MATA54 - Estruturas de Dados e Algoritmos II

Busca de Padrões em Textos

Flávio Assis
Versão gerada a partir de slides do Prof. George Lima

IC - Instituto de Computação

Salvador, outubro de 2021

Motivação: Busca de Padrões em Textos

Aplicações: busca de uma subsequência de caracteres

- **Texto:** sequência de caracteres $T[1..n]$ de tamanho n
- **Alfabeto:** cada caractere do texto é um elemento de um conjunto (alfabeto) Σ
- **Padrão:** sequência de caracteres $P[1..m]$ de tamanho $m \leq n$
- Quais são as posições em T onde P ocorre (se ocorrer)?

Algoritmos a serem estudados:

- Algoritmo baseado em Autômato
- KMP (Knuth-Morris-Pratt)
- BM (Boyer-Moore)

Algoritmo Simples (Ingênuo): Ilustração

Ex: **T** = "abababababaabab" e **P** = "ababb"

1	2	3	4	5	6	7	8	9	10	11	12	13	14
a	b	a	b	a	b	a	b	a	a	b	a	b	b

Algoritmo Simples (Ingênuo): Ilustração

Ex: **T** = "abababababaabab**b**" e **P** = "abab**b**"

1	2	3	4	5	6	7	8	9	10	11	12	13	14
a	b	a	b	a	b	a	b	a	a	b	a	b	b
a	**b**	**a**	**b**	**b**									

Algoritmo Simples (Ingênuo): Ilustração

Ex: **T** = "abababababaabab**b**" e **P** = "abab**b**"

1	2	3	4	5	6	7	8	9	10	11	12	13	14
a	b	a	b	a	b	a	b	a	a	b	a	b	b
a	**b**	**a**	**b**	**b**									
→	**a**	b	a	b	b								

Algoritmo Simples (Ingênuo): Ilustração

Ex: **T** = "abababababaabab" e **P** = "ababb"

1	2	3	4	5	6	7	8	9	10	11	12	13	14
a	b	a	b	a	b	a	b	a	a	b	a	b	b
a	**b**	**a**	**b**	**b**									
$\rightarrow$	**a**	b	a	b	b								
	$\rightarrow$	**a**	**b**	**a**	**b**	**b**							

Algoritmo Simples (Ingênuo): Ilustração

Ex: **T** = "abababababaabab" e **P** = "ababb"

1	2	3	4	5	6	7	8	9	10	11	12	13	14
a	b	a	b	a	b	a	b	a	a	b	a	b	b
a	**b**	**a**	**b**	**b**									
→	**a**	b	a	b	b								
	→	**a**	**b**	**a**	**b**	**b**							
		→	**a**	b	a	b	b						

Algoritmo Simples (Ingênuo): Ilustração

Ex: **T** = "abababababaabab" e **P** = "ababb"

1	2	3	4	5	6	7	8	9	10	11	12	13	14
a	b	a	b	a	b	a	b	a	a	b	a	b	b
a	**b**	**a**	**b**	**b**									
→	**a**	b	a	b	b								
	→	**a**	**b**	**a**	**b**	**b**							
		→	**a**	b	a	b	b						
			→	**a**	**b**	**a**	**b**	**b**					

Algoritmo Simples (Ingênuo): Ilustração

Ex: **T** = "abababababaabab" e **P** = "ababb"

1	2	3	4	5	6	7	8	9	10	11	12	13	14
a	b	a	b	a	b	a	b	a	a	b	a	b	b
a	**b**	**a**	**b**	**b**									
→	**a**	b	a	b	b								
	→	**a**	**b**	**a**	**b**	**b**							
		→	**a**	b	a	b	b						
			→	**a**	**b**	**a**	**b**	**b**					
				→	**a**	b	a	b	b				

Algoritmo Simples (Ingênuo): Ilustração

Ex: **T** = "abababababaabab" e **P** = "ababb"

1	2	3	4	5	6	7	8	9	10	11	12	13	14
a	b	a	b	a	b	a	b	a	a	b	a	b	b
a	**b**	**a**	**b**	**b**									
→	**a**	b	a	b	b								
	→	**a**	**b**	**a**	**b**	**b**							
		→	**a**	b	a	b	b						
			→	**a**	**b**	**a**	**b**	**b**					
				→	**a**	b	a	b	b				
					→	**a**	**b**	**a**	**b**	b			

Algoritmo Simples (Ingênuo): Ilustração

Ex: **T** = "abababababaabababb" e **P** = "ababb"

1	2	3	4	5	6	7	8	9	10	11	12	13	14
a	b	a	b	a	b	a	b	a	a	b	a	b	b
a	**b**	**a**	**b**	**b**									
→	**a**	b	a	b	b								
	→	**a**	**b**	**a**	**b**	**b**							
		→	**a**	b	a	b	b						
			→	**a**	**b**	**a**	**b**	**b**					
				→	**a**	b	a	b	b				
					→	**a**	**b**	**a**	**b**	b			
						→	**a**	b	a	b	b		

Algoritmo Simples (Ingênuo): Ilustração

Ex: **T** = "abababababaabab" e **P** = "ababb"

1	2	3	4	5	6	7	8	9	10	11	12	13	14
a	b	a	b	a	b	a	b	a	a	b	a	b	b
a	**b**	**a**	**b**	**b**									
→	**a**	b	a	b	b								
	→	**a**	**b**	**a**	**b**	**b**							
		→	**a**	b	a	b	b						
			→	**a**	**b**	**a**	**b**	**b**					
				→	**a**	b	a	b	b				
					→	**a**	**b**	**a**	**b**	b			
						→	**a**	b	a	b	b		
							→	**a**	**b**	a	b	b	

Algoritmo Simples (Ingênuo): Ilustração

Ex: **T** = "abababababaabab" e **P** = "ababb"

1	2	3	4	5	6	7	8	9	10	11	12	13	14
a	b	a	b	a	b	a	b	a	a	b	a	b	b
a	**b**	**a**	**b**	**b**									
→	**a**	b	a	b	b								
	→	**a**	**b**	**a**	**b**	**b**							
		→	**a**	b	a	b	b						
			→	**a**	**b**	**a**	**b**	**b**					
				→	**a**	b	a	b	b				
					→	**a**	**b**	**a**	**b**	b			
						→	**a**	b	a	b	b		
							→	**a**	**b**	a	b	b	
								→	**a**	**b**	**a**	**b**	**b**

Algoritmo Simples

Busca de $P[1 \ldots m]$ em $T[1 \ldots n]$, $m \leq n$.

```
1  int simpleStrMatcher(char T[], char P[]) {
2    int i, j, m, n;
3
4    n = strlen(T);
5    m = strlen(P);
6    for (i = 0; i <= n-m; i++) {
7      for (j = 1; j <= m; j++) {
8        if (P[j] != T[i+j]) break;
9      }
10     if (j == m+1) return (i+1);
11   }
12   return (-1);
13 }
```

Complexidade $O(nm)$ – **exemplo?**

É possível melhorar pré-processando o padrão p

Algoritmo baseado em Autômato

Comparações Desnecessárias

Ex: T = “abababababaabab” e P = “ababb”

À medida que se conhece o texto, as seguintes comparações em laranja abaixo são desnecessárias:

já se sabe o resultado da comparação a priori!

1	2	3	4	5	6	7	8	9	10	11	12	13	14
a	b	a	b	a	b	a	b	a	a	b	a	b	b

Comparações Desnecessárias

Ex: T = “abababababaabab” e P = “ababb”

À medida que se conhece o texto, as seguintes comparações em laranja abaixo são desnecessárias:

já se sabe o resultado da comparação a priori!

1	2	3	4	5	6	7	8	9	10	11	12	13	14
a	b	a	b	a	b	a	b	a	a	b	a	b	b
a	**b**	**a**	**b**	**b**									

Comparações Desnecessárias

Ex: T = “abababababaabab” e P = “ababb”

À medida que se conhece o texto, as seguintes comparações em laranja abaixo são desnecessárias:

já se sabe o resultado da comparação a priori!

1	2	3	4	5	6	7	8	9	10	11	12	13	14
a	b	a	b	a	b	a	b	a	a	b	a	b	b
a	**b**	**a**	**b**	**b**									
→	**a**												

Comparações Desnecessárias

Ex: T = “abababababaabab” e P = “ababb”

À medida que se conhece o texto, as seguintes comparações em laranja abaixo são desnecessárias:

já se sabe o resultado da comparação a priori!

1	2	3	4	5	6	7	8	9	10	11	12	13	14
a	b	a	b	a	b	a	b	a	a	b	a	b	b
a	**b**	**a**	**b**	**b**									
→	**a**												
	→	**a**	**b**	**a**	**b**	**b**							

Comparações Desnecessárias

Ex: T = “abababababaabab” e P = “ababb”

À medida que se conhece o texto, as seguintes comparações em laranja abaixo são desnecessárias:

já se sabe o resultado da comparação a priori!

1	2	3	4	5	6	7	8	9	10	11	12	13	14
a	b	a	b	a	b	a	b	a	a	b	a	b	b
a	**b**	**a**	**b**	**b**									
→	**a**												
	→	**a**	**b**	**a**	**b**	**b**							
		→	**a**										

Comparações Desnecessárias

Ex: T = “abababababaabab” e P = “ababb”

À medida que se conhece o texto, as seguintes comparações em laranja abaixo são desnecessárias:

já se sabe o resultado da comparação a priori!

1	2	3	4	5	6	7	8	9	10	11	12	13	14
a	b	a	b	a	b	a	b	a	a	b	a	b	b
a	**b**	**a**	**b**	**b**									
→	**a**												
	→	**a**	**b**	**a**	**b**	**b**							
		→	**a**										
			→	**a**	**b**	**a**	**b**	**b**					

Comparações Desnecessárias

Ex: T = “abababababaabab” e P = “ababb”

À medida que se conhece o texto, as seguintes comparações em laranja abaixo são desnecessárias:

já se sabe o resultado da comparação a priori!

