Análise de Algoritmos

Exercícios

09 de março de 2017

1. Prove que:

   (a) $n \in \mathcal{O}(n)$.

   (b) $f(n) \in \mathcal{O}(f(n))$.

   (c) $n \in \mathcal{O}(n^2)$.

   (d) $n^k \in \mathcal{O}(n^{k+1})$.

   (e) $n^k \in \mathcal{O}(n^\ell)$, para $\ell \geq k$.

   (f) $\alpha n^k \in \mathcal{O}(n^\ell)$, para $\ell \geq k$.

   (g) $\alpha n^k + \beta \in \mathcal{O}(n^\ell)$, para $\ell \geq k$.

   (h) $g(n) + f(n) \in \mathcal{O}(g(n))$, para $f(n) \in \mathcal{O}(g(n))$.

   (i) (transitividade) se $f(n) \in \mathcal{O}(g(n))$ e $g(n) \in \mathcal{O}(h(n))$ então $f(n) \in \mathcal{O}(h(n))$.

   (j) (inclusão) se $f(n) \in \mathcal{O}(g(n))$ então $\mathcal{O}(f(n)) \subseteq \mathcal{O}(g(n))$.

   (k) (polinômios) $\sum_{i=0}^{k} \alpha_i n^i \in \mathcal{O}(n^k)$.

   (l) $\log_b n \in \mathcal{O}(\log_a n)$, para $a, b > 1$
   (lembre que $\log_b n = (\log_b a)(\log_a n)$).

   (m) $\log_b n \in \mathcal{O}(n)$, para $b > 1$.

   (n) $\mathcal{O}(1) \subseteq \mathcal{O}(\log n) \subseteq \mathcal{O}(n) \subseteq \mathcal{O}(n^2) \subseteq \cdots \subseteq \mathcal{O}(n^k)$, para $k \geq 2$.

   (o) (soma) se $f_1(n) \in \mathcal{O}(f(n))$ e $g_1(n) \in \mathcal{O}(g(n))$ então $f_1(n) + g_1(n) \in \mathcal{O}(f(n) + g(n))$.

   (p) (muliplicação) se $f_1(n) \in \mathcal{O}(f(n))$ e $g_1(n) \in \mathcal{O}(g(n))$ então $f_1(n)g_1(n) \in \mathcal{O}(f(n)g(n))$.

2. Prove que:

   (a) $n \in \Omega(n)$.
   (b) $f(n) \in \Omega(f(n))$.
   (c) $n^2 \in \Omega(n)$.
   (d) $n^{k+1} \in \Omega(n^k)$.
   (e) $n^k \in \Omega(n^\ell)$, para $\ell \leq k$.
   (f) $\alpha n^k \in \Omega(n^\ell)$, para $\ell \leq k$.
   (g) $\alpha n^k + \beta \in \Omega(n^\ell)$, para $\ell \leq k$.
   (h) (transitividade) se $f(n) \in \Omega(g(n))$ e $g(n) \in \Omega(h(n))$ então $f(n) \in \Omega(h(n))$.
   (i) (inclusão) se $f(n) \in \Omega(g(n))$ então $\Omega(f(n)) \subseteq \Omega(g(n))$.
   (j) (polinômios) $\sum_{i=0}^{k} \alpha_i n^i \in \Omega(n^k)$, se $\alpha_k > 0$.
   (k) $\log_b n \in \Omega(\log_a n)$, para $a, b > 1$.
   (l) $n \in \Omega(\log_b n)$, para $b > 1$.
   (m) $\Omega(n^k) \subseteq \cdots \subseteq \Omega(n^2) \subseteq \Omega(n) \subseteq \Omega(\log_b n) \subseteq \Omega(1)$, para $b > 1$ e $k \geq 2$.
   (n) (muliplicação) se $f_1(n) \in \Omega(f(n))$ e $g_1(n) \in \Omega(g(n))$ então $f_1(n)g_1(n) \in \Omega(f(n)g(n))$.

3. Prove que:

   (a) $f(n) \in \Theta(f(n))$.
   (b) (simetria) se $f(n) \in \Theta(g(n))$ então $g(n) \in \Theta(f(n))$.
   (c) $\sum_{i=1}^{n} i \in \Theta(n^2)$.

