

# Notação Assintótica e Complexidade

## SCC201/501 - Introdução à Ciência de Computação II

Prof. Moacir A. Ponti  
[www.icmc.usp.br/~moacir](http://www.icmc.usp.br/~moacir)

Instituto de Ciências Matemáticas e de Computação – USP

2020/2



# Sumário



# Eficiência assintótica

- O comportamento assintótico é aquele para o qual o tamanho da entrada tende ao infinito.
- Representa a eficiência em termos da ordem de crescimento, ou apenas ordem, do algoritmo.



## Análise assintótica de algoritmos

- parte da contagem de operações e uma função de eficiência
- caracteriza a **complexidade** como uma **função do tamanho da entrada**, e.g.  $n$
- um algoritmo assintoticamente mais eficiente é a melhor escolha para todas as entradas, *a partir de um certo  $n$* .
- analisar a complexidade independente do sistema computacional utilizado



# Sumário



# Notação assintótica: $O$ (big oh)

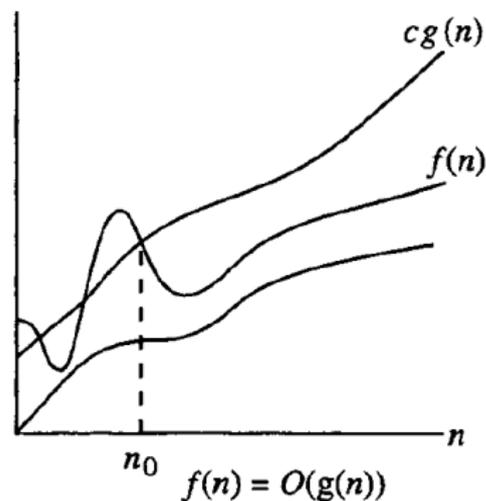
- Para uma dada função  $g(n)$ ,  $O(g(n))$  é um conjunto de funções:

$$O(g(n)) = \{f(n) : \text{existem constantes positivas } c \text{ e } n_0 \text{ tais que} \\ 0 \leq f(n) \leq c \cdot g(n) \text{ para todo } n \geq n_0 \}$$

- $f(n)$  pertence a  $O(g(n))$  se existe  $c > 0$  de forma a estar limitada por  $c \cdot g(n)$  para um valor de  $n$  suficientemente grande



# Notação assintótica: $O$



- com  $n > n_0$ , valores de  $f(n)$  residem em  $c \cdot g(n)$  ou abaixo desse.
- $g(n)$  é um **limite superior** para  $f(n)$
- Exemplo:  $2n^2$  é  $O(n^3)$ 
  - $2n^2$  é limitado superiormente por  $O(n^3)$
  - a taxa de crescimento de  $2n^2$  é **menor ou igual** à taxa de  $n^3$

Fonte da figura: CORMEN et al.(2002)



# Notação assintótica: $O$ (big oh) — exemplos

- Exemplo 1:  $2n + 10$  é  $O(n)$

- podemos realizar uma manipulação para encontrar  $c$  e  $n_0$ :

$$2n + 10 \leq c \cdot n$$

$$c \cdot n - 2n \geq 10$$

$$(c - 2)n \geq 10$$

$$n \geq \frac{10}{c-2}$$

- a afirmação é válida para  $c = 3$  e  $n_0 = 10$ .

- Exemplo 2:  $n^2$  é  $O(n)$

- é preciso encontrar  $c$  que seja sempre maior ou igual a  $n$  para todo valor de um  $n_0$ :

$$n^2 \leq c \cdot n \Rightarrow n \leq c$$

- é impossível pois  $c$  deve ser constante.



