Análise de Algoritmos

Parte destes slides são adaptações de slides

do Prof. Paulo Feofiloff e do Prof. José Coelho de Pina.

Busca de padrão

Dados

- uma palavra $P[1\,..\,m]$ e
- um texto $T[1\,..\,n]$,

uma ocorrência de P em T é um índice s tal que $T[s+j] = P[j]$ para $j = 1, \ldots, m$.

Busca de padrão

Dados

- uma palavra $P[1\,..\,m]$ e
- um texto $T[1\,..\,n]$,

uma ocorrência de P em T é um índice s tal que
$T[s+j] = P[j]$ para $j = 1, \ldots, m$.

Exemplo:

	1	2	3
P	B	R	A

	1	2	3	4	5	6	7	8	9	10	11
T	A	B	R	A	C	A	D	A	B	R	A

Busca de padrão

Dados

- uma palavra $P[1\,..\,m]$ e
- um texto $T[1\,..\,n]$,

uma ocorrência de P em T é um índice s tal que
$T[s+j] = P[j]$ para $j = 1, \ldots, m$.

Exemplo:

	1	2	3
P	B	R	A

	1	2	3	4	5	6	7	8	9	10	11
T	A	B	R	A	C	A	D	A	B	R	A

Problema: Dada uma palavra $P[1\,..\,m]$ e um texto $T[1\,..\,n]$, calcular o número de ocorrências de P em T.

Busca de padrão

Dados

- uma palavra $P[1\,..\,m]$ e
- um texto $T[1\,..\,n]$,

uma ocorrência de P em T é um índice s tal que $T[s+j] = P[j]$ para $j = 1, \ldots, m$.

Exemplo:

	1	2	3
P	B	R	A

	1	2	3	4	5	6	7	8	9	10	11
T	A	B	R	A	C	A	D	A	B	R	A

Problema: Dada uma palavra $P[1\,..\,m]$ e um texto $T[1\,..\,n]$, calcular o número de ocorrências de P em T.

No exemplo, P ocorre duas vezes em T: em 1 e em 8.

Algoritmo ingênuo

BUSCA-TRIVIAL (T, n, P, m)

$c \leftarrow 0$
para $s \leftarrow 0$ **até** $n - m$ **faça**
    $j \leftarrow 1$
    enquanto $j \leq m$ **e** $P[j] = T[s + j]$ **faça**
        $j \leftarrow j + 1$
    se $j > m$
        então $c \leftarrow c + 1$
return c

Algoritmo ingênuo

BUSCA-TRIVIAL (T, n, P, m)

```
c ← 0
para s ← 0 até n − m faça
    j ← 1
    enquanto j ≤ m e P[j] = T[s + j] faça
        j ← j + 1
    se j > m
        então c ← c + 1
return c
```

Consumo de tempo: $\mathrm{O}(mn)$

Algoritmo ingênuo

BUSCA-TRIVIAL (T, n, P, m)

$c \leftarrow 0$
para $s \leftarrow 0$ **até** $n - m$ **faça**
    $j \leftarrow 1$
    enquanto $j \leq m$ **e** $P[j] = T[s + j]$ **faça**
        $j \leftarrow j + 1$
    se $j > m$
        então $c \leftarrow c + 1$
return c

Consumo de tempo: $\Theta(mn)$

Exemplo: $T = a^n$ e $P = a^m$, ou $P = a^{m-1}b$.

Algoritmo ingênuo

BUSCA-TRIVIAL (T, n, P, m)

$c \leftarrow 0$
para $s \leftarrow 0$ **até** $n - m$ **faça**
  $j \leftarrow 1$
  enquanto $j \leq m$ **e** $P[j] = T[s + j]$ **faça**
    $j \leftarrow j + 1$
  se $j > m$
    então $c \leftarrow c + 1$
return c

Consumo de tempo: $\Theta(mn)$

Exemplo: $T = a^n$ e $P = a^m$, ou $P = a^{m-1}b$.

