

- tuindo um selo de três centavos por um selo de quatro centavos ou substituindo dois selos de quatro centavos por três selos de três centavos.
33. Mostre que podemos demonstrar que $P(n, k)$ é verdadeira para todos os pares de números inteiros positivos n e k , se mostrarmos que
- $P(1, 1)$ é verdadeira e $P(n, k) \rightarrow [P(n + 1, k) \wedge P(n, k + 1)]$ é verdadeira para todos os números inteiros positivos n e k .
 - $P(1, k)$ é verdadeira para todos os números inteiros positivos k , e $P(n, k) \rightarrow P(n + 1, k)$ é verdadeira para todos os números inteiros positivos n e k .
 - $P(n, 1)$ é verdadeira para todos os números inteiros positivos n , e $P(n, k) \rightarrow P(n, k + 1)$ é verdadeira para todos os números inteiros positivos n e k .
34. Demonstre que $\sum_{j=1}^n j(j + 1)(j + 2) \cdots (j + k - 1) = n(n + 1)(n + 2) \cdots (n + k)/(k + 1)$ para todos os números inteiros positivos k e n . [Dica: Use a técnica utilizada no Exercício 33.]
- *35. Mostre que, se a_1, a_2, \dots, a_n forem n números reais distintos, exatamente $n - 1$ multiplicações são utilizadas para computar os produtos desses n números, não importando quantos parênteses são inseridos em seus produtos. [Dica: Use a indução completa e considere a última multiplicação.]
- *36. A propriedade de boa ordenação pode ser utilizada para mostrar que há um único máximo divisor comum de dois números inteiros positivos. Considere a e b como números inteiros positivos, e considere S como o conjunto dos números inteiros positivos na forma $as + bt$, em que s e t são números inteiros.
- Mostre que S não é vazio.
 - Use a propriedade da boa ordenação para mostrar que S tem um menor elemento c .
 - Mostre que se d for um divisor comum de a e b , então d é um divisor de c .
 - Mostre que $c \mid a$ e $c \mid b$. [Dica: Primeiro, assuma que $c \nmid a$. Então, $a = qc + r$, em que $0 < r < c$. Mostre que $r \in S$, contradizendo a escolha por c .]
37. Conclua a partir dos itens (c) e (d) que o máximo divisor comum de a e b existe. Termine a demonstração mostrando que este máximo divisor comum é único.
38. Considere a como um número inteiro e d como um número inteiro positivo. Mostre que os inteiros q e r com $a = dq + r$ e $0 \leq r < d$, que foram mostrados como existentes no Exemplo 5, são únicos.
39. Use a indução matemática para mostrar que um tabuleiro de damas retangular com um número par de células e dois quadrados faltando, um branco e um preto, pode ser preenchido por dominós.
- **39. Você pode usar a propriedade da boa ordenação para demonstrar a proposição: “Todo número inteiro positivo pode ser descrito usando não mais que quinze palavras em inglês”? Assuma que as palavras venham de determinado dicionário de língua inglesa. [Dica: Suponha que existam números inteiros positivos que não podem ser descritos usando não mais que quinze palavras em inglês. Pela boa ordenação, o menor número inteiro positivo que não pode ser descrito usando não mais que quinze palavras em inglês deveria então existir.]
40. Use a propriedade da boa ordenação para mostrar que se x e y forem números reais com $x < y$, então existe um número racional r com $x < r < y$. [Dica: Use a propriedade de Arquimedes, dada no Apêndice 1, para encontrar um número inteiro positivo A com $A > 1/(y - x)$. Então, mostre que existe um número racional r com denominador A entre x e y , procurando os números $\lfloor x \rfloor + j/A$, em que j é um número inteiro positivo.]
41. Mostre que a propriedade da boa ordenação pode ser demonstrada quando o princípio da indução matemática for tido como axioma.
42. Mostre que o princípio da indução matemática e a indução completa são equivalentes, ou seja, cada um pode ser mostrado como válido a partir do outro.
43. Mostre que podemos demonstrar a propriedade da boa ordenação quando tomamos o princípio da indução matemática ou o da indução completa como um axioma em vez de tomar a propriedade da boa ordenação como um axioma.

4.3 Definições Recursivas e Indução Estrutural

Introdução

Às vezes é difícil definir um objeto explicitamente. Entretanto, pode ser fácil defini-lo em termos dele próprio. Esse processo é chamado de **recursão**. Por exemplo, a ilustração mostrada na Figura 1 é produzida recursivamente. Primeiro, é dada uma ilustração. Então, é realizado um processo de sobreposições de sucessivas centralizações de fotos menores sobre a ilustração anterior.

Podemos usar a recursão para definir seqüências, funções e conjuntos. Nas discussões anteriores, especificamos os termos de uma seqüência que usa uma fórmula explícita. Por exemplo, a seqüência de potências de 2 é dada por $a_n = 2^n$ para $n = 0, 1, 2, \dots$. Entretanto, esta seqüência pode também ser definida dando-se o primeiro termo da seqüência, ou seja, $a_0 = 1$ e uma regra para encontrar um termo a partir do anterior, ou seja, $a_{n+1} = 2a_n$ para $n = 0, 1, 2, \dots$. Quando definimos uma seqüência *recursivamente*, especificando como os seus termos são

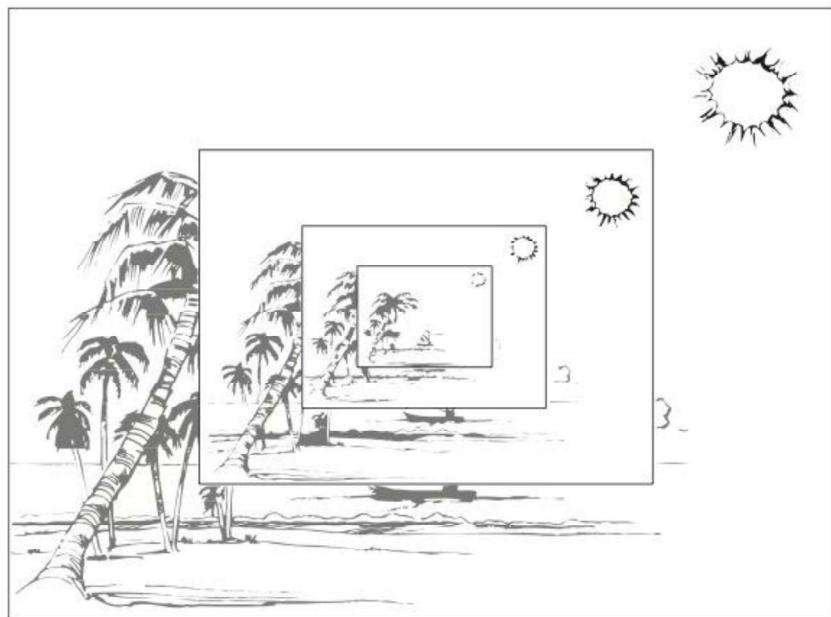


FIGURA 1 Uma Ilustração Definida Recursivamente.

encontrados a partir de termos anteriores, podemos usar a indução para demonstrar os resultados da seqüência.

Podemos definir um conjunto recursivamente, especificando alguns elementos iniciais em um passo base e fornecendo uma regra para a construção de novos elementos a partir daqueles obtidos no passo recursivo. Para demonstrar os resultados de recursividade em conjuntos, usamos um método chamado de *indução estrutural*, ou *recursão*.

Funções Definidas Recursivamente

Usamos duas etapas para definir uma função com o conjunto dos números inteiros não negativos como seu domínio:

PASSO BASE: Especifique o valor da função em zero.



PASSO RECURSIVO: Forneça uma regra para encontrar seu valor em um número inteiro a partir dos valores nos números inteiros menores.

Essa definição é chamada de **recursividade** ou **definição indutiva**.

EXEMPLO 1 Suponha que f seja definida recursivamente por



$$f(0) = 3,$$

$$f(n + 1) = 2f(n) + 3.$$

Encontre $f(1)$, $f(2)$, $f(3)$ e $f(4)$.

Solução: A partir da definição recursiva, temos que

$$\begin{aligned}f(1) &= 2f(0) + 3 = 2 \cdot 3 + 3 = 9, \\f(2) &= 2f(1) + 3 = 2 \cdot 9 + 3 = 21, \\f(3) &= 2f(2) + 3 = 2 \cdot 21 + 3 = 45, \\f(4) &= 2f(3) + 3 = 2 \cdot 45 + 3 = 93.\end{aligned}$$



Muitas funções podem ser estudadas usando suas definições recursivas. A função fatorial é uma delas.

EXEMPLO 2 Dê uma definição recursiva da função fatorial $F(n) = n!$.

Solução: Podemos definir a função fatorial especificando seu valor inicial, ou seja $F(0) = 1$, e dando uma regra para encontrar $F(n+1)$ a partir de $F(n)$. Isso é obtido notando que $(n+1)!$ é computado a partir de $n!$ multiplicado por $n+1$. Assim, a regra desejada é

$$F(n+1) = (n+1)F(n).$$



Para determinar um valor de uma função fatorial, tal como $F(5) = 5!$ a partir da definição recursiva encontrada no Exemplo 2, é necessário usar a regra que mostra como expressar $F(n+1)$ em termos de $F(n)$ muitas vezes:

$$\begin{aligned}F(5) &= 5F(4) = 5 \cdot 4F(3) = 5 \cdot 4 \cdot 3F(2) = 5 \cdot 4 \cdot 3 \cdot 2F(1) \\&= 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 \cdot F(0) = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 \cdot 1 = 120.\end{aligned}$$

Uma vez que $F(0)$ é o único valor da função que aparece, não há necessidade de mais reduções. A única coisa que resta fazer é inserir o valor de $F(0)$ na fórmula.

As funções definidas recursivamente são **bem definidas**, ou seja, para todo número inteiro positivo, o valor da função neste inteiro é determinado de uma forma não ambígua. Isso significa que com qualquer número inteiro positivo, podemos usar as duas partes da definição para encontrar o valor da função naquele inteiro, e também que obtemos o mesmo valor, não importando como aplicamos as duas partes da definição. Isso é uma consequência do princípio da indução matemática. (Veja o Exercício 56 no final desta seção.) Outros exemplos de definições recursivas são dados nos exemplos 3 e 4.

EXEMPLO 3 Dê uma definição recursiva de a^n , em que a é um número real diferente de zero e n é um número inteiro não negativo.

Solução: A definição recursiva contém duas partes. Primeiro a^0 é determinado, ou seja, $a^0 = 1$. Então, é dada a regra pra encontrar a^{n+1} a partir de a^n , ou seja, $a^{n+1} = a \cdot a^n$, para $n = 0, 1, 2, 3, \dots$. Estas duas equações definem unicamente a^n para todos os números inteiros não negativos n .



EXEMPLO 4 Dê uma definição recursiva de

$$\sum_{k=0}^n a_k.$$

Solução: A primeira parte da definição recursiva é

$$\sum_{k=0}^0 a_k = a_0.$$

A segunda parte é

$$\sum_{k=0}^{n+1} a_k = \left(\sum_{k=0}^n a_k \right) + a_{n+1}.$$



Em algumas definições recursivas de funções, os valores da função dos primeiros k números inteiros positivos são especificados e uma regra é dada para determinar o valor da função para números inteiros maiores a partir de seus valores para alguns ou todos os k números inteiros precedentes. Essas definições recursivas são fixadas dessa maneira para produzir funções bem definidas a partir da indução completa (veja o Exercício 57 no final desta seção).

DEFINIÇÃO 1

Os números de Fibonacci, f_0, f_1, f_2, \dots , são definidos pelas equações $f_0 = 0, f_1 = 1$ e

Links

$$f_n = f_{n-1} + f_{n-2}$$

para $n = 2, 3, 4, \dots$

EXEMPLO 5 Encontre os números de Fibonacci f_2, f_3, f_4, f_5 e f_6 .

Solução: Como a primeira parte da definição afirma que $f_0 = 0$ e $f_1 = 1$, temos, a partir da segunda parte da definição, que

$$\begin{aligned} f_2 &= f_1 + f_0 = 1 + 0 = 1, \\ f_3 &= f_2 + f_1 = 1 + 1 = 2, \\ f_4 &= f_3 + f_2 = 2 + 1 = 3, \\ f_5 &= f_4 + f_3 = 3 + 2 = 5, \\ f_6 &= f_5 + f_4 = 5 + 3 = 8. \end{aligned}$$



Podemos usar a definição recursiva dos números de Fibonacci para demonstrar muitas propriedades desses números. Daremos uma dessas propriedades no Exemplo 6.

EXEMPLO 6 Mostre que, sempre que $n \geq 3, f_n > \alpha^{n-2}$, em que $\alpha = (1 + \sqrt{5})/2$.

Exemplos **Extras**

Solução: Podemos usar a indução completa para demonstrar esta inequação. Considere $P(n)$ como a proposição de que $f_n > \alpha^{n-2}$. Queremos mostrar que $P(n)$ é verdadeira sempre que n for um número inteiro maior que ou igual a 3.

PASSO BASE: Primeiro, note que

$$\alpha < 2 = f_3, \quad \alpha^2 = (3 + \sqrt{5})/2 < 3 = f_4,$$

então, $P(3)$ e $P(4)$ são verdadeiras.

PASSO DE INDUÇÃO: Assuma que $P(j)$ seja verdadeira, ou seja, que $f_j \alpha^{j-2}$, para todos os números inteiros j com $3 \leq j \leq k$, em que $k \geq 4$. Devemos mostrar que $P(k+1)$ é verdadeira, ou seja, que $f_{k+1} > \alpha^{k-1}$. Como α é uma solução de $x^2 - x - 1 = 0$ (como a fórmula quadrática mostra), temos que $\alpha^2 = \alpha + 1$. Assim,

$$\alpha^{k-1} = \alpha^2 \cdot \alpha^{k-3} = (\alpha + 1) \alpha^{k-3} = \alpha \cdot \alpha^{k-3} + 1 \cdot \alpha^{k-3} = \alpha^{k-2} + \alpha^{k-3}.$$

Pela hipótese indutiva, se $k \geq 4$, temos que

$$f_{k-1} > \alpha^{k-3}, \quad f_k > \alpha^{k-2}.$$

Assim, temos

$$f_{k+1} = f_k + f_{k-1} > \alpha^{k-2} + \alpha^{k-3} = \alpha^{k-1}.$$

Temos que $P(k+1)$ é verdadeira. Isso completa a demonstração. ◀

Lembre-se: O passo de indução mostra que, sempre que $k \geq 4$, $P(k+1)$ é dado a partir da hipótese de que $P(j)$ seja verdadeira para $3 \leq j \leq k$. Assim, esse passo *não* mostra que $P(3) \rightarrow P(4)$. Além disso, mostramos que $P(4)$ é verdadeira separadamente.

Podemos agora mostrar que o algoritmo de Euclides usa $O(\log b)$ divisões para encontrar o máximo divisor comum dos números inteiros positivos a e b , em que $a \geq b$.

TEOREMA 1

TEOREMA DE LAMÉ Considere a e b como números inteiros positivos com $a \geq b$. Então, o número de divisões utilizado pelo algoritmo de Euclides para encontrar $\text{mdc}(a, b)$ é menor que ou igual a cinco vezes o número de dígitos decimais em b .