# Notação assintótica: $O$ (big oh) — exemplos

- Exemplo 3:  $3n^3 + 20n^2 + 5$  é  $O(n^3)$

- é preciso encontrar  $c > 0$  e  $n_0 \geq 1$  tais que  $3n^3 + 20n^2 + 5 \leq c \cdot n^3$  para  $n \geq n_0$
- como  $3n^3 + 20n^2 + 5 \leq (3 + 20 + 5) \cdot n^3$ , podemos tomar  $c = 28$  e qualquer  $n_0 > 1$

- Exemplo 4:  $3 \log n + 5$  é  $O(\log n)$

- é preciso encontrar  $c > 0$  e  $n_0 \geq 1$  tais que  $3 \log n + 5 \leq c \cdot \log n$  para todo  $n \geq n_0$
- note que  $3 \log n + 5 \leq (3 + 5) \cdot \log n$  se  $n > 1$  ( $\log 1 = 0$ )
- basta tomar, por exemplo,  $c = 8$  e qualquer  $n_0 = 2$

- Exemplo 5:  $2^{n+2}$  é  $O(2^n)$

- é preciso  $c > 0$  e  $n_0 \geq 1$  tais que  $2^{n+2} \leq c \cdot 2^n$  para todo  $n \geq n_0$
- note que  $2^{n+2} = 2^n + 2^2 = 4 \cdot 2^n$
- assim, basta tomar, por exemplo,  $c = 4$  e qualquer  $n_0$



## \*Notação de igualdade para conjuntos de funções\*: $O$

- a igualdade é utilizada no sentido de “representatividade” e pode ser lida como “é”.
- um conjunto em uma fórmula representa uma função anônima.
- Exemplo 6:

$$f(n) = n^3 + O(n^2)$$

significa que existe um  $h(n) \in O(n^2)$  de forma que  $f(n) = n^3 + h(n)$ .

- Exemplo 7:

$$n^2 + O(n) = O(n^2)$$

significa que, para qualquer  $f(n) \in O(n)$  existe  $h(n) \in O(n^2)$  de forma que  $n^2 + f(n) = h(n)$ .



## Notação assintótica: $\Theta$ (theta)

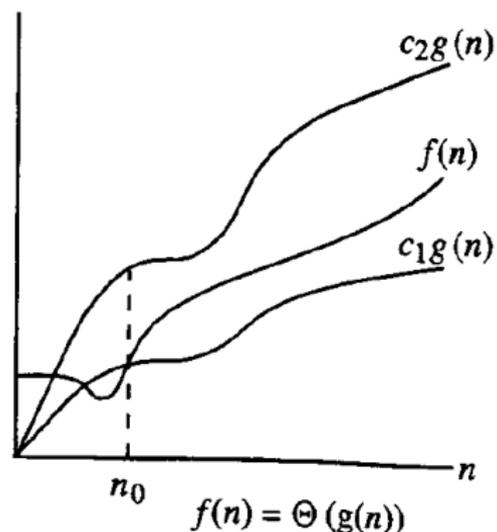
- Para uma dada função  $g(n)$ , denotamos  $\Theta(g(n))$  o conjunto de funções:

$$\Theta(g(n)) = \{f(n) : \text{existem constantes positivas } c_1, c_2 \text{ e } n_0 \text{ tais que } 0 \leq c_1g(n) \leq f(n) \leq c_2g(n) \text{ para todo } n \geq n_0 \}$$

- basta multiplicar  $g(n)$  por duas constantes diferentes para obtermos um limite superior e inferior para  $f(n)$ .
- uma função  $f(n)$  pertence ao conjunto  $\Theta(g(n))$  se existem constantes positivas  $c_1$  e  $c_2$  tais que ela possa estar limitada entre  $c_1g(n)$  e  $c_2g(n)$  para um valor de  $n$  suficientemente grande
- podemos dizer que  $f(n) \in \Theta(g(n))$ , enquanto  $f(n) = \Theta(g(n))$  é abuso da notação de igualdade (não é simétrico)



