Aula 20

Busca de Padrões

Prof. Marco Aurélio Stefanes
marco em dct.ufms.br

www.dct.ufms.br/ ~ marco

Busca de Padrões

- Texto $T[1..n]$: vetor de caracteres
- Padrão $P[1..m]$; vetor de caracteres $m \leq n$
- Σ: Conjunto finito de caracteres (alfabeto)
  Exemplo: $\Sigma = \{a, b, \ldots, z\}$
  $\Sigma = \{0, 1\}$
  $\Sigma = \{a, c, g, t\}$

O Problema de busca de padrões consiste em encontrar em T todas as ocorrências de P.
Dizemos que o padrão P ocorre em T na posição s, se $P[1, ..m] = T[s, \ldots, s + m - 1]$.
Queremos encontrar todos os s com estas propriedades

Busca de Padrões

Algoritmo FB(T, P, n, m)

1: **for** $s = 1$ to $n - m + 1$ **do**
2: **if** $P[1, ..m] = T[s, \ldots, s + m - 1]$ **then**
3: imprima s

Complexidade: $O(nm)$

Notação:

- Σ^*: Conj. de todas as strings finitas formadas por caracteres de Σ

  Exemplo: $\Sigma = \{a, b\}$

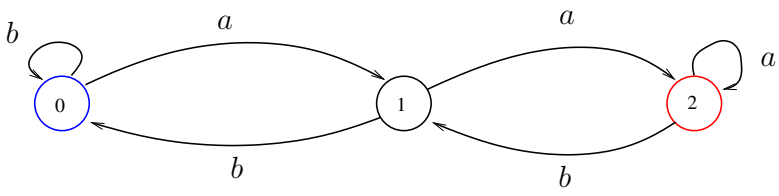
  $\Sigma^* = \{\epsilon, a, b, aa, ab, ba, bb, aaa, \ldots\}$

Busca de Padrões

Notação

- x, y, w, z: stings
- a, b, c, d: caracteres
- $|x|$: Comprimento de x
- a string vazia $\epsilon \in \Sigma^*$ e tem comprimento 0
- w é prefixo de x, denotado por $w \sqsubset x$ se $x = wy$, para algum $y \in \Sigma^*$.
- w é sufixo de x, denotado por $w \sqsupset x$ se $x = yw$, para algum $y \in \Sigma^*$.

Busca com Autômato Finito



- $\Sigma = \{a, b\}$
- $ababa$ não é aceito
- $baaa$ aceito

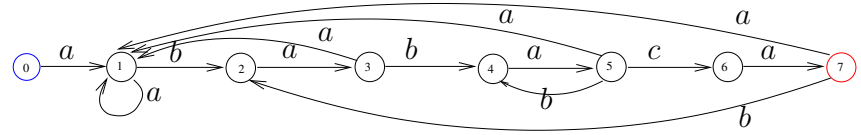
Busca com Autômato Finito

O autômato aceita x se a simulação de x no autômatom começando no estado inicial, termina num estado final. Caso contrário, o autômato rejeita x.
Def.: Um autômato é uma quintupla $(Q, q_0, A, \Sigma, \delta)$:

- Q: conj. finito de estados $\{0, 1, 2\}$
- $q_0 \in Q$: estado inicial 0
- $A \in Q$: Conj. de estados finais $\{0\}$
- Σ: Alfabeto finito $\{a, b\}$
- δ: Função de transição $Q \times \Sigma \to Q$
  $\delta(0, a) = 1 \quad \delta(1, b) = 0$
  $\delta(0, b) = 0 \quad \delta(2, a) = 2$
  $\delta(1, a) = 0 \quad \delta(2, b) = 1$

Busca com Autômato Finito

Um autômato finito induz uma função
$\phi : \Sigma^* \to Q$, a função estado terminal. $\phi(w)$ é o estado que o autômato termina quando usa a string w.
$\phi(\epsilon) = q_0$
$\phi(wa) = \delta(\phi(w), a)$



Convenção: arestas faltando levam ao estado q_0.
Qualquer string x com sufixo $'ababaca'$ é aceito pelo autômato acima

Construção do Autômato

Dado um padrão $P[1, .., m]$, vamos construir um autômato com $Q = \{0, 1, 2, \ldots, m\}$, $q_0 = 0$ e $A = \{m\}$
Se tivermos o autômato para P, podemos resolver o problema da busca de padrão com o seguinte algoritmo

```
1: q = 0
2: for i = 1 to n do
3:    q = δ(q, T[i])
4:    if q = m then
5:       s = i − m + 1
6:       imprima "padrão na posição",s
```

Complexidade $\Theta(n)$

Como construir o autômato para um dado padrão P?