1	2	3	4	5	6	7	8	9	10	11	12	13	14
a	b	a	b	a	b	a	b	a	a	b	a	b	b
a	**b**	**a**	**b**	**b**									
→	**a**												
	→	**a**	**b**	**a**	**b**	**b**							
		→	**a**										
			→	**a**	**b**	**a**	**b**	**b**					
				→	**a**								

Comparações Desnecessárias

Ex: T = “abababababaabab” e P = “ababb”

À medida que se conhece o texto, as seguintes comparações em laranja abaixo são desnecessárias:

já se sabe o resultado da comparação a priori!

1	2	3	4	5	6	7	8	9	10	11	12	13	14
a	b	a	b	a	b	a	b	a	a	b	a	b	b
a	**b**	**a**	**b**	**b**									
→	**a**												
	→	**a**	**b**	**a**	**b**	**b**							
		→	**a**										
			→	**a**	**b**	**a**	**b**	**b**					
				→	**a**								
					→	**a**	**b**	**a**	**b**				

Comparações Desnecessárias

Ex: T = “abababababaabab” e P = “ababb”

À medida que se conhece o texto, as seguintes comparações em laranja abaixo são desnecessárias:

já se sabe o resultado da comparação a priori!

1	2	3	4	5	6	7	8	9	10	11	12	13	14
a	b	a	b	a	b	a	b	a	a	b	a	b	b
a	**b**	**a**	**b**	**b**									
→	**a**												
	→	**a**	**b**	**a**	**b**	**b**							
		→	**a**										
			→	**a**	**b**	**a**	**b**	**b**					
				→	**a**								
					→	**a**	**b**	**a**	**b**				
						→	**a**						

Comparações Desnecessárias

Ex: T = “abababababaabab” e P = “ababb”

À medida que se conhece o texto, as seguintes comparações em laranja abaixo são desnecessárias:

já se sabe o resultado da comparação a priori!

1	2	3	4	5	6	7	8	9	10	11	12	13	14
a	b	a	b	a	b	a	b	a	a	b	a	b	b
a	**b**	**a**	**b**	**b**									
→	**a**												
	→	**a**	**b**	**a**	**b**	**b**							
		→	**a**										
			→	**a**	**b**	**a**	**b**	**b**					
				→	**a**								
					→	**a**	**b**	**a**	**b**				
						→	**a**						
							→	**a**	**b**				

Comparações Desnecessárias

Ex: T = “abababababaabab” e P = “ababb”

À medida que se conhece o texto, as seguintes comparações em laranja abaixo são desnecessárias:

já se sabe o resultado da comparação a priori!

1	2	3	4	5	6	7	8	9	10	11	12	13	14
a	b	a	b	a	b	a	b	a	a	b	a	b	b
a	**b**	**a**	**b**	**b**									
→	**a**												
	→	**a**	**b**	**a**	**b**	**b**							
		→	**a**										
			→	**a**	**b**	**a**	**b**	**b**					
				→	**a**								
					→	**a**	**b**	**a**	**b**				
						→	**a**						
							→	**a**	**b**				
								→	**a**	**b**	**a**	**b**	**b**

O que fazer para evitar as comparações desnecessárias?

Princípio geral do Algoritmo baseado em Autômatos:

Uma vez que se conhece o texto até uma posição i, com qual caractere do padrão deve-se comparar **o próximo caractere do texto**?

1	2	3	4	5	6	7	8	9	10	11	12	13	14
a	b	a	b	a	b	a	b	a	a	b	a	b	b

O que fazer para evitar as comparações desnecessárias?

Princípio geral do Algoritmo baseado em Autômatos:

Uma vez que se conhece o texto até uma posição i, com qual caractere do padrão deve-se comparar **o próximo caractere do texto**?

1	2	3	4	5	6	7	8	9	10	11	12	13	14
a	b	a	b	a	b	a	b	a	a	b	a	b	b
a	**b**	**a**	**b**	**b**									

O que fazer para evitar as comparações desnecessárias?

Princípio geral do Algoritmo baseado em Autômatos:

Uma vez que se conhece o texto até uma posição i, com qual caractere do padrão deve-se comparar **o próximo caractere do texto**?

1	2	3	4	5	6	7	8	9	10	11	12	13	14
a	b	a	b	a	b	a	b	a	a	b	a	b	b

a **b** **a** **b** **b**

a **b** **a** **b** b ← Deslocamento de uma posição? Não!

O que fazer para evitar as comparações desnecessárias?

Princípio geral do Algoritmo baseado em Autômatos:

Uma vez que se conhece o texto até uma posição i, com qual caractere do padrão deve-se comparar **o próximo caractere do texto**?

1	2	3	4	5	6	7	8	9	10	11	12	13	14
a	b	a	b	a	b	a	b	a	a	b	a	b	b

a **b** **a** **b** **b**

a **b** **a** **b** b ← Deslocamento de uma posição? Não!

a **b** **a** **b** b ← De duas? **Sim!**

O que fazer para evitar as comparações desnecessárias?

Princípio geral do Algoritmo baseado em Autômatos:

Uma vez que se conhece o texto até uma posição i, com qual caractere do padrão deve-se comparar **o próximo caractere do texto**?

1	2	3	4	5	6	7	8	9	10	11	12	13	14
a	b	a	b	a	b	a	b	a	a	b	a	b	b

a **b** **a** **b** **b**

a **b** **a** **b** b ← Deslocamento de uma posição? Não!

a **b** **a** **b** b ← De duas? **Sim!**

Próx. comp.: ↑

O que fazer para evitar as comparações desnecessárias?

Princípio geral do Algoritmo baseado em Autômatos:

Uma vez que se conhece o texto até uma posição i, com qual caractere do padrão deve-se comparar **o próximo caractere do texto**?

1	2	3	4	5	6	7	8	9	10	11	12	13	14
a	b	a	b	a	b	a	b	a	a	b	a	b	b

a **b** **a** **b** **b**

a **b** **a** **b** b ← Deslocamento de uma posição? Não!

a **b** **a** **b** b ← De duas? **Sim!**

Próx. comp.: ↑

Procura-se o maior prefixo do padrão que é sufixo da parte do texto conhecida!

O que fazer para evitar as comparações desnecessárias?

Princípio geral do Algoritmo baseado em Autômatos:

Uma vez que se conhece o texto até uma posição i, com qual caractere do padrão deve-se comparar **o próximo caractere do texto**?

1	2	3	4	5	6	7	8	9	10	11	12	13	14
a	b	a	b	a	b	a	b	a	a	b	a	b	b

a **b** **a** **b** **b**

a **b** **a** **b** b ← Deslocamento de uma posição? Não!

a **b** **a** **b** b ← De duas? **Sim!**

Próx. comp.: ↑

Procura-se o maior prefixo do padrão que é sufixo da parte do texto conhecida!

Como saber qual é? Geração de um autômato a partir do padrão!

Pré-processamento do Padrão

Padrão:

$P =$ a b a b a c a

Conjunto de caracteres que podem ocorrer no texto: $\Sigma = \{a, b, c\}$

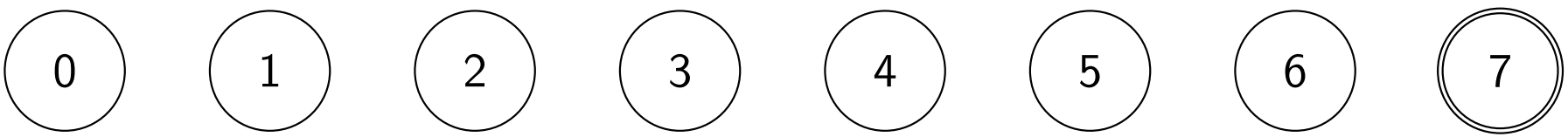
Pré-processamento do Padrão

Padrão:

$P =$ a b a b a c a

Conjunto de caracteres que podem ocorrer no texto: $\Sigma = \{a, b, c\}$

Inicialmente, criam-se estados de 0 até o tamanho do prefixo (7, para o exemplo):



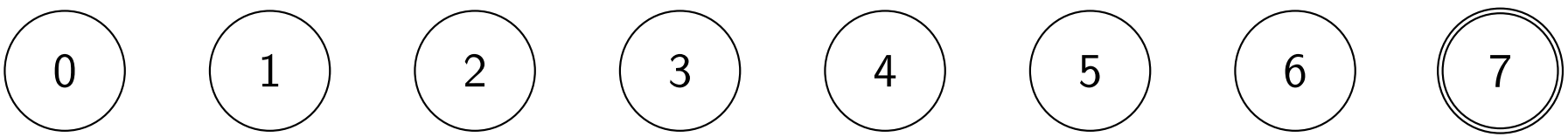
Pré-processamento do Padrão

Padrão:

P = a b a b a c a

Conjunto de caracteres que podem ocorrer no texto: $\Sigma = \{a, b, c\}$

Inicialmente, criam-se estados de 0 até o tamanho do prefixo (7, para o exemplo):



Cada estado do autômato indica **o tamanho do prefixo do padrão que casa com o sufixo conhecido do texto**

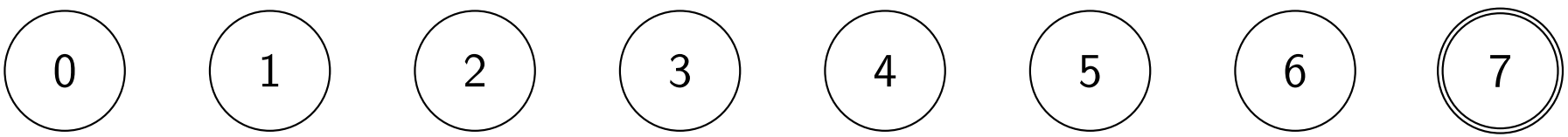
Pré-processamento do Padrão

Padrão:

P = a b a b a c a

Conjunto de caracteres que podem ocorrer no texto: $\Sigma = \{a, b, c\}$

Inicialmente, criam-se estados de 0 até o tamanho do prefixo (7, para o exemplo):



Cada estado do autômato indica **o tamanho do prefixo do padrão que casa com o sufixo conhecido do texto**

Para cada estado, verifica-se para qual estado se vai, considerando-se cada possível caractere do texto.