# Notação assintótica: $\Theta$ (theta)



- Para  $n > n_0$ ,  $f(n)$  reside em  $c_1g(n)$  ou acima dele e em  $c_2g(n)$  ou abaixo desse.
- para todo  $n > n_0$ ,  $f(n) \approx g(n)$  dentro de um fator constante.
- $g(n)$  é um limite “estreito” para  $f(n)$

Fonte da figura: CORMEN et al.(2002)



## Notação assintótica: $\Theta$ (theta)

Para mostrar formalmente que, por exemplo,  $\frac{1}{2}n^2 - 3n = \Theta(n^2)$ :

- definiremos constantes positivas  $c_1$ ,  $c_2$  e  $n_0$  tais que:

$$c_1 n^2 \leq \frac{1}{2} n^2 - 3n \leq c_2 n^2,$$

para todo  $n \geq n_0$ . Dividindo por  $n^2$ :

$$c_1 \leq \frac{1}{2} - \frac{3}{n} \leq c_2,$$

- a desigualdade do lado direito pode ser considerada válida para  $n \geq 1$  escolhendo  $c_2 \geq 1/2$ , e a do lado esquerdo pode ser considerada válida para  $n \geq 7$  escolhendo  $c_1 \geq 1/14$ .
- para  $c_2 = 1/2$ ,  $n = 7$  e  $c_1 = 1/14$ , temos:  $\frac{1}{2}n^2 - 3n = \Theta(n^2)$



# Notação assintótica: $\Theta$ (theta)

- também é possível mostrar que  $6n^3 \neq \Theta(n^2)$ , para isso devemos encontrar:

$$c_1 n^2 \leq 6n^3 \leq c_2 n^2,$$

- dividindo os termos por  $n^2$ , temos:

$$c_1 \leq 6n \leq c_2,$$

- e assim chegamos a:  $n \leq \frac{c_2}{6}$ 
  - mas  $c_2$  é constante e  $n$  não poderia ser suficientemente grande
  - não podemos encontrar também  $c_1$  que atenda à condição.
- assim, por contradição, provamos que  $6n^3 \neq \Theta(n^2)$



# Sumário



# Notação assintótica: $\Omega$ (omega)

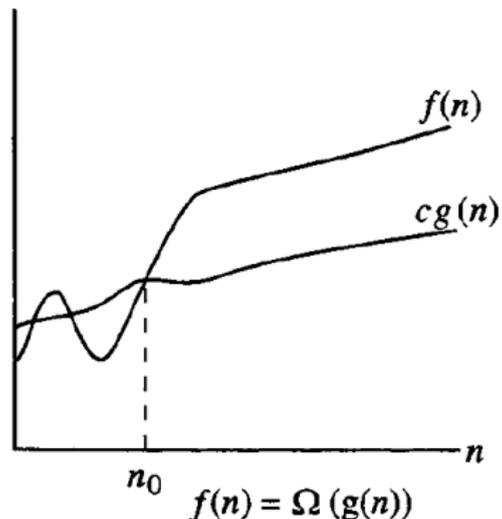
- Na maioria dos casos estamos interessados no limite superior, pois queremos a complexidade no pior caso
- Também podemos analisar o limite assintótico inferior para expressar algo que esteja “pelo menos” em um dado comportamento.
- Para  $g(n)$ ,  $\Omega(g(n))$  é o conjunto de funções:

$$\Omega(g(n)) = \{f(n) : \text{existem constantes positivas } c \text{ e } n_0 \text{ tais que } 0 \leq c \cdot g(n) \leq f(n) \text{ para todo } n \geq n_0 \}$$

- $f(n)$  pertence a  $\Omega(g(n))$  se existem  $c, n_0 > 0$  tais que ela seja limitada inferiormente por  $c \cdot g(n)$  para um valor de  $n$  suficientemente grande



## Notação assintótica: $\Omega$ (omega)



- Para  $n > n_0$ ,  $f(n)$  reside em  $c \cdot g(n)$  ou acima desse.
- Exemplo:  $3n^2 + n = \Omega(n)$ 
  - $n$  é um limite inferior para  $3n^2 + n$ ,
  - a taxa de crescimento de  $3n^2 + n$  é **maior ou igual** à de  $n$

Fonte da figura: CORMEN et al.(2002)



## \*Notação de igualdade para conjuntos de funções\*: $\Omega$

- Exemplo:

$$\sqrt{n} = \Omega(\lg(n))$$

podemos ler: “raiz de  $n$  é, pelo menos, omega de  $\lg(n)$ ” para um  $n$  suficientemente grande ( $n \geq n_0$ ).



