Análise de Algoritmos

Exercícios

24 de fevereiro de 2023

Sumário

1	Introdução	2
2	Notação Assintótica	4
3	Divisão e Conquista	8
4	Algoritmos Gulosos	14
5	Programação Dinâmica	14
A	Solução de Recorrências	18
B	O “Teorema Mestre”	19

1 Introdução

1. Prove¹ que o Algoritmo S abaixo

$S(x, v, a, b)$
Se $a > b$ Devolva $a - 1$
Se $x \geq v[b]$ Devolva b
Devolva $S(x, v, a, b - 1)$

é uma solução para o seguinte problema computacional.

Busca em Vetor Ordenado (BVO)

Instância: (x, v, a, b) onde

x : é um valor,

a, b : são inteiros,

v : é um vetor de valores indexado por $[a..b]$.

Resposta: O “lugar onde x deveria estar em v ”, isto é, o único $m \in [a - 1..b]$ satisfazendo

$$\begin{aligned} v[i] \leq x, & \quad \text{para todo } i \in [a..m], \\ x < v[i], & \quad \text{para todo } i \in [m + 1..b]. \end{aligned}$$

2. Resolva as seguintes recorrências onde $c_1, c_2 \in \mathbb{C}$.

(a)

$$f(n) = \begin{cases} c_1, & \text{se } n = 0, \\ c_2 + f(n - 1), & \text{se } n \geq 1, \end{cases}$$

(b)

$$f(n) = \begin{cases} c_1, & \text{se } n = 0, \\ c_2 + f(\lfloor \frac{n-1}{2} \rfloor), & \text{se } n \geq 1, \end{cases}$$

¹**Sugestão:** Prove por indução em $n := b - a + 1$ que se (x, v, a, b) é uma instância de BVO, e $S(x, v, a, b) = m$, então

$$\begin{aligned} v[i] \leq x, & \quad \text{para todo } i \in [a..m], \text{ e} \\ x < v[i], & \quad \text{para todo } i \in [m + 1..b]. \end{aligned}$$

3. Prove² que o Algoritmo B abaixo

$B(x, v, a, b)$
Se $a > b$ Devolva $a - 1$ $m \leftarrow \lfloor \frac{a+b}{2} \rfloor$ Se $x < v[m]$ Devolva $B(x, v, a, m - 1)$ Devolva $B(x, v, m + 1, b)$

é uma solução para o problema de Busca em Vetor Ordenado (cfr. Exercício 1).

4. Sejam $a, b \in \mathbb{N}$ e sejam

$$m(a, b) := \left\lfloor \frac{a+b}{2} \right\rfloor,$$

$$n(a, b) := b - a + 1.$$

Prove que

$$n(a, m(a, b) - 1) = n(m(a, b) + 1, b) = \left\lfloor \frac{n(a, b) - 1}{2} \right\rfloor.$$

5. Prove que

$$c_1 + c_2 \lg n \leq f(n) \leq c_1 + c_2 \lg(n + 1), \text{ para todo } n \geq 1,$$

onde c_1, c_2 e $f(n)$ são como no Exercício 2b.

6. Uma *árvore (binária)* é uma *árvore vazia*, denotada por Λ , ou é um par $T = (E(T), D(T))$ onde $E(T)$ e $D(T)$ são árvores binárias. A árvore T é uma *folha* se $E(T)$ e $D(T)$ são ambas árvores vazias. A *altura* de T é dada por

$$h(T) = \begin{cases} 1, & \text{se } T = \Lambda, \\ 1 + \max \{h(E(T)), h(D(T))\}, & \text{se } T \neq \Lambda. \end{cases}$$

Prove que³ se T tem $n \geq 1$ folhas então $h(T) \geq \lg n$.

²**Sugestão:** Indução em $b - a + 1$

³**Sugestão:** Observe que se T é uma árvore com n folhas, então a árvore resultante de retirar de T todas as suas folhas tem pelo menos $n/2$ folhas.

2 Notação Assintótica

7. Prove que se $f(n) = \mathcal{O}(1)$ e $g(n) = \mathcal{O}(1)$ então $f(n) + g(n) = \mathcal{O}(1)$, isto é, que

$$\mathcal{O}(1) + \mathcal{O}(1) = \mathcal{O}(1).$$

8. Prove que se $f(n) = \mathcal{O}(1)$ e $\lim g(n) = \infty$, então

$$\frac{f(n)}{g(n)} = \mathcal{O}(1),$$

isto é, que se $\lim g(n) = \infty$, então

$$\frac{\mathcal{O}(1)}{g(n)} = \mathcal{O}(1).$$

9. Sejam $f: \mathbb{N} \rightarrow \mathbb{N}$ e $l(n) = \mathcal{O}(1)$ e $a \in \mathbb{N}$. Prove que

$$\sum_{i=a}^{f(n)} l(i) = \mathcal{O}(1)f(n).$$

10. Prove que

(a) $\frac{1}{n} = \mathcal{O}(1)$.