Pré-processamento do Padrão

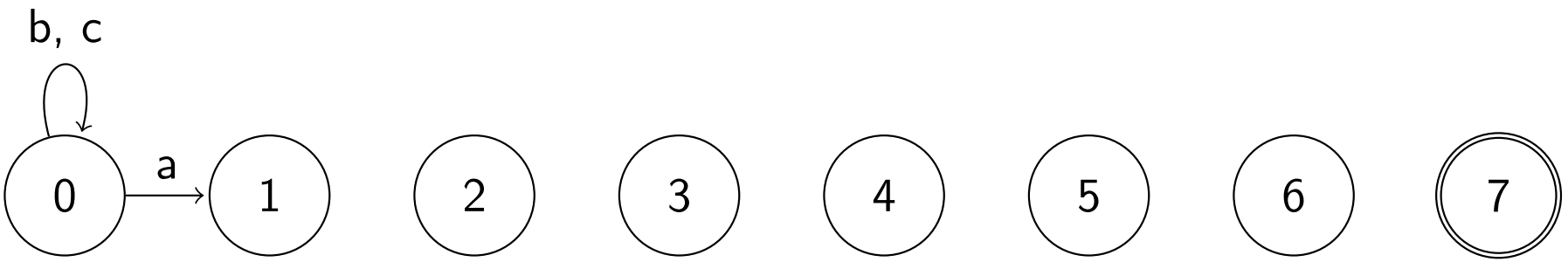
Padrão:

P = a b a b a c a

Conjunto de caracteres que podem ocorrer no texto: $\Sigma = \{a, b, c\}$

Para o caso do estado **0**:

- se o caractere do texto for **a**: vai-se para o estado 1 (expandiu-se o prefixo)
- se o caractere do texto for **b** ou **c**: permanece-se no estado 0 (não se expandiu o prefixo)



Pré-processamento do Padrão

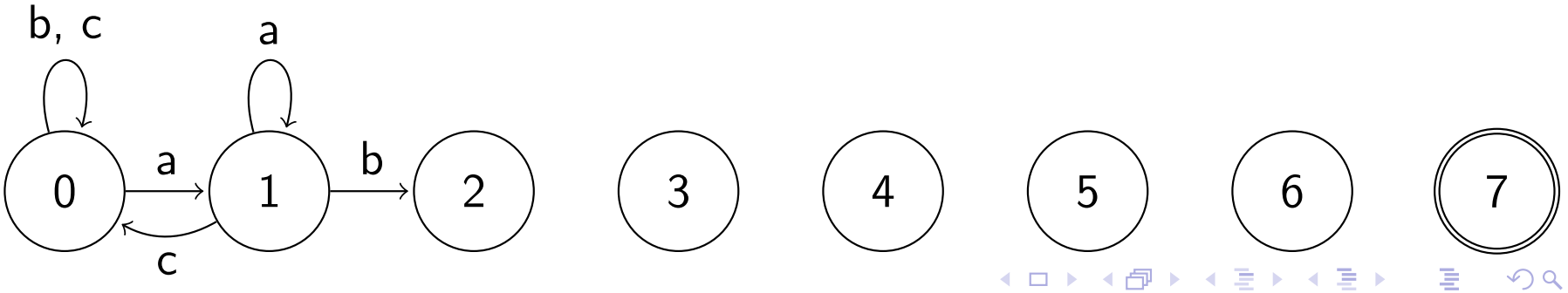
Padrão:

P = a b a b a c a

Conjunto de caracteres que podem ocorrer no texto: $\Sigma = \{a, b, c\}$

Para o caso do estado **1**:

- se o caractere do texto for **a**: permanece-se no estado 1 (não se expandiu o prefixo)
- se o caractere do texto for **b**: vai para estado 2 (expandiu-se o prefixo)
- se o caractere do texto for **c**: volta-se para o estado 0 (não há prefixo igual ao sufixo da parte conhecida do texto)



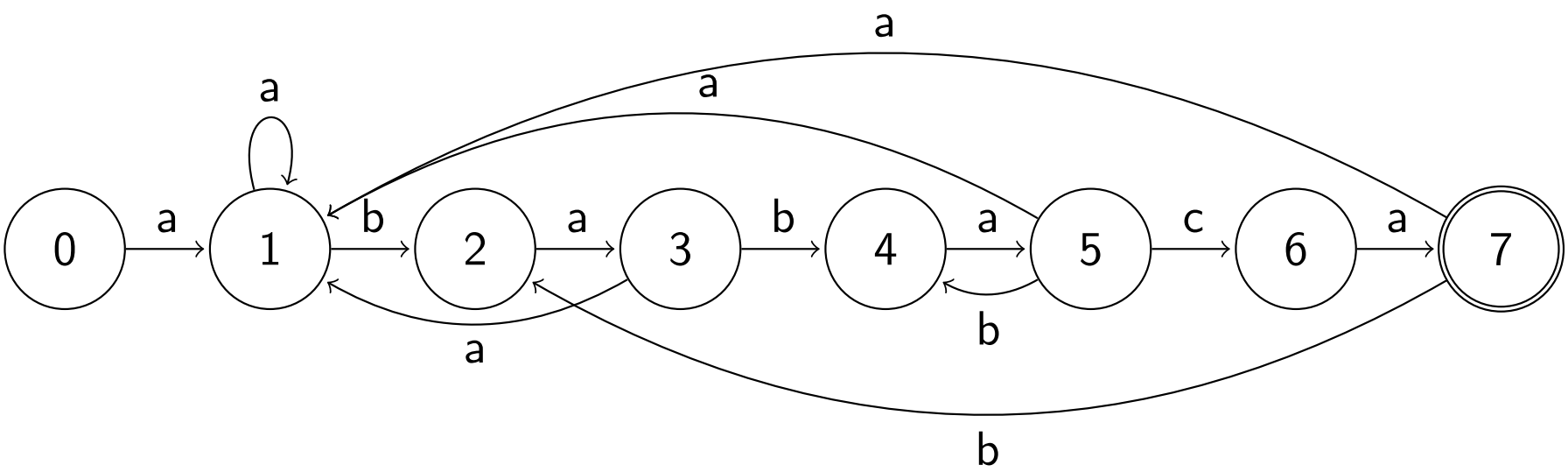
Pré-processamento do Padrão

Padrão:

$P =$ a b a b a c a

Conjunto de caracteres que podem ocorrer no texto: $\Sigma = \{a, b, c\}$

Repetindo-se o processo para todos os estados (para clareza, transições para o estado 0 não são mostradas):



Pré-processamento do Padrão: Tabela δ

Representação do autômato na tabela δ:

		Caracteres		
		a	**b**	**c**
Estados:	**0**	1	0	0
	1	1	2	0
	2	3	0	0
	3	1	4	0
	4	5	0	0
	5	1	4	6
	6	**7**	0	0
	7	1	2	0

Busca do Padrão no Texto

Tabela δ:

		Caracteres		
		a	**b**	**c**
Estados:	**0**	1	0	0
	1	1	2	0
	2	3	0	0
	3	1	4	0
	4	5	0	0
	5	1	4	6
	6	**7**	0	0
	7	1	2	0

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
a	b	a	a	a	b	a	b	a	b	a	c	a	b	b	a

s: 0

Busca do Padrão no Texto

Tabela δ:

Estados:		Caracteres a	b	c
	0	**1**	0	0
	1	1	2	0
	2	3	0	0
	3	1	4	0
	4	5	0	0
	5	1	4	6
	6	**7**	0	0
	7	1	2	0

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	a	b	a	a	a	b	a	b	a	b	a	c	a	b	b	a
s: 0	1															

Busca do Padrão no Texto

Tabela δ:

		Caracteres		
		a	**b**	**c**
Estados:	**0**	1	0	0
	1	1	**2**	0
	2	3	0	0
	3	1	4	0
	4	5	0	0
	5	1	4	6
	6	**7**	0	0
	7	1	2	0

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	a	b	a	a	a	b	a	b	a	b	a	c	a	b	b	a
s: 0	1	2														

Busca do Padrão no Texto

Tabela δ:

		Caracteres		
		a	**b**	**c**
Estados:	**0**	1	0	0
	1	1	2	0
	2	**3**	0	0
	3	1	4	0
	4	5	0	0
	5	1	4	6
	6	**7**	0	0
	7	1	2	0

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	a	b	a	a	a	b	a	b	a	b	a	c	a	b	b	a
s: 0	1	2	3													

Busca do Padrão no Texto

Tabela δ:

		Caracteres		
		a	**b**	**c**
Estados:	**0**	1	0	0
	1	1	2	0
	2	3	0	0
	3	**1**	4	0
	4	5	0	0
	5	1	4	6
	6	**7**	0	0
	7	1	2	0

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	a	b	a	a	a	b	a	b	a	b	a	c	a	b	b	a
s: 0	1	2	3	1												

Busca do Padrão no Texto

Tabela δ:

Estados:		Caracteres a	b	c
	0	1	0	0
	1	1	2	0
	2	3	0	0
	3	1	4	0
	4	5	0	0
	5	1	4	6
	6	**7**	0	0
	7	1	2	0

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	a	b	a	a	a	b	a	b	a	b	a	c	a	b	b	a
s: 0	1	2	3	1	1	2	3	4	5	4	5	6	**7**	2	**0**	**1**

Revendo as Comparações Desnecessárias

Ex: T = “abaaababababaca” (até o padrão) e P = “ababaca”

Comparações desnecessárias em relação ao algoritmo simples (em laranja):