- Exemplo: para dois algoritmos quaisquer, considere as funções de eficiência:
  - $f(n) = 1000n$
  - $g(n) = n^2$
- $f$  é maior do que  $g$  para valores pequenos de  $n$
- $g$  cresce mais rapidamente, e finalmente resultará em maiores valores, sendo o ponto de mudança  $n = 1.000$
- segundo as notações vistas, se existe um  $n_0$  a partir do qual  $c \cdot f(n)$  é pelo menos tão grande quanto  $g(n)$ , então, desprezando os fatores constantes e considerando  $n_0 = 1.000$  e  $c = 1$ :
  - $1000n = O(n^2)$
  - ou  $f(n) = O(n^2)$
- o mesmo aconteceria para  $n_0 = 10$  e  $c = 100$ .



# Notação assintótica: relações e teorema

## Analogias

$$\begin{array}{ccc} O & \Omega & \Theta \\ \leq & \geq & = \end{array}$$

## Teorema (1)

para duas funções  $g(n)$  e  $f(n)$ ,  $f(n) = \Theta(g(n))$  se e somente se:

- $f(n) = O(g(n))$  e
- $f(n) = \Omega(g(n))$ .

## Utilidade

- utilizamos o teorema para demonstrar **limites assintoticamente restritos** a partir de limites assintóticos superiores e inferiores.

# Sumário



# Notações $o$ e $\omega$ : limite estritamente superior ou inferior

- Muito parecidas com as notações  $O$  e  $\Omega$ , respectivamente. No entanto possui uma relação do tipo *menor* e *maior*.
  - ao invés de *menor ou igual* e *maior ou igual*.
- Não representam limites próximos, mas apenas estritamente superiores e inferiores.
- O limite pode até mesmo ser distante. Exemplos:
  - 1  $\log_2(n) \in o(n^2)$ .
  - 2  $n^3 \in \omega(n)$ .
- No entanto, como são estritamente superiores e inferiores
  - 1  $n \notin o(n)$ .
  - 2  $n \log_2 n \notin \omega(n \log_2 n)$ .



- Para uma função  $g(n)$ ,  $o(g(n))$  é o *conjunto de funções*:

$$o(g(n)) = \{f(n) : \text{para qualquer } c > 0 \text{ e } n_0 > 0 \text{ tais que} \\ 0 \leq f(n) < c \cdot g(n) \text{ para todo } n \geq n_0 \}$$

- e  $\omega(g(n))$  o *conjunto de funções*:

$$\omega(g(n)) = \{f(n) : \text{para qualquer } c > 0 \text{ e } n_0 > 0 \text{ tais que} \\ 0 \leq c \cdot g(n) < f(n) \text{ para todo } n \geq n_0 \}$$

- $f(n) \in \omega(g(n))$  se e somente se  $g(n) \in o(f(n))$
- Intuitivamente, (se o limite existe),

$$\text{para } \omega(g(n)), \lim_{n \rightarrow 0} \frac{f(n)}{g(n)} = \infty, \text{ e para } o(g(n)) \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$$