Demonstração: Lembre-se de que, quando o algoritmo de Euclides é aplicado para encontrar o $\text{mdc}(a, b)$ com $a \geq b$, esta seqüência de equações (em que $a = r_0$ e $b = r_1$) é obtida.

$$\begin{aligned} r_0 &= r_1 q_1 + r_2 & 0 \leq r_2 < r_1 \\ r_1 &= r_2 q_2 + r_3 & 0 \leq r_3 < r_2 \\ &\vdots \\ r_{n-2} &= r_{n-1} q_{n-1} + r_n & 0 \leq r_n < r_{n-1} \\ r_{n-1} &= r_n q_n. \end{aligned}$$

Aqui, foram utilizadas n divisões para encontrar $r_n = \text{mdc}(a, b)$. Note que os quocientes q_1, q_2, \dots, q_{n-1} são todos pelo menos 1. Além disso, $q_n \geq 2$, porque $r_n < r_{n-1}$. Isso implica que



FIBONACCI (1170–1250) Fibonacci (abreviação de *filius Bonacci*, ou “filho de Bonacci”) foi também conhecido como Leonardo de Pisa. Ele nasceu no centro comercial italiano de Pisa e foi um comerciante que viajava muito para o Oriente Médio, onde fez contato com matemáticos árabes. Em seu livro, *Liber Abaci*, Fibonacci introduziu ao mundo europeu a notação árabe para numerais e algoritmos na aritmética. Foi em seu livro que seu famoso problema dos coelhos (descrito na Seção 7.1) apareceu. Fibonacci também escreveu livros sobre geometria e trigonometria e sobre equações diofantinas, que basicamente procuram soluções inteiras para as equações.

$$\begin{aligned}
 r_n &\geq 1 = f_2, \\
 r_{n-1} &\geq 2r_n \geq 2f_2 = f_3, \\
 r_{n-2} &\geq r_{n-1} + r_n \geq f_3 + f_2 = f_4, \\
 &\vdots \\
 &\vdots \\
 r_2 &\geq r_3 + r_4 \geq f_{n-1} + f_{n-2} = f_n, \\
 b = r_1 &\geq r_2 + r_3 \geq f_n + f_{n-1} = f_{n+1}.
 \end{aligned}$$

Temos que, se n divisões forem usadas pelo algoritmo de Euclides para encontrar $\text{mdc}(a, b)$ com $a \geq b$, então $b \geq f_{n+1}$. A partir do Exemplo 6, sabemos que $f_{n+1} > \alpha^{n-1}$ para $n > 2$, em que $\alpha = (1 + \sqrt{5})/2$. Assim, temos que $b > \alpha^{n-1}$. Além disso, como $\log_{10} \alpha \sim 0,208 > 1/5$, vemos que

$$\log_{10} b > (n - 1) \log_{10} \alpha > (n - 1)/5.$$

Assim, $n - 1 < 5 \cdot \log_{10} b$. Agora suponha que b tenha k dígitos decimais. Então $b < 10^k$ e $\log_{10} b < k$. Temos que $n - 1 < 5k$, e como k é um número inteiro, temos que $n \leq 5k$. Isso finaliza a demonstração. \triangleleft

Como o número de dígitos decimais de b é igual a $\lfloor \log_{10} b \rfloor + 1$, que é menor que ou igual a $\log_{10} b + 1$, o Teorema 1 nos diz que o número de divisões necessárias para encontrar o $\text{mdc}(a, b)$ com $a > b$ é menor que ou igual a $5(\log_{10} b + 1)$. Como $5(\log_{10} b + 1)$ é $O(\log b)$, vemos que $O(\log b)$ divisões são usadas pelo algoritmo de Euclides para encontrar o $\text{mdc}(a, b)$ sempre que $a > b$.

Conjuntos e Estruturas Definidas Recursivamente



Exploramos como as funções podem ser definidas recursivamente. Agora veremos como os conjuntos podem ser definidos desse mesmo modo. Como na definição recursiva de funções, as definições recursivas de conjuntos têm dois passos, um **passo base** e um **passo recursivo**. No passo base, o grupo inicial de elementos é determinado. No passo recursivo, são fornecidas as regras para formar novos elementos no conjunto a partir daqueles que já são conhecidos. As definições recursivas podem incluir também uma **regra de exclusão**, que especifica que um conjunto definido recursivamente contém não mais que aqueles elementos especificados no passo base ou construídos pelas aplicações do passo recursivo. Em nossas discussões, sempre assumiremos tacitamente que a regra de exclusão é válida e nenhum elemento pertence ao conjunto definido recursivamente, a menos que esteja no grupo inicial especificado no passo base ou que possa ser



GABRIEL LAMÉ (1795–1870) Gabriel Lamé entrou para a Escola Politécnica, na França, em 1813, graduando-se em 1817. Ele continuou sua educação na École des Mines, graduando-se em 1820.

Em 1820, Lamé foi à Rússia, onde trabalhou como diretor das Escolas de Estradas e Transporte em São Petersburgo. Não apenas ensinava, como também projetava estradas e pontes na Rússia. Ele retornou a Paris em 1832, onde ajudou a fundar uma empresa de engenharia. Entretanto, logo deixou a empresa, aceitando a cadeira de física na Escola Politécnica, onde permaneceu até 1844. Enquanto estava nesse cargo, ele atuava fora da academia como consultor em engenharia, trabalhando como engenheiro-chefe de minas e participando de construções de estradas.

A contribuição inicial de Lamé se deu em teoria dos números, matemática aplicada e termodinâmica. Seu trabalho mais conhecido envolve a introdução de coordenadas curvas. Em teoria dos números, seu trabalho inclui demonstrar o Último Teorema de Fermat para $n = 7$, assim como fornecer um limite superior para o número de divisões usado pelo algoritmo de Euclides dado neste texto. Na opinião de Gauss, um dos mais importantes matemáticos da época, Lamé foi o matemático francês que mais avançou em seus estudos naquela época. Entretanto, os matemáticos franceses consideravam-no muito prático, enquanto os cientistas franceses consideravam-no muito teórico.



gerado usando-se o passo recursivo uma ou mais vezes. Depois, veremos como podemos usar uma técnica conhecida como indução estrutural para demonstrar os resultados de conjuntos definidos recursivamente.

Os exemplos 7, 8, 10 e 11 ilustram a definição recursiva de conjuntos. Em cada exemplo, mostramos os elementos gerados pelas primeiras aplicações do passo recursivo.

EXEMPLO 7 Considere o subconjunto S do conjunto dos números inteiros definido por

PASSO BASE: $3 \in S$.

PASSO RECURSIVO: Se $x \in S$ e $y \in S$, então $x + y \in S$.

Exemplos Extras

Os novos elementos encontrados em S são 3, pelo passo base, $3 + 3 = 6$, na primeira aplicação do passo recursivo, $3 + 6 = 6 + 3 = 9$ e $6 + 6 = 12$, na segunda aplicação do passo recursivo, e assim por diante. Mostraremos depois que S é o conjunto dos múltiplos positivos de 3. ◀

As definições recursivas desempenham importante papel no estudo de cadeias. (Veja o Capítulo 12 para uma introdução da teoria das linguagens formais, por exemplo.) Lembre-se da Seção 2.4, em que uma cadeia de um alfabeto Σ é uma seqüência finita de símbolos de Σ . Podemos definir Σ^* , o conjunto das cadeias sobre Σ , recursivamente, como mostra a Definição 2.

DEFINIÇÃO 2

O conjunto Σ^* de *cadeias* do alfabeto Σ pode ser definido recursivamente por

PASSO BASE: $\lambda \in \Sigma^*$ (em que λ é a cadeia vazia que não contém símbolos).

PASSO RECURSIVO: se $w \in \Sigma^*$ e $x \in \Sigma$, então $wx \in \Sigma^*$.

O passo base da definição recursiva de cadeias diz que a cadeia vazia pertence a Σ^* . O passo recursivo afirma que novas cadeias são produzidas pela adição de um símbolo a partir de Σ ao final das cadeias em Σ^* . Em cada aplicação do passo recursivo, cadeias com um símbolo adicional são geradas.

EXEMPLO 8 Se $\Sigma = \{0, 1\}$, as cadeias encontradas que estão em Σ^* , o conjunto de todas as cadeias, são λ , especificadas para pertencer a Σ^* no passo base, 0 e 1, formados durante a primeira aplicação do passo recursivo, 00, 01, 10 e 11, formados durante a segunda aplicação do passo recursivo, e assim por diante. ◀

As definições recursivas podem ser usadas para operações ou funções de elementos de conjuntos definidos recursivamente. Isso é ilustrado na Definição 3 da concatenação de duas cadeias e no Exemplo 9 sobre a extensão de uma cadeia.

DEFINIÇÃO 3

Duas cadeias podem ser combinadas por meio de uma operação de *concatenação*. Considere Σ como o conjunto de símbolos e Σ^* como o conjunto de cadeias formadas a partir dos símbolos em Σ . Podemos definir a concatenação de duas cadeias, indicadas por \cdot , recursivamente, como ilustrado a seguir.

PASSO BASE: Se $w \in \Sigma^*$, então $w \cdot \lambda = w$, em que λ é a cadeia vazia.

PASSO RECURSIVO: Se $w_1 \in \Sigma^*$ e $w_2 \in \Sigma^*$ e $x \in \Sigma$, então $w_1 \cdot (w_2 x) = (w_1 \cdot w_2)x$.

A concatenação das cadeias w_1 e w_2 é geralmente escrita como $w_1 w_2$, em vez de $w_1 \cdot w_2$. Repetindo as aplicações do passo recursivo, temos que a concatenação de duas cadeias w_1 e w_2 consiste

em símbolos em w_1 seguidos de símbolos em w_2 . Por exemplo, a concatenação de $w_1 = \text{abra}$ e $w_2 = \text{cadabra}$ é $w_1w_2 = \text{abracadabra}$.

EXEMPLO 9 Comprimento de uma Cadeia Dê uma definição recursiva de $l(w)$, o comprimento, ou a extensão, da cadeia w .

Solução: A extensão de uma cadeia pode ser definida por

$$\begin{aligned} l(\lambda) &= 0; \\ l(wx) &= l(w) + 1 \text{ se } w \in \Sigma^* \text{ e } x \in \Sigma. \end{aligned}$$

Outro uso importante das definições recursivas é definir **fórmulas bem formadas** de vários tipos. Isso é ilustrado nos exemplos 10 e 11.

EXEMPLO 10 Fórmulas Bem Formadas para Formas de Proposições Compostas Podemos definir o conjunto das fórmulas bem formadas para formas de proposições compostas que envolvem **V**, **F**, variáveis proposicionais e operadores do conjunto $\{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$.

PASSO BASE: **V**, **F** e s , em que s é uma variável proposicional, são fórmulas bem formadas.

PASSO RECURSIVO: Se E e F são fórmulas bem formadas, então $(\neg E)$, $(E \wedge F)$, $(E \vee F)$, $(E \rightarrow F)$ e $(E \leftrightarrow F)$ são fórmulas bem formadas.

Por exemplo, pelo passo base, sabemos que **V**, **F**, p e q são fórmulas bem formadas, em que p e q são variáveis proposicionais. A partir de uma aplicação inicial do passo recursivo, sabemos que $(P \vee q)$, $(P \rightarrow F)$, $(F \rightarrow q)$ e $(q \wedge F)$ são fórmulas bem formadas. Uma segunda aplicação do passo recursivo mostra que $((p \vee q) \rightarrow (q \wedge F))$, $(q \vee (p \vee q))$ e $((p \rightarrow F) \rightarrow V)$ são fórmulas bem formadas. Deixamos para o leitor mostrar que $p \neg q$, $pq \wedge c \neg \wedge pq$ não são fórmulas bem formadas, mostrando que nenhuma delas pode ser obtida usando-se o passo base ou qualquer aplicação do passo recursivo.

EXEMPLO 11 Fórmulas Bem Formadas de Operações Podemos definir o conjunto de fórmulas bem formadas composto de variáveis, numerais e operadores do conjunto $\{+, -, *, /, \uparrow\}$ (em que $*$ indica multiplicação e \uparrow indica exponenciação) recursivamente.

PASSO BASE: x é uma fórmula bem formada, se x for um numeral ou variável.

PASSO RECURSIVO: Se F e G forem fórmulas bem formadas, então $(F + G)$, $(F - G)$, $(F * G)$, (F/G) e $(F \uparrow G)$ são fórmulas bem formadas.

Por exemplo, pelo passo base, vemos que x , y , 0 e 3 são fórmulas bem formadas (assim como qualquer variável ou numeral). Fórmulas bem formadas construídas a partir da aplicação do passo recursivo uma vez incluem $(x + 3)$, $(3 + y)$, $(x - y)$, $(3 - 0)$, $(x * 3)$, $(3 * y)$, $(3/0)$, (x/y) , $(3 \uparrow x)$ e $(0 \uparrow 3)$. Aplicando o passo recursivo duas vezes, vemos que as fórmulas como $((x + 3) + 3)$ e $(x - (3 * y))$ são bem formadas. [Note que $(3/0)$ é uma fórmula bem formada porque queremos que apenas a sintaxe seja relevante aqui.] Deixamos para o leitor mostrar que cada uma das fórmulas $x3 +, y * + x e * x/y$ não são fórmulas bem formadas, mostrando que nenhuma delas pode ser obtida usando o passo base ou qualquer aplicação do passo recursivo.

Estudaremos as árvores exaustivamente no Capítulo 10. Uma árvore é um tipo especial de grafo; um grafo é feito de vértices e arestas que conectam alguns pares de vértices. Estudaremos os grafos no Capítulo 9, mas os introduziremos brevemente aqui para ilustrar como podem ser definidos recursivamente.

DEFINIÇÃO 4

O conjunto de *árvores com raiz*, em que uma árvore com raiz consiste em um conjunto de vértices que contém um vértice distinto, chamado de *raiz*, e arestas que conectam esses vértices, pode ser definido recursivamente por estes passos:

PASSO BASE: Um único vértice r é uma árvore com raiz.

PASSO RECURSIVO: Suponha que T_1, T_2, \dots, T_n são árvores com raízes disjuntas que possuem raízes r_1, r_2, \dots, r_n , respectivamente. Então, o grafo formado começando com uma raiz r , que não esteja em nenhuma das árvores com raízes T_1, T_2, \dots, T_n , e adicionando uma aresta a partir de r a cada um dos vértices r_1, r_2, \dots, r_n , é também uma árvore com raiz.

Na Figura 2 ilustramos as árvores com raízes formadas com o passo base e aplicando-se o passo recursivo uma e duas vezes. Note que infinitas árvores com raízes são formadas em cada aplicação da definição recursiva.

Árvores binárias são um tipo especial de árvores. Forneceremos as definições recursivas de dois tipos de árvores binárias — árvores binárias completas e árvores binárias estendidas. No passo recursivo da definição de cada tipo de árvore binária, duas árvores binárias são combinadas para formar uma nova árvore com uma dessas árvores colocada como subárvore à esquerda e a outra colocada como subárvore à direita. Nas árvores binárias estendidas, as subárvore à esquerda ou à direita podem ser vazias, mas, em árvores binárias completas, isso não é possível. Árvores binárias são um dos mais importantes tipos de estruturas em ciência da computação. No Capítulo 10 veremos de que maneira elas podem ser usadas nos algoritmos de busca e de ordenação, em algoritmos para comprimir dados e em muitas outras aplicações. Primeiro, definiremos as árvores binárias estendidas.

DEFINIÇÃO 5

O conjunto de *árvores binárias estendidas* pode ser definido recursivamente pelos passos abaixo:

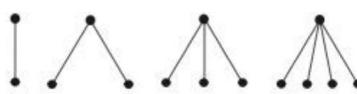
PASSO BASE: O conjunto vazio é uma árvore binária estendida.

PASSO RECURSIVO: Se T_1 e T_2 forem árvores binárias estendidas disjuntas, haverá uma árvore binária estendida, indicada por $T_1 \cdot T_2$, que consiste em uma raiz r junto com arestas que conectam a raiz a cada uma das raízes da subárvore à esquerda T_1 e da subárvore à direita T_2 quando essas árvores não forem vazias.

Passo base



Passo 1



Passo 2

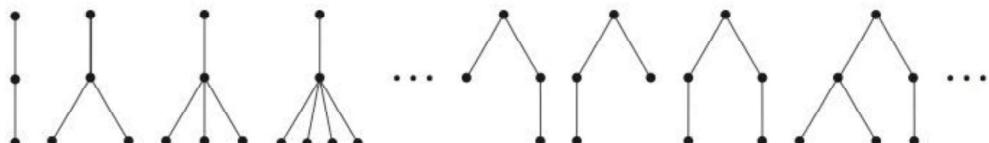


FIGURA 2 Construindo Árvores com Raízes.

