# MAC5711 Análise de Algoritmos

## *DCC–IME–USP, 20 de outubro de 2006*

**Instruções:**

($i$) Esta prova contém seis questões sendo quatro de um ponto e meio e duas de dois pontos.

($ii$) Mencione os teoremas e propriedades usados para justificar suas afirmações.

($iii$) Você pode utilizar como subrotina qualquer algoritmo visto em sala de aula sem reescrevê-lo, mesmo que o algoritmo necessite de alguma **pequena** alteração. No entando, você deve descrever clara e sucintamente o que o algoritmo recebe, devolve ou faz e o seu consumo de tempo e qual é essa alteração. Exemplo

> *"O algoritmo* BLÁ-BLÁ-BLÁ *usa como subrotina o algoritmo* ORDENAÇÃO-LERDA $(A, n)$ *que recebe e rearranja um vetor* $A[1 \,.\,.\, n]$ *de modo que ele fique em ordem crescente. O consumo de tempo do algoritmo* ORDENAÇÃO-LERDA *é* $\mathrm{O}(n^n)$*."*

($iv$) Não é permitida a consulta a livros, anotações, colegas, calculadoras, Internet, computadores ...

## Duração da prova: 2 horas e 30 minutos

**Questão 1** [CLRS 8.3-4]
Descreva um algoritmo ORDENE $(A, n)$ que recebe e ordena um vetor $A[1 \,.\,.\, n]$ em que todos os elementos pertencem a $\{0, 1, \ldots, n^2 - 1\}$. O consumo de tempo do algoritmo deve ser $\mathrm{O}(n)$, onde $n = p - r + 1$. Justifique a sua resposta.

A existência desse algoritmo linear para ordenação contraria em algo o limite inferior para o consumo de tempo de algoritmos de ordenação? Enuncie a asserção desse "limite inferior para ordenação" para justificar a sua resposta.

**Solução:** Estamos habituados a representar números naturais na base decimal que utiliza os algarismos $0, 1, \ldots, 9$. Na solução consideraremos a representação de cada elemento em $A[1 \,.\,.\, n]$ na base $n$, que utiliza os algarismos $0, 1, \ldots, n - 1$. Desta forma, para $j = 1, \ldots, n$, temos que

$$A[j] = a_1 \times n + a_0,$$

onde $a_0$ e $a_1$ são algarismos em $\{0, 1, \ldots, n - 1\}$. Diremos que $a_0$ é o algarismo **menos significativo** de $A[j]$ e que $a_1$ é o algarismo **mais significativo** de $A[j]$.

O algoritmo COUNTING-SORT recebe e ordena um vetor $X[1 \,.\,.\, n]$ em que todos os elementos estão em $\{0, 1, \ldots, k\}$, para um dado $k$. O consumo de tempo do algoritmo é $\Theta(n + k)$. Em particular, se $k$ é $\Theta(n)$, então o consumo de tempo do COUNTING-SORT é $\Theta(n)$.

O algoritmo ORDENE $(A, n)$ é uma mera adaptação do RADIX-SORT.

ORDENE $(A, n)$
1 ordene $A[1 \,.\,.\, n]$ usando como chaves seus algarismos **menos significativos**.
2 ordene $A[1 \,.\,.\, n]$ usando como chaves seus algarismos **mais significativos**.

Utilizando uma adaptação do COUNTING-SORT para $k = n - 1$ nas ordenações das linhas 1 e 2 obtemos um algoritmo de consumo de tempo $\Theta(n + n) = \Theta(n)$. A estabilidade do COUNTING-SORT é fundamental para a correção do algoritmo ORDENE.

Sabe-se que todo algoritmo de ordenação **baseado em comparações** faz

$$\Omega(n \lg n)$$

comparações no **pior caso**. O limite inferior da ordenação, em termos da notação-o, se traduz em

> **Não existe** algoritmo de ordenação **baseado em comparações** de consumo de tempo $\mathrm{o}(n \lg n)$.