(b) $\frac{\log n}{n} = \mathcal{O}(1)$.

11. Prove que se $f: \mathbb{N} \rightarrow \mathbb{R}$ tem limite finito, então $f(n) = \mathcal{O}(1)$.

12. Prove que se $f(n) = \mathcal{O}(1)$ e $g(n) = \mathcal{O}(1)$ então $f(n)g(n) = \mathcal{O}(1)$, isto é, que

$$\mathcal{O}(1)\mathcal{O}(1) = \mathcal{O}(1).$$

13. Seja

$$h(n) := \left\lfloor \frac{n-1}{2} \right\rfloor,$$

e seja

$$u(n) = \min \{k \in \mathbb{N} \mid h^k(n) \leq 0\}.$$

Prove que

$$u(n) = \mathcal{O}(1) \lg n.$$

14. Prove que

$$\log_b n = \mathcal{O}(\log n), \text{ para todo } b > 1.$$

15. Sejam $n_0 \in \mathbb{N}$, $f: \mathbb{N} \rightarrow \mathbb{R}$ e $h: \mathbb{N} \rightarrow \mathbb{N}$ tais que

$$f(n) = \mathcal{O}(1) + f(h(n)), \text{ para todo } n \geq n_0.$$

Prove que

$$f(n) = \mathcal{O}(1)u(n),$$

onde

$$u(n) := \min \{k \in \mathbb{N} \mid h^k(n) < n_0\}.$$

16. Prove que se $f(n) = \mathcal{O}(g(n))$ e $g(n) = \mathcal{O}(h(n))$, então $f(n) = \mathcal{O}(h(n))$.

17. Prove que se $g(n) = \mathcal{O}(1)$ e $f: \mathbb{N} \rightarrow \mathbb{R}$ é não-decrescente, então $f \circ g(n) = \mathcal{O}(1)$, isto é, que

$$f(\mathcal{O}(1)) = \mathcal{O}(1).$$

18. Dê um exemplo de quatro funções $F, f, G, g: \mathbb{N} \rightarrow \mathbb{R}$ tais que

$$\begin{aligned} f(n) &= \mathcal{O}(F(n)), \\ g(n) &= \mathcal{O}(G(n)), \\ F(n) &= \mathcal{O}(G(n)), \text{ e} \\ f(n) &\text{ não é } \mathcal{O}(g(n)). \end{aligned}$$

19. Prove que

- (a) $\mathcal{O}(1) = \mathcal{O}((\log n)^\alpha)$, se e somente se $\alpha > 0$.
- (b) $\mathcal{O}((\log n)^\alpha) = \mathcal{O}(n^\beta)$, para todo $\alpha \in \mathbb{R}$ e todo $\beta > 0$.
- (c) $\mathcal{O}(n^\alpha) = \mathcal{O}(n^\beta)$, se e somente se $\alpha \leq \beta$.
- (d) $\mathcal{O}(n^\alpha) = \mathcal{O}(\beta^n)$, para todo $\alpha \in \mathbb{R}$ e todo $\beta > 1$.
- (e) $\mathcal{O}(\alpha^n) = \mathcal{O}(\beta^n)$, se e somente se $\alpha \leq \beta$.
- (f) $\mathcal{O}(\alpha^n) = \mathcal{O}(n!)$, para todo $\alpha \in \mathbb{R}$.
- (g) $\mathcal{O}(n!) = \mathcal{O}(n^n)$.

20. Prove que

$$(a) \quad n = \Omega(\lfloor \log n \rfloor)$$

21. Prove que se $f: \mathbb{N} \rightarrow \mathbb{R}$ tem limite, então $f(n) = \Omega(1)$ se e somente se $\lim f(n) \neq 0$.

