ é prefixo próprio e sufixo de $P[1\,..\,\pi^2[q-1]]$.

Logo, $P[1\,..\,k]$ é sufixo de

$$P[1\,..\,\pi[\pi[\pi[q-1]]]] = P[1\,..\,\pi^3[q-1]].$$

Subestrutura ótima (cont.)

Se $k \neq \pi^2[q-1]$, então

$P[1\,..\,k]$ é prefixo próprio e sufixo de $P[1\,..\,\pi^2[q-1]]$.

Logo, $P[1\,..\,k]$ é sufixo de

$$P[1\,..\,\pi[\pi[\pi[q-1]]]] = P[1\,..\,\pi^3[q-1]].$$

Se $k \neq \pi^3[q-1]$, então

$P[1\,..\,k]$ é prefixo próprio e sufixo
de $P[1\,..\,\pi^3[q-1]]$.

Logo, $P[1\,..\,k]$ é prefixo e sufixo de $P[1\,..\,\pi^4[q-1]]$.

Se $k \neq \pi^4[q-1]$, então ...

Recorrência

$\pi[q] :=$ 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	0	1	2	3	4	5	6	7	8	9	10	11
P	'?'	a	b	a	b	b	a	b	a	b	b	a
π	-1	0	0	1	2	0	1	2	3	4	5	6

$\pi[0] = -1$

$\pi[1] = 0$

$\pi[q] = \max\{\pi^j[q-1]+1 : P[\pi^j[q-1]+1] = P[q]\}$

Suponha $P[0] =$ "coringa".

Compute-Prefix-Function (grosseiro)

COMPUTE-PREFIX-FUNCTION (P, m)

$\pi[1] \leftarrow 0$
$q \leftarrow 2$
enquanto $q \leq m$ **faça**
    $k \leftarrow q - 1$
    repita
        $i \leftarrow k$
        $j \leftarrow q$
        enquanto $i > 0$ **e** $P[i] = P[j]$ **faça**
            $i \leftarrow i - 1 \quad j \leftarrow j - 1$
        se $i > 0$
            então $k \leftarrow k - 1$
    até que $i = 0$
    $\pi[q] \leftarrow k$
devolva π

Compute-Prefix-Function (melhorado)

COMPUTE-PREFIX-FUNCTION (P, m)

$\pi[1] \leftarrow 0$
$q \leftarrow 2$
enquanto $q \leq m$ **faça**
  $k \leftarrow \pi[q-1]$
  repita
    $i \leftarrow k+1$
    $j \leftarrow q$
    enquanto $i > 0$ **e** $P[i+1] = P[j]$ **faça**
      $i \leftarrow i-1 \quad j \leftarrow j-1$
    se $i > 0$
      então $k \leftarrow \pi[k]$
  até que $i = 0$
  $\pi[q] \leftarrow k+1$
devolva π

Compute-Prefix-Function (melhor ainda)

COMPUTE-PREFIX-FUNCTION (P, m)

$\pi[1] \leftarrow 0$
$q \leftarrow 2$
enquanto $q \leq m$ **faça**
    $k \leftarrow \pi[q-1]$
    repita
        $i \leftarrow k+1$
        $j \leftarrow q$
        se $P[i] = P[j]$
            então $i \leftarrow 0$
            senão $k \leftarrow \pi[k]$
    até que $i = 0$
    $\pi[q] \leftarrow k+1$
devolva π

Compute-Prefix-Function (limpo)

COMPUTE-PREFIX-FUNCTION (P, m)