Construção do Autômato

Vamos definir a função
$\sigma_P : \Sigma^* \rightarrow \{0, 1, \ldots, m\}$, a função sufixo

$$\sigma_P(x) = \max\{k : P_k \sqsupset x\}$$

Isto é, $\sigma_P(x)$ é o tamanho do maior prefixo de P que é sufixo de x
Exemplo:
$P = ab \qquad \sigma(\epsilon) = 0$
$\sigma(ccaca) = 1 \qquad \sigma(ccab) = 2$
$\sigma(x) = m \Leftrightarrow P \sqsupset x$
A função de transição do autômato para P é definida por
$\delta(q, a) = \sigma(P_q a)$

Computação do Autômato

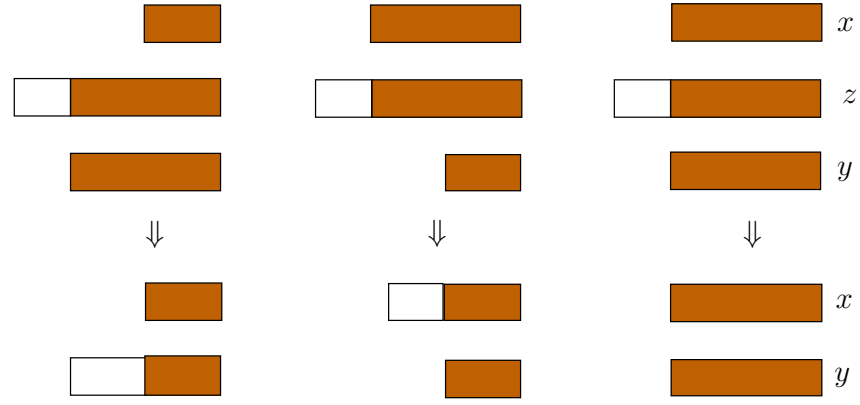
Algoritmo Computa_Delta
Entrada: P, Σ
Saída: Função δ

1: $m = |P|$
2: **for** $q = 0$ to m **do**
3: **for** cada caracter $a \in \Sigma$ **do**
4: $k = \min\{m + 1, q + 2\}$
5: **repeat**
6: $k = k - 1$
7: **until** $P_k \sqsupset P_q a$
8: $\delta(q, a) = k$

Complexidade: $O(m^3|\Sigma|)$
Pode ser melhorada para $O(m|\Sigma|)$

Construção do Autômato

Lema 34.1 Se $x \sqsupset z$ e $y \sqsupset z$ então:
$|x| \leq |y| \Rightarrow x \sqsupset y \qquad |x| \geq |y| \Rightarrow y \sqsupset x \qquad |x| = |y| \Rightarrow x = y$
Prova:



Construção do Autômato

Lema 34.2 $\sigma(xa) \leq \sigma(x) + 1$
Prova:
Seja $r = \sigma(xa)$, então $\sigma(x) \geq r - 1 = \sigma(xa) - 1$

Lema 34.3 Se $q = \sigma(x)$ então $\sigma(xa) = \sigma(P_q a)$
Prova:
Pela def. de σ, temos que $P_q \sqsupset x$. Logo $P_q a \sqsupset xa$.
Seja $r = \sigma(xa)$, pelo Lema 34.2 $r \leq q + 1$.
Assim temos, $P_r \sqsupset xa$, $P_q a \sqsupset xa$ e
$|P_r| \leq |P_q a| \Rightarrow_{Lema 34.1} P_r \sqsupset P_q a$.
Portanto, $r \leq \sigma(P_q a)$, isto é $\sigma(xa) \leq \sigma(P_q a)$. Mas também temos $\sigma(P_q a) \leq \sigma(xa)$, uma vez que $P_q a \sqsupset xa$. Logo
$\sigma(xa) = \sigma(P_q a)$