22. Sejam $n_0 \in \mathbb{N}$, $f: \mathbb{N} \rightarrow \mathbb{R}$ e $h: \mathbb{N} \rightarrow \mathbb{N}$ tais que

$$f(n) = f(h(n)) + \Omega(1), \text{ para todo } n \geq n_0.$$

Prove que

$$f(n) = \Omega(1)u(n),$$

onde

$$u(n) := \min \{k \in \mathbb{N} \mid h^k(n) < n_0\}.$$

23. Seja

$$h(n) := \left\lfloor \frac{n-1}{2} \right\rfloor,$$

e seja

$$u(n) = \min \{k \in \mathbb{N} \mid h^k(n) \leq 0\}.$$

Prove que

$$u(n) = \Omega(1) \lg n.$$

24. Sejam $f(n) = \Omega(1)$ e $g(n) = \Omega(1)$. Prove que $f(n)g(n) = \Omega(1)$, isto é, prove que

$$\Omega(1)\Omega(1) = \Omega(1).$$

25. Prove que se

$$\begin{aligned} g(n) &= \Omega(f(n)), \text{ e} \\ f(n) = 0 &\implies g(n) = 0, \text{ para todo } n \in \mathbb{N}, \end{aligned}$$

então

$$g(n) = \Omega(1)f(n).$$

26. Prove que se $g(n) = \Omega(1)f(n)$ então $g(n) = \Omega(f(n))$ e, além disso, $f(n) = 0 \implies g(n) = 0$, para todo $n \in \mathbb{N}$.

27. Prove que

$$\log_b n = \Omega(\log n), \text{ para todo } b > 1.$$

28. Prove que $f(n) = \Omega(g(n))$ se e somente se $g(n) = \mathcal{O}(f(n))$.

29. Prove que se $g(n) = \Omega(1)$ e $f: \mathbb{N} \rightarrow \mathbb{R}$ é assintoticamente positiva e não-decrescente, então $g \circ f(n) = \Omega(1)$, isto é, que

$$f(\Omega(1)) = \Omega(1).$$

30. Prove que se $\lim f(n) = 0$ então

$$\Omega(1) + f(n) = \Omega(1).$$

31. Dê exemplos de funções $f(n) = \Omega(1)$ e $g(n) = \Omega(1)$, tais que

(a) $f(n)\frac{1}{n} = \Omega(1)$.

(b) $g(n)\frac{1}{n}$ não é $\Omega(1)$.

32. Dê exemplos de funções $f(n) = \Omega(1)$ e $g(n) = \Omega(1)$, tais que

(a) $f(n) + g(n) = \Omega(1)$.

(b) $f(n) + g(n)$ não é $\Omega(1)$.

33. Prove que se $f(n) = \Omega(1)$ e $\lim g(n) = 0$, então

$$f(n) + g(n) = \Omega(1).$$

34. Prove que $g(n) = \Theta(f(n))$ se e somente se existem $c^-, c^+ > 0$ e $n_c \in \mathbb{N}$ tais que

$$c^- |f(n)| \leq |g(n)| \leq c^+ f(n), \text{ para todo } n \geq n_c.$$

35. Prove que se $f(n) = \mathcal{O}(1)$ e $g(n) = o(1)$, então $f(n)g(n) = o(1)$, isto é, que

$$\mathcal{O}(1)o(1) = o(1).$$

36. Prove que se $f(n) = \mathcal{O}(1)$ e $g(n) = \Omega(1)$, então $f(n)/g(n) = \mathcal{O}(1)$, isto é,

$$\frac{\mathcal{O}(1)}{\Omega(1)} = \mathcal{O}(1).$$

37. Use a aproximação de Stirling para provar que

$$\binom{2n}{n} = (1 + o(1)) \frac{2^{2n}}{\sqrt{\pi n}} = \Theta\left(\frac{2^{2n}}{\sqrt{n}}\right)$$

38. Prove que

$$f(n) = o(1) \text{ se e somente se } \lim f(n) = 0.$$

39. Prove que

$$n^{o(1)-\beta} = o(1), \text{ para todo } \beta > 0.$$

40. Prove que

$$\beta^{o(n)-n} = o(1), \text{ para todo } \beta > 1.$$

3 Divisão e Conquista

41. Prove que

(a)
$$\lfloor f(n) \rfloor = f(n) + \mathcal{O}(1), \text{ para todo } f: \mathbb{N} \rightarrow \mathbb{R}.$$

(b) se $b > 1$ e $n_0 \in \mathbb{N}$, então

$$\left\lfloor \log_b \frac{n}{n_0} \right\rfloor + 1 = \log_b n + \mathcal{O}(1).$$

42. Use o “Teorema Mestre” para obter soluções para as seguintes recorrências.

(a) $T(n) = 2T(n/4) + 1,$

(b) $T(n) = 2T(n/4) + \sqrt{n},$

(c) $T(n) = 2T(n/4) + n,$

(d) $T(n) = 2T(n/4) + n^2,$

(e) $T(n) = T(7n/10) + n,$

(f) $T(n) = 16T(n/4) + n^2,$

(g) $T(n) = 7T(n/3) + n^2,$

(h) $T(n) = 7T(n/2) + n^2.$

43. Use o “Teorema Mestre” para provar que o tempo de execução de $(B(x,v,a,b))$ (onde B é o algoritmo do Exercício 3) é $\Theta(\log n)$.