A existência do algoritmo linear ORDENE$(A, n)$ não contraria em nada o limite inferior para ordenação já que esse algoritmo não é baseado em comparações, ou seja, para determinar a ordem relativa entre os elemento do vetor $A[1 \,..\, n]$ o algoritmo não utiliza apenas testes como

$$A[i] < A[j], A[i] \leq A[j], A[i] = A[j], A[i] \geq A[j] \text{ e } A[i] > A[j].$$

O algoritmo ORDENE examina o valor de cada elemento e também utiliza operações de "mascara" para extrair um determinado dígito de cada elemento de $A$.

**Questão 2** [CLRS 9.3-5]
Suponha que MEDIANA$(A, p, r)$ é um algoritmo que rearranja os elementos de um dado vetor $A[p\,..\,r]$ de números inteiros e devolve um índice $q$, $p \leq q \leq r$, tal que

$$A[p\,..\,q-1] \leq A[q] \leq A[q+1\,..\,r]$$

e $A[q]$ é a mediana de $A[p\,..\,r]$. Suponha ainda que o consumo de tempo do algoritmo MEDIANA é linear.

Escreva um algoritmo SELECT$(A, p, r, i)$ que recebe um vetor $A[p\,..\,r]$ de números inteiros e um número inteiro $i$, $1 \leq i \leq r - p + 1$, e devolve o valor do $i$-ésimo menor elemento de $A[p\,..\,r]$. O algoritmo deve utilizar o algoritmo MEDIANA como subrotina e deve consumir tempo linear. Explique sucintamente porque seu algoritmo está correto e tem o consumo de tempo pedido.

**Solução:** Eis uma versão de algoritmo SELECT que faz o serviço:

SELECT $(A, p, r, i)$
1 **se** $p = r$
2     **então devolva** $A[p]$
3 $q \leftarrow$ MEDIANA $(A, p, r)$
4 $k \leftarrow q - p + 1$
5 **se** $k = i$
6     **então devolva** $A[q]$
7 **se** $k > i$
8     **então devolva** SELECT $(A, p, q-1, i)$
9     **senão devolva** SELECT $(A, q+1, r, i-k)$

Como $1 \leq i \leq r - p + 1$, se $p = q$, então $i = 1$ e o algoritmo devolve na linha 2 o único elemento do vetor $A[p\,..\,q]$. O algoritmo determina na linha 4 o número de elementos no vetor $A[p\,..\,q]$. Devido a especificação do algoritmo MEDIANA vemos que:

- se $k = i$, então o $i$-ésimo menor elemento do vetor $A[p\,..\,q]$ é $A[q]$;
- se $k > i$, então o elemento desejado é o $i$-ésimo menor elemento do vetor $A[p\,..\,q-1]$; e
- se $k < i$, então o elemento procurado é o $(i-k)$-ésimo menor elemento do vetor $A[q+1\,..\,r]$.

A correção do algoritmo SELECT é devida a este ser uma mera implementação dos fatos acima.

Seja $T(n)$ o consumo de tempo máximo quando $n = r - p + 1$. O consumo de tempo pela execução de cada linha do algoritmo está na tabela a seguir.

| linha | consumo de tempo da linha |
|---|---|
| 1-2 | $= 2\,\Theta(1)$ |
| 3 | $= \Theta(n)$ |
| 4-7 | $= 4\,\Theta(1)$ |
| 8-9 | $= T(\lfloor n/2 \rfloor)$ |
| $T(n)$ | $= \Theta(n+6) + T(\lfloor n/2 \rfloor)$ |
| | $= \Theta(n) + T(\lfloor n/2 \rfloor)$ |

Obtemos desta forma a recorrência que descreve o consumo de tempo do algoritmo Select no pior caso:

$$\begin{aligned} &T(1) = \Theta(1) \\ &T(n) = \Theta(n) + T(\lfloor n/2 \rfloor) \quad \text{para } n = 2, 3, \ldots \end{aligned}$$

