Departamento de Ciência de Computadores — FCUP

Desenho e Análise de Algoritmos CC2001 — 2017/2018

Folha 2 – Ordens de grandeza e análise da complexidade assintótica de algoritmos

Para recordar:

As ordens de grandeza O, Θ e Ω são assim definidas:

$$O(g(n)) = \{f(n) \mid \text{existem } c > 0 \text{ e } n_0 > 0 \text{ tais que } f(n) \leq cg(n) \text{ para todo } n \geq n_0\}$$
$$\Omega(g(n)) = \{f(n) \mid \text{existem } c > 0 \text{ e } n_0 > 0 \text{ tais que } f(n) \geq cg(n) \text{ para todo } n \geq n_0\}$$
$$\Theta(g(n)) = \{f(n) \mid \text{existem } c_1 > 0, c_2 > 0 \text{ e } n_0 > 0 \text{ tais que } c_1g(n) \leq f(n) \leq c_2g(n) \text{ para todo } n \geq n_0\}$$

As notações $f(n)$ e $g(n)$ estão a ser usadas com as duas interpretações. Habitualmente, $f(n)$ designa a imagem de n pela função f, mas, nesta definição, $f(n)$ designa também a função $f : \mathbb{N} \to \mathbb{R}_0^+$ que a cada n associa $f(n)$. Do mesmo modo, $g(n)$ designa a função $g : \mathbb{N} \to \mathbb{R}_0^+$ que a cada n associa $g(n)$. Assim, em $f(n) \in O(g(n))$ estamos a classificar as funções f e g e em $f(n) \leq cg(n)$ estamos a referir as imagens de n por f e por g.

- $f(n) \in O(g(n))$ sse $f(n)$ é **majorada** por $cg(n)$ para alguma constante $c > 0$, a partir de uma certa ordem (definida por n_0).
- $f(n) \in \Omega(g(n))$ sse $f(n)$ é **minorada** por $cg(n)$ para alguma constante $c > 0$, a partir de uma certa ordem (definida por n_0).
- $f(n) \in \Theta(g(n))$ sse $f(n)$ é **majorada** por $c_2g(n)$ e **minorada** por $c_1g(n)$ para algum $c_1 > 0$ e algum $c_2 > 0$, a partir de uma certa ordem (definida por n_0).

Exercícios

1. Considere as sucessões de termo geral $s_n = 30n+70$, $r_n = 15n+2000$, $u_n = 3n^3+n^2$, $v_n = 7n\log_2(n)$ e $w_n = 2^n$, $t_n = 3000000$, e $q_n = 1000\log_2(n)$, para $n \geq 1$.

a) Usando diretamente a definição das ordens de grandeza, prove que:

1. $s_n \in \Theta(r_n)$ e $r_n \in \Theta(s_n)$.
2. $s_n \in \Omega(t_n)$ e $t_n \notin \Omega(s_n)$
3. $q_n \in \Omega(t_n)$ e $t_n \notin \Omega(q_n)$
4. $q_n \in O(w_n)$ e $w_n \in \Omega(q_n)$.
5. $u_n \notin O(w_n)$ e $u_n \in \Omega(w_n)$
6. $u_n \notin \Omega(q_n)$ e $q_n \notin \Omega(v_n)$

b) Classifique a ordem de grandeza de cada uma das sucessões indicadas usando $O(1)$, $O(\log_2(n))$, $O(n)$, $O(n\log_2(n))$, $O(n^2)$, $O(n^3)$, $O(2^n)$, e $O(3^n)$, e também as classes $\Omega(\cdot)$ e $\Theta(\cdot)$ correspondentes.

2. Sendo $w_n = 2^n$, para $n \geq 1$, justificar que $2^n \notin O(n^k)$ para nenhum $k \geq 1$.

3. O algoritmo seguinte resolve o problema de imprimir a soma dos dois últimos valores de uma sequência de valores dada pelo utilizador, sendo lidos pelo menos dois valores além do valor -1, o qual indica que a sequência terminou.

ler($penultimo$);	c_1: leitura e atribuição do valor
ler($ultimo$);	c_1: leitura e atribuição do valor
ler($novo$);	c_1: leitura e atribuição do valor
Enquanto ($novo \neq -1$) fazer	c_2: avaliação da condição e transferências de controlo
$penultimo \leftarrow ultimo$;	c_3: atribuição do valor de uma variável a outra
$ultimo \leftarrow novo$;	c_3: atribuição do valor de uma variável a outra
ler($novo$);	c_1: leitura e atribuição do valor
escrever($penultimo + ultimo$);	c_4: avaliação da expressão e escrita do valor

a) Indicar um invariante de ciclo que permita concluir que o algoritmo está correto.

b) Considerando o modelo de custos indicado, determine a expressão que define o tempo de execução do algoritmo no melhor e no pior caso, quando aplicado a instâncias de tamanho n (onde n denota o número total de inteiros lidos). Justifique se trata de um algoritmo $\Theta(n)$ no pior caso e no melhor caso.

c) Justificar que o algoritmo é *assintoticamente ótimo*, ou seja, que qualquer algoritmo que resolva o problema tem a mesma complexidade assintótica ou pior.

4. Seja $T_{(v,n,x)}(n)$ o tempo que a função seguinte requer quando aplicada a uma instância (v, n, x) supondo que $n \geq 1$ e $v[0]$ designa o primeiro elemento de v.

```
PROCURA(v, n, x)
  i ← 0;
  Enquanto (i < n ∧ v[i] ≠ x) fazer
      i ← i + 1;
  Se (i < n) então retorna i;
  retorna −1;
```

a) Justificar que, no melhor caso, $T_{(v,n,x)}(n) \in O(1)$.

b) Justificar que, no pior caso, $T_{(v,n,x)}(n) \in \Omega(n)$.

c) Averiguar a veracidade de:

- $T_{(v,n,x)}(n) \in \Theta(n)$ qualquer que seja (v, n, x).
- O tempo máximo que a função demora em instâncias de tamanho n é $O(n)$.

5. Admita que Mat é uma matriz, com pelo menos n linhas e n colunas, indexadas a partir de 0. Determine o tempo de execução da função seguinte no melhor caso e no pior caso, e indique a sua ordem de grandeza. Comece por definir o modelo de custos e a propriedade que carateriza as instâncias nas duas situações.

```
SIMETRICA(Mat, n)
  i ← 1
  Enquanto(i < n) fazer
      j ← 0;
      Enquanto(j < i) fazer
          Se (Mat[i][j] ≠ Mat[j][i]) então
              retorna 0;
          j ← j + 1;
      i ← i + 1;
  retorna 1;
```

6. Considerar os problemas “Disse que disse”, “Sentar ou não sentar?” e “Plano de Férias”. Para cada um dos problemas:

a) Escrever, em pseudocódigo, um algoritmo eficiente para o resolver.

b) Justificar a correção desse algoritmo (sucintamente mas com rigor).

c) Analisar a sua complexidade assintótica.

d) Implementar o algoritmo em linguagem C ou Java (ou C++).