Nesta aula, vamos ver um algoritmo cujo consumo de tempo no pior caso é $\Theta(mn)$, porém na prática é $\mathrm{O}(n)$, podendo inclusive ter um comportamento sublinear.

Algoritmo BM – primeira versão

BM: Boyer-Moore.

Algoritmo BM – primeira versão

BM: Boyer-Moore.

$v_1[c]$: menor t em $0\,..\,m-1$ tal que $P[m-t] = c$
(se c não aparece em P, então $v_1[c] = m$)

Algoritmo BM – primeira versão

BM: Boyer-Moore.

$v_1[c]$: menor t em $0\,..\,m-1$ tal que $P[m-t] = c$
(se c não aparece em P, então $v_1[c] = m$)

$v_1[c]$ corresponde à última ocorrência de c em P.

Algoritmo BM – primeira versão

BM: Boyer-Moore.

$v_1[c]$: menor t em $0\,..\,m-1$ tal que $P[m-t] = c$
(se c não aparece em P, então $v_1[c] = m$)

$v_1[c]$ corresponde à última ocorrência de c em P.

Exemplo:

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

	a	b	c	d
v_1	1	5	2	0

Algoritmo BM – primeira versão

BM: Boyer-Moore.

$v_1[c]$: menor t em $0 \mathinner{.\,.} m-1$ tal que $P[m-t] = c$
(se c não aparece em P, então $v_1[c] = m$)

$v_1[c]$ corresponde à última ocorrência de c em P.

Exemplo:

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

	a	b	c	d
v_1	1	5	2	0

Seja $k \geq m$. Suponha que $c = T[k+1]$ e que $v_1[c] = 4$. Então $P[1 \mathinner{.\,.} m]$ não é um sufixo de $T[1 \mathinner{.\,.} k{+}1], \ldots, T[1 \mathinner{.\,.} k{+}4]$.
(Supondo que $k + 4 \leq n$.)

Algoritmo BM – primeira versão

BM: Boyer-Moore.

$v_1[c]$: menor t em $0\,..\,m-1$ tal que $P[m-t] = c$
(se c não aparece em P, então $v_1[c] = m$)

$v_1[c]$ corresponde à última ocorrência de c em P.

Exemplo:

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

	a	b	c	d
v_1	1	5	2	0

Seja $k \geq m$. Suponha que $c = T[k+1]$. Então $P[1\,..\,m]$ não é um sufixo de $T[1\,..\,k{+}1], \ldots, T[1\,..\,k{+}\mu]$, onde $\mu = \min\{n-k, v_1[c]\}$.

Algoritmo Boyer-Moore 1

Supondo que o alfabeto é o conjunto $\Sigma = \{0, \ldots, 255\}$.

BM1 (P, m, T, n)

para $i \leftarrow 0$ **até** 255 **faça** $v_1[i] \leftarrow m$
para $i \leftarrow 1$ **até** m **faça** $v_1[P[i]] \leftarrow m - i$
$c \leftarrow 0 \qquad k \leftarrow m$
enquanto $k \leq n$ **faça**
    $r \leftarrow 0$
    enquanto $m - r \geq 1$ **e** $P[m-r] = T[k-r]$ **faça**
        $r \leftarrow r + 1$
    se $m - r < 1$ **então** $c \leftarrow c + 1$
    se $k = n$
        então $k \leftarrow k + 1$
        senão $k \leftarrow k + 1 + v_1[T[k+1]]$
devolva c

Algoritmo Boyer-Moore 1

Supondo que o alfabeto é o conjunto $\Sigma = \{0, \ldots, 255\}$.