Seja $c$ uma constante tal que o consumo de tempo $\Theta(n)$ correspondente às linhas 1–7 é não superior a $c\,n$ para $n = 1, 2, \ldots$. Desta forma, para $n \geq 2$,

$$\begin{aligned} T(n) &= \Theta(n) + T\big(\lfloor \tfrac{n}{2} \rfloor\big) \\ &\leq c\,n + T\big(\lfloor \tfrac{n}{2} \rfloor\big) \\ &\leq c\,n + c\,\lfloor \tfrac{n}{2} \rfloor + T\big(\lfloor \tfrac{n}{2^2} \rfloor\big) \\ &\leq c\,n + c\,\lfloor \tfrac{n}{2} \rfloor + c\,\lfloor \tfrac{n}{2^2} \rfloor + T\big(\lfloor \tfrac{n}{2^3} \rfloor\big) \\ &\leq \ldots \\ &\leq c\,n + c\,\lfloor \tfrac{n}{2} \rfloor + c\,\lfloor \tfrac{n}{2^2} \rfloor + \cdots + c\,\lfloor \tfrac{n}{2^k} \rfloor \\ &\leq c\,n + c\,\frac{n}{2} + c\,\frac{n}{2^2} + \cdots + c\,\frac{n}{2^k} \\ &= c\,n\,\big(1 + \frac{1}{2} + \frac{1}{2^2} + \cdots + \frac{1}{2^k}\big) \\ &< c\,n\,\big(1 + \frac{1}{2} + \frac{1}{2^2} + \cdots + \frac{1}{2^k} + \cdots\big) \\ &= 2c\,n \end{aligned}$$

onde $k = \lfloor \lg n \rfloor$. Como $T(n) \leq 2c\,n$ para $n = 2, 3, \ldots$ concluímos que $T(n)$ é $\mathrm{O}(n)$ (e portanto $\Theta(n)$).

**Questão 3** [CLRS 9.2]
Escreva um algoritmo SELECT$(A, W, p, r, i)$ que recebe vetores $A[p\,..\,r]$ e $W[p\,..\,r]$ de números inteiros positivos e um número inteiro $i$ tal que $1 \leq i \leq \sum_{j=p}^{r} W[j]$ e devolve o valor do $i$-ésimo menor elemento da lista formada por $W[p]$ cópias de $A[p]$, $W[p+1]$ cópias de $A[p+1], \ldots, W[r]$ cópias de $A[r]$. Por exemplo, para $p = 6, r = 8, i = 6$,

| 5 | 6 | 7 | 8 | 9 | |
|---|---|---|---|---|---|
| ... | 4 | 2 | 8 | ... | $A$ , |

e

| 5 | 6 | 7 | 8 | 9 | |
|---|---|---|---|---|---|
| ... | 3 | 5 | 4 | ... | $W$ |

o algoritmo deve devolver 4 que é o $6^{\underline{o}}$ menor elemento da lista formada por

$$4 \;\; 4 \;\; 4 \;\; 2 \;\; 2 \;\; 2 \;\; 2 \;\; 2 \;\; 8 \;\; 8 \;\; 8 \;\; 8\,.$$

O consumo de tempo do algoritmo SELECT deve ser $\mathrm{O}(n)$, onde $n = r - p + 1$. Explique sucintamente porque seu algoritmo está correto e tem o consumo de tempo pedido.