1	2	3	4	5	6	7	8	9	10	11	12	13
a	b	a	a	a	b	a	b	a	b	a	c	a

a b a b
a
a b
a b
a b a b a c
a
a b a b a c a

Algoritmo para Calcular a Função de Transição

$P[1..m]$: padrão de tamanho m
P_k : prefixo de P de tamanho k, $1 \leq k \leq m$
$P_k \cdot a$: concatenação de P_k com caractere a
Σ: alfabeto de caracteres que podem ocorrer no texto

Algorithm 1: Algoritmo para calcular a função de transição δ

entrada : Padrão $P[1..m]$ e alfabeto Σ
saida : Função de transição δ

```
1 m ← P.length;
2 for q = 0 to m do
3     for each character a ∈ Σ do
4         k ← min(m + 1, q + 2);
5         repeat
6             k ← k − 1;
7         until P_k is a suffix of P_q · a;
8         δ(q, a) ← k;
9 return δ;
```

Algoritmo para Calcular a Função de Transição

$P[1..m]$: padrão de tamanho m
P_k : prefixo de P de tamanho k, $1 \leq k \leq m$
$P_k \cdot a$: concatenação de P_k com caractere a
Σ: alfabeto de caracteres que podem ocorrer no texto

Algorithm 2: Algoritmo para calcular a função de transição δ

entrada : Padrão $P[1..m]$ e alfabeto Σ
saida : Função de transição δ

```
1 m ← P.length;
2 for q = 0 to m do
3     for each character a ∈ Σ do
4         k ← min(m + 1, q + 2);
5         repeat
6             k ← k − 1;
7         until P_k is a suffix of P_q · a;
8         δ(q, a) ← k;
9 return δ;
```

Complexidade: $O(m^3|\Sigma|)$. Mas δ pode ser calculada em $O(m|\Sigma|)$

Algoritmo para Encontrar o Padrão no Texto

Algorithm 3: Algoritmo para encontrar o padrão no texto

entrada : Texto $T[1..n]$, Função de Transição δ, tamanho do padrão m

```
1 n ← T.length;
2 q ← 0;
3 for i = 1 to n do
4 |   q ← δ(q, T[i]);
5 |   if q = m then
6 |   |   print "Padrão ocorre com deslocamento" i − m;
```

Algoritmo para Encontrar o Padrão no Texto

Algorithm 4: Algoritmo para encontrar o padrão no texto

entrada : Texto $T[1..n]$, Função de Transição δ, tamanho do padrão m

```
1 n ← T.length;
2 q ← 0;
3 for i = 1 to n do
4 |   q ← δ(q, T[i]);
5 |   if q = m then
6 |   |   print "Padrão ocorre com deslocamento" i − m;
```

Complexidade: $\Theta(n)$

Exercício

Qual seria a tabela δ e quais seriam as transições de estado para se procurar o padrão P no texto T abaixo:

$T =$ abababcbaababababcbab

$P =$ ababcbab

Considere que o alfabeto $\Sigma = \{a, b, c\}$.

Exercício: Resposta

δ

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

	1	2	3	4	5	6	7	8	9	10	11	12
	a	b	a	b	a	b	c	b	a	a	b	a
s: 0	1	2	3	4	3	4	5	6	7	1	2	3

	13	14	15	16	17	18	19	20	21
	b	**a**	**b**	**a**	**b**	**c**	**b**	**a**	**b**
s:	4	3	4	3	4	5	6	7	**8**

Como diminuir a complexidade do cálculo de δ?

- A tabela δ inclui todos os elementos do alfabeto Σ

Como diminuir a complexidade do cálculo de δ?

- A tabela δ inclui todos os elementos do alfabeto Σ
- Para cada estado, consideram-se todos os possíveis elementos do alfabeto $\rightarrow m \cdot |\Sigma|$

Como diminuir a complexidade do cálculo de δ?

- A tabela δ inclui todos os elementos do alfabeto Σ
- Para cada estado, consideram-se todos os possíveis elementos do alfabeto $\rightarrow m \cdot |\Sigma|$
- Idéia para diminuir a complexidade: não considerar o caractere atual do texto ao se pré-processar o padrão $\rightarrow$ **apenas** m

Como diminuir a complexidade do cálculo de δ?

- A tabela δ inclui todos os elementos do alfabeto Σ
- Para cada estado, consideram-se todos os possíveis elementos do alfabeto $\rightarrow m \cdot |\Sigma|$
- Idéia para diminuir a complexidade: não considerar o caractere atual do texto ao se pré-processar o padrão $\rightarrow$ **apenas** m
- Essa é a idéia do **KMP!**

Algoritmo KMP (Knuth-Morris-Pratt)

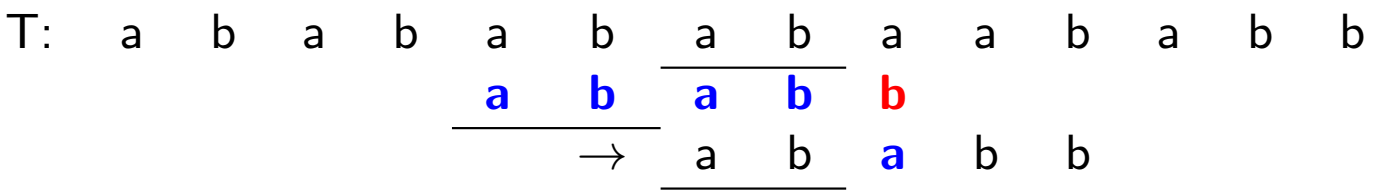
Algoritmo KMP

Considerações:

- No algoritmo baseado em autômatos, o pré-processamento do padrão requer uma tabela de tamanho $m|\Sigma|$
- Quando um elemento do padrão difere do caractere do texto, define-se um deslocamento de acordo com o maior prefixo do padrão que é igual ao sufixo do texto conhecido

Idéia: Diminuir o custo de pré-processamento do padrão considerando apenas o próprio padrão

Seja P = a b a b b



KMP: Ilustração

Ex: $T =$ "abababababaabab" e $P =$ "ababb"

1	2	3	4	5	6	7	8	9	10	11	12	13	14
a	b	a	b	a	b	a	b	a	a	b	a	b	b
a	**b**	**a**	**b**	**b**									
	→	**a**	**b**	**a**	**b**	**b**							
			→	**a**	**b**	**a**	**b**	**b**					
					→	**a**	**b**	**a**	**b**	b			
							→	**a**	**b**	a	b	b	
								→	**a**	**b**	**a**	**b**	**b**

KMP: Pré-processamento do Padrão

Objetivo

Para cada posição j de P, encontrar o tamanho do maior sufixo que é igual a um prefixo de $P[1 \ldots j]$ (de tamanho menor que j)

Ex: $P =$ "ababbababa"

j	1	2	3	4	5	6	7	8	9	10
P	**a**	**b**	**a**	**b**	**b**	**a**	**b**	**a**	**b**	**a**
π										

KMP: Pré-processamento do Padrão

Objetivo

Para cada posição j de P, encontrar o tamanho do maior sufixo que é igual a um prefixo de $P[1 \ldots j]$ (de tamanho menor que j)

Ex: $P =$ "ababbababa"

j	1	2	3	4	5	6	7	8	9	10
P	**a**	**b**	**a**	**b**	**b**	**a**	**b**	**a**	**b**	**a**
π	0									

KMP: Pré-processamento do Padrão

Objetivo

Para cada posição j de P, encontrar o tamanho do maior sufixo que é igual a um prefixo de $P[1 \ldots j]$ (de tamanho menor que j)

Ex: $P =$ "ababbababa"

j	1	2	3	4	5	6	7	8	9	10
P	**a**	**b**	**a**	**b**	**b**	**a**	**b**	**a**	**b**	**a**
π	0	0								

KMP: Pré-processamento do Padrão

Objetivo

Para cada posição j de P, encontrar o tamanho do maior sufixo que é igual a um prefixo de $P[1 \ldots j]$ (de tamanho menor que j)

Ex: $P =$ "ababbababa"

j	1	2	3	4	5	6	7	8	9	10
P	**a**	**b**	**a**	**b**	**b**	**a**	**b**	**a**	**b**	**a**
π	0	0	1							

KMP: Pré-processamento do Padrão

Objetivo

Para cada posição j de P, encontrar o tamanho do maior sufixo que é igual a um prefixo de $P[1 \ldots j]$ (de tamanho menor que j)

Ex: $P =$ "ababbababa"

j	1	2	3	4	5	6	7	8	9	10
P	**a**	**b**	**a**	**b**	**b**	**a**	**b**	**a**	**b**	**a**
π	0	0	1	2						

KMP: Pré-processamento do Padrão

Objetivo

Para cada posição j de P, encontrar o tamanho do maior sufixo que é igual a um prefixo de $P[1 \ldots j]$ (de tamanho menor que j)

Ex: $P =$ "ababbababa"

j	1	2	3	4	5	6	7	8	9	10
P	**a**	**b**	**a**	**b**	**b**	**a**	**b**	**a**	**b**	**a**
π	0	0	1	2	0					

KMP: Pré-processamento do Padrão

Objetivo

Para cada posição j de P, encontrar o tamanho do maior sufixo que é igual a um prefixo de $P[1 \ldots j]$ (de tamanho menor que j)

Ex: $P =$ "ababbababa"

j	1	2	3	4	5	6	7	8	9	10
P	**a**	**b**	**a**	**b**	**b**	**a**	**b**	**a**	**b**	**a**
π	0	0	1	2	0	1				

KMP: Pré-processamento do Padrão

Objetivo

Para cada posição j de P, encontrar o tamanho do maior sufixo que é igual a um prefixo de $P[1 \ldots j]$ (de tamanho menor que j)

Ex: $P =$ "ababbababa"

j	1	2	3	4	5	6	7	8	9	10
P	**a**	**b**	**a**	**b**	**b**	**a**	**b**	**a**	**b**	**a**
π	0	0	1	2	0	1	2			

