Projeto e Análise de Algoritmos

Crescimento de funções - Aula 3

Diane Castonguay
diane@inf.ufg.br

Instituto de Informática
Universidade Federal de Goiás

Notação Assintótica: Definição *O*

Sejam f, g : $\mathbb{R} \rightarrow \mathbb{R}$ (ou $\mathbb{N} \rightarrow \mathbb{N}$) duas funções.
Dizemos que:

$$f(n) \in O(g(n))$$

se exitem duas constantes positivas

$$c,\ n_0 > 0 \text{ tais que}$$

$$0 \leq f(n) \leq c * g(n) \quad \forall n \geq n_0.$$

Notação Assintótica: Definição Ω

Sejam f, g : $\mathbb{R} \rightarrow \mathbb{R}$ (ou $\mathbb{N} \rightarrow \mathbb{N}$) duas funções.
Dizemos que:

$f(n) \in \Omega(g(n))$
se exitem duas constantes positivas
c, $n_0 > 0$ tais que

$$0 \leq c * g(n) \leq f(n) \quad \forall n \geq n_0.$$

Notação Assintótica: Definição Θ

Sejam f, g : $\mathbb{R} \to \mathbb{R}$ (ou $\mathbb{N} \to \mathbb{N}$) duas funções.
Dizemos que:

$$f(n) \in \Theta(g(n))$$

se

$$f(n) \in O(g(n)) \text{ e } f(n) \in \Omega(g(n))$$

Notação: Por abuso de linguagem, dizemos que

$$f(n) = \Theta(g(n))$$

Notação Assintótica: Exemplos

$$n! = O(n^n)$$
$$n! = \Omega(2^n)$$
$$\log(n!) = \Theta(n \log n)$$

para qualquer a > 0.

$$\lg n = O(n^a)$$
$$\lg^k n = (\lg n)^k = O(n^a) \text{ para qualquer } k$$

Para todos reais a e b, a >1

$$n^b = O(a^n)$$

```
/*
 * Ordena um vetor.
 * O comprimento do vetor é denotado por comprimento[A].
 */
INSERTION-SORT(A)
1.      para j ← 2 até comprimento[A] faça
2.              chave ← A[j]
3.              // Inserir A[j] na seqüência ordenada A[1..j-1]
4.              i ← j-1
5.              enquanto i > 0 e A[i] > chave faça
6.                      A[i+1] ← A[i]
7.                      i ← i-1
8.              fim-enquanto
9.              A[i+1] ← chave
10.     fim-para
```

Linha	Custo	Vezes
1. <u>para</u> j←2 <u>até</u> n	c1	n
2. *chave* ← A[j]	c2	n-1
3. // Comentário	0	n-1
4. i ← j-1	c4	n-1
5. <u>enquanto</u> i > 0 e A[i] > *chave* <u>faça</u>	c5	$\sum_j t_j$
6. A[i+1] ← A[i]	c6	$\sum_j (t_j - 1)$
7. i ← i-1	c7	"
8. <u>fim-enquanto</u>	0	$\sum_j t_j$
9. A[i+1] ←*chave*	c9	n-1
10. <u>fim-para</u>	0	n-1

```
SELEÇÃO-SORT(A)
1.      para i ← 1 até comprimento[A]-1 faça
2.              //achamos o iesimo menor valor do vetor
3.              menor ← i
4.              para j ← i+1 até comprimento[A] faça
5.                      se V[j] < V[menor]
6.                          então menor ← j
7.                      fim-se
8.              fim-para
9.              se menor ≠ i
10.                     //trocamos os valores de V[menor] e V[i]
11.                 então aux ← V[menor]
12.                         V[menor] ← V[i]
13.                         V[i] ← aux
14.             fim-se
15.     fim-para
```

```
BOLHA(A)
1.  para i ← 1 até comprimento[A]-1 faça
2.          para j ← 1 até comprimento[A]-i faça
3.            se A[j] > A[j+1])
4.              //troca A[j] com A[j+1]
5.                  então aux ← A[j]
6.                        A[j] ← A[j+1]
7.                        A[j+1] ← aux
8.            fim-se
9.        fim-para
10. fim-para
```

```
BOLHA-MELHOR(A)
1.  ultimatroca ← comprimento[A]
2.  troca ← 1
3.  enquanto ultimatroca > 1 faça
2.          para j ← 1 até ultimatroca-1 faça
3.                  se A[j] > A[j+1]
4.                      //troca A[j] com A[j+1]
5.                      então aux ← A[j]
6.                              A[j] ← A[j+1]
7.                              A[j+1] ← aux
8.                              troca ← j
8.                  fim-se
9.          fim-para
10.         ultimatroca ← troca
11.         troca ← 1
10.fim-enquanto
```