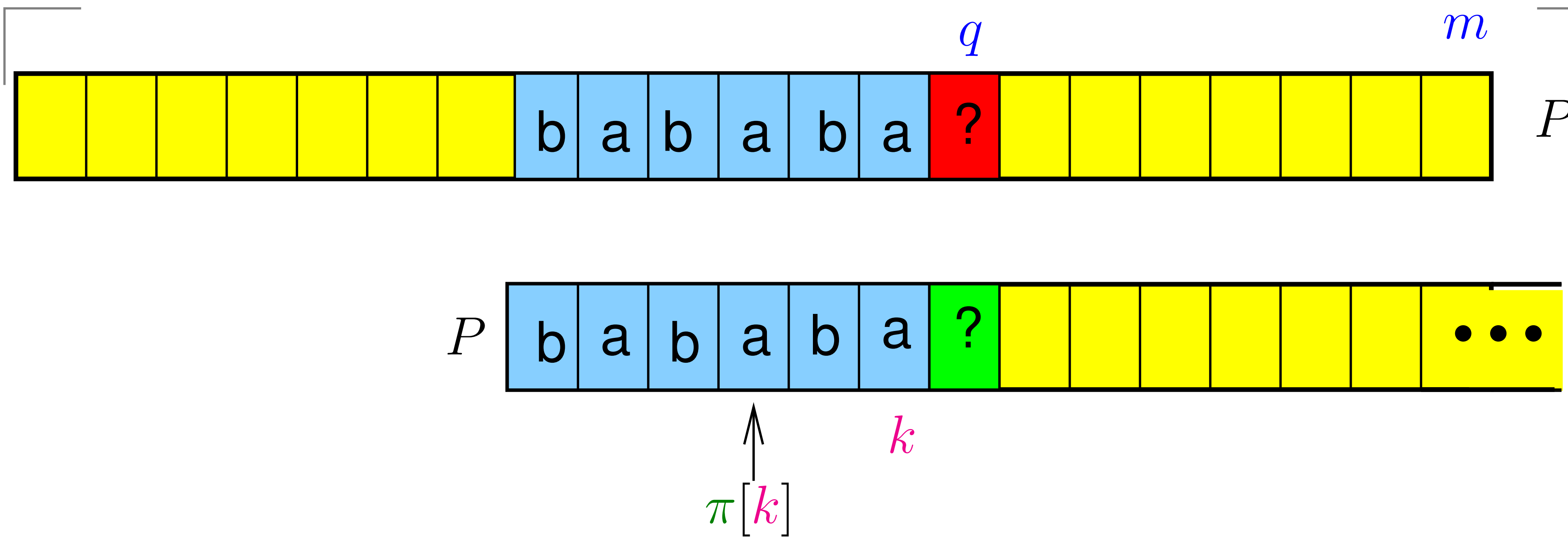
$\pi[1] \leftarrow 0$
$q \leftarrow 2 \quad k \leftarrow 0$
enquanto $q \leq m$ **faça**
    enquanto $k > 0$ **e** $P[k+1] \neq P[q]$ **faça**
        $k \leftarrow \pi[k]$
    se $P[k+1] = P[q]$
        então $k \leftarrow k+1$
    $\pi[q] \leftarrow k$
devolva π

Compute-Prefix-Function (limpo)

COMPUTE-PREFIX-FUNCTION (P, m)

$\pi[1] \leftarrow 0$
$q \leftarrow 2 \quad k \leftarrow 0$
enquanto $q \leq m$ **faça**
    se $P[k+1] = P[q]$ **então** $\triangleright$ caso 1
        $\pi[q] \leftarrow k + 1$
        $q \leftarrow q + 1$
        $k \leftarrow k + 1$
    senão se $P[k+1] \neq P[q]$ **e** $k > 0$ **então** $\triangleright$ caso 2
        $k \leftarrow \pi[k]$
    senão se $P[k+1] \neq P[q]$ **e** $k = 0$ **então** $\triangleright$ caso 3
        $\pi[q] \leftarrow 0$
        $q \leftarrow q + 1$
devolva π

Correção



Relações invariantes: na linha 2 vale que

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

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

Consumo de tempo

Cada linha do algoritmo consome tempo $\Theta(1)$, exceto a linha 12 que pode consumir tempo $\mathrm{O}(m)$.

Caso 1 e caso 3 ocorrem m vezes (total).

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

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

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

O consumo de tempo do algoritmo é $\Theta(m)$.

Conclusões

O consumo de tempo do algoritmo COMPUTE-PREFIX-FUNCTION é $\Theta(m)$.

O consumo de tempo do algoritmo KMP-MATCHER é $\Theta(n + m)$.