**Solução:** O algoritmo SELECT é semelhante ao algoritmo da questão anterior:

SELECT $(A, W, p, r, i)$
**se** $p = r$
    **então devolva** $A[p]$
$q \leftarrow$ MEDIANA $(A, W, p, r)$
$k \leftarrow 0$
**para** $i \leftarrow 1$ **até** $q$ **faça**
    $k \leftarrow k + W[i]$
**se** $k - W[q] < i$ **e** $i \leq k$
    **então devolva** $A[q]$
**se** $k > i$    $\rhd$ portanto $k - W[q] > i$
    **então devolva** SELECT $(A, p, q - 1, i)$
    **senão devolva** SELECT $(A, q + 1, r, i - k)$

O algoritmo SELECT utiliza como subrotina o algoritmo:

MEDIANA $(A, W, p, r)$
$n \leftarrow r - p + 1$
$i \leftarrow \lfloor (n + 1)/2 \rfloor$
$q \leftarrow$ SELECT-BFPRT $(A, W, p, r, i)$
**devolva** $q$

O algoritmo MEDIANA, por sua vez, utiliza como subrotina o algoritmo SELECT-BFPRT, visto em sala de aula, com uma **pequena** alteração. O algoritmo SELECT-BFPRT recebe o vetor $A[p\,..\,r]$ e um número inteiro $i$ tal que $1 \leq i \leq r - p + 1$ e devolve um índice $q$ tal que $A[q]$ é o $i$-ésimo menor elemento de $A[p\,..\,r]$. A pequena alteração necessária é a seguinte:

> o algoritmo passa a receber também um vetor $W[p\,..\,r]$ e toda operação "$A[j] \leftrightarrow A[k]$" realizada deve ser seguida pela operação "$W[j] \leftrightarrow W[k]$."

Assim, o algoritmo Mediana devolve um índice $q$, $p \leq q \leq r$, tal que

$$A[p\,..\,q-1] \leq A[q] \leq A[q+1\,..\,r]$$

e $A[q]$ é a mediana de $A[p\,..\,r]$. Como o consumo de tempo do algoritmo Select-BFPRT é $\Theta(n)$, então o consumo de tempo do algoritmo Mediana também é $\Theta(n)$.

Como $1 \leq i \leq \sum_{j=p}^{r} W[j]$, se $p = q$, então, na linha 2, o algoritmo corretamente devolve o único elemento da lista. O algoritmo determina nas linha 4 e 5 o número de elementos da lista formada pelos elementos do vetor $A[p\,..\,q]$. A correção do algoritmo é devido a especificação do algoritmo Mediana já que:

- se $k - W[q] < i \leq k$, então o $i$-ésimo menor elemento da lista é $A[q]$ (linhas 7 e 8);
- se $k - W[q] > i$, então o elemento desejado é o $i$-ésimo menor elemento da lista formada por cópias dos elementos no vetor $A[p\,..\,q{-}1]$ (linha 10); e
- se $k < i$, então o elemento procurado é o $(i-k)$-ésimo menor elemento da lista formada por cópias dos elementos no vetor $A[q{+}1\,..\,r]$ (linha 11).

Seja $T(n)$ o consumo de tempo máximo quando $n = r - p + 1$. O consumo de tempo pela execução de cada linha do algoritmo está na tabela a seguir.

| linha | consumo de tempo da linha |
|---|---|
| 1-2 | $= 2\,\Theta(1)$ |
| 3 | $= \Theta(n)$ |
| 4 | $= \Theta(1)$ |
| 5-6 | $= 2\,\Theta(\lceil n/2 \rceil)$ |
| 7-9 | $= 3\,\Theta(1)$ |
| 10-11 | $= T(\lfloor n/2 \rfloor)$ |
| $T(n)$ | $= \Theta(n + 2\lceil n/2 \rceil + 6) + T(\lfloor n/2 \rfloor)$ |
| | $= \Theta(n) + T(\lfloor n/2 \rfloor)$ |

Obtemos assim a recorrência que descreve o consumo de tempo do algoritmo Select no pior caso:

$$T(1) = \Theta(1)$$
$$T(n) = \Theta(n) + T(\lfloor n/2 \rfloor) \quad \text{para } n = 2, 3, \ldots$$

Como visto na questão 2, $T(n)$ é $\Theta(n)$.