- Exemplo 1 :  $2n^2 = o(n^3)$ 
  - $n^2$  é sempre menor que  $n^3$  para um  $n$  suficientemente grande.
  - é preciso apenas determinar  $n_0$  em função de  $c$
- Exemplo 2 :  $2n^3 \neq o(n^3)$ 
  - ignorando as constantes, não podemos dizer que  $n^3$  é sempre menor que  $n^3$  para um  $n$  suficientemente grande.
- Exemplo 3:  $\frac{1}{2}n^2 = \Theta(n^2)$ , mas
  - $\frac{1}{2}n^2 \neq o(n^2)$ , e
  - $\frac{1}{2}n^2 \neq \omega(n^2)$



# Sumário



## Algumas regras

- Se  $T_1(n) = O(f(n))$  e  $T_2(n) = O(g(n))$ , então:  
 $T_1(n) + T_2(n) = \max [O(f(n)), O(g(n))]$  e  
 $T_1(n) \cdot T_2(n) = O(f(n) \cdot g(n))$ .
- $\log_k n = o(n)$  para qualquer  $k$  pois logaritmos crescem muito lentamente



## ... Algumas regras

- Se  $T(x)$  é um polinômio de grau  $n$ , então:  
$$T(x) = \Theta(x^n).$$

## Relembrando

- um polinômio de grau  $n$  é uma função na forma:  
$$f(x) = a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + \cdots + a_0 \cdot x + a_0$$
- classificação em função do grau
  - 0: constante
  - 1: linear
  - 2: quadrático
  - 3: cúbico

# Funções importantes (1/3)

- **Constante:**  $\approx 1$ 
  - independente do tamanho de  $n$ , operações executadas um número fixo de vezes.
- **Logarítmica:**  $\approx \log_b n$ 
  - típica de algoritmos que resolvem um problema transformando-o em problemas menores.
  - para dobrar  $\log_2 n$  é preciso fazer  $\log_2 n^2$ .
  - a base também muda pouco os valores:  $\log_2 n \approx 20$  e  $\log_{10} n \approx 6$  para  $n = 1.000.000$ .
- **Linear:**  $\approx n$ 
  - em geral, uma certa quantidade de operações é realizada sobre cada um dos elementos de entrada.
  - melhor situação para quando é preciso processar  $n$  elementos de entrada e obter  $n$  elementos de saída.



# Funções importantes (2/3)

- **Log linear (ou n-log-n):**  $\approx n \cdot \log_b n$ 
  - típico de algoritmos que resolvem um problema transformando-o em problemas menores, resolvem cada um de forma independente e depois ajunta as soluções.
  - para dobrar  $n \cdot \log_2 n$  é preciso fazer aproximadamente  $n \cdot \log_2 2n$ .
- **Quadrática:**  $\approx n^2$ 
  - ocorre frequentemente quando os dados são processados aos pares, com laços de repetição aninhados.
  - sempre que  $n$  dobra, o tempo de execução é multiplicado por 4.
  - podem ser úteis para resolver problemas de tamanho relativamente pequeno.
- **Cúbica:**  $\approx n^3$ 
  - ocorre em multiplicações de matrizes, com três estruturas de repetição aninhadas.
  - sempre que  $n$  dobra, o tempo de execução é multiplicado por 8.
  - podem ser úteis para resolver problemas de tamanho relativamente pequeno (ou quando não se tem outra opção!).



# Funções importantes (3/3)

- **Exponencial:**  $\approx a^n$ 
  - geralmente ocorre quando se usa uma solução de força bruta.
  - para o caso  $2^n$ , sempre que  $n$  dobra, o tempo de execução é elevado ao quadrado.
  - não são úteis do ponto de vista prático.
- **Fatorial:**  $\approx n!$ 
  - pode ser considerado “exponencial”, mas tem um comportamento muito pior.
  - geralmente ocorre quando se usa uma solução de força bruta.
  - para  $n = 20$ ,  $n! \approx 2,4 \times 10^{18}$ ,
  - – para o dobro  $n = 40$ ,  $n! \approx 8,2 \times 10^{47}$ .
  - definitivamente, **não** são úteis do ponto de vista prático.