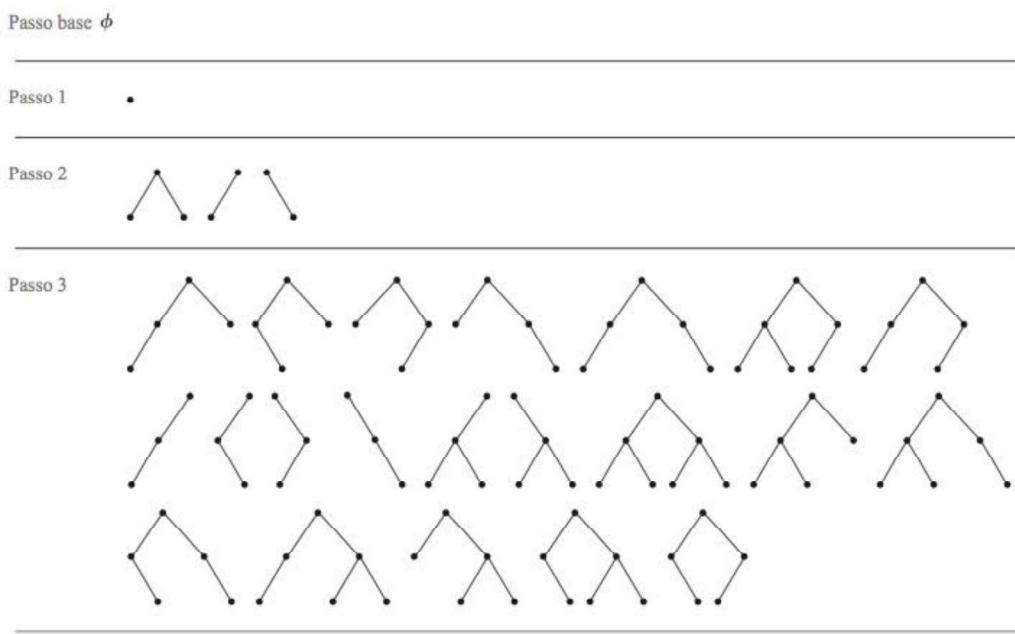


FIGURA 3 Construindo Árvores Binárias Estendidas.

A Figura 3 mostra como árvores binárias estendidas são construídas aplicando-se o passo recursivo de uma a três vezes.

Mostraremos agora como definir o conjunto das árvores binárias completas. Note que a diferença entre esta definição recursiva e a das árvores binárias estendidas fundamenta-se inteiramente no passo base.

DEFINIÇÃO 6

O conjunto das árvores binárias completas pode ser definido recursivamente pelos passos abaixo:

PASSO BASE: Há uma árvore binária completa que contém apenas um único vértice r .

PASSO RECURSIVO: Se T_1 e T_2 forem árvores binárias completas disjuntas, haverá uma árvore binária completa, indicada por $T_1 \cdot T_2$, que é formada por uma raiz r junto com as arestas que ligam a raiz a cada uma das raízes das subárvores à esquerda T_1 e à direita T_2 .

A Figura 4 mostra como as árvores binárias completas são construídas aplicando-se o passo recursivo uma e duas vezes.

Indução Estrutural

Para demonstrar os resultados sobre conjuntos definidos recursivamente, geralmente usamos formas da indução matemática. O Exemplo 12 ilustra a conexão entre conjuntos definidos recursivamente e a indução matemática.

EXEMPLO 12 Mostre que o conjunto S definido no Exemplo 7, ao especificar que $3 \in S$ e que se $x \in S$ e $y \in S$, então $x + y \in S$, é o conjunto de todos os números inteiros positivos que são múltiplos de 3.

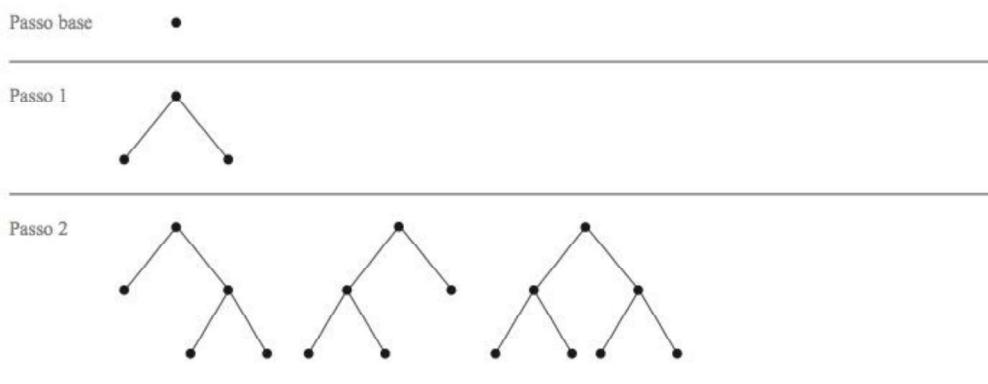


FIGURA 4 Construindo Árvores Binárias Completas.

Solução: Considere A como o conjunto de todos os números inteiros positivos divisíveis por 3. Para demonstrar que $A = S$, devemos mostrar que A é um subconjunto de S e que S é um subconjunto de A . Para demonstrar que A é um subconjunto de S , devemos mostrar que todo número inteiro positivo divisível por 3 está em S . Usaremos a indução matemática para demonstrar isso.

Considere $P(n)$ como a proposição de que $3n$ pertence a S . O passo base é mantido, pois, pela primeira parte da definição recursiva de S , $3 \cdot 1 = 3$ está em S . Para estabelecer o passo de indução, assuma que $P(k)$ seja verdadeira, ou seja, que $3k$ está em S . Como $3k$ está em S e como 3 está em S , temos que, a partir da segunda parte da definição recursiva de S , $3k + 3 = 3(k + 1)$ está também em S .

Para demonstrar que S é um subconjunto de A , usamos a definição recursiva de S . Primeiro, o passo base da definição específica que 3 está em S . Como $3 = 3 \cdot 1$, todos os elementos considerados como estando em S neste passo são divisíveis por 3 e, portanto, estão em A . Para finalizar a demonstração, devemos mostrar que todos os números inteiros em S gerados por meio da segunda parte da definição recursiva estão em A . Isso consiste em mostrar que $x + y$ está em A sempre que x e y forem elementos de S também considerados como estando em A . Agora, se x e y estiverem em A , temos que $3 \mid x$ e $3 \mid y$. Pelo item (i) do Teorema 1 da Seção 3.4, temos que $3 \mid x + y$, completam a demonstração. ◀

No Exemplo 12 usamos a indução matemática sobre o conjunto dos números inteiros positivos e uma definição recursiva para demonstrar um resultado de conjuntos definidos recursivamente. Entretanto, em vez de usar a indução matemática diretamente para demonstrar resultados sobre conjuntos definidos recursivamente, podemos usar uma forma mais conveniente de indução, conhecida como **indução estrutural**. Uma demonstração por indução estrutural é formada por dois passos, que são:

PASSO BASE: Mostre como os resultados se mantêm para todos os elementos especificados no passo base da definição recursiva de pertencerem a um conjunto.

PASSO RECURSIVO: Mostre que, se a proposição for verdadeira para cada um dos elementos usados para formar novos elementos no passo recursivo da definição, o resultado se mantém para novos elementos.

A validade da indução estrutural ocorre a partir do princípio da indução matemática para números inteiros não negativos. Para verificar isso, considere $P(n)$ como a afirmação verdadeira para todos os elementos do conjunto que são gerados por n ou menos aplicações das regras no passo recursivo de uma definição recursiva. Teremos estabelecido que o princípio da indução matemática implica o princípio da indução estrutural se pudermos mostrar que $P(n)$ é verdadeira sempre que n for um número inteiro positivo. No passo base de uma demonstração por indução estrutural, mostramos que $P(0)$ é verdadeira, ou seja, mostramos que o resultado é verdadeiro para todos os elementos especificados como sendo do conjunto no passo base da definição. Uma consequência do passo de indução é que, se

assumimos que $P(k)$ seja verdadeira, temos que $P(k + 1)$ é verdadeira. Quando completamos uma demonstração usando a indução estrutural, mostramos que $P(0)$ é verdadeira e que $P(k)$ implica $P(k + 1)$. Pela indução matemática, temos que $P(n)$ é verdadeira para todos os números inteiros não negativos n . Isso também mostra que o resultado é verdadeira para todos os elementos gerados na definição recursiva e mostra que a indução estrutural é uma técnica válida de demonstração.

EXEMPLOS DE DEMONSTRAÇÕES QUE USAM A INDUÇÃO ESTRUTURAL A indução estrutural pode ser usada para demonstrar que todos os membros de um conjunto construídos recursivamente têm uma propriedade particular. Ilustraremos essa idéia usando a indução estrutural para demonstrar os resultados de fórmulas bem formadas, cadeias e árvores binárias. Para cada demonstração, teremos de tomar cuidado com os passos base e recursivo apropriados. Por exemplo, usar a indução estrutural para demonstrar um resultado sobre o conjunto de fórmulas bem formadas definidas no Exemplo 10, em que especificamos que \mathbf{V} , \mathbf{F} e toda variável proposicional s são fórmulas bem formadas e especificamos que, se E e F forem fórmulas bem formadas, então $(\neg E)$, $(E \wedge F)$, $(E \vee F)$, $(E \rightarrow F)$ e $(E \leftrightarrow F)$ são também fórmulas bem formadas, precisamos completar o passo base e o recursivo abaixo.

PASSO BASE: Mostre que o resultado é verdadeiro para \mathbf{V} , \mathbf{F} e s sempre que s for uma variável proposicional.

PASSO RECURSIVO: Mostre que, se o resultado for verdadeiro para as proposições compostas p e q , também será verdadeiro para $(\neg p)$, $(p \vee q)$, $(p \wedge q)$, $(p \rightarrow q)$ e $(p \leftrightarrow q)$.

O Exemplo 13 ilustra como podemos demonstrar os resultados de fórmulas bem formadas usando a indução estrutural.

EXEMPLO 13 Mostre que todas as fórmulas bem formadas para proposições compostas, como definidas no Exemplo 10, contêm um número igual de parênteses do lado esquerdo e do lado direito.

Solução:

PASSO BASE: Cada uma das formulas \mathbf{V} , \mathbf{F} e s não contém parênteses, assim, elas contêm um número igual de parênteses do lado esquerdo e do lado direito.

PASSO RECURSIVO: Assuma p e q como fórmulas bem formadas, cada uma com um número igual de parênteses à esquerda e à direita, ou seja, se l_p e l_q forem o número de parênteses à esquerda de p e q , respectivamente, e r_p e r_q forem o número de parênteses à direita de p e q , respectivamente, então $l_p = r_p$ e $l_q = r_q$. Para completar o passo de indução, precisamos mostrar que cada um de $(\neg p)$, $(p \vee q)$, $(p \wedge q)$, $(p \rightarrow q)$ e $(p \leftrightarrow q)$ também contém um número igual de parênteses à esquerda e à direita. O número de parênteses à esquerda na primeira dessas proposições compostas é igual a $l_p + 1$ e em cada uma das outras proposições compostas é igual a $l_p + l_q + 1$. Do mesmo modo, o número de parênteses à direita na primeira dessas proposições compostas é igual a $r_p + 1$ e em cada uma das outras proposições é igual a $r_p + r_q + 1$. Como $l_p = r_p$ e $l_q = r_q$, temos que cada uma das expressões compostas contém o mesmo número de parênteses à direita e à esquerda. Isso completa a demonstração indutiva. ◀

Suponha que $P(w)$ é uma função proposicional do conjunto de cadeias $w \in \Sigma^*$. Para usar a indução estrutural a fim de demonstrar que $P(w)$ é mantida para todas as cadeias $w \in \Sigma^*$, precisamos completar o passo base e o passo recursivo. Esses passos são:

PASSO BASE: Mostre que $P(\lambda)$ é verdadeira.

PASSO RECURSIVO: Assuma que $P(w)$ seja verdadeira, em que $w \in \Sigma^*$. Mostre que, se $x \in \Sigma$, então $P(wx)$ deve ser também verdadeira.

O Exemplo 14 ilustra como a indução estrutural pode ser usada em demonstrações de cadeias.

EXEMPLO 14 Use a indução estrutural para demonstrar que $l(xy) = l(x) + l(y)$, em que x e y pertencem a Σ^* , o conjunto de cadeias do alfabeto Σ .

Solução: Fundamentaremos nossa demonstração na definição recursiva do conjunto Σ^* , dado na Definição 2, e na definição de extensão de uma cadeia do Exemplo 9, que especifica que $l(\lambda) = 0$ e $l(wx) = l(w) + 1$ quando $w \in \Sigma^*$ e $x \in \Sigma$. Considere $P(y)$ como a proposição que afirma que $l(xy) = l(x) + l(y)$ sempre que x pertencer a Σ^* .

PASSO BASE: Para completar o passo base, devemos mostrar que $P(\lambda)$ é verdadeira, ou seja, devemos mostrar que $l(x\lambda) = l(x) + l(\lambda)$ para todo $x \in \Sigma^*$. Como $l(x\lambda) = l(x) = l(x) + 0 = l(x) + l(\lambda)$ para toda cadeia x , temos que $P(\lambda)$ é verdadeira.

PASSO RECURSIVO: Para completar o passo de indução, assumimos que $P(y)$ seja verdadeira e mostramos como isso implica que $P(ya)$ seja verdadeira sempre que $a \in \Sigma$. O que nós precisamos fazer é mostrar que $l(xya) = l(x) + l(ya)$ para todo $a \in \Sigma$. Para mostrar isso, note que, pela definição de recursividade de $l(w)$ (dada no Exemplo 9), temos que $l(xya) = l(xy) + 1$ e $l(ya) = l(y) + 1$. E, pela hipótese indutiva, $l(xy) = l(x) + l(y)$. Concluímos que $l(xya) = l(x) + l(y) + 1 = l(x) + l(ya)$. ◀

Podemos demonstrar os resultados sobre árvores ou classes especiais de árvores usando a indução estrutural. Por exemplo, para demonstrar um resultado de árvores binárias completas usando a indução estrutural, precisamos completar o passo base e o passo recursivo a seguir.

PASSO BASE: Mostre que o resultado é verdadeiro para a árvore que tiver um único vértice.

PASSO RECURSIVO: Mostre que se o resultado for verdadeiro para as árvores T_1 e T_2 , então é verdadeiro para a árvore $T_1 \cdot T_2$ que possui uma raiz r , e que tem a subárvore T_1 à sua esquerda e a subárvore T_2 à sua direita.

Antes de fornecermos um exemplo que mostra como a indução estrutural pode ser usada para demonstrar um resultado de árvores binárias completas, precisamos de algumas definições. Definiremos recursivamente a altura $h(T)$ e o número de vértices $n(T)$ de uma árvore binária completa T . Começamos com a definição de altura de uma árvore binária completa.

DEFINIÇÃO 7

Definimos recursivamente a altura $h(T)$ de uma árvore binária completa T .

PASSO BASE: A altura da árvore binária completa T com apenas uma raiz r é $h(T) = 0$.

PASSO RECURSIVO: Se T_1 e T_2 forem árvores binárias completas, então a árvore binária completa $T = T_1 \cdot T_2$ tem altura $h(T) = 1 + \max(h(T_1), h(T_2))$.

Se considerarmos $n(T)$ como o número de vértices em uma árvore binária completa, observamos que $n(T)$ satisfaz à seguinte fórmula recursiva:

PASSO BASE: O número de vértices $n(T)$ da árvore binária completa T que tem apenas uma raiz r é $n(T) = 1$.

PASSO RECURSIVO: Se T_1 e T_2 forem árvores binárias completas, então o número de vértices da árvore binária completa $T = T_1 \cdot T_2$ é $n(T) = 1 + n(T_1) + n(T_2)$.

Mostraremos agora como a indução estrutural pode ser usada para demonstrar um resultado de árvores binárias completas.

TEOREMA 2

Se T for uma árvore binária completa T , então $n(T) \leq 2^{h(T)+1} - 1$.

Demonstração: Demonstramos essa inequação usando a indução estrutural.

PASSO BASE: Para a árvore binária completa que tem apenas a raiz r , o resultado é verdadeiro porque $n(T) = 1$ e $h(T) = 0$, assim, $n(T) = 1 \leq 2^{0+1} - 1 = 1$.

PASSO DE INDUÇÃO: Para a hipótese induativa, assumimos que $n(T_1) \leq 2^{h(T_1)+1} - 1$ e $n(T_2) \leq 2^{h(T_2)+1} - 1$ sempre que T_1 e T_2 forem árvores binárias completas. Pelas fórmulas recursivas para $n(T)$ e $h(T)$, temos que $n(T) = 1 + n(T_1) + n(T_2)$ e $h(T) = 1 + \max(h(T_1), h(T_2))$.

