2

Demonstrações, Recorrência e Análise de Algoritmos

Objetivos do Capítulo

Após o estudo deste capítulo, você será capaz de:

- Atacar as demonstrações de conjecturas usando as técnicas de demonstração direta, demonstração por contraposição e demonstração por absurdo.
- Reconhecer quando uma demonstração por indução é apropriada e fazê-la usando o primeiro ou o segundo princípio de indução.
- Demonstrar matematicamente a correção de programas usando proposições em laço.
- Compreender definições recorrentes de seqüências, coleções de objetos e operações sobre objetos.
- Escrever definições recorrentes para determinadas seqüências, coleções de objetos e operações sobre objetos.
- Compreender como os algoritmos recorrentes funcionam.
- Escrever algoritmos recorrentes para gerar seqüências definidas recorrentemente.
- Encontrar soluções em forma fechada para determinadas relações de recorrência encontradas na análise de algoritmos.

Você está servindo no Conselho Municipal de Obras, que está considerando uma proposta de uma firma para gerenciar a eliminação de material químico. O material a ser estocado decai a uma taxa de 5% ao ano. O empreiteiro afirma que, a essa taxa, apenas aproximadamente um terço do material ativo original restará ao final de 20 anos.

Pergunta: Essa estimativa está correta?

É possível verificar esta estimativa através de cálculos extensos: se existe uma certa quantidade inicialmente, então sobrará um tanto no próximo ano, outro tanto no ano seguinte e assim por diante, até completar os 20 anos. Mas pode-se obter uma solução rápida e elegante resolvendo-se uma relação de recorrência; as relações de recorrência são discutidas na Seção 2.4.

Primeiro, no entanto, vamos ver como provar argumentos "do mundo real", em vez de argumentos formais como no Cap. 1. Vai ser útil ter um arsenal de técnicas para atacar uma demonstração. Demonstrações diretas, por contraposição e por absurdo são examinadas na Seção 2.1. A Seção 2.2 trata de indução matemática, uma técnica de demonstração com muitas aplicações em ciência da computação. Na Seção 2.3 veremos como podemos estender a demonstração de correção a proposições envolvendo laços, usando indução.

A Seção 2.4 discute relações de recorrência, que estão intimamente relacionadas à indução matemática e que são importantes para expressar muitas definições e até algoritmos. Algumas seqüências definidas de maneira recorrente também podem ser definidas através de uma fórmula; encontrar tal fórmula envolve a solução de uma relação de recorrência e a utilização de indução para verificar a correção da fórmula. O uso de relações de recorrência para determinar a quantidade de operações que um algoritmo particular tem que fazer é explorado na Seção 2.5.

Seção 2.1 Técnicas de Demonstração

Teoremas e Demonstrações Informais

Os argumentos formais do Cap. 1 têm a forma $P \rightarrow Q$, onde P e Q podem representar proposições compostas. Lá, o ponto era demonstrar a validade de um argumento — verdadeiro para todas as interpretações possíveis devido à natureza interna de sua forma ou estrutura, e não por causa de seu conteúdo ou do significado de suas componentes. No entanto, muitas vezes queremos provar argumentos que não são universalmente verdadeiros mas que são verdadeiros em determinados contextos. O significado torna-se importante porque estamos discutindo um assunto particular — grafos de algoritmos, ou álgebra de Boole, ou compiladores, ou qualquer outra coisa — e queremos provar que, se P é verdadeiro nesse contexto, então Q também o é. Se pudermos fazer isso, então $P \rightarrow Q$ torna-se um *teorema* sobre aquele assunto. Para provar um teorema sobre o assunto XXX, podemos inserir fatos sobre XXX na demonstração; esses fatos agem como hipóteses adicionais.

Pode não ser fácil reconhecer quais fatos específicos sobre o assunto são relevantes, ou arrumar uma seqüência de demonstração que nos leve logicamente de P a Q. Infelizmente, não existe uma fórmula para a construção de demonstrações e não existe algoritmo geral prático ou programa de computador para provar teoremas. A experiência ajuda, não apenas porque você vai melhorando com a prática, mas também porque uma demonstração que funciona para um teorema pode ser algumas vezes modificada para funcionar para um outro teorema semelhante.