BM1 (P, m, T, n)

para $i \leftarrow 0$ **até** 255 **faça** $v_1[i] \leftarrow m$
para $i \leftarrow 1$ **até** m **faça** $v_1[P[i]] \leftarrow m - i$
$c \leftarrow 0$ $\quad k \leftarrow m$
enquanto $k \leq n$ **faça**
  $r \leftarrow 0$
  enquanto $m - r \geq 1$ **e** $P[m-r] = T[k-r]$ **faça**
    $r \leftarrow r + 1$
  se $m - r < 1$ **então** $c \leftarrow c + 1$
  se $k = n$
    então $k \leftarrow k + 1$
    senão $k \leftarrow k + 1 + v_1[T[k+1]]$
devolva c

Consumo de tempo: $\Theta(|\Sigma| + mn)$

Algoritmo BM – segunda versão

j em $1\,..\,m-1$ é bom para i se $P[m-\mu\,..\,m] = P[j-\mu\,..\,j]$,
onde $\mu = \min\{m-i, j-1\}$.

Algoritmo BM – segunda versão

j em $1\,..\,m{-}1$ é bom para i se $P[m{-}\mu\,..\,m] = P[j{-}\mu\,..\,j]$,
onde $\mu = \min\{m{-}i, j{-}1\}$.

j é bom para i se o mais curto entre $P[i\,..\,m]$ e $P[1\,..\,j]$
é sufixo do mais longo dos dois.

Algoritmo BM – segunda versão

j em $1\,..\,m-1$ é bom para i se $P[m-\mu\,..\,m] = P[j-\mu\,..\,j]$,
onde $\mu = \min\{m-i, j-1\}$.

j é bom para i se o mais curto entre $P[i\,..\,m]$ e $P[1\,..\,j]$
é sufixo do mais longo dos dois.

Exemplos:

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

Algoritmo BM – segunda versão

j em $1\,.\,.\,m{-}1$ é bom para i se $P[m{-}\mu\,.\,.\,m] = P[j{-}\mu\,.\,.\,j]$,
onde $\mu = \min\{m{-}i, j{-}1\}$.

j é bom para i se o mais curto entre $P[i\,.\,.\,m]$ e $P[1\,.\,.\,j]$
é sufixo do mais longo dos dois.

Exemplos:

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

$j = 3$ é bom para $i = 5$, pois aa é sufixo de caa.

Algoritmo BM – segunda versão

j em $1 \mathinner{..} m-1$ é bom para i se $P[m-\mu \mathinner{..} m] = P[j-\mu \mathinner{..} j]$,
onde $\mu = \min\{m-i, j-1\}$.

j é bom para i se o mais curto entre $P[i \mathinner{..} m]$ e $P[1 \mathinner{..} j]$
é sufixo do mais longo dos dois.

Exemplos:

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

Algoritmo BM – segunda versão

j em $1\,..\,m{-}1$ é bom para i se $P[m{-}\mu\,..\,m] = P[j{-}\mu\,..\,j]$,
onde $\mu = \min\{m{-}i, j{-}1\}$.

j é bom para i se o mais curto entre $P[i\,..\,m]$ e $P[1\,..\,j]$
é sufixo do mais longo dos dois.

Exemplos:

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

$j = 6$ é bom para $i = 8$, pois aa é sufixo de $caabaa$.

Algoritmo BM – segunda versão

j em $1\,..\,m{-}1$ é bom para i se $P[m{-}\mu\,..\,m] = P[j{-}\mu\,..\,j]$,
onde $\mu = \min\{m{-}i, j{-}1\}$.

j é bom para i se o mais curto entre $P[i\,..\,m]$ e $P[1\,..\,j]$
é sufixo do mais longo dos dois.

Exemplos:

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

$j = 6$ é bom para $i = 8$, pois aa é sufixo de $caabaa$.

$j = 6$ é bom para $i = 5$, pois $aabaa$ é sufixo de $caabaa$.

Algoritmo BM – segunda versão

j em $1\,..\,m{-}1$ é bom para i se $P[m{-}\mu\,..\,m] = P[j{-}\mu\,..\,j]$,
onde $\mu = \min\{m{-}i, j{-}1\}$.

j é bom para i se o mais curto entre $P[i\,..\,m]$ e $P[1\,..\,j]$
é sufixo do mais longo dos dois.

Exemplos:

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

$j = 6$ é bom para $i = 8$, pois aa é sufixo de $caabaa$.