Verificamos que

$$\begin{aligned}
 n(T) &= 1 + n(T_1) + n(T_2) && \text{pela fórmula recursiva para } n(T) \\
 &\leq 1 + (2^{h(T_1)+1}-1) + (2^{h(T_2)+1}-1) && \text{pela hipótese induativa} \\
 &\leq 2 \cdot \max(2^{h(T_1)+1}, 2^{h(T_2)+1}) - 1 && \text{porque a soma de dois termos é no máximo 2} \\
 &&& \text{vezes o maior} \\
 &= 2 \cdot 2^{\max(h(T_1), h(T_2))+1} - 1 && \text{porque } \max(2^x, 2^y) = 2^{\max(x,y)} \\
 &= 2 \cdot 2^{h(T)} - 1 && \text{pela definição recursiva de } h(T) \\
 &= 2^{h(T)+1} - 1.
 \end{aligned}$$

Isso completa o passo de indução. \triangleleft

Indução Generalizada

Podemos estender a indução matemática para demonstrar os resultados de outros conjuntos que têm a propriedade da boa ordenação além do conjunto dos números inteiros. Discutiremos este conceito em detalhes na Seção 8.6, mas daremos alguns exemplos aqui para ilustrar a utilidade dessas aproximações.

Como exemplo, note que podemos definir uma ordem de $\mathbb{N} \times \mathbb{N}$, os pares ordenados dos números inteiros não negativos, especificando que (x_1, y_1) é menor que ou igual a (x_2, y_2) se $x_1 < x_2$ ou $x_1 = x_2$ e $y_1 < y_2$; isto é chamado de **ordem lexicográfica** (ordenação alfabética). O conjunto $\mathbb{N} \times \mathbb{N}$ com essa ordenação tem a propriedade de que todo subconjunto $\mathbb{N} \times \mathbb{N}$ tem pelo menos um elemento (veja o Exercício Complementar 53 da Seção 8.6). Isso implica que podemos definir recursivamente os termos $a_{m,n}$, com $m \in \mathbb{N}$ e $n \in \mathbb{N}$, e demonstrar os resultados deles usando uma variante da indução matemática, como ilustrado no Exemplo 15.

EXEMPLO 15 Suponha que $a_{m,n}$ seja definido recursivamente por $(m, n) \in \mathbb{N} \times \mathbb{N}$ por $a_{0,0} = 0$ e

$$a_{m,n} = \begin{cases} a_{m-1,n} + 1 & \text{se } n = 0 \text{ e } m > 0 \\ a_{m,n-1} + n & \text{se } n > 0. \end{cases}$$

Mostre que $a_{m,n} = m + n(n+1)/2$ para todo $(m, n) \in \mathbb{N} \times \mathbb{N}$, ou seja, para todos os pares de números inteiros não negativos.

Solução: Podemos demonstrar que $a_{m,n} = m + n(n+1)/2$ usando uma versão generalizada da indução matemática. O passo base requer que mostremos que esta fórmula é válida quando $(m, n) = (0, 0)$. O passo de indução requer que mostremos que se a fórmula for mantida para todos os pares menores que (m, n) em ordem lexicográfica de $\mathbb{N} \times \mathbb{N}$, então ela é também mantida para (m, n) .

PASSO BASE: Considere $(m, n) = (0, 0)$. Então, pela passo base da definição recursiva de $a_{m,n}$, temos que $a_{0,0} = 0$. Além disso, quando $m = n = 0$, $m + n(n+1)/2 = 0 + (0 \cdot 1)/2 = 0$. Isso completa o passo base.

PASSO DE INDUÇÃO: Suponha que $a_{m',n'} = m' + n'(n'+1)/2$ sempre que (m', n') for menor que (m, n) na ordem lexicográfica de $\mathbb{N} \times \mathbb{N}$. Pela definição recursiva, se $n = 0$, então $a_{m,n} = a_{m-1,n} + 1$. Como $(m-1, n)$ é menor que (m, n) , a hipótese induativa nos diz que $a_{m-1,n} = m-1 + n(n+1)/2$, assim, $a_{m,n} = m-1 + n(n+1)/2 + 1 = m + n(n+1)/2$, resultando na equação desejada.

Agora suponha que $n > 0$, assim $a_{m,n} = a_{m,n-1} + n$. Como $(m, n - 1)$ é menor que (m, n) , a hipótese indutiva nos diz que $a_{m,n-1} = m + (n - 1)n/2$, assim $a_{m,n} = m + (n - 1)n/2 + n = m + (n^2 - n + 2n)/2 = m + n(n + 1)/2$. Isso finaliza o passo de indução. ◀

Conforme mencionado, justificaremos esta técnica de demonstração na Seção 8.6.

Exercícios

1. Encontre $f(1), f(2), f(3)$ e $f(4)$ se $f(n)$ for definido recursivamente por $f(0) = 1$ e para $n = 0, 1, 2, \dots$
 - $f(n+1) = f(n) + 2$.
 - $f(n+1) = 3f(n)$.
 - $f(n+1) = 2^{f(n)}$.
 - $f(n+1) = f(n)^2 + f(n) + 1$.
 2. Encontre $f(1), f(2), f(3), f(4)$ e $f(5)$ se $f(n)$ for definido recursivamente por $f(0) = 3$ e para $n = 0, 1, 2, \dots$
 - $f(n+1) = -2f(n)$.
 - $f(n+1) = 3f(n) + 7$.
 - $f(n+1) = f(n)^2 - 2f(n) - 2$.
 - $f(n+1) = 3^{f(n)/3}$.
 3. Encontre $f(2), f(3), f(4)$ e $f(5)$ se f for definido recursivamente por $f(0) = -1, f(1) = 2$ e para $n = 1, 2, \dots$
 - $f(n+1) = f(n) + 3f(n-1)$.
 - $f(n+1) = f(n)^2 f(n-1)$.
 - $f(n+1) = 3f(n)^2 - 4f(n-1)^2$.
 - $f(n+1) = f(n-1)/f(n)$.
 4. Encontre $f(2), f(3), f(4)$ e $f(5)$ se f for definido recursivamente por $f(0) = f(1) = 1$ e para $n = 1, 2, \dots$
 - $f(n+1) = f(n) - f(n-1)$.
 - $f(n+1) = f(n)f(n-1)$.
 - $f(n+1) = f(n)^2 + f(n-1)^3$.
 - $f(n+1) = f(n)/f(n-1)$.
 5. Determine se cada uma das definições propostas abaixo é uma definição recursiva válida de uma função f a partir do conjunto dos números inteiros não negativos para o conjunto dos números inteiros. Se f for bem definida, encontre uma fórmula para $f(n)$ quando n for um número inteiro não negativo e demonstre que sua fórmula é válida.
 - $f(0) = 0, f(n) = 2f(n-2)$ para $n \geq 1$
 - $f(0) = 1, f(n) = f(n-1) - 1$ para $n \geq 1$
 - $f(0) = 2, f(1) = 3, f(n) = f(n-1) - 1$ para $n \geq 2$
 - $f(0) = 1, f(1) = 2, f(n) = 2f(n-2)$ para $n \geq 2$
 - $f(0) = 1, f(n) = 3f(n-1)$ se n for ímpar e $n \geq 1$ e $f(n) = 9f(n-2)$ se n for par e $n \geq 2$
 6. Determine se cada uma das definições propostas a seguir é uma definição recursiva válida de uma função f a partir do conjunto dos números inteiros não negativos para o conjunto dos números inteiros. Se f for bem definida, encontre uma fórmula para $f(n)$ quando n for um número inteiro não negativo e demonstre que sua fórmula é válida.
 - $f(0) = 1, f(n) = -f(n-1)$ para $n \geq 1$
 - $f(0) = 1, f(1) = 0, f(2) = 2, f(n) = 2f(n-3)$ para $n \geq 3$
 - c) $f(0) = 0, f(1) = 1, f(n) = 2f(n+1)$ para $n \geq 2$
 - d) $f(0) = 0, f(1) = 1, f(n) = 2f(n-1)$ para $n \geq 1$
 - e) $f(0) = 2, f(n) = f(n-1)$ se n for ímpar e $n \geq 1$ e $f(n) = 2f(n-2)$ se $n \geq 2$
 7. Dê uma definição recursiva da seqüência $\{a_n\}, n = 1, 2, 3, \dots$ se
 - $a_n = 6n$.
 - $a_n = 2n + 1$.
 - $a_n = 10^n$.
 - $a_n = 5$.
 8. Dê uma definição recursiva da seqüência $\{a_n\}, n = 1, 2, 3, \dots$ se
 - $a_n = 4n - 2$.
 - $a_n = 1 + (-1)^n$.
 - $a_n = n(n+1)$.
 - $a_n = n^2$.
 9. Consulte F como uma função, tal que $F(n)$ é a soma dos primeiros n números inteiros positivos. Dê uma definição recursiva de $F(n)$.
 10. Dê uma definição recursiva de $S_m(n)$, a soma do número inteiro m e do número inteiro não negativo n .
 11. Dê uma definição recursiva de $P_m(n)$, o produto do número inteiro m pelo número inteiro não negativo n .
- Nos exercícios 12 a 19, f_n é o n -ésimo número de Fibonacci.
12. Demonstre que $f_1^2 + f_2^2 + \dots + f_n^2 = f_n f_{n+1}$ quando n é um número inteiro positivo.
 13. Demonstre que $f_1 + f_3 + \dots + f_{2n-1} = f_{2n}$ quando n é um número inteiro positivo.
 - *14. Mostre que $f_{n+1} f_{n-1} - f_n^2 = (-1)^n$ quando n é um número inteiro positivo.
 - *15. Mostre que $f_0 f_1 + f_1 f_2 + \dots + f_{2n-1} f_{2n} = f_{2n}^2$ quando n é um número inteiro positivo.
 - *16. Mostre que $f_0 - f_1 + f_2 - \dots - f_{2n-1} + f_{2n} = f_{2n-1} - 1$ quando n é um número inteiro positivo.
 17. Determine o número de divisões usadas pelo algoritmo de Euclides para encontrar o máximo divisor comum dos números de Fibonacci f_n e f_{n+1} , em que n é um número inteiro não negativo. Verifique sua resposta usando a indução matemática.
 18. Considere
- $$\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$
- Mostre que
- $$\mathbf{A}^n = \begin{bmatrix} f_{n+1} & f_n \\ f_n & f_{n-1} \end{bmatrix}$$
- quando n é um número inteiro positivo.

19. Considerando determinantes de ambos os lados da equação do Exercício 18, demonstre a identidade dada no Exercício 14. (Lembre-se de que o determinante de uma matriz $\begin{vmatrix} a & b \\ c & d \end{vmatrix}$ é $ad - bc$.)
- *20. Dê uma definição recursiva das funções max e min, para que $\max(a_1, a_2, \dots, a_n)$ e $\min(a_1, a_2, \dots, a_n)$ sejam o máximo e o mínimo de n números a_1, a_2, \dots, a_n , respectivamente.
- *21. Considere a_1, a_2, \dots, a_n e b_1, b_2, \dots, b_n como números reais. Use as definições recursivas do Exercício 20 para demonstrar as equações abaixo.
- $\max(-a_1, -a_2, \dots, -a_n) = -\min(a_1, a_2, \dots, a_n)$
 - $\max(a_1 + b_1, a_2 + b_2, \dots, a_n + b_n)$
 $\leq \max(a_1, a_2, \dots, a_n) + \max(b_1, b_2, \dots, b_n)$
 - $\min(a_1 + b_1, a_2 + b_2, \dots, a_n + b_n)$
 $\geq \min(a_1, a_2, \dots, a_n) + \min(b_1, b_2, \dots, b_n)$
22. Mostre que o conjunto S , definido por $1 \in S$ e $s + t \in S$ sempre que $s \in S$ e $t \in S$, é o conjunto dos números inteiros positivos.
23. Dê uma definição recursiva do conjunto dos números inteiros positivos que são múltiplos de 5.
24. Dê uma definição recursiva do
- conjunto de números inteiros positivos e ímpares.
 - conjunto dos números inteiros positivos que são potências de 3.
 - conjunto de polinômios com coeficientes inteiros.
25. Dê uma definição recursiva do
- conjunto de números inteiros pares.
 - conjunto de números inteiros positivos congruentes a 2 módulo 3.
 - conjunto de números inteiros positivos não divisíveis por 5.
26. Consulte S como o subconjunto do conjunto de pares ordenados de números inteiros definido recursivamente por
Passo base: $(0, 0) \in S$.
Passo recursivo: Se $(a, b) \in S$, então $(a + 2, b + 3) \in S$ e $(a + 3, b + 2) \in S$.
- Liste os elementos de S produzidos pelas primeiras cinco aplicações do passo recursivo.
 - Use a indução completa no número de aplicações do passo recursivo da definição para mostrar que $5 \mid a + b$ quando $(a, b) \in S$.
 - Use a indução estrutural para mostrar que $5 \mid a + b$ quando $(a, b) \in S$.
27. Consulte S como o subconjunto do conjunto de pares ordenados de números inteiros definido recursivamente por
Passo base: $(0, 0) \in S$.
Passo recursivo: Se $(a, b) \in S$, então $(a, b + 1) \in S$, $(a + 1, b + 1) \in S$, e $(a + 2, b + 1) \in S$.
- Liste os elementos de S produzidos pelas primeiras quatro aplicações do passo recursivo.
 - Use a indução completa sobre o número de aplicações do passo recursivo da definição para mostrar que $a \leq 2b$ sempre que $(a, b) \in S$.
 - Use a indução estrutural para mostrar que $a \leq 2b$ sempre que $(a, b) \in S$.
28. Dê uma definição recursiva para cada um dos conjuntos de pares ordenados de números inteiros positivos abaixo. [Dica: Organize os pontos do conjunto no plano e procure por linhas que contenham pontos no conjunto.]
- $S = \{(a, b) \mid a \in \mathbb{Z}^+, b \in \mathbb{Z}^+ \text{ e } a + b \text{ é ímpar}\}$
 - $S = \{(a, b) \mid a \in \mathbb{Z}^+, b \in \mathbb{Z}^+ \text{ e } a \mid b\}$
 - $S = \{(a, b) \mid a \in \mathbb{Z}^+, b \in \mathbb{Z}^+ \text{ e } 3 \mid a + b\}$
29. Dê uma definição recursiva para cada um dos conjuntos de pares ordenados de números inteiros positivos abaixo. Use a indução estrutural para demonstrar que a definição recursiva que você encontrou está correta. [Dica: Para encontrar uma definição recursiva, organize os pontos do conjunto no plano e procure por padrões.]
- $S = \{(a, b) \mid a \in \mathbb{Z}^+, b \in \mathbb{Z}^+ \text{ e } a + b \text{ é par}\}$
 - $S = \{(a, b) \mid a \in \mathbb{Z}^+, b \in \mathbb{Z}^+ \text{ e } a \text{ ou } b \text{ é ímpar}\}$
 - $S = \{(a, b) \mid a \in \mathbb{Z}^+, b \in \mathbb{Z}^+, a + b \text{ é ímpar e } 3 \mid b\}$
30. Demonstre que em uma cadeia de bits, a seqüência 01 aparece no máximo uma vez mais que a seqüência 10.
31. Defina as fórmulas bem formadas de conjuntos, variáveis que representam os conjuntos e os operadores de $\{\setminus, \cup, \cap, \neg\}$.
32. a) Dê uma definição recursiva da função $uns(s)$, que contém o número de uns em uma cadeia de bits s .
b) Use a indução estrutural para demonstrar que $uns(st) = uns(s) + uns(t)$.
33. a) Dê uma definição recursiva da função $m(s)$, que é igual ao menor dígito em uma cadeia não vazia de dígitos decimais.
b) Use a indução estrutural para demonstrar que $m(st) = \min(m(s), m(t))$.
- O **reverso** de uma cadeia é a cadeia que contém os símbolos daquela cadeia em ordem reversa. O reverso de uma cadeia w é indicado por w^R .
34. Encontre o reverso das cadeias de bits a seguir.
- 0101
 - 1 1011
 - 1000 1001 0111
35. Dê uma definição recursiva do reverso de uma cadeia. [Dica: Primeiro, defina o reverso da cadeia vazia. Então, escreva uma cadeia w de extensão $n + 1$ como xy , em que x é uma cadeia de extensão n , e expresse o reverso de w em termos de x^R e y .]
- *36. Use a indução estrutural para demonstrar que $(w_1 w_2)^R = w_2^R w_1^R$.
37. Dê uma definição recursiva de w^i , em que w é uma cadeia e i é um número inteiro não negativo. (Aqui, w^i representa a concatenação de i cópias da cadeia w .)
- *38. Dê uma definição recursiva do conjunto de cadeias de bits que são palíndromos.
39. Quando uma cadeia pertence ao conjunto A de cadeias de bits definidas recursivamente por
- $$\lambda \in A,$$
- $$0x1 \in A \text{ se } x \in A,$$
- em que λ é a cadeia vazia?
- *40. Defina recursivamente o conjunto de cadeias de bits que têm mais zeros que uns.
41. Use o Exercício 37 e a indução matemática para mostrar que $l(w^i) = i \cdot l(w)$, em que w é uma cadeia e i é um número inteiro não negativo.