44. Considere o seguinte algoritmo.

Minimo(v, a, b)

Entrada: um vetor v indexado por $[a..b]$, com $a \leq b$

Saída : um índice $m \in [a..b]$ tal que $v[m] \leq v[i]$ para todo $i \in [a..b]$.

Se $a = b$

 Devolva a

$m \leftarrow \lfloor \frac{a+b}{2} \rfloor$

$m_1 \leftarrow \text{Minimo}(v, a, m)$

$m_2 \leftarrow \text{Minimo}(v, m+1, b)$

Se $v[m_1] \leq v[m_2]$

 Devolva m_1

 Devolva m_2

- (a) Prove que o algoritmo está correto.
- (b) Use o “Teorema Mestre” para obter uma expressão assintótica para o tempo de execução de $\text{Minimo}(v, a, b)$ em função de $n = b - a + 1$.
- (c) Explique por que não é possível existir algoritmo assintoticamente mais eficiente que este (em análise de pior caso) para o problema de determinar o mínimo de um vetor.

45. Considere o seguinte algoritmo

$\text{Multiplica}(x, n)$
Entrada: uma “coisa somável” ^a x e um inteiro n Saída : O valor de $n \times x$ Se $n = 0$ Devolva 0 Se n é par Devolva $\text{Multiplica}(x + x, \frac{n}{2})$ Devolva $\text{Multiplica}(x + x, \frac{n-1}{2}) + x$

^a x pode ser qualquer coisa para a qual exista uma operação de soma definida, como por exemplo, um número, uma matriz, uma função etc.

- (a) Prove que o algoritmo está correto.
 - (b) Use o “Teorema Mestre” para obter uma expressão assintótica para o tempo de execução de $\text{Multiplica}(x, n)$ em função de n nos casos em que $x + x$ pode ser computado em tempo $\mathcal{O}(1)$
46. Um estudante diz que é possível obter um algoritmo melhor do que o Algoritmo de Karatsuba, dividindo os inteiros em três partes em vez de somente duas, da seguinte maneira.

Dada uma sequência $x = (x_{n-1}, \dots, x_0) \in \{0, 1\}^+$, sejam x_L , x_C e x_R as partes “esquerda”, “central” e “direita” de x , isto é,

$$\begin{aligned}
 x_L &= (x_{n-1}, \dots, x_{2n/3}), \\
 x_C &= (x_{2n/3-1}, \dots, x_{n/3}), \\
 x_R &= (x_{n/3-1}, \dots, x_0).
 \end{aligned}$$

Então

$$\begin{aligned}\overline{xy} &= x_L \times y_L \times 2^{4n/3} \\ &+ (x_L \times y_C + x_C \times x_C \times y_L) \times 2^n \\ &+ (x_L \times y_R + x_R \times y_L + x_C \times y_C) \times 2^{2n/3} \\ &+ (x_C \times y_L + x_R \times y_C) \times 2^{n/3} \\ &+ x_R \times y_R,\end{aligned}$$

e fazendo

$$\begin{aligned}r_1 &= x_L \times y_L, \\ r_2 &= (x_L + x_C) \times (y_L + y_C), \\ r_3 &= x_C \times y_C, \\ r_4 &= (x_L + x_R) \times (y_L + y_R), \\ r_5 &= x_R \times y_R, \\ r_6 &= (x_C + x_R) \times (y_C + y_R),\end{aligned}$$

temos

$$x \times y = r_1 \times 2^{4n/3} + (r_2 - r_1 - r_3) \times 2^n + (r_3 + r_4 - r_1 - r_5) \times 2^{2n/3} + (r_6 - r_3 - r_5) \times 2^{n/3} + r_5.$$

- (a) Escreva o algoritmo correspondente às observações acima, para o caso em que x e y tem tamanho n onde n é potência de 3.
 - (b) O algoritmo do estudante é assintoticamente mais rápido que o Algoritmo de Karatsuba? Justifique.
 - (c) O algoritmo do estudante está correto? Justifique.
47. Uma α -pseudo-mediana de um conjunto de n valores é um elemento m do conjunto que é menor ou igual a pelo menos n^α elementos do conjunto e maior ou igual a pelo menos n^α elementos do conjunto. Uma pseudo-mediana de um conjunto de n valores é uma α -pseudo-mediana desse conjunto para algum $0 \leq \alpha \leq 1$.