Os teoremas podem ser, muitas vezes, enunciados e demonstrados de maneira menos formal do que usando argumentos de lógica proposicional e de predicados como no Cap. 1. Por exemplo, um teorema pode expressar o fato de que todos os objetos em um domínio de interpretação (o assunto em consideração) que tenham a propriedade P também têm a propriedade Q. A proposição formal desse teorema seria $(\forall x)[P(x) \rightarrow Q(x)]$ mas o teorema poderia ser enunciado informalmente como $P(x) \rightarrow Q(x)$. Se pudermos provar que $P(x) \rightarrow Q(x)$, onde x é considerado um elemento arbitrário do conjunto universo, a generalização universal implica, então, em $(\forall x)[P(x) \rightarrow Q(x)]$.

Como outro exemplo, poderíamos saber que todos os objetos no conjunto universo têm alguma propriedade, isto é, algo da forma $(\forall x)P(x)$ pode ser considerado como um fato específico sobre o assunto em questão. Uma demonstração informal poderia incluir, então, sentenças como "Seja x um elemento arbitrário do conjunto universo. Então x tem a propriedade P". (Formalmente, estamos usando a particularização universal para obter $P(x)$ de $(\forall x)P(x)$.

Da mesma forma, demonstrações normalmente não são escritas de uma vez com justificativas formais para cada passo. Em vez disso, os passos importantes e suas razões são esboçados na forma de narrativa. Tal narrativa, entretanto, pode ser traduzida em uma demonstração formal, se necessário. Na verdade, o valor de uma demonstração formal é que ela serve como um tipo de seguro — se uma demonstração em narrativa não puder ser traduzida em uma demonstração formal, ela deve ser vista com grande suspeita.

Provar ou Não Provar

Um livro texto contém, muitas vezes, frases como "Prove o seguinte teorema" e o leitor sabe, então, que o teorema é verdade; além disso, ele está provavelmente enunciado na sua forma mais sofisticada. Mas suponha que você está pesquisando um determinado assunto e observa diversos casos onde, sempre que P é verdade, Q também o é. Baseado nessas experiências, você pode formular uma conjectura: $P \rightarrow Q$. Quanto mais casos você encontra onde Q segue de P, mais confiante você se sente de que sua conjectura é verdadeira. Esse processo ilustra o **raciocínio indutivo**, concluir algo baseado na experiência.

Independentemente de quão razoável pareça ser sua conjectura, no entanto, você não vai ficar satisfeito até aplicar um **raciocínio dedutivo**, também, à sua conjectura. Neste processo, você tenta verificar se sua conjectura é verdadeira ou falsa. Então você produz uma demonstração de que $P \rightarrow Q$ (transformando-a em teorema) ou encontra um **contra-exemplo**, mostrando que a conjectura está errada, com um caso onde P é verdadeiro e Q é falso. (Estávamos usando um raciocínio dedutivo em lógica de predicados quando ou provávamos que uma fbf era válida ou encontrávamos uma interpretação para a qual ela era falsa.)

Se você for simplesmente apresentado com uma conjectura, pode ser difícil decidir qual das duas abordagens tentar — tentar prová-la ou procurar um contra-exemplo! Um único contra-exemplo é suficiente para provar a falsidade. É claro que, se a procura por um contra-exemplo não tiver sucesso, isto não significa que a conjectura seja verdadeira.

LEMBRETE:
Um contra-exemplo é suficiente para provar que uma conjectura é falsa.

EXEMPLO 1

Para um inteiro positivo n, n **fatorial** é definido como sendo $n(n-1)(n-2)\ldots 1$ e denotado por $n!$. Prove ou encontre um contra-exemplo para a conjectura "Para todo inteiro positivo n, $n! \leq n^2$".

Vamos começar testando alguns casos:

n	$n!$	n^2	$n! \leq n^2$
1	1	1	sim
2	2	4	sim
3	6	9	sim

Até agora essa conjectura parece boa. Mas o próximo caso,

n	$n!$	n^2	$n! \leq n^2$
4	24	16	não

mostra um contra-exemplo. O fato de que a conjectura é verdadeira para $n = 1, 2$ e 3 não prova nada sobre a conjectura mas o caso $n = 4$ é suficiente para provar sua falsidade. ●

PROBLEMA PRÁTICO 1

Dê contra-exemplos para as seguintes conjecturas:

a. Todos os animais vivendo no oceano são peixes.
b. Todo inteiro menor do que 10 é maior do que 5. ●

Se está difícil encontrar um contra-exemplo, que técnicas podemos tentar usar para provar uma conjectura? No resto desta seção vamos examinar diversos métodos para fazer uma demonstração.