- *42. Mostre que $(w^R)^i = (w^i)^R$ sempre que w for uma cadeia e i for um número inteiro não negativo; ou seja, mostre que a i -ésima potência da reversa de uma cadeia é a reversa da i -ésima potência da cadeia.
43. Use a indução estrutural para mostrar que $n(T) \geq 2h(T) + 1$, em que T é uma árvore binária completa, $n(T)$ é igual ao número de vértices de T e $h(T)$ é a altura de T .

O conjunto de folhas e o conjunto de vértices internos de uma árvore binária completa podem ser definidos recursivamente.

Passo base: A raiz r é uma folha da árvore binária completa com um vértice r . Esta árvore não tem vértices internos.

Passo recursivo: O conjunto de folhas da árvore $T = T_1 \cdot T_2$ é a união do conjunto de folhas de T_1 e o conjunto de folhas de T_2 . Os vértices internos de T são a raiz r de T e a união do conjunto de vértices internos de T_1 e o conjunto de vértices internos de T_2 .

44. Use a indução estrutural para mostrar que $l(T)$, o número de folhas de uma árvore binária completa T , é 1 mais $i(T)$, o número de vértices internos de T .

45. Use a indução generalizada, como foi feito no Exemplo 15, para mostrar que se $a_{m,n}$ for definido recursivamente por $a_{0,0} = 0$ e

$$a_{m,n} = \begin{cases} a_{m-1,n} + 1 & \text{se } n = 0 \text{ e } m > 0 \\ a_{m,n-1} + 1 & \text{se } n > 0, \end{cases}$$

então $a_{m,n} = m + n$ para todo $(m, n) \in \mathbb{N} \times \mathbb{N}$.

46. Use a indução generalizada, como foi feito no Exemplo 15, para mostrar que se $a_{m,n}$ for definido recursivamente por $a_{1,1} = 5$ e

$$a_{m,n} = \begin{cases} a_{m-1,n} + 2 & \text{se } n = 1 \text{ e } m > 1 \\ a_{m,n-1} + 2 & \text{se } n > 1, \end{cases}$$

então $a_{m,n} = 2(m+n) + 1$ para todo $(m, n) \in \mathbb{Z}^+ \times \mathbb{Z}^+$.

*47. Uma **partição** de um número inteiro positivo n é uma forma de escrever n como a soma de números inteiros positivos, em que a ordem dos termos na soma não importa. Por exemplo, $7 = 3 + 2 + 1 + 1$ é uma partição de 7. Considere P_m como o número de partições diferentes de m , e considere $P_{m,n}$ como os números de maneiras diferentes de expressar m como a soma dos números inteiros positivos não excedentes a n .

- a) Mostre que $P_{m,m} = P_m$.
 b) Mostre que a definição recursiva a seguir para $P_{m,n}$ está correta:

$$P_{m,n} = \begin{cases} 1 & \text{se } m = 1 \\ 1 & \text{se } n = 1 \\ P_{m,m} & \text{se } m < n \\ 1 + P_{m,m-1} & \text{se } m = n > 1 \\ P_{m,n-1} + P_{m-n,n} & \text{se } m > n > 1. \end{cases}$$

- c) Encontre o número de partições de 5 e de 6 usando essa definição recursiva.

Considere a definição inductiva de uma versão da **função de Ackermann**. Essa função foi assim chamada em homenagem a Wilhelm Ackermann, um matemático alemão que foi aluno do grande matemático David Hilbert. A função de Ackermann tem um papel importante na teoria das funções recursivas e no estudo da complexidade de determinados algoritmos que envolvem a

união de conjuntos. (Há muitas variações dessa função. Todas são chamadas de função de Ackermann e têm propriedades semelhantes, mesmo que seus valores nem sempre coincidam.)

$$A(m,n) = \begin{cases} 2n & \text{se } m = 0 \\ 0 & \text{se } m \geq 1 \text{ e } n = 0 \\ 2 & \text{se } m \geq 1 \text{ e } n = 1 \\ A(m-1, A(m,n-1)) & \text{se } m \geq 1 \text{ e } n \geq 2 \end{cases}$$

Os exercícios 48 a 55 envolvem essa versão da função de Ackermann.

48. Encontre os valores abaixo para a função de Ackermann.

- a) $A(1, 0)$ b) $A(0, 1)$
 c) $A(1, 1)$ d) $A(2, 2)$

49. Mostre que $A(m, 2) = 4$ sempre que $m \geq 1$.

50. Mostre que $A(1, n) = 2^n$ sempre que $n \geq 1$.

51. Encontre os valores abaixo para a função de Ackermann.

- a) $A(2, 3)$ *b) $A(3, 3)$

*52. Encontre $A(3, 4)$.

**53. Demonstre que $A(m, n+1) > A(m, n)$ sempre que m e n forem números inteiros não negativos.

*54. Demonstre que $A(m+1, n) \geq A(m, n)$ sempre que m e n forem números inteiros não negativos.

55. Demonstre que $A(i, j) \geq j$ sempre que i e j forem números inteiros não negativos.

**56. Use a indução matemática para demonstrar que uma função F definida especificando-se $F(0)$ e uma regra para obter $F(n+1)$ de $F(n)$ é bem definida.

57. Use a indução completa para demonstrar que uma função F definida especificando-se $F(0)$ e uma regra para obter $F(n+1)$ a partir dos valores de $F(k)$ para $k = 0, 1, 2, \dots, n$ é bem definida.

58. Mostre que cada uma das definições recursivas propostas abaixo de uma função no conjunto dos números inteiros positivos não produz uma função bem definida.

- a) $F(n) = 1 + F(\lfloor n/2 \rfloor)$ para $n \geq 1$ e $F(1) = 1$.
 b) $F(n) = 1 + F(n-3)$ para $n \geq 2$, $F(1) = 2$ e $F(2) = 3$.
 c) $F(n) = 1 + F(n/2)$ para $n \geq 2$, $F(1) = 1$ e $F(2) = 2$.
 d) $F(n) = 1 + F(n/2)$ se n for par e $n \geq 2$, $F(n) = 1 - F(n-1)$ se n for ímpar e $F(1) = 1$.
 e) $F(n) = 1 + F(n/2)$ se n for par e $n \geq 2$, $F(n) = F(3n-1)$ se n for ímpar e $n \geq 3$ e $F(1) = 1$.

59. Mostre que cada uma das definições recursivas propostas abaixo de uma função no conjunto dos números inteiros positivos não produz uma função bem definida.

- a) $F(n) = 1 + F(\lfloor (n+1)/2 \rfloor)$ para $n \geq 1$ e $F(1) = 1$.
 b) $F(n) = 1 + F(n-2)$ para $n \geq 2$ e $F(1) = 0$.
 c) $F(n) = 1 + F(n/3)$ para $n \geq 3$, $F(1) = 1$, $F(2) = 2$ e $F(3) = 3$.
 d) $F(n) = 1 + F(n/2)$ se n for par e $n \geq 2$, $F(n) = 1 + F(n-2)$ se n for ímpar e $F(1) = 1$.
 e) $F(n) = 1 + F(F(n-1))$ se $n \geq 2$ e $F(1) = 2$.



Os exercícios 60 a 62 lidam com iterações da função logarítmica. Considere $\log n$ como o logaritmo de n na base 2. A função $\log^{(k)} n$ é definida recursivamente por

$$\log^{(k)} n = \begin{cases} n & \text{se } k = 0 \\ \log(\log^{(k-1)} n) & \text{se } \log^{(k-1)} n \text{ é definido} \\ & \text{e positivo} \\ \text{indefinido} & \text{caso contrário.} \end{cases}$$

O **logaritmo iterado** é a função $\log^* n$ cujo valor em n é o menor número inteiro não negativo k , tal que $\log^{(k)} n \leq 1$.

60. Encontre cada um dos valores abaixo:

- a) $\log^{(2)} 16$ b) $\log^{(3)} 256$
 c) $\log^{(3)} 2^{65536}$ d) $\log^{(4)} 2^{2^{65536}}$

61. Encontre os valores de $\log^* n$ para cada um dos valores abaixo de n :

- a) 2 b) 4 c) 8 d) 16
 e) 256 f) 65536 g) 2^{2048}

62. Encontre o maior número inteiro n , tal que $\log^* n = 5$. Determine o número de dígitos decimais nesse número.

Os exercícios 63 a 65 tratam de funções com valores iterados. Suponha que $f(n)$ seja uma função do conjunto de números reais

— ou números reais positivos ou algum outro conjunto de números reais — para o conjunto de números reais, tal que $f(n)$ é monotonicamente crescente [ou seja, $f(n) < f(m)$ quando $n < m$ e $f(n) < n$ para todo n no domínio de f .] A função $f^{(k)}(n)$ é definida recursivamente por

$$f^{(k)}(n) = \begin{cases} n & \text{se } k = 0 \\ f(f^{(k-1)}(n)) & \text{se } k > 0. \end{cases}$$

Além disso, considere c como um número real positivo. A função iterada f_c^* é o número de repetições de f necessário para reduzir seu argumento a c ou menos, assim $f_c^*(n)$ é o menor número inteiro não negativo k , tal que $f^k(n) \leq c$.

63. Considere $f(n) = n - a$, em que a é um número inteiro positivo. Encontre uma fórmula para $f^{(k)}(n)$. Qual o valor de $f_0^*(n)$ quando n for um número inteiro positivo?

64. Considere $f(n) = n/2$. Encontre uma fórmula para $f^{(k)}(n)$. Qual o valor de $f_1^*(n)$ quando n for um número inteiro positivo?

65. Considere $f(n) = \sqrt{n}$. Encontre uma fórmula para $f^{(k)}(n)$. Qual o valor de $f_2^*(n)$ quando n for um número inteiro positivo?

4.4 Algoritmos Recursivos

Introdução

Às vezes podemos reduzir a solução de um problema com um determinado conjunto de valores iniciais para a solução do mesmo problema com valores iniciais menores. Por exemplo, o problema para encontrar o máximo divisor comum de dois números inteiros positivos a e b , em que $b > a$, pode ser reduzido a encontrar o máximo divisor comum de um par de números inteiros menores, ou seja, $b \bmod a$ e a , pois $\text{mdc}(b \bmod a, a) = \text{mdc}(a, b)$. Quando essa redução pode ser feita, a solução do problema original pode ser encontrada com uma seqüência de reduções, até que o problema seja reduzido ao caso inicial para qualquer solução conhecida. Por exemplo, para encontrar o máximo divisor comum, a redução continua até que o menor dos dois números seja zero, porque $\text{mdc}(a, 0) = a$ quando $a > 0$.

Veremos os algoritmos que reduzem sucessivamente um problema ao mesmo problema com valores iniciais menores que são usados para resolver uma grande variedade de problemas.

DEFINIÇÃO 1

Um algoritmo é chamado de *recursivo* se resolver um problema reduzindo-o a um mesmo problema com valores iniciais menores.



Descreveremos vários algoritmos recursivos diferentes nesta seção.

EXEMPLO 1

Dê um algoritmo recursivo para computar $n!$, em que n é um número inteiro não negativo.



Solução: Podemos construir um algoritmo recursivo que encontre $n!$, em que n é um número inteiro não negativo, com base na definição recursiva de $n!$, que determina que $n! = n \cdot (n - 1)!$ quando n for um número inteiro positivo e que $0! = 1$. Para encontrar $n!$ para determinado número inteiro, usamos o passo recursivo n vezes, cada vez substituindo um valor da função

fatorial pelo valor da função fatorial do próximo número inteiro menor. Nesta última etapa, inserimos o valor de $0!$. O algoritmo recursivo que obtemos é mostrado como Algoritmo 1.

Para ajudar a entender como esse algoritmo funciona, traçamos estas etapas usadas pelo algoritmo para computar $4!$. Primeiro, usamos o passo recursivo para escrever $4! = 4 \cdot 3!$. Então, usamos o mesmo passo recursivo repetidamente para escrever $3! = 3 \cdot 2!$, $2! = 2 \cdot 1!$, e $1! = 1 \cdot 0!$. Inserindo o valor de $0! = 1$ e trabalhando com os passos, vemos que $1! = 1 \cdot 1 = 1$, $2! = 2 \cdot 1! = 2$, $3! = 3 \cdot 2! = 3 \cdot 2 = 6$ e $4! = 4 \cdot 3! = 4 \cdot 6 = 24$. ◀

ALGORITMO 1 Um Algoritmo Recursivo para Computar $n!$.

```
procedure fatorial(n: número inteiro não negativo)
if n = 0 then fatorial(n) := 1
else fatorial(n) := n · fatorial(n - 1)
```

O Exemplo 2 mostra como um algoritmo recursivo pode ser construído para avaliar uma função a partir de sua definição recursiva.

EXEMPLO 2 Dê um algoritmo recursivo para computar a^n , em que a é um número real diferente de zero e n é um número inteiro não negativo.

Solução: Podemos basear um algoritmo recursivo em uma definição recursiva de a^n . Essa definição afirma que $a^{n+1} = a \cdot a^n$ para $n > 0$ e a condição inicial $a^0 = 1$. Para encontrar a^n , usamos sucessivamente o passo recursivo para reduzir o expoente até que ele se torne zero. Apresentamos esse procedimento no Algoritmo 2. ◀

ALGORITMO 2 Um Algoritmo Recursivo para Computar a^n .

```
procedure potencia(a: número real diferente de zero, n: número inteiro não negativo)
if n = 0 then potencia(a, n) := 1
else potencia(a, n) := a · potencia(a, n - 1)
```

EXEMPLO 3 Construa um algoritmo recursivo para computar $b^n \bmod m$, em que b , n e m são números inteiros com $m \geq 2$, $n \geq 0$ e $1 \leq b < m$.

Solução: Podemos basear um algoritmo recursivo no fato de que

$$b^n \bmod m = (b \cdot (b^{n-1} \bmod m)) \bmod m,$$

que é mantido pelo Corolário 2 da Seção 3.4 e com condição inicial $b^0 \bmod m = 1$. Apresentamos essa questão no Exercício 12, ao final desta seção.

Entretanto, podemos construir algoritmos recursivos muito mais eficientes com base na observação de que

$$b^n \bmod m = (b^{n/2} \bmod m)^2 \bmod m$$

quando n é par e

$$b^n \bmod m = ((b^{\lfloor n/2 \rfloor} \bmod m)^2 \bmod m \cdot b \bmod m) \bmod m$$

quando n é ímpar, como descrito em pseudocódigo no Algoritmo 3.