O seguinte algoritmo calcula uma pseudo-mediana dos elementos de um vetor.

$P(v, a, b)$

Entrada: Vetor v indexado por $a..b$

Saída : Uma pseudo-mediana de $\{v[i] : i \in [a..b]\}$

$n \leftarrow b - a + 1$

Se $n < 3$

 Devolva $v[a]$

Se $n \leq 3$

 Devolva a mediana de $\{v[a], v[a + 1], v[a + 2]\}$

$u \leftarrow$ vetor indexado por $[1.. \lceil n/3 \rceil]$

$j \leftarrow 1$

Para i de a até $b - 3$

$u[j] \leftarrow P(v, i, i + 2)$

$j \leftarrow j + 1$

$i \leftarrow i + 3$

Se $i \leq b$

$u[j] \leftarrow P(v, i, b)$

Devolva $P(u, 1, \lceil n/3 \rceil)$

- (a) Use o “Teorema Mestre” para expressar o tempo de execução de $P(v, a, b)$ em termos assintóticos em função do número $n = b - a + 1$ de elementos do vetor.
- (b) É suficiente para que o algoritmo QuickSort execute em tempo $\mathcal{O}(n \log n)$ (onde n é o número de elementos do vetor) que o pivô escolhido a cada iteração seja uma pseudo-mediana do vetor sendo ordenado. É possível computar uma pseudo-mediana a cada iteração e ainda assim ordenar o vetor em tempo $\mathcal{O}(n \log n)$? Justifique⁴

48. O seguinte algoritmo resolve o conhecido quebra-cabeça das **Torres de Hanói**. A execução de $Hanoi(n, a, b, c)$ move n discos da torre a para a torre b usando a torre c como torre auxiliar, de acordo com as regras do jogo.

$Hanoi(n, a, b, c)$

Se $n = 0$

 Termine

$Hanoi(n - 1, a, c, b)$

 mova o disco no topo da torre a para o topo da torre b

$Hanoi(n - 1, c, b, a)$

⁴Sugestão: Use o “Teorema Mestre” de novo.

Seja $M(n)$ o número de movimentos (passagem de um disco de uma torre para outra) na execução de $\text{Hanoi}(n, a, b, c)$.

- (a) Descreva $M(n)$ por meio de uma recorrência.
- (b) Resolva esta recorrência.
- (c) É possível usar o “Teorema Mestre” para obter uma expressão assintótica para o tempo de execução do algoritmo?
- (d) Prove⁵ que não é possível resolver uma instância (n, a, b, c) do problema das Torres de Hanói com menos do que $2^n - 1$ movimentos.
- (e) O algoritmo acima é uma solução ótima para o problema das Torres de Hanói do ponto de vista do tempo de execução assintótico?

49. Considere o seguinte problema computacional.

Subvetor de Soma Máxima (SSM)

Instância: Uma tripla (v, a, b) onde v é um vetor indexado por $[a..b]$.

Resposta: Um par (p, q) de índices de v tal que a soma

$$\sum_{i=p}^q v[i]$$

é máxima.

Brutus, um programador, sugere o seguinte algoritmo: para cada par (p, q) de índices em $[a..b]$ com $p \leq q$, compute a soma dos valores de $v[p..q]$ e devolva um par cuja soma é máxima.

Outro programador, Júlio, sugere o seguinte algoritmo: fazendo $m = \lfloor \frac{a+b}{2} \rfloor$,

- (a) recursivamente compute uma resposta (p_1, q_1) da instância (v, a, m) ,
- (b) recursivamente compute uma resposta (p_2, q_2) da instância $(v, m+1, b)$,
- (c) compute o par (p', q') onde $p' \leq m \leq q'$ são tais que a soma $\sum_{i=p'}^{q'} v[i]$ é máxima.
- (d) devolva o par (p, q) com soma máxima dentre estes três.

⁵**Sugestão:** Indução no número de discos.

- (a) Expresse o tempo de execução do algoritmo de Brutus em termos assintóticos, em função do número de elementos do vetor⁶.
- (b) Confiando na afirmação de Júlio de que o passo 49c do algoritmo pode ser executado em tempo (de pior caso) $\Theta(n)$ onde n é o número de elementos do vetor, expresse o tempo de execução de seu algoritmo em termos assintóticos⁷, em função de n .
- (c) Compare o tempo de execução dos algoritmos obtidos nos itens anteriores e diga se são assintoticamente equivalentes ou qual é mais eficiente segundo esta análise.
- (d) Explique como executar o passo 49c do algoritmo de Júlio em tempo de pior caso $\Theta(n)$.