**Questão 4** [CLRS 5.3-4]
Considere o algoritmo a seguir que recebe um vetor $A[1\,..\,n]$ e devolve no vetor $B[1\,..\,n]$ uma permutação dos elementos de $A$.

```
Permute-By-Cyclic(A, B, n)
1  k ← Random(1, n)
2  para i ← 1 até n faça
3      j ← i + k
4      se j > n
5          então j ← j − n
6      B[j] ← A[i]
```

Suponha que o vetor $A[1\,..\,n]$ contém uma permutação dos inteiros $1, 2, \ldots, n$. Se $i$ e $j$ são números inteiros em $\{1, \ldots, n\}$, então quanto vale $\Pr\{B[j] = A[i]\}$? O professor McSperto afirma que o vetor $B[1\,..\,n]$ produzido pelo algoritmo é uma permutação aleatória uniforme de $A[1\,..\,n]$. O professor tem razão? Justifique as suas respostas.

**Solução:** Se $j > i$, então

$$\begin{aligned}\Pr\{B[j] = A[i]\} &= \Pr\{j = i + k\} &(1)\\ &= \Pr\{k = j - i\}\\ &= 1/n\ , &(2)\end{aligned}$$

onde a igualdade (1) segue da linha 3 e do fato de $A[1\,..\,n]$ conter elementos distintos e a igualdade (2) é devida à especificação do algoritmo Random. Analogamente, se $j \leq i$

$$\begin{aligned}\Pr\{B[j] = A[i]\} &= \Pr\{j = i + k - n\} &(3)\\ &= \Pr\{k = j - i + n\}\\ &= 1/n\ ,\end{aligned}$$

onde a igualdade (3) é conseqüência das linhas 3, 4 e 5 e do fato de $A[1\,..\,n]$ conter elementos distintos.

O algoritmo não produz uma permutação aleatória uniforme para $n > 2$. Em virtude da linha 1 vê-se que o algoritmo é capaz de produzir $n$ permutação de $A[1\,..\,n]$ enquanto o número de permutações é $n!$ ($n! > n$ para $n > 2$).

**Questão 5** [CLRS 5.1-2]
Considere o algoritmo abaixo que recebe dois números inteiros $a$ e $b$, $0 \leq a \leq b$ e devolve um número inteiro $i$, $a \leq i \leq b$. O algoritmo usa a subrotina Bit-Aleatório() que devolve 1 ou 0, independentemente do valores previamente devolvidos, com probabilidade uniforme.

Random$(a, b)$
1 $n \leftarrow b - a$
2 $t \leftarrow \lceil \lg(n+1) \rceil$
3 **repita**
4 $\quad r \leftarrow 0$
5 $\quad$ **para** $j \leftarrow 1$ **até** $t$ **faça**
6 $\quad\quad r \leftarrow 2 \times r +$ Bit-Aleatório()
7 **até que** $r \leq n$
8 $i \leftarrow a + r$
9 **devolva** $i$

Supondo que o consumo de tempo da subrotina Bit-Aleatório() é constante, qual o consumo de tempo esperado do algoritmo Random? Use a notação O, mas procure dar a resposta o mais justa possível e justifique-a.

**Solução:** Seja $X$ a variável aleatória representando o número de execuções das linhas 3-7. O consumo de tempo pela execução de cada linha do algoritmo está na tabela a seguir.

| linha | consumo de tempo da linha |
|---|---|
| 1-2 | $= 2\,\Theta(1)$ |
| 3-4 | $= \Theta(X)$ |
| 5-6 | $= 2\,\Theta(X\lceil \lg(n+1) \rceil)$ |
| 7 | $= \Theta(X)$ |
| 8-9 | $= 2\,\Theta(1)$ |
| $T(n)$ | $= \Theta(X\,\lceil \lg(n+1) \rceil + 2\,X + 4)$ |
| | $= \Theta(X\,\lceil \lg(n+1) \rceil)$ |
| | $= \Theta(X\ \lg n)$ |

O consumo de tempo do algoritmo é $\Theta(X \lg n)$, onde $n = b - a$, e o consumo de tempo esperado do algoritmo é $\Theta(\mathrm{E}[X] \lg n)$. Determinaremos $\mathrm{E}[X]$.