KMP: Pré-processamento do Padrão

Objetivo

Para cada posição j de P, encontrar o tamanho do maior sufixo que é igual a um prefixo de $P[1 \ldots j]$ (de tamanho menor que j)

Ex: $P =$ "ababbababa"

j	1	2	3	4	5	6	7	8	9	10
P	**a**	**b**	**a**	**b**	**b**	**a**	**b**	**a**	**b**	**a**
π	0	0	1	2	0	1	2	3		

KMP: Pré-processamento do Padrão

Objetivo

Para cada posição j de P, encontrar o tamanho do maior sufixo que é igual a um prefixo de $P[1 \ldots j]$ (de tamanho menor que j)

Ex: $P =$ "ababbababa"

j	1	2	3	4	5	6	7	8	9	10
P	**a**	**b**	**a**	**b**	**b**	**a**	**b**	**a**	**b**	**a**
π	0	0	1	2	0	1	2	3	4	

KMP: Pré-processamento do Padrão

Objetivo

Para cada posição j de P, encontrar o tamanho do maior sufixo que é igual a um prefixo de $P[1 \ldots j]$ (de tamanho menor que j)

Ex: $P =$ "ababbababa"

j	1	2	3	4	5	6	7	8	9	10
P	**a**	**b**	**a**	**b**	**b**	**a**	**b**	**a**	**b**	**a**
π	0	0	1	2	0	1	2	3	4	3

KMP: Ilustração

Ex: T = “abaaababababaca” e P = “ababaca”

j	1	2	3	4	5	6	7
P	**a**	**b**	**a**	**b**	**a**	**c**	**a**
π	0	0	1	2	3	0	1

1	2	3	4	5	6	7	8	9	10	11	12	13
a	b	a	a	a	b	a	b	a	b	a	c	a
a	**b**	**a**	**b**									
		a	**b**									
			a	**b**								
				a	**b**	**a**	**b**	**a**	**c**			
						a	**b**	**a**	**b**	**a**	**c**	**a**

KMP: Ilustração

Ex: T = “abaaababababaca” e P = “ababaca”

j	1	2	3	4	5	6	7
P	**a**	**b**	**a**	**b**	**a**	**c**	**a**
π	0	0	1	2	3	0	1

1	2	3	4	5	6	7	8	9	10	11	12	13
a	b	a	a	a	b	a	b	a	b	a	c	a
a	**b**	**a**	**b**									
		a	**b**									
			a	**b**								
				a	**b**	**a**	**b**	**a**	**c**			
						a	**b**	**a**	**b**	**a**	**c**	**a**

Qual é a lógica de definição da próxima comparação?

KMP: Pré-processamento do Padrão

Idéia geral:

O valor de $\pi[1] = 0$. Para se determinar o valor de $\pi[i]$, $i > 1$:

KMP: Pré-processamento do Padrão

Idéia geral:

O valor de $\pi[1] = 0$. Para se determinar o valor de $\pi[i]$, $i > 1$:

- verifica-se se é possível "estender"o maior prefixo igual a sufixo encontrado para a posição $i - 1$, ou seja, se $P[\pi[i-1]+1] = P[i]$

KMP: Pré-processamento do Padrão

Idéia geral:

O valor de $\pi[1] = 0$. Para se determinar o valor de $\pi[i]$, $i > 1$:

- verifica-se se é possível "estender"o maior prefixo igual a sufixo encontrado para a posição $i - 1$, ou seja, se $P[\pi[i-1]+1] = P[i]$
  - se sim, $\pi[i] \leftarrow \pi[i-1]+1$

KMP: Pré-processamento do Padrão

Idéia geral:

O valor de $\pi[1] = 0$. Para se determinar o valor de $\pi[i]$, $i > 1$:

- verifica-se se é possível "estender"o maior prefixo igual a sufixo encontrado para a posição $i - 1$, ou seja, se $P[\pi[i-1]+1] = P[i]$
  - se sim, $\pi[i] \leftarrow \pi[i-1]+1$
  - se não, verifica-se se é possível "estender"o próximo maior prefixo igual a sufixo, utilizando o valor de $\pi[\pi[i-1]]$

KMP: Pré-processamento do Padrão

Idéia geral:

O valor de $\pi[1] = 0$. Para se determinar o valor de $\pi[i]$, $i > 1$:

- verifica-se se é possível "estender"o maior prefixo igual a sufixo encontrado para a posição $i - 1$, ou seja, se $P[\pi[i-1]+1] = P[i]$
  - se sim, $\pi[i] \leftarrow \pi[i-1]+1$
  - se não, verifica-se se é possível "estender"o próximo maior prefixo igual a sufixo, utilizando o valor de $\pi[\pi[i-1]]$
  - repete-se o processo até ser possível "estender"um prefixo igual a sufixo ou se chegar ao fim da tabela

j	1	2	3	4	5	6	7	8	9	10	11
P	**a**	**b**	**a**	**b**	**a**	**a**	**b**	**a**	**b**	**c**	**a**
π											

KMP: Pré-processamento do Padrão

Idéia geral:

O valor de $\pi[1] = 0$. Para se determinar o valor de $\pi[i]$, $i > 1$:

- verifica-se se é possível "estender"o maior prefixo igual a sufixo encontrado para a posição $i - 1$, ou seja, se $P[\pi[i-1]+1] = P[i]$
  - se sim, $\pi[i] \leftarrow \pi[i-1]+1$
  - se não, verifica-se se é possível "estender"o próximo maior prefixo igual a sufixo, utilizando o valor de $\pi[\pi[i-1]]$
  - repete-se o processo até ser possível "estender"um prefixo igual a sufixo ou se chegar ao fim da tabela

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

KMP: Pré-processamento do Padrão

Idéia geral:

O valor de $\pi[1] = 0$. Para se determinar o valor de $\pi[i]$, $i > 1$:

- verifica-se se é possível "estender"o maior prefixo igual a sufixo encontrado para a posição $i - 1$, ou seja, se $P[\pi[i-1]+1] = P[i]$
  - se sim, $\pi[i] \leftarrow \pi[i-1]+1$
  - se não, verifica-se se é possível "estender"o próximo maior prefixo igual a sufixo, utilizando o valor de $\pi[\pi[i-1]]$
  - repete-se o processo até ser possível "estender"um prefixo igual a sufixo ou se chegar ao fim da tabela

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

KMP: Pré-processamento do Padrão

Idéia geral:

O valor de $\pi[1] = 0$. Para se determinar o valor de $\pi[i]$, $i > 1$:

- verifica-se se é possível "estender"o maior prefixo igual a sufixo encontrado para a posição $i - 1$, ou seja, se $P[\pi[i-1]+1] = P[i]$
  - se sim, $\pi[i] \leftarrow \pi[i-1]+1$
  - se não, verifica-se se é possível "estender"o próximo maior prefixo igual a sufixo, utilizando o valor de $\pi[\pi[i-1]]$
  - repete-se o processo até ser possível "estender"um prefixo igual a sufixo ou se chegar ao fim da tabela

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

KMP: Pré-processamento do Padrão

Idéia geral:

O valor de $\pi[1] = 0$. Para se determinar o valor de $\pi[i]$, $i > 1$:

- verifica-se se é possível "estender"o maior prefixo igual a sufixo encontrado para a posição $i - 1$, ou seja, se $P[\pi[i-1]+1] = P[i]$
  - se sim, $\pi[i] \leftarrow \pi[i-1]+1$
  - se não, verifica-se se é possível "estender"o próximo maior prefixo igual a sufixo, utilizando o valor de $\pi[\pi[i-1]]$
  - repete-se o processo até ser possível "estender"um prefixo igual a sufixo ou se chegar ao fim da tabela

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

KMP: Pré-processamento do Padrão

Idéia geral:

O valor de $\pi[1] = 0$. Para se determinar o valor de $\pi[i]$, $i > 1$:

- verifica-se se é possível "estender"o maior prefixo igual a sufixo encontrado para a posição $i - 1$, ou seja, se $P[\pi[i-1]+1] = P[i]$
  - se sim, $\pi[i] \leftarrow \pi[i-1]+1$
  - se não, verifica-se se é possível "estender"o próximo maior prefixo igual a sufixo, utilizando o valor de $\pi[\pi[i-1]]$
  - repete-se o processo até ser possível "estender"um prefixo igual a sufixo ou se chegar ao fim da tabela

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

KMP: Pré-processamento do Padrão

Idéia geral:

O valor de $\pi[1] = 0$. Para se determinar o valor de $\pi[i]$, $i > 1$:

- verifica-se se é possível "estender"o maior prefixo igual a sufixo encontrado para a posição $i - 1$, ou seja, se $P[\pi[i-1]+1] = P[i]$
  - se sim, $\pi[i] \leftarrow \pi[i-1]+1$
  - se não, verifica-se se é possível "estender"o próximo maior prefixo igual a sufixo, utilizando o valor de $\pi[\pi[i-1]]$
  - repete-se o processo até ser possível "estender"um prefixo igual a sufixo ou se chegar ao fim da tabela

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

KMP: Pré-processamento do Padrão

Idéia geral:

O valor de $\pi[1] = 0$. Para se determinar o valor de $\pi[i]$, $i > 1$:

- verifica-se se é possível "estender"o maior prefixo igual a sufixo encontrado para a posição $i-1$, ou seja, se $P[\pi[i-1]+1] = P[i]$
  - se sim, $\pi[i] \leftarrow \pi[i-1]+1$
  - se não, verifica-se se é possível "estender"o próximo maior prefixo igual a sufixo, utilizando o valor de $\pi[\pi[i-1]]$
  - repete-se o processo até ser possível "estender"um prefixo igual a sufixo ou se chegar ao fim da tabela

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

KMP: Pré-processamento do Padrão

Idéia geral:

O valor de $\pi[1] = 0$. Para se determinar o valor de $\pi[i]$, $i > 1$:

- verifica-se se é possível "estender"o maior prefixo igual a sufixo encontrado para a posição $i - 1$, ou seja, se $P[\pi[i-1]+1] = P[i]$
  - se sim, $\pi[i] \leftarrow \pi[i-1]+1$
  - se não, verifica-se se é possível "estender"o próximo maior prefixo igual a sufixo, utilizando o valor de $\pi[\pi[i-1]]$
  - repete-se o processo até ser possível "estender"um prefixo igual a sufixo ou se chegar ao fim da tabela

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

KMP: Pré-processamento do Padrão

Idéia geral:

O valor de $\pi[1] = 0$. Para se determinar o valor de $\pi[i]$, $i > 1$:

- verifica-se se é possível "estender"o maior prefixo igual a sufixo encontrado para a posição $i - 1$, ou seja, se $P[\pi[i-1]+1] = P[i]$
  - se sim, $\pi[i] \leftarrow \pi[i-1]+1$
  - se não, verifica-se se é possível "estender"o próximo maior prefixo igual a sufixo, utilizando o valor de $\pi[\pi[i-1]]$
  - repete-se o processo até ser possível "estender"um prefixo igual a sufixo ou se chegar ao fim da tabela

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

KMP: Pré-processamento do Padrão

Idéia geral:

O valor de $\pi[1] = 0$. Para se determinar o valor de $\pi[i]$, $i > 1$:

- verifica-se se é possível "estender"o maior prefixo igual a sufixo encontrado para a posição $i - 1$, ou seja, se $P[\pi[i-1]+1] = P[i]$
  - se sim, $\pi[i] \leftarrow \pi[i-1]+1$
  - se não, verifica-se se é possível "estender"o próximo maior prefixo igual a sufixo, utilizando o valor de $\pi[\pi[i-1]]$
  - repete-se o processo até ser possível "estender"um prefixo igual a sufixo ou se chegar ao fim da tabela

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

KMP: Pré-processamento do Padrão

Idéia geral:

O valor de $\pi[1] = 0$. Para se determinar o valor de $\pi[i]$, $i > 1$:

- verifica-se se é possível "estender"o maior prefixo igual a sufixo encontrado para a posição $i-1$, ou seja, se $P[\pi[i-1]+1] = P[i]$
  - se sim, $\pi[i] \leftarrow \pi[i-1]+1$
  - se não, verifica-se se é possível "estender"o próximo maior prefixo igual a sufixo, utilizando o valor de $\pi[\pi[i-1]]$
  - repete-se o processo até ser possível "estender"um prefixo igual a sufixo ou se chegar ao fim da tabela

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

KMP: Pré-processamento do Padrão

Pergunta de fixação:

Quais comparações o algoritmo KMP realiza para encontrar o valor 0 para a posição $\pi[12]$ e quais os resultados das comparações?

i	1	2	3	4	5	6	7	8	9	10	11	12
p	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}
π	0	0	1	2	0	1	2	3	4	5	6	0

KMP: Pré-processamento do Padrão

Pergunta de fixação:

Quais comparações o algoritmo KMP realiza para encontrar o valor 0 para a posição $\pi[12]$ e quais os resultados das comparações?

i	1	2	3	4	5	6	7	8	9	10	11	12
p	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}
π	0	0	1	2	0	1	2	3	4	5	6	0

$$c_{12} \neq c_7$$

KMP: Pré-processamento do Padrão

Pergunta de fixação:

Quais comparações o algoritmo KMP realiza para encontrar o valor 0 para a posição $\pi[12]$ e quais os resultados das comparações?

i	1	2	3	4	5	6	7	8	9	10	11	12
p	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}
π	0	0	1	2	0	1	2	3	4	5	6	0

$$c_{12} \neq c_2$$

KMP: Pré-processamento do Padrão

Pergunta de fixação:

Quais comparações o algoritmo KMP realiza para encontrar o valor 0 para a posição $\pi[12]$ e quais os resultados das comparações?

i	1	2	3	4	5	6	7	8	9	10	11	12
p	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}
π	0	0	1	2	0	1	2	3	4	5	6	0

$c_{12} \neq c_1$ Portanto $\pi[12] \leftarrow 0$

KMP: Pré-processamento do Padrão

Algorithm 5: COMPUTE_PREFIX_FUNCTION (P)

entrada : Padrão $P[1..m]$
saida : Tabela π

```
1  m ← P.length;
2  let π[1..m] be a new array;
3  π[0] ← 0
4  q ← 0;
5  for i ← 2 to m do
6      while q > 0 and P[q + 1] ≠ P[i] do
7          q ← π[q];
8      if P[q + 1] = P[i] then
9          q ← q + 1;
10     π[i] ← q;
11 return π;
```

KMP: Pré-processamento do Padrão

Encontre os valores da tabela π para o padrão $P =$ "ababbababa", seguindo o algoritmo *COMPUTE_PREFIX_FUNCTION*.

Resposta:

j	1	2	3	4	5	6	7	8	9	10
P	**a**	**b**	**a**	**b**	**b**	**a**	**b**	**a**	**b**	**a**
π										

KMP: Pré-processamento do Padrão

Encontre os valores da tabela π para o padrão $P =$ "ababbababa", seguindo o algoritmo *COMPUTE_PREFIX_FUNCTION*.

Resposta:

j	1	2	3	4	5	6	7	8	9	10
P	**a**	**b**	**a**	**b**	**b**	**a**	**b**	**a**	**b**	**a**
π	0									

KMP: Pré-processamento do Padrão

Encontre os valores da tabela π para o padrão $P =$ "ababbababa", seguindo o algoritmo *COMPUTE_PREFIX_FUNCTION*.

Resposta:

j	1	2	3	4	5	6	7	8	9	10
P	**a**	**b**	**a**	**b**	**b**	**a**	**b**	**a**	**b**	**a**
π	0	0								

KMP: Pré-processamento do Padrão

Encontre os valores da tabela π para o padrão $P =$ "ababbababa", seguindo o algoritmo *COMPUTE_PREFIX_FUNCTION*.

Resposta:

j	1	2	3	4	5	6	7	8	9	10
P	**a**	**b**	**a**	**b**	**b**	**a**	**b**	**a**	**b**	**a**
π	0	0	1							

KMP: Pré-processamento do Padrão

Encontre os valores da tabela π para o padrão $P =$ "ababbababa", seguindo o algoritmo *COMPUTE_PREFIX_FUNCTION*.

Resposta:

j	1	2	3	4	5	6	7	8	9	10
P	**a**	**b**	**a**	**b**	**b**	**a**	**b**	**a**	**b**	**a**
π	0	0	1	2						

KMP: Pré-processamento do Padrão

Encontre os valores da tabela π para o padrão $P =$ "ababbababa", seguindo o algoritmo *COMPUTE_PREFIX_FUNCTION*.

Resposta:

j	1	2	3	4	5	6	7	8	9	10
P	**a**	**b**	**a**	**b**	**b**	**a**	**b**	**a**	**b**	**a**
π	0	0	1	2	0					

KMP: Pré-processamento do Padrão

Encontre os valores da tabela π para o padrão $P =$ "ababbababa", seguindo o algoritmo *COMPUTE_PREFIX_FUNCTION*.

Resposta:

j	1	2	3	4	5	6	7	8	9	10
P	**a**	**b**	**a**	**b**	**b**	**a**	**b**	**a**	**b**	**a**
π	0	0	1	2	0	1				

KMP: Pré-processamento do Padrão

Encontre os valores da tabela π para o padrão $P =$ "ababbababa", seguindo o algoritmo *COMPUTE_PREFIX_FUNCTION*.

Resposta:

j	1	2	3	4	5	6	7	8	9	10
P	**a**	**b**	**a**	**b**	**b**	**a**	**b**	**a**	**b**	**a**
π	0	0	1	2	0	1	2			

KMP: Pré-processamento do Padrão

Encontre os valores da tabela π para o padrão $P =$ "ababbababa", seguindo o algoritmo *COMPUTE_PREFIX_FUNCTION*.

Resposta:

j	1	2	3	4	5	6	7	8	9	10
P	**a**	**b**	**a**	**b**	**b**	**a**	**b**	**a**	**b**	**a**
π	0	0	1	2	0	1	2	3		

KMP: Pré-processamento do Padrão

Encontre os valores da tabela π para o padrão $P =$ "ababbababa", seguindo o algoritmo *COMPUTE_PREFIX_FUNCTION*.

Resposta:

j	1	2	3	4	5	6	7	8	9	10
P	**a**	**b**	**a**	**b**	**b**	**a**	**b**	**a**	**b**	**a**
π	0	0	1	2	0	1	2	3	4	

KMP: Pré-processamento do Padrão

Encontre os valores da tabela π para o padrão $P =$ "ababbababa", seguindo o algoritmo *COMPUTE_PREFIX_FUNCTION*.