# Funções e tempo cronológico

segundos
minutos
séculos

Características Aproximadas do Hardware	
Número de Instruções executadas por Ciclo do relógio (IPC)	8
Frequência (1 / período do ciclo em min.)	3E+09
No. de Instruções por minuto	24E+09

$T(n)$	$n = 20$	$n = 40$	$n = 60$	$n = 80$
$n$	5,3E-08	1,1E-07	1,6E-07	2,1E-07
$n \log n$	2,3E-07	5,7E-07	9,5E-07	1,3E-06
$n^2$	1,1E-06	4,3E-06	9,6E-06	1,7E-05
$n^3$	2,1E-05	1,7E-04	5,8E-04	1,4E-03
$2^n$	2,8E-03	48,9	1,0	1,0E+06
$3^n$	0,2	5,4E+08	1,9E+18	6,6E+27

Fonte da figura: notas de aula do Prof. Ricardo Campello



# Exercício

- Um algoritmo tradicional e muito utilizado possui complexidade  $n^{1,5}$ , enquanto um algoritmo novo proposto é da ordem de  $n \log n$ :
  - $f(n) = n^{1,5}$
  - $g(n) = n \log n$

- Qual algoritmo adotar?

- Uma possível solução:

$$f(n) = \frac{n^{1,5}}{n} = n^{0,5} \quad \Rightarrow \quad (n^{0,5})^2 = n$$

$$g(n) = \frac{n \log n}{n} = \log n \quad \Rightarrow \quad (\log n)^2 = \log^2 n$$

- Como  $n$  cresce mais rapidamente do que qualquer potência de  $\log$ , o algoritmo novo é mais eficiente.

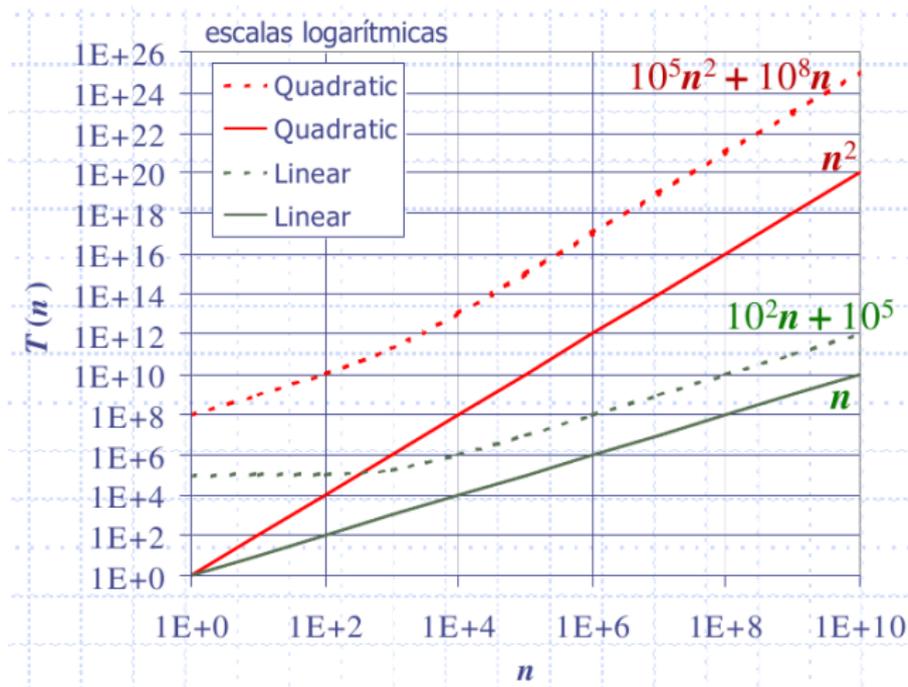


# Dicas de análise na prática

- Se  $f(n)$  for um polinômio de grau  $d$  então  $f(n)$  é  $O(n^d)$ 
  - despreze os termos de menor ordem
  - despreze os fatores constantes
- Use a menor classe de funções possível
  - $2n$  é  $O(n)$ , ao invés de  $2n$  é  $O(2n)$
- Use a expressão mais simples
  - $3n + 5$  é  $O(n)$ , ao invés de  $3n + 5$  é  $O(3n)$