4. Prove que para toda $f, g \colon \mathbb{N} \to \mathbb{R}$,

$$\mathcal{O}(f(n))\mathcal{O}(g(n)) = \mathcal{O}((fg)(n)).$$

5. Prove que para toda $f, g \colon \mathbb{N} \to \mathbb{R}$, se $f(n) = \mathcal{O}(g(n))$, então

$$\mathcal{O}((f+g)(n)) = \mathcal{O}(g(n)).$$

6. Sejam $f, g \colon \mathbb{N} \to \mathbb{R}$

(a) Prove que
$$\Omega(f(n))\Omega(g(n)) = \Omega((fg)(n)).$$

(b) Sejam $f, g\colon \mathbb{N} \to \mathbb{R}$ e seja $m\colon \mathbb{N} \to \mathbb{R}$ dada por
$$m(n) = \max\{f(n), g(n)\}.$$
Prove que
$$m(n) = \Theta(f(n) + g(n)).$$

(c) Sejam $a, b \in \mathbb{R}$. Prove que
$$(n+a)^b = \Theta(n^b).$$

(d) Prove que $\lceil x \rceil$ é o único inteiro que satisfaz
$$x \leq \lceil x \rceil < x + 1.$$

(e) Prove que
$$z - \lfloor x \rfloor = \lceil z - x \rceil,$$
para todo $x \in \mathbb{R}$ e todo $z \in \mathbb{Z}$.

(f) Sejam $a, b \in \mathbb{Z}$ com $a \leq b$, e sejam
$$\begin{aligned} n &= b - a + 1, \\ m &= \left\lfloor \frac{a+b}{2} \right\rfloor. \end{aligned}$$
Prove que
$$\begin{aligned} m - a + 1 &= \left\lfloor \frac{n+1}{2} \right\rfloor, \\ b - (m+1) + 1 &= \left\lceil \frac{n-1}{2} \right\rceil. \end{aligned}$$

(g) Sejam $f\colon \mathbb{N} \to \mathbb{R}$, $n_0 \in \mathbb{N}$ e $\epsilon < 1$ tais que
$$\begin{aligned} f(n) &= \mathcal{O}(1), \text{ para todo } n \leq n_0 \\ f(n) &= \mathcal{O}(1) + \max\{f(\lfloor \epsilon n \rfloor), f(\lceil (\epsilon) n \rceil)\}, \text{ para todo } n \geq n_0. \end{aligned}$$
Prove que
$$f(n) = \mathcal{O}(\log n).$$

(h) Sejam $f\colon \mathbb{N} \to \mathbb{R}$, $n_0 \in \mathbb{N}$ e $0 < \epsilon < 1$ tais que

$$\begin{aligned} f(n) &= \Omega(1), \text{ para todo } n \le n_0 \\ f(n) &= \Omega(1) + \max\{f(\lfloor \epsilon n \rfloor), f(\lceil \epsilon n \rceil)\}, \text{ para todo } n \ge n_0. \end{aligned}$$

Prove que

$$f(n) = \Omega(\log n).$$

7. Prove que:

   (a) $3(n + \log n) + 5 \in \Theta(n)$;
   (b) $g(n) + f(n) \in \mathcal{O}(g(n))$, para $f(n) \in \mathcal{O}(g(n))$;
   (c) se $f(n) \in \mathcal{O}(g(n))$ e $g(n) \in \mathcal{O}(h(n))$ então $f(n) \in \mathcal{O}(h(n))$;
   (d) $\sum_{i=0}^{k} \alpha_i n^i \in \mathcal{O}(n^k)$.
   (e) $6(n + \log(n^3)) \in \Theta(n)$;
   (f) se $f(n) \in \mathcal{O}(n)$ e $h(n) \in \mathcal{O}(n^2)$ e $(fh)(n) = f(n)h(n)$, então $(fh)(n) \in \mathcal{O}(n^3)$;
   (g) $n^2 + 3n - 5 \in \Theta(n^2)$.

8. Prove a seguinte afirmação ou apresente um contra-exemplo: para quaisquer funções $f, g : \mathbb{N} \to \mathbb{R}$, $f(n) \in \mathcal{O}(g(n))$ se, e somente se, $g(n) \in \Omega(f(n))$.

9. Considere o algoritmo X, abaixo, que recebe um vetor $v[a..b]$. Faça a análise de complexidade deste algoritmo e use a notação Θ para descrever sua complexidade. Não esqueça de deixar claro qual o tamanho da entrada.

