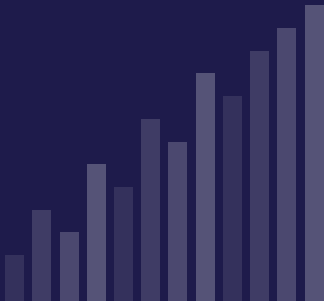

AULA 35 · BLOCO 4: GRAFOS

Casamento de cadeias: o algoritmo de Knuth-Morris-Pratt

Seções 32.1 e 32.4

Estruturas de dados e seus algoritmos · José Dihego · IFBA



Roteiro da aula

Ao final, você será capaz de

- ▶ Definir o problema de casamento de cadeias e o algoritmo ingênuo
- ▶ Calcular a função de prefixo π de um padrão
- ▶ Executar o KMP e mostrar que ele é $\Theta(n + m)$
- ▶ Medir o pior caso do ingênuo contra o KMP



Teoria

O problema, o ingênuo, a função de prefixo e o KMP



Prática em C

KMP em C



Experimento

Texto a^n e padrão $a^{99}b$: $(n - m + 1)m$ contra $\leq 3n$



Conexões

Buscadores, segurança, bioinformática e editores

O problema

Dado um texto $T[1..n]$ e um padrão $P[1..m]$, achar todos os deslocamentos s com $T[s + 1..s + m] = P$. **Ingênuo:** testa cada um dos $n - m + 1$ deslocamentos comparando até m caracteres: $O((n - m + 1)m)$. Pior caso: $T = a^n$, $P = a^{m-1}b$.

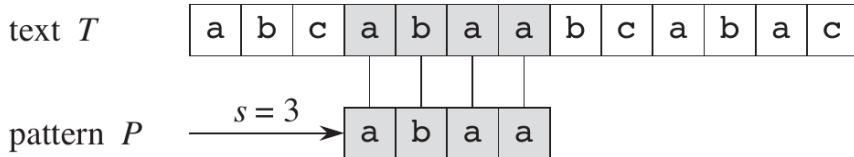


Figura 32.1 de Cormen et al. (2009) — o padrão $P = abaa$ ocorre no texto T com deslocamento $s = 3$.

Os algoritmos do capítulo

- ▶ **Rabin-Karp:** compara hashes das janelas (aula 20); rápido em média, $O((n - m + 1)m)$ no pior caso.
- ▶ **Autômato finito:** $\Theta(n)$ para casar, mas pré-processamento $O(m|\Sigma|)$.
- ▶ **KMP:** $\Theta(m)$ de pré-processamento e $\Theta(n)$ para casar, sem depender do alfabeto.

Algorithm	Preprocessing time	Matching time
Naive	0	$O((n - m + 1)m)$
Rabin-Karp	$\Theta(m)$	$O((n - m + 1)m)$
Finite automaton	$O(m \Sigma)$	$\Theta(n)$
Knuth-Morris-Pratt	$\Theta(m)$	$\Theta(n)$

Figura 32.2 de Cormen et al. (2009) — tempos de pré-processamento e de casamento.

A função de prefixo

$\pi[q]$ = tamanho do maior prefixo **próprio** de P_q que também é **sufixo** de P_q .

- ▶ Depois de casar q caracteres e falhar, não é preciso voltar no texto: o padrão continua da posição $\pi[q]$.
- ▶ Os caracteres do texto nunca são relidos para trás.
- ▶ Calcular π é o KMP do padrão contra ele mesmo: $\Theta(m)$.

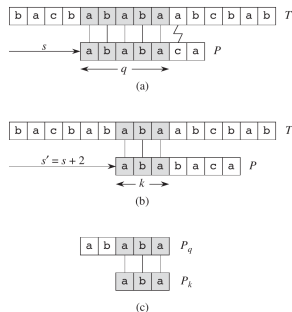


Figura 32.10 de Cormen et al. (2009) — (a) $q = 5$ caracteres casaram e o próximo falha; (b) desloca-se P para o maior prefixo que é também sufixo; (c) $\pi[5] = 3$.

KMP

Análise amortizada: q aumenta no máximo n vezes (uma por caractere); cada volta do **enquanto** diminui q . Então o **enquanto** roda no máximo n vezes no total.

Casamento: $\Theta(n)$. Total com o pré-processamento: $\Theta(n + m)$.

KMP-MATCHER(T, P)

```

1  $\pi \leftarrow \text{Compute-Prefix-FUNCTION}(P)$ 
2  $q \leftarrow 0$  // caracteres casados
3 para  $i \leftarrow 1$  até  $n$  faça
4   enquanto  $q > 0$  e  $P[q + 1] \neq T[i]$  faça
5      $q \leftarrow \pi[q]$ 
6   se  $P[q + 1] = T[i]$  então
7      $q \leftarrow q + 1$ 
8   se  $q = m$  então
9     imprima "ocorrência com deslocamento"
       $i - m$ 
10     $q \leftarrow \pi[q]$ 

```



Prática em C

- ▶ Função de prefixo com índices de 0
- ▶ KMP que relata todas as ocorrências (inclusive sobrepostas)
- ▶ Testes com o exemplo da Figura 32.10