Podemos rascunhar o Algoritmo 3 com a entrada $b = 2$, $n = 5$ e $m = 3$ para ilustrar como ele funciona. Primeiro, como $n = 5$ é ímpar, usamos a cláusula “else” para estabelecer que $mpotência(2, 5, 3) = (mpotência(2, 2, 3)^2 \text{ mod } 3 \cdot 2 \text{ mod } 3) \text{ mod } 3$. Depois, usamos a cláusula “else if” para ver que $mpotência(2, 2, 3) = mpotência(2, 1, 3)^2 \text{ mod } 3$. Usando novamente “else”, vemos que $mpotência(2, 1, 3) = (mpotência(2, 0, 3)^2 \text{ mod } 3 \cdot 2 \text{ mod } 3) \text{ mod } 3$. Por fim, usamos a sentença “if”, para ver que $mpotência(2, 0, 3) = 1$. Assim, temos que $mpotência(2, 1, 3) = (1^2 \text{ mod } 3 \cdot 2 \text{ mod } 3) \text{ mod } 3 = 2$, então $mpotência(2, 2, 3) = 2^2 \text{ mod } 3 = 1$ e, finalmente, $mpotência(2, 5, 3) = (1^2 \text{ mod } 3 \cdot 2 \text{ mod } 3) \text{ mod } 3 = 2$. \blacktriangleleft

ALGORITMO 3 Potenciação Modular Recursiva.

```

procedure mpotencia(b, n, m: números inteiros com  $m \geq 2$ ,  $n \geq 0$ )
  if  $n = 0$  then
    mpotencia(b, n, m) = 1
  else if  $n$  é par then
    mpotencia(b, n, m) = mpotencia(b,  $n/2$ , m)2 mod m
  else
    mpotencia(b, n, m) = (mpotencia(b,  $\lfloor n/2 \rfloor$ , m)2 mod m · b mod m) mod m
  {mpotencia(b, n, m) =  $b^n \text{ mod } m$ }

```

Agora, apresentaremos um algoritmo recursivo para encontrar o máximo divisor comum.

EXEMPLO 4 Dê um algoritmo recursivo para computar o máximo divisor comum de dois números inteiros não negativos a e b com $a < b$.

Solução: Podemos basear o algoritmo recursivo na redução $mdc(a, b) = mdc(b \text{ mod } a, a)$ e na condição que $mdc(0, b) = b$ quando $b > 0$. Isso é feito no procedimento do Algoritmo 4, que é uma versão recursiva do algoritmo de Euclides.

Ilustramos o funcionamento do Algoritmo 4 calculando-o quando as entradas são $a = 5$, $b = 8$. Com essas entradas, o algoritmo usa a sentença “else” para encontrar $mdc(5, 8) = mdc(8 \text{ mod } 5, 5) = mdc(3, 5)$. Ele usa esta sentença novamente para encontrar $mdc(3, 5) = mdc(5 \text{ mod } 3, 3) = mdc(2, 3)$, então para ter $mdc(2, 3) = mdc(3 \text{ mod } 2, 2) = mdc(1, 2)$, então para ter $mdc(1, 2) = mdc(2 \text{ mod } 1, 1) = mdc(0, 1)$. Por fim, para encontrar $mdc(0, 1)$ ele usa o primeiro passo com $a = 0$ para encontrar $mdc(0, 1) = 1$. Conseqüentemente, o algoritmo encontra $mdc(5, 8) = 1$. \blacktriangleleft

ALGORITMO 4 Um Algoritmo Recursivo para Computar $mdc(a, b)$.

```

procedure mdc(a, b: números inteiros não negativos com  $a < b$ )
  if  $a = 0$  then  $mdc(a, b) := b$ 
  else  $mdc(a, b) := mdc(b \text{ mod } a, a)$ 

```

Vamos fornecer agora versões recursivas dos algoritmos de busca que foram introduzidos na Seção 3.1.

EXEMPLO 5 Expresse o algoritmo de busca linear como um procedimento recursivo.

Solução: Para a busca por x na seqüência de busca a_1, a_2, \dots, a_n , no i -ésimo passo do algoritmo, x e a_i são comparados. Se x for igual a a_i , então i é a localização de x . Por outro lado, a busca por x é reduzida a uma busca em uma seqüência com um elemento a menos, ou seja, a seqüência a_{i+1}, \dots, a_n . Podemos agora fornecer um procedimento recursivo, que está apresentado em pseudo-código no Algoritmo 5.

Considere a busca (i, j, x) como o procedimento que procura por x na seqüência a_i, a_{i+1}, \dots, a_j . A entrada do algoritmo consiste no trio $(1, n, x)$. O procedimento termina em um passo se o primeiro termo da seqüência restante for x ou se houver apenas um termo na seqüência e este não for x . Se x não for o primeiro termo e houver termos adicionais, o mesmo procedimento é realizado, mas com uma seqüência de busca com um termo a menos, obtido apagando-se o primeiro termo da seqüência de busca. ◀

ALGORITMO 5 Um Algoritmo de Busca Linear Recursivo.

```
procedure busca(i, j, x: i, j, x números inteiros, 1 ≤ i ≤ n, 1 ≤ j ≤ n)
if  $a_i = x$  then
    localizacao := i
else if  $i = j$  then
    localizacao := 0
else
    busca(i + 1, j, x)
```

EXEMPLO 6 Construa uma versão recursiva do algoritmo de busca binária.

Solução: Suponha que queiramos localizar x na seqüência a_1, a_2, \dots, a_n , números inteiros em ordem crescente. Para realizar uma busca binária, começamos pela comparação de x com o termo do meio, $a_{\lfloor(n+1)/2\rfloor}$. Nossa algoritmo terminará se x for igual a este termo. Por outro lado, reduzimos a busca a uma seqüência de busca menor, ou seja, a primeira metade da seqüência se x for menor que o termo do meio da seqüência original, ou a segunda metade se ele for maior. Reduzimos a solução do problema de busca para a solução do mesmo problema como uma seqüência aproximadamente igual à metade da original. Expressamos esta versão recursiva do algoritmo de busca binária como Algoritmo 6. ◀

ALGORITMO 6 Um Algoritmo de Busca Binária Recursivo.

```
procedure busca binaria(i, j, x: i, j, x números inteiros, 1 ≤ i ≤ n, 1 ≤ j ≤ n)
m :=  $\lfloor(i + j)/2\rfloor$ 
if  $x = a_m$  then
    localizacao := m
else if ( $x < a_m$  e  $i < m$ ) then
    busca binaria(x, i, m - 1)
else if ( $x > a_m$  e  $j > m$ ) then
    busca binaria(x, m + 1, j)
else
    localizacao := 0
```

Demonstrando que os Algoritmos Recursivos Estão Corretos

A indução matemática e sua variante, a indução completa, podem ser usadas para demonstrar que um algoritmo recursivo está correto, ou seja, que ele produz a saída desejada para todos os valores de entrada possíveis. Os exemplos 7 e 8 ilustram como a indução matemática ou a completa podem ser usadas para demonstrar que os algoritmos recursivos estão corretos. Primeiro mostraremos que o Algoritmo 2 está correto.

EXEMPLO 7 Demonstre que o Algoritmo 2, que computa potências de números reais, está correto.

Solução: Usamos a indução matemática no expoente n .

PASSO BASE: Se $n = 0$, o primeiro passo do algoritmo nos diz que $\text{potência}(a, 0) = 1$. Isso é correto porque $a^0 = 1$ para todo número real diferente de zero a . Isso completa o passo base.

PASSO DE INDUÇÃO: A hipótese indutiva é a proposição de que a $\text{potência}(a, k) = a^k$ para todo $a \neq 0$ para o número inteiro não negativo k , ou seja, a hipótese indutiva é a proposição de que o algoritmo computará corretamente a^k . Para completar o passo de indução, mostramos que se a hipótese indutiva for verdadeira, então o algoritmo vai computar corretamente a^{k+1} . Como $k + 1$ é um número inteiro positivo, quando o algoritmo computar a^{k+1} , ele executará a $\text{potência}(a, k + 1) = a \cdot \text{potência}(a, k)$. Pela hipótese indutiva, temos $\text{potência}(a, k) = a^k$, assim, $\text{potência}(a, k + 1) = a \cdot \text{potência}(a, k) = a \cdot a^k = a^{k+1}$. Isso completa o passo de indução.

Completamos os passos base e de indução; assim, podemos concluir que o Algoritmo 2 sempre computará a^n corretamente quando $a \neq 0$ e n for um número inteiro não negativo. ◀

Geralmente, precisamos usar a indução completa para demonstrar que os algoritmos recursivos estão corretos, em vez de utilizar apenas a indução matemática. O Exemplo 8 ilustra esse fato; ele mostra como a indução completa pode ser usada para demonstrar que o Algoritmo 3 está correto.

EXEMPLO 8 Demonstre que o Algoritmo 3, que faz a computação de potências modulares, está correto.



Solução: Usamos a indução completa no expoente n .

PASSO BASE: Considere b como um número inteiro e m como um número inteiro com $m \geq 2$. Quando $n = 0$, o algoritmo executa $\text{mpotência}(b, n, m)$ igual a 1. Isto está correto porque $b^0 \bmod m = 1$. O passo base está completo.

PASSO DE INDUÇÃO: Para a hipótese indutiva, assuma que $\text{mpotência}(b, j, m) = b^j \bmod m$ para todos os números inteiros $0 \leq j < k$ sempre que b for um número inteiro positivo e m for um número inteiro com $m \geq 2$. Para completar o passo de indução, mostramos que se ele estiver correto, então $\text{mpotência}(b, k, m) = b^k \bmod m$. Como o algoritmo recursivo trata de valores pares e ímpares de k de forma diferente, separamos o passo de indução em dois casos.

Quando k é par, temos

$$\text{mpotência}(b, k, m) = \text{mpotência}(b, k/2, m)^2 \bmod m = (b^{k/2} \bmod m)^2 \bmod m = b^k \bmod m,$$

em que usamos a hipótese indutiva para substituir $\text{mpotência}(b, k/2, m)$ por $b^{k/2} \bmod m$.

Quando k é ímpar, temos

$$\begin{aligned} \text{mpotência}(b, k, m) &= ((\text{mpotência}(b, \lfloor k/2 \rfloor, m))^2 \bmod m \cdot b \bmod m) \bmod m \\ &= ((b^{\lfloor k/2 \rfloor} \bmod m)^2 \bmod m \cdot b \bmod m) \bmod m \\ &= b^{2\lfloor k/2 \rfloor + 1} \bmod m = b^k \bmod m, \end{aligned}$$

usando o Corolário 2 da Seção 3.4, pois $2\lfloor k/2 \rfloor + 1 = 2(k-1)/2 + 1 = k$ quando k é ímpar. Aqui nós usamos a hipótese indutiva para substituir $\text{mpotência}(b, \lfloor k/2 \rfloor, m)$ por $b^{\lfloor k/2 \rfloor} \bmod m$. Isso completa o passo de indução.

deira. Assim, $m - 1$ não pode ser maior que n , de modo que $m - 1 = n$ e $P(n)$ é, de fato, verdadeira. Esta contradição mostra que $P(n)$ é verdadeira para todo n . 29. O erro está em ir do caso base $n = 0$ para o próximo caso, $n = 1$; não podemos escrever 1 como a soma de dois números naturais menores. 31. Suponha que a propriedade da boa ordenação seja válida. Suponha que $P(1)$ seja verdadeira e que a afirmação condicional $[P(1) \wedge P(2) \wedge \dots \wedge P(n)] \rightarrow P(n + 1)$ seja verdadeira para todo inteiro positivo n . Seja S o conjunto dos inteiros positivos n para os quais $P(n)$ é falsa. Mostraremos que $S = \emptyset$. Suponha que $S \neq \emptyset$. Então, pela propriedade da boa ordenação, existe um menor inteiro m em S . Sabemos que m não pode ser 1 porque $P(1)$ é verdadeira. Como $n = m$ é o menor inteiro, tal que $P(n)$ é falsa, $P(1), P(2), \dots, P(m - 1)$ são verdadeiras, e $m - 1 \geq 1$. Como $[P(1) \wedge P(2) \wedge \dots \wedge P(m - 1)] \rightarrow P(m)$ é verdadeira, segue que $P(m)$ também deve ser verdadeira, o que é uma contradição. Logo, $S = \emptyset$. 33. Em cada caso, dê uma demonstração por contradição baseada em um “menor contra-exemplo”, ou seja, valores de n e k tal que $P(n, k)$ não seja verdadeira e n e k sejam menores em algum sentido. a) Escolha um contra-exemplo com $n + k$ tão pequeno quanto possível. Não podemos ter $n = 1$ e $k = 1$, porque nos foi dado que $P(1, 1)$ é verdadeira. Portanto, ou $n > 1$ ou $k > 1$. No primeiro caso, pela nossa escolha de contra-exemplo, sabemos que $P(n - 1, k)$ é verdadeira. Mas o passo de indução então força $P(n, k)$ a ser verdadeira, uma contradição. b) Escolha um contra-exemplo com n tão pequeno quanto possível. Não podemos ter $n = 1$, porque nos foi dado que $P(1, k)$ é verdadeira para todo k . Portanto, $n > 1$. Pela nossa escolha de contra-exemplo, sabemos que $P(n - 1, k)$ é verdadeira. Mas o passo de indução então força $P(n, k)$ a ser verdadeira, uma contradição. c) Escolha um contra-exemplo com k tão pequeno quanto possível. Não podemos ter $k = 1$, porque nos foi dito que $P(n, 1)$ é verdadeira para todo n . Portanto, $k > 1$. Pela nossa escolha de contra-exemplo, sabemos que $P(n, k - 1)$ é verdadeira. Mas o passo de indução então força $P(n, k)$ a ser verdadeira, uma contradição. 35. Seja $P(n)$ a afirmação de que, se x_1, x_2, \dots, x_n forem n números reais distintos, então são usadas $n - 1$ multiplicações para encontrar o produto destes números, não importando como os parênteses são inseridos neste produto. Demonstraremos que $P(n)$ é verdadeira usando indução completa. O caso base $P(1)$ é verdadeiro porque são necessárias $1 - 1 = 0$ multiplicações para encontrar o produto de x_1 , um produto com apenas um fator. Suponha que $P(k)$ seja verdadeira para $1 \leq k \leq n$. A última multiplicação usada para encontrar o produto dos $n + 1$ números reais distintos $x_1, x_2, \dots, x_n, x_{n+1}$ é uma multiplicação do produto dos primeiros k destes números para algum k e o produto dos últimos $n + 1 - k$ deles. Pela hipótese de indução, são usadas $k - 1$ multiplicações para encontrar o produto de k destes números, não importando como os parênteses foram inseridos no produto destes números, e são usadas $n - k$ multiplicações para encontrar o produto dos outros $n + 1 - k$ deles, não importando como os parênteses foram inseridos no produto destes números. Como é necessária mais uma multiplicação para encontrar o produto de todos os $n + 1$ números, o número total de multiplicações usadas é igual a $(k - 1) + (n - k) + 1 = n$. Logo, $P(n + 1)$ é verdadeira. 37. Suponha

que $a = dq + r = dq' + r'$ com $0 \leq r < d$ e $0 \leq r' < d$. Então $d(q - q') = r' - r$. Segue que d divide $r' - r$. Como $-d < r' - r < d$, temos $r' - r = 0$. Logo, $r' = r$. Segue que $q = q'$. 39. Isto é um paradoxo causado pela auto-referência. A resposta é claramente “não”. Existe um número finito de palavras em inglês, logo, apenas um número finito de seqüências de 15 palavras ou menos; portanto, apenas um número finito de números inteiros positivos pode ser descrito, não todos eles. 41. Suponha que a propriedade da boa ordenação fosse falsa. Seja S um conjunto não vazio de inteiros não negativos que não tivesse um menor elemento. Seja $P(n)$ a afirmação “ $i \notin S$ para $i = 0, 1, \dots, n$ ”. $P(0)$ é verdadeira porque se $0 \in S$ então S tem um menor elemento, a saber, 0. Agora, suponha que $P(n)$ seja verdadeira. Logo, $0 \notin S, 1 \notin S, \dots, n \notin S$. Claramente, $n + 1$ não pode estar em S , pois, se estivesse, ele seria seu menor elemento. Logo, $P(n + 1)$ é verdadeira. Assim, pelo princípio da indução matemática, $n \notin S$ para todo inteiro não negativo n . Logo, $S = \emptyset$, uma contradição. 43. Isto segue imediatamente do Exercício 41 (se considerarmos o princípio da indução matemática como axioma) e desse exercício junto com a discussão a seguir do enunciado formal de indução completa no texto, o qual mostrou que a indução completa implica o princípio de indução matemática (se considerarmos a indução completa como axioma).