50. Considere o Algoritmo $\text{Exp}(x, n)$ dado por

$\text{Exp}(x, n)$

Entrada: uma “coisa multiplicável”^a x e um inteiro n

Saída : O valor de x^n

Se $n = 0$

Devolva o elemento neutro da multiplicação

$e \leftarrow \text{Exp}(x, \lfloor n/2 \rfloor)$

$e \leftarrow e \times e$

Se n é par

Devolva e

Devolva $x \times e$

^a x pode ser qualquer coisa para a qual exista uma operação de multiplicação definida, como por exemplo, um número, uma matriz, uma função etc.

- (a) Prove que o algoritmo está correto.
- (b) Dê uma expressão não recorrente para o número de multiplicações na execução de $\text{Exp}(x, n)$ em função de n .
- (c) Assumindo que cada multiplicação na execução de $\text{Exp}(x, n)$ é efetuada em tempo $\mathcal{O}(1)$, expresse o tempo de execução de $\text{Exp}(x, n)$ em termos assintóticos em função de n .

⁶**Sugestão:** O tempo de execução será $\Theta(S(n))$ onde $S(n)$ é o número de somas efetuadas e n é o número de elementos do vetor

⁷**Sugestão:** Use o “Teorema Mestre”

4 Algoritmos Gulosos

51. Seja (Σ, f) uma instância do problema de Codificação de Huffman e sejam σ_1 e σ_2 duas letras frequência mínima nessa instância. Seja $\Sigma' = \Sigma - \{\sigma_1, \sigma_2\} \cup \tau$, onde $\tau \notin \Sigma$ e seja $f': \Sigma' \rightarrow \mathbb{Q}$ dada por

$$f'(\sigma) = \begin{cases} f(\sigma), & \text{se } \sigma \neq \tau, \\ f(\sigma_1) + f(\sigma_2), & \text{se } \sigma = \tau. \end{cases}$$

Prove que as respostas c e c' das instâncias (Σ, f) e (Σ', f') tem mesmo custo, isto é,

$$\sum_{\sigma \in \Sigma} |c(\sigma)|f(\sigma) = \sum_{\sigma \in \Sigma'} |c'(\sigma)|f'(\sigma).$$

52. Use os resultados provados em aula para provar que Algoritmo Codificação de Huffman⁸ é uma solução para o problema de Codificação de Huffman⁸.

5 Programação Dinâmica

53. Considere o seguinte algoritmo para computar o n -ésimo termo da Sequência de Fibonacci.

Fib(n)

Se $n \leq 1$
 Devolva n
 Devolva $Fib_R(n - 2) + Fib_R(n - 1)$

Numa execução de $Fib_R(n)$, quantas vezes se executa $Fib_R(k)$, se $k \leq n$?

54. Prove que o seguinte algoritmo para computar o n -ésimo termo da Sequência de Fibonacci está correto.

Fib(n)

$v \leftarrow$ vetor indexado por $[0..1]$
 $v[0] \leftarrow 0$
 $v[1] \leftarrow 1$
Para k de 2 até n
 $v[k \bmod 2] \leftarrow v[0] + v[1]$
 Devolva $v[n \bmod 2]$

⁸Sugestão: Indução no tamanho do alfabeto da instância.

55. Considere o seguinte algoritmo para computar o n -ésimo termo da Sequência de Fibonacci.

Fib(n)
$M \leftarrow \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$ $M \leftarrow \text{Exp}(M, n)$ Devolva $M[1, 1]$

onde **Exp** é o algoritmo do Exercício 50.

- Prove que o algoritmo está correto.
 - Dê uma expressão não recorrente para o número de operações aritméticas efetuadas (somadas e multiplicações de inteiros) na execução de $\text{Fib}(n)$ em função de n .
 - Expresse o tempo de execução de $\text{Fib}(n)$ em termos assintóticos em função de n .
56. Considere o seguinte algoritmo para o cálculo do coeficiente binomial $\binom{n}{k}$.

B(n, k)
Devolva $\frac{\text{Fatorial}(n)}{\text{Fatorial}(k) \times \text{Fatorial}(n-k)}$

- Sabendo que $\text{Fatorial}(n)$ executa $n - 1$ multiplicações para todo $n \in \mathbb{N}$, dê uma expressão para o número de operações aritméticas (multiplicações e divisões) na execução de $\text{B}(n, k)$.
 - Dê uma expressão para o tempo de execução de $\text{B}(n, k)$ em função de n .
57. Proponha um algoritmo de programação dinâmica para o cálculo do coeficiente binomial $\binom{n}{k}$ baseado na relação de Stifel

$$\binom{n}{k} = \begin{cases} 1, & \text{se } k = 0, \\ \binom{n-1}{k} + \binom{n-1}{k-1}, & \text{se } 1 \leq k \leq n, \\ 0, & \text{caso contrário.} \end{cases}$$

- Quantas operações aritméticas entre inteiros faz seu algoritmo?
- Qual seu tempo de execução em termos assintóticos em função de n e k . E somente em função de n ?