Algoritmo 1: $X(a, b, v)$

```
Entrada: vetor v[a..b]
Saída: um número
R ← 0
i ← a
enquanto i ≤ b faça
    se v[i] > R então
        R ← v[i] − R
    senão
        R ← R − v[i]
    i ← i + 1
Devolva R
```

10. Considere o algoritmo Z, abaixo, que recebe um inteiro x. Faça a análise de complexidade deste algoritmo e use a notação $\mathcal{O}$, Ω ou Θ, conforme achar necessário. Deixe claro qual o tamanho da entrada e se é preciso dividir em pior caso e melhor caso.

Algoritmo 2: $Z(x)$

Entrada: $x \in \mathbb{N}$
Saída: um número
se $x \leq 1$ então
    Devolva x
Devolva $2 * Z(x-2)$

11. Qual o valor retornado pelo algoritmo abaixo em função de n?

Algoritmo 3: $Loops(n)$

Entrada: $n \in \mathbb{N}$
Saída : um número
$r \leftarrow 0$
Para $i \leftarrow 1$ *até* n
    Para $j \leftarrow 1$ *até* i
        Para $k \leftarrow j$ *até* $n - i$
            $r \leftarrow r + 1$
Devolva r

12. Considere três funções $f, g, h : \mathbb{N} \to \mathbb{R}$ tais que $f(n) \in \mathcal{O}(g(n))$ e $h(n) \in \Omega(g(n))$. É verdade que $f(n) \notin \Theta(h(n))$? Justifique.

13. Considere o algoritmo abaixo. Faça a análise de complexidade deste algoritmo e use a notação $\mathcal{O}$, Ω ou Θ, conforme achar necessário. Deixe claro qual o tamanho da entrada e se é preciso dividir em pior caso e

melhor caso.

Algoritmo 4: $A(a, b, v)$

Entrada: vetor $v[a..b]$
Saída: um número
$R \leftarrow 0$
$i \leftarrow a$
$j \leftarrow b$
enquanto $i \leq j$ faça
    $k \leftarrow i$
    enquanto $k \leq j$ faça
        $R \leftarrow R + v[k]$
        $k \leftarrow k + 1$
    $i \leftarrow i + 1$
    $j \leftarrow j - 1$
Devolva R

---

14. Considere o algoritmo abaixo. Faça a análise de complexidade deste algoritmo e use a notação $\mathcal{O}$, Ω ou Θ, conforme achar necessário. Deixe claro qual o tamanho da entrada e se é preciso dividir em pior caso e melhor caso.

Algoritmo 5: $B(x)$

Entrada: $x \in \mathbb{N}$
Saída: um número
se $x \leq 1$ então
    Devolva x
Devolva $4 * B(x - 1) + 2$

---

15. Seja um algoritmo A (com instância I) que usa divisão e conquista. Suponha que $|I| = n$, que os casos base ($n \leq n_0$) são resolvidos em tempo $\mathcal{O}(1)$, que a instância I é dividida em duas partes, I_1 e I_2, tais que $|I_1| + |I_2| = |I|$, e que a complexidade de tempo de dividir a instância e juntar as respostas é $\mathcal{O}(n)$. Responda:

    a) Qual a complexidade de tempo de A se $|I_1| = |I_2|$?

    b) Qual a complexidade de tempo de A se $|I_1| = 2|I_2|$?

    c) Qual a complexidade de tempo de A se $|I_1| = c|I_2|$, para uma constante c?

16. Seja a função $F : \mathbb{N}^2 \to \mathbb{N}$ definida pela seguinte recorrência:

$$F(i,j) = \begin{cases} 1, & \text{se } i = 0 \text{ ou } j = 0, \\ F(i,j-1) + F(i-1,j-1) + F(i-1,j) + i + j, & \text{caso contrário,} \end{cases}$$

Responda:

a) Qual a complexidade de tempo de calcular $F(i,j)$ usando Programação Dinâmica?

b) Qual a complexidade de tempo de calcular $F(i,j)$ usando recursão direta?

17. O que significa a verificação de corretude de um algoritmo? Qual a importância de se fazer esta verificação? Exemplifique apresentando uma verificação de corretude de um algoritmo (simples) a sua escolha.

18. Explique, em linhas gerais, como se deve fazer o cálculo da complexidade de caso médio de um algoritmo.

19. Sejam A e B dois problemas de decisão em `NP`. Temos uma redução polinomial de A para B (transformação das instâncias de A em instâncias de B e transformação das respostas de B em respostas de A, em tempo polinomial). Para qual dos dois problemas é verdade que se está em `NP-completo` o outro também estará? Por que?