Para $i = 1, 2, 3, \ldots$, seja $X_i$ a variável aleatória que indica se as linhas 3-7 foram executadas (*pelo menos*) $i$ vezes, ou seja,

$$X_i = \begin{cases} 1, & \text{se as linhas 3-7 foram executadas } i \text{ vezes,} \\ 0, & \text{em caso contrário.} \end{cases}$$

Portanto,

$$X = X_1 + X_2 + X_3 + \cdots \tag{4}$$

Temos que

$$\mathrm{E}[X_i] = 1 \times \Pr\{X_i = 1\} + 0 \times \Pr\{X_i = 0\} = \Pr\{X_i = 1\} = \Pr\{X \geq i\}.$$

Em palavras, $\mathrm{E}[X_i]$ é igual a probabilidade das linhas 3-7 serem executada $i$ vezes.

Seja $r_i$ o valor de $r$ na $i$-ésima execução da linha 7, para $i = 1, 2, \ldots$ As linha 3-7 são executadas $i$ vezes se e somente se

$$r_1 > n \ \text{ e } \ r_2 > n \ \text{ e } \ \ldots \ \text{ e } \ r_{i-1} > n$$

Assim, $\Pr\{X \geq 1\} = 1$ e, para $i \geq 2$,

$$\begin{aligned}
\Pr\{X \geq i\} &= \Pr\{r_1 > n \ \text{ e } \ r_2 > n \ \text{ e } \ \ldots \ \text{ e } \ r_{i-1} > n\} \\
&= \Pr\{r_1 > n\} \times \Pr\{r_2 > n\} \times \cdots \times \Pr\{r_{i-1} > n\} \\
&< \frac{1}{2} \times \frac{1}{2} \times \cdots \times \frac{1}{2} \qquad (5) \\
&= \frac{1}{2^{i-1}}\,,
\end{aligned}$$

onde a desigualdade (5) será demonstrada mais adiante.

Agora, podemos determinar $\mathrm{E}[X]$. De (4) sabemos que

$$\begin{aligned}
\mathrm{E}[X] &= \mathrm{E}[X_1 + X_2 + X_3 + \cdots] \\
&= \mathrm{E}[X_1] + \mathrm{E}[X_2] + \mathrm{E}[X_3] + \cdots \\
&= \Pr\{X \geq 1\} + \Pr\{X \geq 2\} + \Pr\{X \geq 3\} + \cdots \\
&< 1 + \frac{1}{2} + \frac{1}{2^2} + \cdots \\
&= 1 + 1 \\
&= 2.
\end{aligned}$$

Portanto, $\mathrm{E}[X]$ é $\Theta(1)$ e o consumo de tempo esperado do algoritmo é $\Theta(\lg n)$.

Demonstraremos agora a desigualdade utilizada em (5): na linha 7 vale que

$$\Pr\{r > n\} < \frac{1}{2}\,.$$

Primeiro, notemos que na linha 5:

(i0) $r$ é um número em $0\,..\,2^{j-1}-1$; e
(i1) $\Pr\{r = k\} = 1/2^{j-1}$ para cada $k$ em $0\,..\,2^{j-1}-1$.

Em particular, (i0) e (i1) também valem quando $j = t + 1$, e portanto na linha 7:

(i2) $r$ é um número em $0\,..\,2^t-1$; e
(i3) $\Pr\{r = k\} = 1/2^t$ para cada $k$ em $0\,..\,2^t-1$.