Resposta:

j	1	2	3	4	5	6	7	8	9	10
P	**a**	**b**	**a**	**b**	**b**	**a**	**b**	**a**	**b**	**a**
π	0	0	1	2	0	1	2	3	4	3

KMP: Busca

Algorithm 6: KMP_Matcher(T, P)

entrada : Texto $T[1..n]$ e Padrão $P[1..m]$

```
1  n ← T.length;
2  m ← P.length;
3  π ← COMPUTE_PREFIX_FUNCTION(P);
4  q ← 0;
5  for i = 1 to n do
6      while q > 0 and P[q + 1] ≠ T[i] do
7          q ← π[q];
8      if P[q + 1] = T[i] then
9          q ← q + 1;
10     if q = m then
11         print "Pattern occurs with shift" i − m;
12         q ← π[q];
```

Exercício

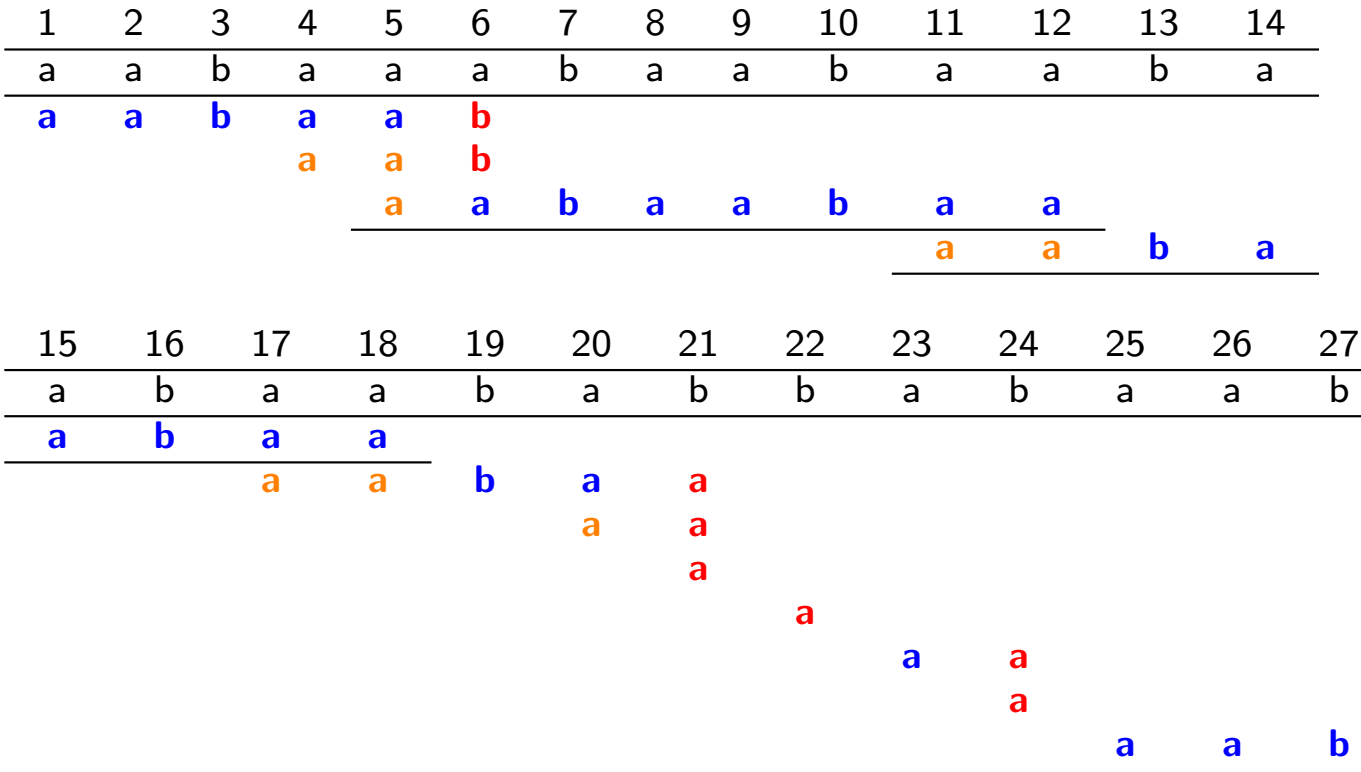
Para o texto e o padrão abaixo:

$T =$ "aabaaabaabaabaabaababbabaab"
$P =$ "aabaabaa"

apresente a tabela π e as comparações realizadas para se encontrar as ocorrências de P em T.

Exercício - Resposta

i	1	2	3	4	5	6	7	8
P	a	a	b	a	a	b	a	a
π	0	1	0	1	2	3	1	2



KMP: Complexidade

- O pré-processamento do padrão possui complexidade $\Theta(m)$

  Argumentos informais:

  - q é sempre positivo e somente se acrescenta seu valor no máximo $m-1$ vezes (*COMPUTE_PREFIX_FUNCTION*, linha 9)
  - para cada posição i somente se pode diminuir o valor de q até zero
  - portanto, não se fazem mais do que $2m$ comparações

- Por argumento análogo, o procedimento de busca é $\Theta(n)$

Algoritmo de Boyer-Moore

Idéia Básica

Ao invés de se comparar os caracteres do texto com os caracteres do padrão a partir do início do padrão ... **compara-se a partir do fim!**

Considere o texto abaixo e o padrão $P = abacabb$, tamanho $m = 7$.

Se compararmos o primeiro caractere do padrão com o primeiro caractere do texto, qual deslocamento podemos fazer?

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
T :	**b**	a	b	b	a	b	d	a	b	b	a	a	c	a	b	a	c	a	b	b
	a	b	a	c	a	b	b													

Idéia Básica

Ao invés de se comparar os caracteres do texto com os caracteres do padrão a partir do início do padrão ... **compara-se a partir do fim!**

Considere o texto abaixo e o padrão $P = abacabb$, tamanho $m = 7$.

Se compararmos o primeiro caractere do padrão com o primeiro caractere do texto, qual deslocamento podemos fazer?

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
T :	**b**	a	b	b	a	b	d	a	b	b	a	a	c	a	b	a	c	a	b	b
	a	b	a	c	a	b	b													
	→	a	b	a	c	a	b	b												

Apenas avanço de uma posição

Idéia Básica

Ao invés de se comparar os caracteres do texto com os caracteres do padrão a partir do início do padrão ... **compara-se a partir do fim!**

Considere o texto abaixo e o padrão $P = abacabb$, tamanho $m = 7$.

E se compararmos o sétimo elemento do padrão b com o sétimo caractere do texto?

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
T :	b	a	b	b	a	b	**d**	a	b	b	a	a	c	a	b	a	c	a	b	b
	a	b	a	c	a	b	**b**													

Idéia Básica

Ao invés de se comparar os caracteres do texto com os caracteres do padrão a partir do início do padrão ... **compara-se a partir do fim!**

Considere o texto abaixo e o padrão $P = abacabb$, tamanho $m = 7$.

E se compararmos o sétimo elemento do padrão b com o sétimo caractere do texto?

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
T :	b	a	b	b	a	b	**d**	a	b	b	a	a	c	a	b	a	c	a	b	b
	a	b	a	c	a	b	**b**													
							$\rightarrow$	a	b	a	c	a	b	b						

Avanço de 7 posições, pois **d** não ocorre no padrão!

Idéia Básica

Ao invés de se comparar os caracteres do texto com os caracteres do padrão a partir do início do padrão ... **compara-se a partir do fim!**

Considere o texto abaixo e o padrão $P = abacabb$, tamanho $m = 7$.

E qual descolamento pode ser feito agora?

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
T :	b	a	b	b	a	b	d	a	b	b	a	a	c	**a**	b	a	c	a	b	b
	a	b	a	c	a	b	b													
								a	b	a	c	a	b	**b**						

Idéia Básica

Ao invés de se comparar os caracteres do texto com os caracteres do padrão a partir do início do padrão ... **compara-se a partir do fim!**

Considere o texto abaixo e o padrão $P = abacabb$, tamanho $m = 7$.

E se compararmos o sétimo elemento do padrão b com o sétimo caractere do texto?

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
T :	b	a	b	b	a	b	d	a	b	b	a	a	c	a	b	a	c	a	b	b
	a	b	a	c	a	b	b													
								a	b	a	c	a	b	**b**						
									→	a	b	a	c	**a**	b	b				

Avanço de 2 posições, para se alinhar o **a** do texto (posição 14) com o **a** mais à direita do padrão (posição 5)!

Idéia Básica

Ao invés de se comparar os caracteres do texto com os caracteres do padrão a partir do início do padrão ... **compara-se a partir do fim!**

Considere o texto abaixo e o padrão $P = abacabb$, tamanho $m = 7$.

Seguindo o mesmo princípio:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
T :	b	a	b	b	a	b	d	a	b	b	a	a	c	a	b	a	c	a	b	b
	a	b	a	c	a	b	b													
								a	b	a	c	a	b	b						
										a	b	a	c	a	b	**b**				
											→	a	b	a	c	a	b	**b**		
													→	**a**	**b**	**a**	**c**	**a**	**b**	**b**

Algoritmo Boyer-Moore

Observe que foram feitas **11** comparações para se encontrar o padrão na posição **14**!

Ou seja, menos comparações do que o número de caracteres até a posição em que o padrão ocorre!

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
T :	b	a	b	b	a	b	**d**	a	b	b	a	a	c	a	b	a	c	a	b	b
	a	b	a	c	a	b	**b**													
								a	b	a	c	a	b	**b**						
										a	b	a	c	a	b	**b**				
											→	a	b	a	c	a	b	**b**		
													→	**a**	**b**	**a**	**c**	**a**	**b**	**b**

Tabela *Delta 1*: Δ_1

O deslocamento relativo a esse critério é definido pela tabela Δ_1.

Δ_1 tem uma entrada para cada símbolo do alfabeto.

Tabela Δ_1:

Seja Σ um alfabeto, $P[1..m]$ um padrão e $c \in \Sigma$:

$$\Delta_1(c) = \begin{cases} m & \text{if } c \text{ não ocorre em } P \\ m - j & \text{onde } j \text{ é o máximo inteiro tal que } P[j] = c \end{cases}$$

Para o exemplo anterior:

Seja $\Sigma = \{a, b, c, d\}$ e $P = abacabb$ $(m = 7)$:

i	1	2	3	4	5	6	7
P	a	b	a	c	a	b	b

$\Delta_1(a) = 7 - 5 = 2$ $\quad$ $\Delta_1(b) = 7 - 7 = 0$

$\Delta_1(c) = 7 - 4 = 3$ $\quad$ $\Delta_1(char) = 7$, para todo $char \notin \{a, b, c\}$

Segundo Critério para Deslocamento

Além do critério usado para deslocamento anterior, o algoritmo também considera o deslocamento que pode ser feito considerando-se o sufixo do padrão já encontrado no texto.