# Dicas de análise na prática



Exemplo:  $n^2$  vs.  $10^5 n^2 + 10^8 n$  e  $n$  vs.  $10^2 n^2 + 10^5$

Fonte da figura: notas de aula do Prof. Ricardo Campello



- Há casos em que a análise assintótica ignora fatores assintoticamente irrelevantes, mas relevantes na prática: em especial quando temos interesse em entradas relativamente pequenas.
- Ao comparar dois algoritmos com tempo de execução:

- $f(n) = 10^{100}n$ , e

- $g(n) = 10n \log n$

pela análise assintótica, o primeiro é mais eficiente

- No entanto,  $10^{100}$  é o número estimado (por alguns astrônomos) como o limite superior para a quantidade de átomos no universo observável
  - $10n \log n > 10^{100}n$  apenas para  $n > 2^{10^{99}}$



- **Repetições:** o tempo de execução é pelo menos o tempo dos comandos dentro da repetição multiplicada pelo número de vezes que é executada.

- o exemplo abaixo é  $O(n)$

```
para i de 1 ate n faça
```

```
  a = a*i
```

- **Repetições aninhadas:** análise feita de dentro para fora

- o tempo total é o tempo de execução dos comandos multiplicado pelo produto do tamanho de todas as repetições.
- o exemplo abaixo é  $O(n^2)$

```
para i de 1 ate n faça
```

```
  para j de 0 ate n-1 faça
```

```
    a = a*(i+j)
```



- **Condições:** o tempo nunca é maior do que o tempo do teste mais o tempo do maior entre os comandos dentro do bloco do “então” e do “senão”

- o exemplo abaixo é  $O(n)$ , mas não é  $\Theta(n)$ .

```
se (a < b) entao
```

```
  a = a + 1
```

```
senao
```

```
  para i de 1 ate n-1 faca
```

```
    a = a*i
```

- **Chamadas à subrotinas:**

- a subrotina deve ser analisada primeiro e depois ter suas unidades de tempo incorporadas ao programa que a chamou



# Exercício

- Quantas unidades de tempo são necessárias para rodar o algoritmo abaixo? Qual a ordem de complexidade de tempo?

```
01  inicio
02      i, j: inteiro
03      A: vetor inteiro de n posicoes
04      i = 1
05
06      enquanto (i < n) faca
07          A[i] = 0
08          i = i + 1
09
10      para i = 1 ate n faca
11          para j = 1 ate n faca
12              A[i] = A[i] + (i*j)
13  fim
```



- CORMEN, T.H. et al. **Algoritmos: Teoria e Prática** (Caps. 1–3). Campus. 2002.
- ZIVIANI, N. **Projeto de algoritmos**: com implementações em Pascal e C (Cap. 1). 2.ed. Thomson, 2004.
- FEOFILOFF, P. **Minicurso de Análise de Algoritmos**, 2010. Disponível em: <http://www.ime.usp.br/~pf/livrinho-AA/>.
- DOWNEY, A.B. **Analysis of algorithms** (Cap. 2), Em: Computational Modeling and Complexity Science. Disponível em: <http://www.greenteapress.com/compmo/html/book003.html>
- ROSA, J.L. **Notas de Aula de Introdução a Ciência de Computação II**. Universidade de São Paulo. Disponível em: <http://coteia.icmc.usp.br/mostra.php?ident=639>
- CAMPELLO, R. **Notas de Aula de Introdução a Ciência de Computação II**. Universidade de São Paulo. Disponível em: <http://coteia.icmc.usp.br/mostra.php?ident=611>