$j = 6$ é bom para $i = 5$, pois $aabaa$ é sufixo de $caabaa$.

$j = 3$ é bom para $i = 8$, pois aa é sufixo de caa.

Algoritmo BM – segunda versão

j em $1 \, . \, . \, m{-}1$ é bom para i se $P[m{-}\mu \, . \, . \, m] = P[j{-}\mu \, . \, . \, j]$,
onde $\mu = \min\{m{-}i, j{-}1\}$.

j é bom para i se o mais curto entre $P[i \, . \, . \, m]$ e $P[1 \, . \, . \, j]$
é sufixo do mais longo dos dois.

Exemplos:

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

$j = 3$ é bom para $i = 6$, pois caa é sufixo de $acaa$.

Algoritmo BM – segunda versão

j em $1\,..\,m{-}1$ é bom para i se $P[m{-}\mu\,..\,m] = P[j{-}\mu\,..\,j]$,
onde $\mu = \min\{m{-}i, j{-}1\}$.

j é bom para i se o mais curto entre $P[i\,..\,m]$ e $P[1\,..\,j]$
é sufixo do mais longo dos dois.

Exemplos:

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

$j = 3$ é bom para $i = 6$, pois caa é sufixo de $acaa$.

$j = 3$ é bom para $i = 1, \ldots, 7$, pois caa é sufixo de $caabaacaa$.

Algoritmo BM – segunda versão

j é bom para i se o mais curto entre $P[i\,..\,m]$ e $P[1\,..\,j]$
é sufixo do mais longo dos dois.

Algoritmo BM – segunda versão

j é bom para i se o mais curto entre $P[i \,..\, m]$ e $P[1 \,..\, j]$ é sufixo do mais longo dos dois.

$v_2[i]$: menor t em $0 \,..\, m-1$ tal que $m-t$ é bom para i.
(se não há j bom para i, então $v_2[i] = m$)

Algoritmo BM – segunda versão

j é bom para i se o mais curto entre $P[i\,..\,m]$ e $P[1\,..\,j]$ é sufixo do mais longo dos dois.

$v_2[i]$: menor t em $0\,..\,m-1$ tal que $m-t$ é bom para i.
(se não há j bom para i, então $v_2[i] = m$)

Exemplos:

1	2	3	4	5	6
c	a	a	b	a	a

i	6	5	4	3	2	1
v_2	1	3	6	6	6	6

Algoritmo BM – segunda versão

j é bom para i se o mais curto entre $P[i\,..\,m]$ e $P[1\,..\,j]$
é sufixo do mais longo dos dois.

$v_2[i]$: menor t em $0\,..\,m-1$ tal que $m-t$ é bom para i.
(se não há j bom para i, então $v_2[i] = m$)

Exemplos:

1	2	3	4	5	6
c	a	a	b	a	a

i	6	5	4	3	2	1
v_2	1	3	6	6	6	6

1	2	3	4	5	6	7	8
b	a	-	b	a	.	b	a

i	8	7	6	5	4	3	2	1
v_2	3	3	6	6	6	6	6	6

Algoritmo BM – segunda versão

j é bom para i se o mais curto entre $P[i\,..\,m]$ e $P[1\,..\,j]$ é sufixo do mais longo dos dois.

$v_2[i]$: menor t em $0\,..\,m-1$ tal que $m-t$ é bom para i.
(se não há j bom para i, então $v_2[i] = m$)

Exemplos:

1	2	3	4	5	6
c	a	a	b	a	a

i	6	5	4	3	2	1
v_2	1	3	6	6	6	6

1	2	3	4	5	6	7	8
b	a	-	b	a	.	b	a

i	8	7	6	5	4	3	2	1
v_2	3	3	6	6	6	6	6	6

	1	2	3	4	5	6	7	8	9	10	11
	b	a	-	b	a	*	b	a	*	b	a
i	11	10	9	8	7	6	5	4	3	2	1
v_2	3	3	3	3	3	9	9	9	9	9	9