Exemplo:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
T :	b	a	b	b	a	d	**b**	**b**	**b**	a	b	b	c	b	b	c	b	b
	a	b	b	c	b	b	**c**	**b**	**b**									

Segundo Critério para Deslocamento

Além do critério usado para deslocamento anterior, o algoritmo também considera o deslocamento que pode ser feito considerando-se o sufixo do padrão já encontrado no texto.

Exemplo:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
T :	b	a	b	b	a	d	**b**	**b**	**b**	a	b	b	c	b	b	c	b	b
	a	b	b	c	b	b	**c**	**b**	**b**									
						→	**a**	**b**	**b**	c	b	b	c	b	b			

Procura-se pela próxima ocorrência de **bb** no padrão, sendo que o caractere anterior seja diferente de **c**

Segundo Critério para Deslocamento

Além do critério usado para deslocamento anterior, o algoritmo também considera o deslocamento que pode ser feito considerando-se o sufixo do padrão já encontrado no texto.

Exemplo:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
T :	b	a	b	b	a	d	b	b	b	**a**	**b**	**b**	**c**	**b**	**b**	c	b	b
	a	b	b	c	b	b	c	b	b									
						→	a	b	b	**c**	**b**	**b**	**c**	**b**	**b**			

Segundo Critério para Deslocamento

Além do critério usado para deslocamento anterior, o algoritmo também considera o deslocamento que pode ser feito considerando-se o sufixo do padrão já encontrado no texto.

Exemplo:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
T :	b	a	b	b	a	d	b	b	b	**a**	**b**	**b**	**c**	**b**	**b**	c	b	b
	a	b	b	c	b	b	c	b	b									
						→	a	b	b	**c**	**b**	**b**	**c**	**b**	**b**			

Procura-se pela próxima ocorrência de **bbcbb** no padrão, sendo que o caractere anterior seja diferente de **c**

Segundo Critério para Deslocamento

Além do critério usado para deslocamento anterior, o algoritmo também considera o deslocamento que pode ser feito considerando-se o sufixo do padrão já encontrado no texto.

Exemplo:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
T :	b	a	b	b	a	d	b	b	b	**a**	**b**	**b**	**c**	**b**	**b**	**c**	**b**	**b**
	a	b	b	c	b	b	c	b	b									
						→	a	b	b	c	b	b	c	b	b			
									→	**a**	**b**	**b**	**c**	**b**	**b**	c	b	b

Procura-se pela próxima ocorrência de **bbcbb** no padrão, sendo que o caractere anterior seja diferente de **c**

Segundo Critério para Deslocamento

Além do critério usado para deslocamento anterior, o algoritmo também considera o deslocamento que pode ser feito considerando-se o sufixo do padrão já encontrado no texto.

Exemplo:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
T :	b	a	b	b	a	d	b	b	b	**a**	**b**	**b**	**c**	**b**	**b**	**c**	**b**	**b**
	a	b	b	c	b	b	c	b	b									
						→	a	b	b	c	b	b	c	b	b			
									→	**a**	**b**	**b**	**c**	**b**	**b**	**c**	**b**	**b**

Tabela *Delta 2*: Δ_2

O segundo critério de deslocamento é definido na tabela Δ_2

A tabela Δ_2 possui uma entrada para cada elemento do padrão.

Tabela *Delta 2*: Δ_2

O segundo critério de deslocamento é definido na tabela Δ_2

A tabela Δ_2 possui uma entrada para cada elemento do padrão.

Considere:

- que o padrão P seja estendido com elementos de índice 0, -1, -2, ... e que essas posições terão um caractere especial **$**, que não ocorre em P

Tabela *Delta 2*: Δ_2

O segundo critério de deslocamento é definido na tabela Δ_2

A tabela Δ_2 possui uma entrada para cada elemento do padrão.

Considere:

- que o padrão P seja estendido com elementos de índice 0, -1, -2, ... e que essas posições terão um caractere especial **$**, que não ocorre em P
- que duas sequências $[c_1, c_2, ..., c_k]$ e $[d_1, d_2, ..., d_k]$ se "unificam", se:

  para todo $i = 1...k$: ou $c_i = d_i$ ou $c_i = \$$ ou $d_i = \$$

Tabela *Delta 2*: Δ_2

O segundo critério de deslocamento é definido na tabela Δ_2

A tabela Δ_2 possui uma entrada para cada elemento do padrão.

Considere:

- que o padrão P seja estendido com elementos de índice 0, -1, -2, ... e que essas posições terão um caractere especial **$**, que não ocorre em P
- que duas sequências $[c_1, c_2, ..., c_k]$ e $[d_1, d_2, ..., d_k]$ se "unificam", se:

  para todo $i = 1...k$: ou $c_i = d_i$ ou $c_i = \$$ ou $d_i = \$$

  Por exemplo, as sequências abaixo unificam-se:

  Sequência 1: $ $ $ $ a b c d
  Sequência 2: a b a b a b c d

Tabela *Delta 2*: Δ_2

Rightmost Plausible Reoccurrence (rpr)

Para a *substring* que se inicia na posição $j+1$, $rpr(j)$, para $j = 1 \cdots m-1$, é o maior k menor que ou igual a m tal que:

$$P[j+1...m] \text{ e } P[k...k+m-j-1] \text{ se unificam}$$

e

$$\text{ou } k \leq 1 \text{ ou } P[k-1] \neq P[j]$$

Tabela Δ_2:

$$\Delta_2(j) = m + 1 - rpr(j)$$

Tabela Δ_2: Exemplo

j	1	2	3	4	5	6	7	8	9
P	a	b	y	x	c	d	e	y	x
Δ_2	17	16	15	14	13	12	7	10	1

$\Delta_2(m)$ pode ser sempre 1

Por que $\Delta_2(7) = 7$?

$rpr(7) = 3$, pois a próxima ocorrência de yx no padrão em que o caractere anterior não é e é na posição 3.

Com isso, $\Delta_2(7) = m + 1 - rpr(7) = 9 + 1 - 3 = 7$

Mas qual é a razão? Porque o deslocamento possível, quando a comparação com o $P[7]$ der sem sucesso, corresponde a um deslocamento de 7:

j	1	2	3	4	5	6	7	8	9					
P	a	b	y	x	c	d	**e**	**y**	**x**					
						a	**b**	**y**	**x**	c	d	e	y	**x**
							↑	deslocamento de 7						↑

Tabela Δ_2: Exemplo

j	1	2	3	4	5	6	7	8	9
P	a	b	y	x	c	d	e	y	x
Δ_2	17	16	15	14	13	12	7	10	1

Por que $\Delta_2(8) = 10$?

$rpr8 = 0$:

j	-1	0	1	2	3	4	5	6	7	8	9
P	\$	\$	a	b	y	x	c	d	e	**y**	**x**
		x									

\$	a	b	y	x	c	d	e	y	x	
↑		deslocamento de 10							↑	

$\Delta_2(8) = m + 1 - rpr(8) = 9 + 1 - 0 = 10$

Tabela Δ_2: Exemplo

j	1	2	3	4	5	6	7	8	9
P	a	b	y	x	c	d	e	y	x
Δ_2	17	16	15	14	13	12	7	10	1

Por que $\Delta_2(6) = 12$?

$rpr6 = -2$:

j	-3	-2	-1	0	1	2	3	4	5	6	7	8	9
P	\$	\$	\$	\$	a	b	y	x	c	**d**	**e**	**y**	**x**
		e	**y**	**x**									

				\$	\$	\$	a	b	y	x	c	d	e	y	x
				↑	deslocamento de 12										↑

$\Delta_2(6) = m + 1 - rpr(8) = 9 + 1 - (-2) = 12$

Exercício

Encontre a tabela Δ_2 para o padrão:

j	1	2	3	4	5	6	7	8	9
P	a	b	c	x	x	x	a	b	c

Resposta:

j	1	2	3	4	5	6	7	8	9
P	a	b	c	x	x	x	a	b	c
Δ_2	14	13	12	11	10	9	11	10	1

Algoritmo de Boyer-Moore

Comparam-se os caracteres a partir do último caractere do padrão em direção ao primeiro. Quando o caractere do padrão em uma posição i for diferente do caractere do texto c, calcula-se o deslocamento definido por $\Delta_1(c)$ e $\Delta_2(i)$. Soma-se o maior destes valores à posição atual do texto e recomeça o processo a partir desta posição resultante.

Busca no Texto

Exemplo: Padrão: $P = aababbaa$ e $\Sigma = \{a, b, c\}$

Tabela Δ_1:

$$\Delta_1(a) = 8 - 8 = 0 \qquad \Delta_1(b) = 8 - 6 = 2 \qquad \Delta_1(c) = 8$$

Tabela Δ_2:

j	1	2	3	4	5	6	7	8
P	a	a	b	a	b	b	a	a
Δ_2	13	12	11	10	9	8	5	1

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

$\Delta_1(c) = 8, \Delta_2(7) = 5$: maior: 8

a	a	b	a	**b**	**b**	**a**	**a**

$\Delta_1(a) = 0, \Delta_2(5) = 9$: maior: 9

a	**a**	**b**	**a**	**b**	**b**	**a**	**a**

Apenas 14 comparações!

Complexidade

- A construção de Δ_1 pode ser feita inicializando-se um vetor do tamanho do alfabeto com o valor m e, em seguida, percorrendo o padrão linearmente. Portanto, é $O(|\Sigma| + m)$

- A construção de Δ_2 pode ser feita em $O(m)$ (D.E. Knuth, J.H. Morris, V.R. Pratt. Fast Pattern Matching in Strings. *SIAM Journal of Computing*. Vol. 6, No. 2. Junho, 1977)

- O algoritmo de busca é $O(n + m)$. O algoritmo, no entanto, tem um comportamento esperado *sublinear*, ou seja, fazem-se menos do que i comparações para se encontrar o padrão na posição i ([Boyer, Moore])