- (c) Qual o espaço consumido em termos assintóticos em função de n e k . E somente em função de n ?
- (d) Compare-o com o algoritmo do Exercício 56.
- (e) No algoritmo do Exercício 56 os valores intermediários calculados podem ser muito grandes, mesmo quando o resultado final não seja. Isso torna possível, por exemplo, que a execução do algoritmo incorra em erro de “overflow” mesmo num caso em que o resultado final não ultrapasse a capacidade de armazenamento de um inteiro.
 - i. Prove que seu algoritmo não sofre deste problema.
 - ii. Prove que o algoritmo do Exercício 56 pode computar valores intermediários arbitrariamente maiores que os do resultado final, provando que para todo $N \in \mathbb{N}$ existem $n, k \in \mathbb{N}$ tais que

$$n! - \binom{n}{k} \geq N.$$

- 58. Prove que um algoritmo para o problema da **Subsequência Crescente Máxima** que examina cada uma das subsequências possíveis e escolhe a maior consome tempo $\Omega(2^{|X|})$ para computar uma instância $|X|$.
- 59. Prove que o tempo de execução de⁹ $\text{DistEdit}_R(x, y)$ é $\Theta((1 + \sqrt{2})^n)$ onde $n = |x| + |y|$.
- 60. Seja A um conjunto. Uma função $d: A \times A \rightarrow \mathbb{R}$ é uma *distância* (ou *métrica*) sobre o conjunto A se tem as seguintes propriedades.
 - (a) $d(a, b) = 0$ se e somente se $a = b$, para todo $a, b \in A$,
 - (b) $d(a, b) \geq 0$, para todo $a, b \in A$,
 - (c) $d(a, b) = d(b, a)$, para todo $a, b \in A$, e
 - (d) $d(a, c) \leq d(a, b) + d(b, c)$ para todo $a, b, c \in A$.

Seja Σ um alfabeto.

- (a) Prove que a Distância de Hamming é uma distância sobre Σ^n , para todo $n \in \mathbb{N}$.
- (b) Prove que a Distância de Edição é uma distância sobre Σ^* .

⁹ DistEdit_R é o algoritmo recursivo para o problema da Distância de Edição discutido em aula.

61. Execute o algoritmo $\text{DistEdit}_D(x, y)$ (discutido em aula) para $x = \text{cara}$ e $y = \text{caldo}$ preenchendo a matriz abaixo.

		0	1	2	3	4	5
		-	c	a	l	d	o
0	-						
1	c						
2	a						
3	r						
4	a						

62. Considere as seguintes soluções para o Problema do Caixeiro Viajante tal como discutido em aula.

- A:** O algoritmo que computa o custo de cada uma das $n!$ permutações sobre $[1..n]$, e devolve uma que apresenta custo mínimo.
- B:** O algoritmo que, a partir da observação de que o custo de permutações circulares (como por exemplo $(p_0, \dots, p_{n-1})$ e $(p_1, \dots, p_0)$) de $[1..n]$ tem mesmo custo, computa o custo de cada uma das $(n - 1)!$ permutações circulares sobre $[1..n]$, e devolve uma que apresenta custo mínimo.

Sejam $T_{\mathbf{A}}(n)$ e $T_{\mathbf{B}}(n)$ os tempos de execução de cada um destes algoritmos expressos em função de n .

- (a) Expresse $T_{\mathbf{A}}(n)$ em termos assintóticos.
- (b) Expresse $T_{\mathbf{B}}(n)$ em termos assintóticos.
- (c) $T_{\mathbf{A}}(n)$ e $T_{\mathbf{B}}(n)$ são assintoticamente equivalentes ou um deles é assintoticamente menor que o outro? Justifique.
63. Prove que a execução de $\text{CustoMenorCaminho}_R(D)$ ¹⁰ toma tempo $\Theta((n-1)!)$ e espaço $\Theta(n^2)$ onde $n = \dim(D)$.

64. Prove que¹¹

$$\sum_{k=2}^n \binom{n}{k} k^2 = n^2 2^{n-2} + n^2 2^{n-2} + n \text{ para todo } n \in \mathbb{N},$$

e que

$$\sum_{k=2}^n \binom{n}{k} k^2 = \Theta(n^2 2^n).$$

¹⁰O algoritmo discutido em aula.