Assim, como $2^{t-1} \leq n \leq 2^t - 1$, na linha 7 vale que

$$\begin{aligned}
\Pr\{r > n\} &= \Pr\{r = n+1\} + \Pr\{r = n+2\} + \cdots + \Pr\{r = 2^t - 1\} \\
&= \frac{1}{2^t} + \frac{1}{2^t} + \cdots + \frac{1}{2^t} \\
&< \frac{2^t - 2^{t-1}}{2^t} \\
&= \frac{2^{t-1}}{2^t} \\
&= \frac{1}{2}\ .
\end{aligned}$$

**Observação:** É ainda possível verificar que na linha 8

(i4) $r$ é um número em $0\,..\,n$; e
(i5) $\Pr\{r = k\} = 1/(n+1)$ para cada $k$ em $0\,..\,n$.

Portanto, o valor devolvido pelo algoritmo é um número em $a\,..\,b$ gerado com probabilidade uniforme $1/(b - a + 1)$.

**Questão 6**
Uma seqüência $X[1\,..\,m]$ é subseqüência de uma seqüência $Y[1\,..\,n]$ se existem índices $i_1 < i_2 < \ldots < i_m$ tais que

$$X[j] = Y[i_j] \quad \text{para } j = 1, \ldots, m.$$

Por exemplo,

$$X = \begin{array}{|c|c|c|c|} \multicolumn{1}{c}{1} & \multicolumn{1}{c}{2} & \multicolumn{1}{c}{3} & \multicolumn{1}{c}{4} \\ \hline \mathtt{B} & \mathtt{C} & \mathtt{D} & \mathtt{B} \\ \hline \end{array} \quad \text{é uma subseqüência de} \quad Y = \begin{array}{|c|c|c|c|c|c|c|} \multicolumn{1}{c}{1} & \multicolumn{1}{c}{2} & \multicolumn{1}{c}{3} & \multicolumn{1}{c}{4} & \multicolumn{1}{c}{5} & \multicolumn{1}{c}{6} & \multicolumn{1}{c}{7} \\ \hline \mathtt{A} & \underline{\mathtt{B}} & \underline{\mathtt{C}} & \mathtt{D} & \underline{\mathtt{D}} & \mathtt{A} & \underline{\mathtt{B}} \\ \hline \end{array}$$

com índices $2 < 3 < 5 < 7$.

Escreva um algoritmo SUBSEQ-CONT$(X, m, Y, n)$ que recebe uma seqüência $X[1\,..\,m]$ e uma seqüência $Y[1\,..\,n]$, e devolve o número de ocorrências de $X$ como subseqüência de $Y$. Sua função deve consumir tempo $O(mn)$.

**Exemplos:**

a) Se

$$X = \begin{array}{|c|c|c|c|c|c|} \multicolumn{1}{c}{1} & \multicolumn{1}{c}{2} & \multicolumn{1}{c}{3} & \multicolumn{1}{c}{4} & \multicolumn{1}{c}{5} & \multicolumn{1}{c}{6} \\ \hline \mathtt{R} & \mathtt{A} & \mathtt{B} & \mathtt{B} & \mathtt{I} & \mathtt{T} \\ \hline \end{array} \quad \text{e} \quad Y = \begin{array}{|c|c|c|c|c|c|c|} \multicolumn{1}{c}{1} & \multicolumn{1}{c}{2} & \multicolumn{1}{c}{3} & \multicolumn{1}{c}{4} & \multicolumn{1}{c}{5} & \multicolumn{1}{c}{6} & \multicolumn{1}{c}{7} \\ \hline \mathtt{R} & \mathtt{A} & \mathtt{B} & \mathtt{B} & \mathtt{B} & \mathtt{I} & \mathtt{T} \\ \hline \end{array},$$

o algoritmo deve devolver 3.