Knuth-Morris-Pratt em C (1/2)

```
1  #include <stdio.h>
2  #include <string.h>
3
4  /* pi[q] = tamanho do maior prefixo próprio de P[0..q] que também é sufixo dele.
5     Quando um caractere não casa, voltamos para pi[q-1] em vez de recomeçar. */
6  void prefixo(const char *P, int m, int pi[]) {    // COMPUTE-PREFIX-FUNCTION:  $\Theta(m)$ 
7      pi[0] = 0;
8      for (int q = 1, k = 0; q < m; q++) {
9          while (k > 0 && P[k] != P[q]) k = pi[k - 1];
10         if (P[k] == P[q]) k++;
11         pi[q] = k;
12     }
13 }
14
15 int main(void) {
```

Knuth-Morris-Pratt em C (2/2)

```
16 0
17 char T[1001], P[101];
18 if (scanf("%1000s %100s", T, P) != 2) return 1;
19 int n = strlen(T), m = strlen(P), pi[101], achou = 0;
20 prefixo(P, m, pi);
21 for (int i = 0, q = 0; i < n; i++) {           // KMP-MATCHER:  $\Theta(n)$ 
22     while (q > 0 && P[q] != T[i]) q = pi[q - 1];
23     if (P[q] == T[i]) q++;
24     if (q == m) {                             // casou P inteiro terminando em i
25         printf("ocorrência no deslocamento %d\n", i - m + 1);
26         achou = 1;
27         q = pi[q - 1];                         // procura a próxima
28     }
29 }
30 if (!achou) printf("nenhuma ocorrência\n");
31 return 0;
```



Experimento

- ▶ Texto a^n , padrão $a^{99}b$ ($m = 100$)
- ▶ Ingênuo: exatamente $(n - m + 1)m$
- ▶ KMP: no máximo $3n$ (contando o teste que encerra cada laço)

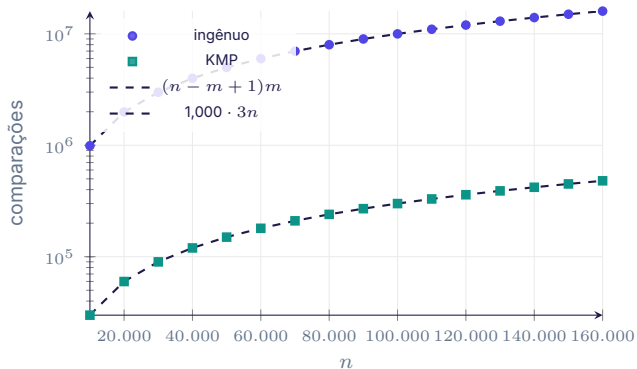
O pior caso do ingênuo

Hipótese. No ingênuo, cada deslocamento casa 99 caracteres antes de falhar no último: exatamente $(n - 99) \cdot 100$ comparações. No KMP, cada caractere do texto custa no máximo uma comparação no **se**, uma que encerra o **enquanto** e, amortizadas, as voltas do **enquanto**: no máximo $3n$.

Método

1. n de 10.000 a 160.000
2. Contador em toda comparação entre caracteres
3. Mesma entrada para os dois algoritmos
4. Padrão ausente: nenhuma ocorrência

Comparações: ingênuo contra KMP



n	medido	razão
10.000	990.100	1,000
40.000	3.990.100	1,000
70.000	6.990.100	1,000
100.000	9.990.100	1,000
130.000	12.990.100	1,000
160.000	15.990.100	1,000

razão = medido $\div$ $(n - m + 1)m$

✓ desvio máx. 0,0%

Para $n = 160.000$: 16 milhões de comparações no ingênuo, 480 mil no KMP: 33 vezes menos, e a razão cresce com m .

EXPERIMENTO

O que o experimento mostrou

O ingênuo faz exatamente $(n - m + 1)m$ comparações no pior caso; o KMP fica abaixo de $3n$, **independente de m** . A função de prefixo é uma pequena tabela que evita reler o texto: mais uma vez, pré-processar troca memória por tempo.

>_ Reproduza: `gcc -O2 aula35_experimento.c -o exp -lm && ./exp`



Conexões

- ▶ Busca em texto
- ▶ Segurança
- ▶ Bioinformática
- ▶ Compiladores

Onde isso aparece na computação

Busca em texto

grep, o "localizar dos editores e navegadores e as bibliotecas de strings usam KMP, Boyer-Moore ou variações para buscar em gigabytes.

Segurança

Antivírus e sistemas de detecção de intrusão (Snort, Suricata) procuram milhares de assinaturas ao mesmo tempo com Aho-Corasick, a generalização do KMP.

Bioinformática

Localizar motivos em DNA, sítios de restrição e sequências repetidas: casamento de cadeias sobre o alfabeto $\{A, C, G, T\}$.

Compiladores

Analisadores léxicos reconhecem palavras-chave e tokens com autômatos finitos, o mesmo modelo da seção 32.3.

Para casa

Exercícios

1. (32.1-2) Se todos os caracteres de P são diferentes, acelere o ingênuo para $O(n)$.
2. (32.4-1) Calcule a função de prefixo de ababbabbabbababbabb.
3. (32.4-7) Decida se T é uma rotação cíclica de T' em tempo linear.
4. (32.2-1) Simule o Rabin-Karp com módulo 11 em $T = 3141592653589793$ e $P = 26$: quantos alarmes falsos?
5. Meça o experimento com texto aleatório em português. O ingênuo ainda perde? Por quê?

Leitura

Cormen et al., seções 32.1, 32.2 e 32.4.

Código

josedihego.net/estruturas