Algoritmo Boyer-Moore 2

BM2 (P, m, T, n)

para $i \leftarrow m$ **decrescendo até** 1 **faça**
    $j \leftarrow m - 1 \qquad r \leftarrow 0$
    enquanto $m - r \geq i$ **e** $j - r \geq 1$ **faça**
        se $P[m - r] = P[j - r]$ **então** $r \leftarrow r + 1$
            senão $j \leftarrow j - 1 \qquad r \leftarrow 0$
    $v_2[i] \leftarrow m - j$

Algoritmo Boyer-Moore 2

BM2 (P, m, T, n)

para $i \leftarrow m$ **decrescendo até** 1 **faça**

    $\ldots$

    $v_2[i] \leftarrow m - j$

$c \leftarrow 0$ $k \leftarrow m$

enquanto $k \leq n$ **faça**

    $r \leftarrow 0$

    enquanto $m - r \geq 1$ **e** $P[m - r] = T[k - r]$ **faça**

        $r \leftarrow r + 1$

    se $m - r < 1$ **então** $c \leftarrow c + 1$

    se $r = 0$ **então** $k \leftarrow k + 1$

        senão $k \leftarrow k + v_2[m - r + 1]$

devolva c

Algoritmo Boyer-Moore 2

BM2 (P, m, T, n)

para $i \leftarrow m$ **decrescendo até** 1 **faça**

$\quad\quad \ldots$

$\quad\quad v_2[i] \leftarrow m - j$

$c \leftarrow 0 \qquad k \leftarrow m$

enquanto $k \leq n$ **faça**

$\quad\quad r \leftarrow 0$

$\quad\quad$ **enquanto** $m - r \geq 1$ **e** $P[m - r] = T[k - r]$ **faça**

$\quad\quad\quad\quad r \leftarrow r + 1$

$\quad\quad$ **se** $m - r < 1$ **então** $c \leftarrow c + 1$

$\quad\quad$ **se** $r = 0$ **então** $k \leftarrow k + 1$

$\quad\quad\quad\quad$ **senão** $k \leftarrow k + v_2[m - r + 1]$

devolva c

Consumo de tempo: $\Theta(mn)$

Algoritmo Boyer-Moore

Calcula v_1 e v_2, e
atualiza fazendo sempre o maior dos dois deslocamentos.

Algoritmo Boyer-Moore

Calcula v_1 e v_2, e
atualiza fazendo sempre o maior dos dois deslocamentos.

Consumo de tempo: $\Theta(mn)$
(fazendo melhora para inicializar v_1)

Algoritmo Boyer-Moore

Calcula v_1 e v_2, e
atualiza fazendo sempre o maior dos dois deslocamentos.

Consumo de tempo: $\Theta(mn)$
(fazendo melhora para inicializar v_1)

No caso médio entretanto, ele consome apenas $\mathrm{O}(n)$.

É o mais rápido algoritmo de busca de padrão na prática.

Algoritmo Boyer-Moore

Calcula v_1 e v_2, e
atualiza fazendo sempre o maior dos dois deslocamentos.

Consumo de tempo: $\Theta(mn)$
(fazendo melhora para inicializar v_1)

No caso médio entretanto, ele consome apenas $\mathrm{O}(n)$.

É o mais rápido algoritmo de busca de padrão na prática.

É possível aperfeiçoar a tabela v_2 para que
este algoritmo consuma apenas $\mathrm{O}(m+n)$ no pior caso.

Algoritmo Boyer-Moore

Calcula v_1 e v_2, e
atualiza fazendo sempre o maior dos dois deslocamentos.

Consumo de tempo: $\Theta(mn)$
(fazendo melhora para inicializar v_1)

No caso médio entretanto, ele consome apenas $\mathrm{O}(n)$.

É o mais rápido algoritmo de busca de padrão na prática.

É possível aperfeiçoar a tabela v_2 para que
este algoritmo consuma apenas $\mathrm{O}(m+n)$ no pior caso.

Veja os exercícios nas notas de aula indicadas para leitura.