Demonstração Exaustiva

Embora "provar a falsidade por um contra-exemplo" sempre funcione, "provar por um exemplo" quase nunca funciona. Uma exceção ocorre quando a conjectura é uma asserção sobre uma coleção finita. Nesse caso, a conjectura pode ser provada verificando-se que ela é verdadeira para cada elemento da coleção. Uma **demonstração por exaustão** significa que foram exauridos todos os casos possíveis, embora, com freqüência, signifique que a pessoa fazendo a demonstração também esteja exaurida!

EXEMPLO 2

Prove a conjectura "Se um inteiro entre 1 e 20 é divisível por 6, então é também divisível por 3". ("Divisibilidade por 6" significa "divisibilidade inteira por 6", isto é, que o número é um múltiplo inteiro de 6.)

Como existe apenas um número finito de casos, a conjectura pode ser provada simplesmente mostrando-se que é verdadeira para todos os inteiros entre 1 e 20. A demonstração é a Tabela 2.1. ●

Número	Divisível por 6	Divisível por 3
1	não	
2	não	
3	não	
4	não	
5	não	
6	sim: $6 = 1 \times 6$	sim: $6 = 2 \times 3$
7	não	
8	não	
9	não	
10	não	

Número	Divisível por 6	Divisível por 3
11	não	
12	sim: $12 = 2 \times 6$	sim: $12 = 4 \times 3$
13	não	
14	não	
15	não	
16	não	
17	não	
18	sim: $18 = 3 \times 6$	sim: $18 = 6 \times 3$
19	não	
20	não	

Tabela 2.1

EXEMPLO 3

Prove a conjectura "Não é possível traçar todas as retas na Fig. 2.1 sem levantar o lápis e sem redesenhar alguma reta".

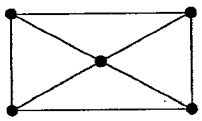


Fig. 2.1

Existe apenas um número finito de maneiras de traçar as retas na figura. Fazendo anotações cuidadosas, cada uma das possibilidades pode ser tentada e cada uma delas vai falhar. No Cap. 6 aprenderemos um método para resolver este problema que é menos tedioso do que o da exaustão. ●

PROBLEMA PRÁTICO 2

a. Prove a conjectura "Para qualquer inteiro positivo menor ou igual a 5, o quadrado deste inteiro é menor ou igual à soma de 10 mais 5 vezes o inteiro".
b. Dê um contra-exemplo para a conjectura "Para qualquer inteiro positivo, o quadrado deste inteiro é menor ou igual à soma de 10 mais 5 vezes o inteiro". ●

Demonstração Direta

Em geral (quando a demonstração por exaustão não funciona), como você pode provar que $P \rightarrow Q$ é verdadeiro? A abordagem óbvia é a **demonstração direta** — suponha a hipótese P e deduza a conclusão Q. Uma demonstração formal necessitaria de uma seqüência de demonstração partindo de P e chegando a Q.

EXEMPLO 4

Considere a conjectura

$(\forall x)(\forall y)(x$ é um inteiro par $\wedge\ y$ é um inteiro par $\rightarrow$ o produto xy é um inteiro par)

Uma seqüência de demonstração formal completa poderia ser do seguinte tipo:

1. $(\forall x)[x$ é um inteiro par $\rightarrow (\exists k)(k$ é um inteiro $\wedge\ x = 2k)]$	fato sobre o número (definição de inteiro par)
2. x é um inteiro par $\rightarrow (\exists k)(k$ é um inteiro $\wedge\ x = 2k)$	1, pu
3. y é um inteiro par $\rightarrow (\exists k)(k$ é um inteiro $\wedge\ y = 2k)$	1, pu
4. x é um inteiro par $\wedge\ y$ é um inteiro par	hip temporária
5. x é um inteiro par	4, simp
6. $(\exists k)(k$ é um inteiro $\wedge\ x = 2k)$	2, 5, mp
7. m é um inteiro $\wedge\ x = 2m$	6, pe
8. y é um inteiro par	4, simp
9. $(\exists k)(k$ é um inteiro $\wedge\ y = 2k)$	3, 8, mp
10. n é um inteiro e $y = 2n$	9, pe
11. $x = 2m$	7, simp
12. $y = 2n$	10, simp
13. $xy = (2m)(2n)$	11, 12, substituição de iguais
14. $xy = 2(2mn)$	13, fato sobre a multiplicação
15. m é um inteiro	7, simp
16. n é um inteiro	10, simp
17. $2mn$ é um inteiro	15, 16, fatos sobre os números
18. $xy = 2(2mn) \wedge 2mn$ é um inteiro	14, 17, conj
19. $(\exists k)(k$ é um inteiro $\wedge\ xy = 2k)$	18, ge
20. $(\forall x)((\exists k)(k$ é um inteiro $\wedge\ x = 2k) \rightarrow x$ é um inteiro par)	fato sobre os números (definição de inteiro par)
21. $(\exists k)(k$ é um inteiro $\wedge\ xy = 2k) \rightarrow xy$ é um inteiro par	20, pu
22. xy é um inteiro par	19, 21, mp
23. x é um inteiro par $\wedge\ y$ é um inteiro par $\rightarrow xy$ é um inteiro par	retirada da hip temporária
24. $(\forall x)(\forall y)$ (x é um inteiro par $\wedge\ y$ é um inteiro par $\rightarrow$ o produto xy é um inteiro par)	23, gu duas vezes ●

Nunca mais faremos uma tal demonstração de novo, nem você vai precisar fazer isto! Uma demonstração muito mais informal seria perfeitamente aceitável na maior parte dos casos.

EXEMPLO 5

Segue uma prova informal direta do fato de que o produto de dois inteiros pares é par. Sejam $x = 2m$ e $y = 2n$, onde m e n são inteiros. Então $xy = (2m)(2n) = 2(2mn)$, onde $2mn$ é um inteiro. Logo, xy tem a forma $2k$, onde k é um inteiro e, portanto, é par. ●

A demonstração no Exemplo 5 não enuncia explicitamente a hipótese (que x e y são pares) e usa, implicitamente, a definição de inteiro par. Mesmo em uma demonstração informal, no entanto, é importante identificar a hipótese e a conclusão, não apenas em palavras, mas o que elas realmente significam, aplicando as definições apropriadas. Se não compreendemos claramente o que temos (a hipótese) ou o que queremos (a conclusão), não podemos esperar construir uma ponte de uma para outra. É por isso que é importante conhecer as definições.

PROBLEMA PRÁTICO 3 Dê uma demonstração direta (informal) do teorema "Se um inteiro é divisível por 6, então duas vezes este inteiro é divisível por 4". ●

Contraposição

Se você tentou, diligentemente, produzir uma demonstração direta da conjectura $P \rightarrow Q$, não conseguiu, mas ainda acha que a conjectura é verdadeira, você pode tentar algumas variantes da técnica de demonstração direta. Se você puder provar o teorema $Q' \rightarrow P'$, pode concluir que $P \rightarrow Q$ usando a tautologia $(Q' \rightarrow P') \rightarrow (P \rightarrow Q)$. $Q' \rightarrow P'$ é a **contrapositiva** de $P \rightarrow Q$, e a técnica de provar $P \rightarrow Q$ através de uma demonstração direta de $Q' \rightarrow P'$ é chamada uma **demonstração por contraposição** (ou **demonstração indireta pela contrapositiva**). (A regra de inferência de contraposição na lógica proposicional, Tabela 1.14, diz que $P \rightarrow Q$ pode ser deduzida de $Q' \rightarrow P'$.)

EXEMPLO 6 Prove que, se o quadrado de um inteiro é ímpar, então o inteiro tem que ser ímpar.

A conjectura é n^2 ímpar $\rightarrow n$ ímpar. Vamos fazer uma demonstração por contraposição e provar que n par $\rightarrow n^2$ par. Seja n par. Então $n^2 = n(n)$ é par pelo Exemplo 5. ●

EXEMPLO 7 Prove que, se $n + 1$ senhas diferentes forem distribuídas a n alunos, então algum aluno recebe ≥ 2 senhas.