b) Se

$$X = \begin{array}{|c|c|c|} \multicolumn{1}{c}{1} & \multicolumn{1}{c}{2} & \multicolumn{1}{c}{3} \\ \hline \mathtt{B} & \mathtt{A} & \mathtt{G} \\ \hline \end{array} \quad \text{e} \quad Y = \begin{array}{|c|c|c|c|c|c|c|} \multicolumn{1}{c}{1} & \multicolumn{1}{c}{2} & \multicolumn{1}{c}{3} & \multicolumn{1}{c}{4} & \multicolumn{1}{c}{5} & \multicolumn{1}{c}{6} & \multicolumn{1}{c}{7} \\ \hline \mathtt{B} & \mathtt{A} & \mathtt{B} & \mathtt{G} & \mathtt{B} & \mathtt{A} & \mathtt{G} \\ \hline \end{array},$$

o algoritmo deve devolver 5.

Explique sucintamente porque seu algoritmo está correto e tem o consumo de tempo pedido.

**Solução:** Seja $c[i, j]$ o número de ocorrências de $X[1\,..\,i]$ como subseqüência de $Y[1\,..\,j]$. Se $X[i] = Y[j]$ então $c[i, j] = c[i-1, j-1] + c[i, j-1]$, senão $c[i, j] = c[i, j-1]$. Assim, obtemos a recorrência:

$$\begin{array}{lcll} c[0, j] & = & 1, & \text{para } j = 0, 1, \ldots, n; \\ c[i, 0] & = & 0, & \text{para } i = 1, 2, \ldots, m; \\ c[i, j] & = & c[i-1, j-1] + c[i, j-1], & \text{se } i \geq 1, j \geq 1 \text{ e } X[i] = Y[j]; \\ c[i, j] & = & c[i, j-1], & \text{se } i \geq 1, j \geq 1 \text{ e } X[i] \neq Y[j]. \end{array}$$

O algoritmo a seguir baseia-se nessa recorrência.

```
Subseq-Cont(X, m, Y, n)
 1  para j ← 0 até n faça
 2      c[0, j] ← 1
 3  para i ← 1 até m faça
 4      c[i, 0] ← 0
 5  para i ← 1 até m faça
 6      para j ← 1 até n faça
 7          se X[i] = Y[j]
 8              então c[i, j] ← c[i − 1, j − 1] + c[i, j − 1]
 9              senão c[i, j] ← c[i, j − 1]
10  devolva c[m, n]
```

O consumo de tempo pela execução de cada linha do algoritmo está na tabela a seguir.

| linha | consumo de tempo da linha |
|---|---|
| 1-2 | $2\,\Theta(n)$ |
| 3-4 | $2\,\Theta(m)$ |
| 5 | $\Theta(m)$ |
| 6 | $\Theta(mn)$ |
| 7 | $\Theta(mn)$ |
| 8-9 | $\Theta(mn)$ |
| 10 | $\Theta(1)$ |
| **total** | $\Theta(3\,mn + 3\,m + 2\,n + 1) = \Theta(mn)$ |

Eis uma versão *memoized* do algoritmo:

```
Subseq-Cont(X, m, Y, n)
 1  para i ← 0 até m faça
 2      para j ← 0 até n faça
 3          c[i, j] ← indefinido
 4  devolva Lookup (c, m, n)
```

LOOKUP $(c, i, j)$ $\quad\triangleright$ $X$ e $Y$ estão implícitos

**se** $c[i, j] \neq$ INDEFINIDO<br>
    **então** devolva $c[i, j]$<br>
**se** $i = 0$ **então** $c[i, j] = 1$<br>
**senão se** $j = 0$ **então** $c[i, j] = 0$<br>
**senão se** $X[i] = Y[j]$<br>
        **então** $c[i, j] \leftarrow$ LOOKUP $(c, i-1, j-1) +$ LOOKUP $(c, i, j-1)$<br>
        **senão** $c[i, j] \leftarrow$ LOOKUP $(c, i, j-1)$<br>
**devolva** $c[i, j]$