¹¹**Sugestão:** Use indução em n .

A Solução de Recorrências

Este apêndice traz um apanhado de resultados a respeito de solução de recorrências.

Teorema 1. *Sejam $n_0 \in \mathbb{N}$, $h: \mathbb{N} \rightarrow \mathbb{N}$ e $f, m, s: \mathbb{N} \rightarrow \mathbb{C}$ tais que*

$$f(n) = m(n)f(h(n)) + s(n), \text{ para todo } n \geq n_0.$$

Então

$$f(n) = f(h^u(n)) \prod_{i=0}^{u-1} m(h^i(n)) + \sum_{i=0}^{u-1} s(h^i(n)) \prod_{j=0}^{i-1} m(h^j(n)), \text{ para todo } n \geq n_0,$$

onde

$$u = \min \{k \in \mathbb{N} \mid h^k(n) < n_0\}.$$

Teorema 2. *Se a função $f: \mathbb{N} \rightarrow \mathbb{C}$ satisfaz uma recorrência linear homogênea cujo polinômio característico é*

$$(X - r_1)^{m_1}(X - r_2)^{m_2} \dots (X - r_k)^{m_k}$$

então

$$\begin{aligned} f(n) &= c_{1,1}n^0r_1^n + \dots + c_{1,m_1}n^{m_1-1}r_1^n \\ &+ c_{2,1}n^0r_2^n + \dots + c_{2,m_2}n^{m_2-1}r_2^n \\ &+ \dots \\ &+ c_{k,1}n^0r_k^n + \dots + c_{k,m_k}n^{m_k-1}r_k^n \end{aligned}$$

onde $c_{1,1}, \dots, c_{1,m_1}, c_{2,1}, \dots, c_{2,m_2}, \dots, c_{k,1}, \dots, c_{k,m_k}$ são a solução de um sistema linear dado por

$$\begin{aligned} f(a) &= c_{1,1}a^0r_1^a + \dots + c_{1,m_1}a^{m_1-1}r_1^a \\ &+ c_{2,1}a^0r_2^a + \dots + c_{2,m_2}a^{m_2-1}r_2^a \\ &+ \dots \\ &+ c_{k,1}a^0r_k^a + \dots + c_{k,m_k}a^{m_k-1}r_k^a, \end{aligned}$$

para k valores distintos de a .

Teorema 3. *Se $f: \mathbb{N} \rightarrow \mathbb{C}$ satisfaz a recorrência linear homogênea*

$$f(n) = a_1f(n-1) + a_2f(n-2) + \dots + a_kf(n-k), \text{ para todo } n \geq k,$$

então $f(n)$ satisfaz uma recorrência linear homogênea cujo polinômio característico é

$$X^k - a_1X^{k-1} - \dots - a_{k-1}X^1 - a_k.$$

Teorema 4. Se $f: \mathbb{N} \rightarrow \mathbb{C}$ satisfaz a recorrência linear não homogênea

$$f(n) = a_1 f(n-1) + a_2 f(n-2) + \dots + a_k f(n-k) + g(n), \text{ para todo } n \geq k,$$

e $g(n)$ satisfaz uma recorrência linear homogênea cujo polinômio característico é G , então $f(n)$ satisfaz uma recorrência linear homogênea cujo polinômio característico é

$$(X^k - a_1 X^{k-1} - \dots - a_{k-1} X - a_k)G.$$

Teorema 5. Dados $k \in \mathbb{N}$ e $c, r \in \mathbb{C}$, a função

$$g(n) = cn^k r^n,$$

satisfaz uma recorrência linear homogênea cujo polinômio característico é

$$(X - r)^{k+1}.$$

B O “Teorema Mestre”

Teorema (“Teorema Mestre”). Sejam $a \geq 1$, $b > 1$, $T, f: \mathbb{N} \rightarrow \mathbb{R}$ tais que

$$T(n) = aT(n/b) + f(n),$$

então

$$T(n) = \begin{cases} \Theta(n^{\log_b a}), & \text{se } f(n) = \mathcal{O}(n^{\log_b a - \epsilon}), \text{ para algum } \epsilon > 0, \\ \Theta(n^{\log_b a} \log n), & \text{se } f(n) = \Theta(n^{\log_b a}), \\ \Theta(f(n)), & \text{se } f(n) = \Omega(n^{\log_b a + \epsilon}), \text{ para algum } \epsilon > 0 \text{ e} \\ & af(n/b) < cf(n) \text{ assintoticamente para algum } c < 1. \end{cases}$$