Seção 4.3

1. a) $f(1) = 3, f(2) = 5, f(3) = 7, f(4) = 9$ b) $f(1) = 3, f(2) = 9, f(3) = 27, f(4) = 81$ c) $f(1) = 2, f(2) = 4, f(3) = 16, f(4) = 65\,536$ d) $f(1) = 3, f(2) = 13, f(3) = 183, f(4) = 33\,673$ 3. a) $f(2) = -1, f(3) = 5, f(4) = 2, f(5) = 17$ b) $f(2) = -4, f(3) = 32, f(4) = -4096, f(5) = 53\,687\,0912$ c) $f(2) = 8, f(3) = 176, f(4) = 92\,672, f(5) = 25\,764\,174\,848$ d) $f(2) = -\frac{1}{2}, f(3) = -4, f(4) = \frac{1}{8}, f(5) = -32$ 5. a) Não é válida. b) $f(n) = 1 - n$. *Passo de indução:* se $f(k) = 1 - k$, então $f(k + 1) = f(k) - 1 = 1 - k - 1 = 1 - (k + 1)$. c) $f(n) = 4 - n$ se $n > 0$ e $f(0) = 2$. *Passo base:* $f(0) = 2$ e $f(1) = 3 = 4 - 1$. *Passo de indução* (com $k \geq 1$): $f(k + 1) = f(k) - 1 = (4 - k) - 1 = 4 - (k + 1)$. d) $f(n) = 2^{\lfloor \frac{(n+1)}{2} \rfloor}$. *Passo base:* $f(0) = 1 = 2^{\lfloor \frac{(0+1)}{2} \rfloor}$ e $f(1) = 2 = 2^{\lfloor \frac{(1+1)}{2} \rfloor}$. *Passo de indução* (com $k \geq 1$): $f(k + 1) = 2f(k - 1) = 2 \cdot 2^{\lfloor \frac{k}{2} \rfloor} = 2^{\lfloor \frac{(k+1)+1}{2} \rfloor}$. e) $f(n) = 3^n$. *Passo base:* Trivial. *Passo de indução:* Para n ímpar, $f(n) = 3f(n - 1) = 3 \cdot 3^{n-1} = 3^n$; e para $n > 1$ par, $f(n) = 9f(n - 2) = 9 \cdot 3^{n-2} = 3^n$. 7. Existem muitas possibilidades corretas de resposta. Daremos algumas relativamente simples. a) $a_{n+1} = a_n + 6$ para $n \geq 1$ e $a_1 = 6$ b) $a_{n+1} = a_n + 2$ para $n \geq 1$ e $a_1 = 3$ c) $a_{n+1} = 10a_n$ para $n \geq 1$ e $a_1 = 10$ d) $a_{n+1} = a_n$ para $n \geq 1$ e $a_1 = 5$ 9. $F(0) = 0, F(n) = F(n - 1) + n$ para $n \geq 1$ 11. $P_m(0) = 0, P_m(n + 1) = P_m(n) + m$ 13. Seja $P(n)$ a afirmação “ $f_1 + f_3 + \dots + f_{2n-1} = f_{2n}$ ”. *Passo base:* $P(1)$ é verdadeira porque $f_1 = 1 = f_2$. *Passo de indução:* Suponha que $P(k)$ seja verdadeira. Então $f_1 + f_3 + \dots + f_{2k-1} + f_{2k+1} = f_{2k} + f_{2k+1} = f_{2k+2} + f_{2(k+1)}$. 15. *Passo base:* $f_0f_1 + f_1f_2 = 0 \cdot 1 + 1 \cdot 1 = 1^2 = f_2^2$. *Passo de indução:* Suponha que $f_0f_1 + f_1f_2 + \dots + f_{2k-1}f_{2k} = f_{2k}^2$. Então $f_0f_1 + f_1f_2 + \dots + f_{2k-1}f_{2k} + f_{2k}f_{2k+1} + f_{2k+1}f_{2k+2} = f_{2k}^2 + f_{2k}f_{2k+1} + f_{2k+1}f_{2k+2} = f_{2k+2}$

$(f_{2k} + f_{2k+1}) + f_{2k+1}f_{2k+2} = f_{2k}f_{2k+2} + f_{2k+1}f_{2k+2} = (f_{2k} + f_{2k+1})f_{2k+2} = f_{2k+1}^2$. 17. O número de divisões usadas pelo algoritmo de Euclides para determinar o $\text{mdc}(f_{n+1}, f_n)$ é 0 para $n = 0$, 1 para $n = 1$ e $n - 1$ para $n \geq 2$. Para demonstrar este resultado para $n \geq 2$ usamos indução matemática. Para $n = 2$, uma divisão mostra que $\text{mdc}(f_3, f_2) = \text{mdc}(2, 1) = \text{mdc}(1, 0) = 1$. Agora, suponha que $k - 1$ divisões sejam usadas para encontrar $\text{mdc}(f_{k+1}, f_k)$. Para encontrar $\text{mdc}(f_{k+2}, f_{k+1})$, primeiro divida f_{k+2} por f_{k+1} para obter $f_{k+2} = 1 \cdot f_{k+1} + f_k$. Depois de uma divisão temos $\text{mdc}(f_{k+2}, f_{k+1}) = \text{mdc}(f_{k+1}, f_k)$. Pela hipótese de indução, segue que são necessárias exatamente mais $k - 1$ divisões. Isto mostra que são necessárias k divisões para encontrar $\text{mdc}(f_{k+2}, f_{k+1})$, terminando a demonstração por indução. 19. $|A| = -1$. Logo, $|A^n| = (-1)^n$. Segue que $f_{n+1}f_{n-1} - f_n^2 = (-1)^n$. 21. a) Demonstração por indução. Passo base: Para $n = 1$, $\max(-a_1) = -a_1 = -\min(a_1)$. Para $n = 2$, existem dois casos. Se $a_2 \geq a_1$, então $-\bar{a}_1 \geq -a_2$, de modo que $\max(-a_1, -a_2) = -a_1 = -\min(a_1, a_2)$. Se $a_2 < a_1$, então $-\bar{a}_1 < -a_2$, de modo que $\max(-a_1, -a_2) = -a_2 = -\min(a_1, a_2)$. Passo de indução: Suponha que seja verdade para k com $k \geq 2$. Então $\max(-a_1, -a_2, \dots, -a_k, -a_{k+1}) = \max(\max(-a_1, \dots, -a_k), -a_{k+1}) = \max(-\min(a_1, \dots, a_k), -a_{k+1}) = -\min(\min(a_1, \dots, a_k), a_{k+1}) = -\min(a_1, \dots, a_{k+1})$. b) Demonstração por indução matemática. Passo base: Para $n = 1$, o resultado é a identidade $a_1 + b_1 = a_1 + b_1$. Para $n = 2$, considere primeiro o caso no qual $a_1 + b_1 \geq a_2 + b_2$. Então $\max(a_1 + b_1, a_2 + b_2) = a_1 + b_1$. Observe também que $a_1 \leq \max(a_1, a_2)$ e $b_1 \leq \max(b_1, b_2)$, de modo que $a_1 + b_1 \leq \max(a_1, a_2) + \max(b_1, b_2)$. Portanto, $\max(a_1 + b_1, a_2 + b_2) = a_1 + b_1 \leq \max(a_1, a_2) + \max(b_1, b_2)$. O caso com $a_1 + b_1 < a_2 + b_2$ é análogo. Passo de indução: Suponha que o resultado seja verdadeiro para k . Então $\max(a_1 + b_1, a_2 + b_2, \dots, a_k + b_k, a_{k+1} + b_{k+1}) = \max(\max(a_1 + b_1, a_2 + b_2, \dots, a_k + b_k), a_{k+1} + b_{k+1}) \leq \max(\max(a_1, a_2, \dots, a_k) + \max(b_1, b_2, \dots, b_k), a_{k+1} + b_{k+1}) \leq \max(\max(a_1, a_2, \dots, a_k), a_{k+1}) + \max(\max(b_1, b_2, \dots, b_k), b_{k+1}) = \max(a_1, a_2, \dots, a_k, a_{k+1}) + \max(b_1, b_2, \dots, b_k, b_{k+1})$. c) O mesmo que na parte (b), mas substitua toda ocorrência de “max” por “min” e inverta cada desigualdade. 23. $5 \in S$ e $x + y \in S$ se $x, y \in S$. 25. a) $0 \in S$ e, se $x \in S$, então $x + 2 \in S$ e $x - 2 \in S$. b) $2 \in S$ e, se $x \in S$, então $x + 3 \in S$. c) $1 \in S$, $2 \in S$, $3 \in S$, $4 \in S$ e, se $x \in S$, então $x + 5 \in S$. 27. a) $(0, 1), (1, 1), (2, 1); (0, 2), (1, 2), (2, 2), (3, 2), (4, 2); (0, 3), (1, 3), (2, 3), (3, 3), (4, 3), (5, 3), (6, 3); (0, 4), (1, 4), (2, 4), (3, 4), (4, 4), (5, 4), (6, 4), (7, 4), (8, 4)$. b) Seja $P(n)$ a afirmação de que $a \leq 2b$ sempre que $(a, b) \in S$ for obtido por n aplicações do passo recursivo. Passo base: $P(0)$ é verdadeira, porque o único elemento de S obtido sem nenhuma aplicação do passo recursivo é $(0, 0)$ e, de fato, $0 \leq 2 \cdot 0$. Passo de indução: Suponha que $a \leq 2b$ sempre que $(a, b) \in S$ seja obtido por k ou menos aplicações do passo recursivo, e considere um elemento obtido com $k + 1$ aplicações do passo recursivo. Como a aplicação final do passo recursivo a um elemento (a, b) deve ser aplicada a um elemento obtido com menos aplicações do passo recursivo, sabemos que $a \leq 2b$. Adicione $0 \leq 2$, $1 \leq 2$ e $2 \leq 2$, respectivamente, para obter $a \leq 2(b + 1)$, $a + 1 \leq 2(b + 1)$ e $a + 2 \leq 2(b + 1)$, como desejado. c) Isto vale para o passo base, porque $0 \leq 0$. Se isto valesse para (a, b) , então também valeria para os elemen-

tos obtidos de (a, b) no passo recursivo, porque adicionar $0 \leq 2$, $1 \leq 2$ e $2 \leq 2$, respectivamente, a $a \leq 2b$ fornece $a \leq 2(b + 1)$, $a + 1 \leq 2(b + 1)$ e $a + 2 \leq 2(b + 1)$. 29. a) Defina S por $(1, 1) \in S$, e, se $(a, b) \in S$, então $(a + 2, b) \in S$, $(a, b + 2) \in S$ e $(a + 1, b + 1) \in S$. Todos os elementos colocados em S satisfazem a condição, porque $(1, 1)$ tem uma soma par de coordenadas, e, se (a, b) tiver uma soma par de coordenadas, então o mesmo ocorre com $(a + 2, b)$, $(a, b + 2)$ e $(a + 1, b + 1)$. Reciprocamente, mostramos por indução na soma das coordenadas que, se $a + b$ for par, então $(a, b) \in S$. Se a soma for 2, então $(a, b) = (1, 1)$, e o passo base põe (a, b) em S . Caso contrário, a soma é pelo menos 4, e pelo menos um entre $(a - 2, b)$, $(a, b - 2)$ e $(a - 1, b - 1)$ deve ter coordenadas inteiras positivas cuja soma é um número par menor que $a + b$ e, portanto, deve estar em S . Então uma aplicação do passo recursivo mostra que $(a, b) \in S$. b) Defina S por $(1, 1), (1, 2)$ e $(2, 1)$ estão em S , e, se $(a, b) \in S$, então $(a + 2, b) \in S$ e $(a, b + 2) \in S$ estão em S . Para demonstrar que nossa definição funciona, observamos primeiro que $(1, 1), (1, 2)$ e $(2, 1)$ têm coordenadas ímpares, e, se (a, b) tiver uma coordenada ímpar, então o mesmo ocorre com $(a + 2, b)$ e $(a, b + 2)$. Reciprocamente, mostramos por indução na soma das coordenadas que, se (a, b) tiver pelo menos uma coordenada ímpar, então $(a, b) \in S$. Se $(a, b) = (1, 1)$ ou $(a, b) = (1, 2)$ ou $(a, b) = (2, 1)$, então o passo base põe (a, b) em S . Caso contrário, a ou b é pelo menos 3, de modo que pelo menos um entre $(a - 2, b)$ e $(a, b - 2)$ deve ter coordenadas inteiras positivas cuja soma é menor que $a + b$ e, portanto, deve estar em S . Então, uma aplicação do passo de recursão mostra que $(a, b) \in S$. c) $(1, 6) \in S$ e $(2, 3) \in S$, e, se $(a, b) \in S$, então $(a + 2, b) \in S$ e $(a, b + 6) \in S$. Para demonstrar que nossa definição funciona, observamos primeiro que $(1, 6)$ e $(2, 3)$ satisfazem a condição, e, se (a, b) satisfizer a condição, então o mesmo ocorre com $(a + 2, b)$ e $(a, b + 6)$. Reciprocamente, mostramos por indução na soma das coordenadas que, se (a, b) satisfizer a condição, então $(a, b) \in S$. Para somas 5 e 7, os únicos pontos são $(1, 6)$, que o passo base põe em S , $(2, 3)$, que o passo base põe em S e $(4, 3) = (2 + 2, 3)$, que está em S por uma aplicação da definição recursiva. Para uma soma maior que 7, ou $a \geq 3$ ou $a \leq 2$ e $b \geq 9$, em cujo caso ou $(a - 2, b)$ ou $(a, b - 6)$ devem ter coordenadas inteiras positivas cuja soma é menor que $a + b$ e satisfaça a condição para estar em S . Então, uma aplicação do passo recursivo mostra que $(a, b) \in S$. 31. Se x for um conjunto ou uma variável que representa um conjunto, então x é uma fórmula bem formada. Se x e y forem fórmulas bem formadas, então o mesmo ocorre com \bar{x} , $(x \cup y)$, $(x \cap y)$ e $(x - y)$. 33. a) Se $x \in D = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, então $m(x) = x$; se $s = tx$, em que $t \in D^*$ e $x \in D$, então $m(s) = \min(m(s), x)$. b) Seja $t = wx$, em que $w \in D^*$ e $x \in D$. Se $w = \lambda$, então $m(st) = m(sx) = \min(m(s), x) = \min(m(s), m(x))$ pelo passo recursivo e pelo passo base da definição de m . Caso contrário, $m(st) = m(sw)x = \min(m(sw), x)$ pela definição de m . Agora, $m(sw) = \min(m(s), m(w))$ pela hipótese de indução da indução estrutural, de modo que $m(st) = \min(\min(m(s), m(w)), x) = \min(m(s), \min(m(w), x))$ pelo significado de \min . Mas $\min(m(w), x) = m(wx) = m(t)$ pelo passo recursivo da definição de m . Logo, $m(st) = \min(m(s), m(t))$. 35. $\lambda^R = \lambda$ e $(ux)^R = xu^R$ para $x \in \Sigma$, $u \in \Sigma^*$. 37. $w^0 = \lambda$ e $w^{n+1} =$