Construção do Autômato

Teorema Considere o autômato finito de um padrão $P[1..m]$ constituido da seguinte forma:
$Q = \{0, 1, .., m\}, \quad q_0 = 0, \quad A = \{m\} \quad \delta(q, a) = \sigma(P_q a)$
Então, para qualquer texto $T[1..n]$ temos que $\phi(T_i) = \sigma(T_i)$, $i = \{0, 1, \ldots, n\}$

Prova: Indução em i

Base: $i = 0 \quad T_0 = \epsilon, \quad \phi(T_0) = 0 = \sigma(T_0)$

PI: Supor $\phi(T_i) = \sigma(T_i)$ e mostramos que $\phi(T_{i+1}) = \sigma(T_{i+1})$

Seja $q = \phi(T_i)$ e seja $a = T[i+1]$ então

$$\begin{aligned}\phi(T_{i+1}) &= \phi(T_i a) \\ &= \delta(\phi(T_i), a) \\ &= \delta(q, a) \\ &= \sigma(P_q a) (\text{def. de } \delta) \\ &= \sigma(T_i a) = \sigma(T_{i+1}) (\text{Lema 34.3})\end{aligned}$$

Algoritmo KMP

π: Função prefixo

$\pi(q) = \max\{k : k < q \text{ e } P_k \sqsupset P_q\}$

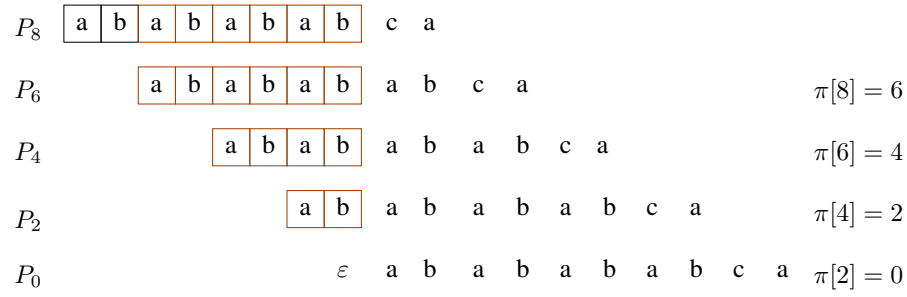
Exemplo: $\pi(5) = 3$, pois aba é o maior sufixo de $ababa$

Algoritmo Calcula_$\pi(P, n)$

1: $\pi(1) = 0$
2: $k = 0$
3: **for** $q = 2$ to m **do**
4: **while** $k > 0$ e $P[k+1] \neq P[q]$ **do**
5: $k = \pi[k]$
6: **if** $P[k+1] = P[q]$ **then**
7: $k = k + 1$
8: $\pi[q] = k$
9: devolva π

Exemplo do Calcula_π

i	1	2	3	4	5	6	7	8	9	10
P_i	a	b	a	b	a	b	a	b	c	a
$\pi[i]$	0	0	1	2	3	4	5	6	0	1



Complexidade do Calcula_π

Análise Amortizada

Cada vez que k é incrementado na linha 7 um crédito extra é dado à variável k. Cada execução da linha 5 reduz o valor de k, e é feita em tempo constante. O custo desta operação é pago com o crédito associado a k. Como o valor de k nunca é negativo, sempre poderemos pagar as operações da linha 5.

Como o número de vezes que k é incrementado é limitado por $m - 1$, o custo total das operações da linha 5 é $O(m)$. O restante das operações também é $O(m)$. Portanto o cálculo de π é executado em tempo $O(m)$.

Algoritmo KMP

Algoritmo KMP(T, P, n, m)

```
 1: π = Calcula_π(P, n)
 2: q = 0
 3: for i = 1 to n do
 4:   while q > 0 e P[q + 1] ≠ T[i] do
 5:     q = π[q]
 6:   if P[q + 1] = P[i] then
 7:     q = q + 1
 8:   if q = m then
 9:     imprima i − m + 1
10:     q = π[q]
```

Complexidade: $O(n + m)$ Similar ao Calcula_π