ww^n . 39. Quando a sequência consistir em n 0s seguidos por n 1s para algum inteiro não negativo n . 41. Seja $P(i)$ a afirmação " $l(w^i) = i \cdot l(w)$ ". $P(0)$ é verdadeira porque $l(w^0) = 0 = 0 \cdot l(w)$. Suponha que $P(i)$ seja verdadeira. Então $l(w^{i+1}) = l(ww^i) = l(w) + l(w^i) = l(w) + i \cdot l(w) = (i+1) \cdot l(w)$. 43. *Passo base:* Para a árvore binária completa consistindo apenas na raiz o resultado é verdadeiro porque $n(T) = 1$ e $h(T) = 0$, e $1 \geq 2 \cdot 0 + 1$. *Passo de indução:* Suponha que $n(T_1) \geq 2h(T_1) + 1$ e $n(T_2) \geq 2h(T_2) + 1$. Pelas definições recursivas de $n(T)$ e $h(T)$, temos $n(T) = 1 + n(T_1) + n(T_2)$ e $h(T) = 1 + \max(h(T_1), h(T_2))$. Portanto, $n(T) = 1 + n(T_1) + n(T_2) \geq 1 + 2h(T_1) + 1 + 2h(T_2) + 1 \geq 1 + 2 \cdot \max(h(T_1), h(T_2)) + 2 = 1 + 2(\max(h(T_1), h(T_2)) + 1) = 1 + 2h(T)$. 45. *Passo base:* $a_{0,0} = 0 = 0 + 0$. *Passo de indução:* Suponha que $a_{m',n'} = m' + n'$ sempre que (m',n') for menor que (m,n) na ordem lexicográfica de $\mathbb{N} \times \mathbb{N}$. Se $n = 0$, então $a_{m,n} = a_{m-1,n} + 1 = m - 1 + n + 1 = m + n$. Se $n > 0$, então $a_{m,n} = a_{m,n-1} + 1 = m + n - 1 + 1 = m + n$. 47. a) $P_{m,m} = P_m$ porque um número que excede m não pode ser usado em uma partição de m . b) Como existe apenas uma maneira de obter uma partição de 1, a saber, $1 = 1$, segue que $P_{1,1} = 1$. Como existe apenas uma maneira de obter uma partição de m em 1s, $P_{m,1} = 1$. Quando $n > m$, segue que $P_{m,n} = P_{m,m}$ porque um número que excede m não pode ser usado. $P_{m,m} = 1 + P_{m,m-1}$ porque uma partição extra, a saber, $m = m$, aparece quando m é permitido na partição. $P_{m,n} = P_{m,n-1} + P_{m-n,n}$ se $m > n$, porque uma partição de m em inteiros que não excedam n ou não usa nenhum n , e, portanto, está contada em $P_{m,n-1}$, ou então usa um n e uma partição de $m - n$, e, portanto, está contada em $P_{m-n,n}$. c) $P_5 = 7, P_6 = 11$ 49. Seja $P(n)$ a afirmação " $A(n, 2) = 4$ ". *Passo base:* $P(1)$ é verdadeira porque $A(1, 2) = A(0, A(1, 1)) = A(0, 2) = 2 \cdot 2 = 4$. *Passo de indução:* Suponha que $P(n)$ seja verdadeira, ou seja, $A(n, 2) = 4$. Então $A(n + 1, 2) = A(n, A(n + 1, 1)) = A(n, 2) = 4$. 51. a) 16 b) 65536 53. Use um argumento de indução dupla para demonstrar a afirmação mais forte: $A(m, k) > A(m, l)$ quando $k > l$. *Passo base:* Quando $m = 0$, a afirmação é verdadeira porque $k > l$ implica que $A(0, k) = 2k > 2l = A(0, l)$. *Passo de indução:* Suponha que $A(m, x) > A(m, y)$ para todos os inteiros não negativos x e y com $x > y$. Mostraremos que isto implica que $A(m + 1, k) > A(m + 1, l)$ se $k > l$. *Passo base:* Quando $l = 0$ e $k > 0$, $A(m + 1, l) = 0$ e ou $A(m + 1, k) = 2$ ou $A(m + 1, k) = A(m, A(m + 1, k - 1))$. Se $m = 0$, isto é $2A(1, k - 1) = 2^k$. Se $m > 0$, isto é maior que 0 pela hipótese de indução. Em todos os casos, $A(m + 1, k) > 0$, e, de fato, $A(m + 1, k) \geq 2$. Se $l = 1$ e $k > 1$, então $A(m + 1, l) = 2$ e $A(m + 1, k) = A(m, A(m + 1, k - 1))$, com $A(m + 1, k - 1) \geq 2$. Logo, pela hipótese de indução, $A(m, A(m + 1, k - 1)) \geq A(m, 2) > A(m, 1) = 2$. *Passo de indução:* Suponha que $A(m + 1, r) > A(m + 1, s)$ para todo $r > s$, $s = 0, 1, \dots, l$. Então, se $k + 1 > l + 1$, segue que $A(m + 1, k + 1) = A(m, A(m + 1, k)) > A(m, A(m + 1, k)) = A(m + 1, l + 1)$. 55. Do Exercício 54 segue que $A(i, j) \geq A(i - 1, j) \geq \dots \geq A(0, j) = 2j \geq j$. 57. Seja $P(n)$ a afirmação " $F(n)$ está bem definida". Então $P(0)$ é verdadeira porque $F(0)$ foi especificado. Suponha que $P(k)$ seja verdadeira para todo $k < n$. Então $F(n)$ está bem definida em n porque $F(n)$ é dado em termos de $F(0), F(1), \dots, F(n - 1)$. Assim, $P(n)$ é verdadeira para todos os inteiros n . 59. a) O valor de $F(1)$ é ambíguo. b) $F(2)$

não está definido porque $F(0)$ não está definido. c) $F(3)$ é ambíguo e $F(4)$ não está definido porque $F\left(\frac{4}{3}\right)$ não faz sentido. d) A definição de $F(1)$ é ambígua porque ambas, a segunda e a terceira cláusula, parecem se aplicar. e) $F(2)$ não pode ser calculado porque a tentativa de calcular $F(2)$ fornece $F(2) = 1 + F(F(1)) = 1 + F(2)$. 61. a) 1 b) 2 c) 3 d) 3 e) 4 f) 4 g) 5 63. $f_0^*(n) = \lceil n/a \rceil$ 65. $f_2^*(n) = \lceil \log \log n \rceil$ para $n \geq 2$, $f_2^*(1) = 0$

Seção 4.4

1. Primeiro, usamos o passo recursivo para escrever $5! = 5 \cdot 4!$. Então, usamos o passo recursivo repetidamente para escrever $4! = 4 \cdot 3!, 3! = 3 \cdot 2!, 2! = 2 \cdot 1!$ e $1! = 1 \cdot 0!$. Inserindo o valor de $0! = 1$, e trabalhando de trás para frente nos passos, vemos que $1! = 1 \cdot 1 = 1, 2! = 2 \cdot 1! = 2, 3! = 3 \cdot 2! = 3 \cdot 2 = 6, 4! = 4 \cdot 3! = 4 \cdot 6 = 24$ e $5! = 5 \cdot 4! = 5 \cdot 24 = 120$. 3. Primeiro, como $n = 11$ é ímpar, usamos a cláusula `else` para ver que $mpotencia(3, 11, 5) = (mpotencia(3, 5, 5)^2 \bmod 5 \cdot 3 \bmod 5) \bmod 5$. A seguir, usamos a cláusula `else` novamente para ver que $mpotencia(3, 5, 5) = (mpotencia(3, 2, 5)^2 \bmod 5 \cdot 3 \bmod 5) \bmod 5$. Então, usamos a cláusula `else if` para ver que $mpotencia(3, 2, 5) = mpotencia(3, 1, 5)^2 \bmod 5$. Usando a cláusula `else` novamente, temos $mpotencia(3, 1, 5) = (mpotencia(3, 0, 5)^2 \bmod 5 \cdot 3 \bmod 5) \bmod 5$. Finalmente, usando a cláusula `if`, vemos que $mpotencia(3, 0, 5) = 1$. Trabalhando de trás para frente, segue que $mpotencia(3, 1, 5) = (1^2 \bmod 5 \cdot 3 \bmod 5) \bmod 5 = 3, mpotencia(3, 2, 5) = 3^2 \bmod 5 = 4, mpotencia(3, 5, 5) = (4^2 \bmod 5 \cdot 3 \bmod 5) \bmod 5 = 3$ e, finalmente, $mpotencia(3, 11, 5) = (3^2 \bmod 5 \cdot 3 \bmod 5) \bmod 5 = 2$. Concluímos que $3^{11} \bmod 5 = 2$.

5. Com esta entrada, o algoritmo usa a cláusula `else` para encontrar que $\text{mdc}(8, 13) = \text{mdc}(13 \bmod 8, 8) = \text{mdc}(5, 8)$. Ele usa esta cláusula novamente para determinar que $\text{mdc}(5, 8) = \text{mdc}(8 \bmod 5, 5) = \text{mdc}(3, 5)$, então para obter $\text{mdc}(3, 5) = \text{mdc}(5 \bmod 3, 3) = \text{mdc}(2, 3)$, então $\text{mdc}(2, 3) = \text{mdc}(3 \bmod 2, 2) = \text{mdc}(1, 2)$, e mais uma vez para obter $\text{mdc}(1, 2) = \text{mdc}(2 \bmod 1, 1) = \text{mdc}(0, 1)$. Finalmente, para determinar $\text{mdc}(0, 1)$ ele usa o primeiro passo com $a = 0$ para encontrar que $\text{mdc}(0, 1) = 1$. Consequentemente, o algoritmo determina que $\text{mdc}(8, 13) = 1$.

7. **procedure** *mult* (*n*: inteiro positivo, *x*: inteiro)
 - if** *n* = 1 **then** *mult* (*n*, *x*) := *x*
 - else** *mult* (*n*, *x*) := *x* + *mult* (*n* - 1, *x*)
9. **procedure** *soma de impares* (*n*: inteiro positivo)
 - if** *n* = 1 **then** *soma de impares* (*n*) := 1
 - else** *soma de impares* (*n*) := *soma de impares* (*n* - 1) + 2*n* - 1
11. **procedure** *menor* (*a₁*, ..., *a_n*: inteiros)
 - if** *n* = 1 **then** *menor* (*a₁*, ..., *a_n*) = *a₁*
 - else** *menor* (*a₁*, ..., *a_n*) := *min*(*menor* (*a₁*, ..., *a_{n-1}*), *a_n*)
13. **procedure** *fatorialmod* (*n*, *m*: inteiros positivos)
 - if** *n* = 1 **then** *fatorialmod* (*n*, *m*) := 1
 - else** *fatorialmod* (*n*, *m*) := (*n* · *fatorialmod* (*n* - 1, *m*)) **mod** *m*
15. **procedure** *mdc* (*a*, *b*: inteiros não negativos)
 - {*ab* é suposto válido}

```

if  $a = 0$  then  $\text{mdc}(a, b) := b$ 
else if  $a = b - a$  then  $\text{mdc}(a, b) := a$ 
else if  $a < b - a$  then  $\text{mdc}(a, b) := \text{mdc}(a, b - a)$ 
else  $\text{mdc}(a, b) := \text{mdc}(b - a, a)$ 
17. procedure multiplica ( $x, y$ : inteiros não negativos)
   if  $y = 0$  then multiplica ( $x, y$ ) := 0
   else if  $y$  é par then
      multiplica ( $x, y$ ) :=  $2 \cdot \text{multiplica}(x, y/2)$ 
   else multiplica ( $x, y$ ) :=  $2 \cdot \text{multiplica}(x, (y - 1)/2) + x$ 
19. Usamos indução completa em  $a$ . Passo base: se  $a = 0$ , sabemos que  $\text{mdc}(0, b) = b$  para todo  $b > 0$ , e isto é precisamente o que a cláusula if faz. Passo de indução: Fixe  $k > 0$ , suponha válida a hipótese de indução — de que o algoritmo funciona corretamente para todos os valores de seu primeiro argumento menor que  $k$  — e considere o que acontece com a entrada  $(k, b)$ , em que  $k < b$ . Como  $k > 0$ , a cláusula else é executada, e a resposta é o que o algoritmo der como saída para a entrada  $(b \bmod k, k)$ . Como  $b \bmod k < k$ , o par de entrada é verdadeiro. Por nossa hipótese de indução, esta saída é de fato  $\text{mdc}(b \bmod k, k)$ , que é igual a  $\text{mdc}(k, b)$  pelo Lema 1 da Seção 3.6. 21. Se  $n = 1$ , então  $nx = x$ , e o algoritmo retorna  $x$  corretamente. Suponha que o algoritmo calcule corretamente  $kx$ . Para calcular  $(k + 1)x$ , ele calcula recursivamente o produto de  $k + 1 - 1 = k$  e  $x$ , e então soma  $x$ . Pela hipótese de indução, ele calcula o produto corretamente, de modo que a resposta retornada é  $kx + x = (k + 1)x$ , que é correta.
23. procedure quadrado ( $n$ : inteiro não negativo)
   if  $n = 0$  then quadrado ( $n$ ) := 0
   else quadrado ( $n$ ) := quadrado ( $n - 1$ ) +  $2(n - 1) + 1$ 
Seja  $P(n)$  a afirmação de que este algoritmo calcula corretamente  $n^2$ . Como  $0^2 = 0$ , o algoritmo funciona corretamente (usando a cláusula if) se a entrada for 0. Suponha que o algoritmo funcione corretamente para entrada  $k$ . Então, para entrada  $k + 1$ , ele dá como saída (por causa da cláusula else) sua saída quando a entrada é  $k$ , mais  $2(k + 1 - 1) + 1$ . Pela hipótese de indução, sua saída em  $k$  é  $k^2$ , de modo que sua saída em  $k + 1$  é  $k^2 + 2(k + 1 - 1) + 1 = k^2 + 2k + 1 = (k + 1)^2$ , como desejado. 25.  $n$  multiplicações versus  $2^n$  27.  $O(\log n)$  versus  $n$ 
29. procedure a ( $n$ : inteiro não negativo)
   if  $n = 0$  then a ( $n$ ) := 1
   else if  $n = 1$  then a ( $n$ ) := 2
   else a ( $n$ ) := a ( $n - 1$ ) * a ( $n - 2$ )
31. Iterativo
33. procedure iterativo ( $n$ : inteiro não negativo)
   if  $n = 0$  then  $z := 1$ 
   else if  $n = 1$  then  $z := 2$ 
   else
   begin
       $x := 1$ 
       $y := 2$ 
       $z := 3$ 
      for  $i := 1$  to  $n - 2$  do
         begin
             $w := x + y + z$ 
             $x := y$ 
             $y := z$ 
             $z := w$ 
         end
      end

```

```

end
end

```

{ z é o n -ésimo termo da seqüência}

35. Primeiro damos um procedimento recursivo e então um procedimento iterativo.

```

procedure r ( $n$ : inteiro não negativo)
   if  $n < 3$  then r ( $n$ ) :=  $2n + 1$ 
   else r ( $n$ ) = r ( $n - 1$ ) · r ( $n - 2$ ) $^2$  · r ( $n - 3$ ) $^3$ 

```

```

procedure i ( $n$ : inteiro não negativo)

```

```

   if  $n = 0$  then  $z := 1$ 

```

```

   else if  $n = 1$  then  $z := 3$ 

```

```

   else

```

```

   begin

```

```

       $x := 1$ 

```

```

       $y := 3$ 

```

```

       $z := 5$ 

```

```

      for  $i := 1$  to  $n - 2$  do

```

```

         begin

```

```

             $w := z * y^2 * x^3$ 

```

```

             $x := y$ 

```

```

             $y := z$ 

```

```

             $z := w$ 

```

```

         end

```

```

      end

```

{ z é o n -ésimo termo da seqüência}

A versão iterativa é mais eficiente.

37. **procedure** *inverte* (w : seqüência de bits)

```

    $n := \text{comprimento}(w)$ 

```

```

   if  $n \leq 1$  then inverte ( $w$ ) :=  $w$ 

```

```

   else inverte ( $w$ ) :=

```

```

      subseq ( $w, n, n$ ) inverte (subseq ( $w, 1, n - 1$ ))

```

{*subseq* (w, a, b) é a sub-seqüência de w que consiste nos símbolos da posição a até a posição b }

39. O procedimento dá corretamente o inverso de λ como λ (passo base), e, como o inverso de uma seqüência consiste em seu último caractere seguido pelo inverso de seus primeiros $n - 1$ caracteres (veja o Exercício 35 da Seção 4.3), o algoritmo se comporta corretamente quando $n > 0$, pela hipótese de indução. 41. O algoritmo implementa a idéia do Exemplo 13 da Seção 4.1. Se $n = 1$ (passo base), coloque o único triominó direito de modo que o quadrado entre suas pontas corresponda ao buraco no tabuleiro 2×2 . Se $n > 1$, então divida o tabuleiro em quatro tabuleiros, cada um de tamanho $2^{n-1} \times 2^{n-1}$, observe em qual quarto ocorre o buraco, posicione um triominó direito no centro do tabuleiro com o quadrado entre suas pontas no quarto onde estiver o quadrado faltando (veja a Figura 8 na Seção 4.1) e chame o algoritmo recursivamente quatro vezes — uma para cada tabuleiro $2^{n-1} \times 2^{n-1}$, cada um dos quais tem um quadrado faltando (ou porque ele estava faltando no começo, ou porque ele foi coberto pelo triominó central).

43. **procedure** *A* (m, n : inteiros não negativos)

```

   if  $m = 0$  then A ( $m, n$ ) :=  $2n$ 

```

```

   else if  $n = 0$  then A ( $m, n$ ) := 0

```

```

   else if  $n = 1$  then A ( $m, n$ ) := 2

```