A contrapositiva é "Se todo aluno recebe < 2 senhas, então não foram distribuídas $n + 1$ senhas". Suponha que todo aluno recebe < 2 senhas; então cada um dos n alunos tem, no máximo, 1 senha. O número total de senhas é, no máximo, n, e não $n + 1$. ●

O Exemplo 7 é uma ilustração do Princípio das Casas de Pombo, que veremos no Cap. 3.

PROBLEMA PRÁTICO 4 Escreva a contrapositiva de cada proposição no Problema Prático 5 do Cap. 1. ●

O Problema Prático 7 no Cap. 1 mostrou que as fbfs $A \rightarrow B$ e $B \rightarrow A$ não são equivalentes. $B \rightarrow A$ é a **recíproca** de $A \rightarrow B$. Se um condicional é verdadeiro, sua recíproca pode ser verdadeira ou falsa. Portanto, você não pode provar $P \rightarrow Q$ considerando $Q \rightarrow P$.

EXEMPLO 8 A implicação "Se $a > 5$, então $a > 2$" é verdadeira mas sua recíproca, "se $a > 2$, então $a > 5$", é falsa. ●

PROBLEMA PRÁTICO 5 Escreva a recíproca de cada proposição no Problema Prático 5 do Cap. 1. ●

LEMBRETE: "Se e somente se" necessita de duas demonstrações, uma em cada direção.

Teoremas são enunciados, muitas vezes, na forma "P se, e somente se, Q", o que significa que P se Q e P só se Q, ou seja, $Q \rightarrow P$ e $P \rightarrow Q$. Para provar tal teorema, precisamos demonstrar um condicional e sua recíproca. Novamente, a verdade de um não implica na verdade do outro.

EXEMPLO 9 Prove que o produto xy é ímpar se, e somente se, ambos x e y são inteiros ímpares.

Vamos provar primeiro que, se x e y forem ímpares, então xy também o é. Uma demonstração direta vai funcionar. Suponha que tanto x quanto y são ímpares. Então $x = 2n + 1$ e $y = 2m + 1$, onde m e n são inteiros. Logo, $xy = (2n + 1)(2m + 1) = 4nm + 2m + 2n + 1 = 2(2nm + m + n) + 1$. Isso tem a forma $2k + 1$, onde k é um inteiro, portanto xy é ímpar.

A seguir, vamos provar que, se xy é ímpar, então ambos, x e y, têm que ser ímpares, ou

xy ímpar $\rightarrow x$ ímpar e y ímpar

Uma demonstração direta começaria com a hipótese de que xy é ímpar, o que não nos deixa muito a dizer. Uma demonstração por contraposição vai funcionar melhor, pois vamos ter informações mais úteis como hipóteses. Vamos provar, então, que

(x ímpar e y ímpar)' $\rightarrow$ (xy ímpar)'

Pela lei de De Morgan, $(A \wedge B)' \Leftrightarrow A' \vee B'$, vemos que essa expressão pode ser escrita na forma

x par ou y par $\rightarrow$ xy par (1)

A hipótese "x par ou y par" pode ser quebrada em três casos. Vamos considerar cada um deles.

1. x par, y ímpar: então $x = 2m$ e $y = 2n + 1$, logo $xy = (2m)(2n + 1) = 2(2mn + m)$, que é par.
2. x ímpar, y par: funciona como no primeiro caso.
3. x par, y par: então xy é par pelo Exemplo 5.

Isso completa a demonstração de (1) e, portanto, do teorema. ●

A segunda parte da demonstração do Exemplo 9 usa uma **demonstração por casos**, uma forma de demonstração por exaustão. Ela envolve a identificação de todos os casos possíveis consistentes com a informação dada e a posterior demonstração de cada caso separadamente.

Demonstração por Absurdo

Além da demonstração direta e da demonstração por contraposição, você pode usar a técnica de **demonstração por absurdo**. (Uma demonstração por absurdo é chamada, algumas vezes, de *demonstração indireta*, mas esse termo é mais apropriado para qualquer argumento que não seja uma demonstração direta.) Como no Cap. 1, denotaremos qualquer contradição por 0, isto é, qualquer fbf cujo valor lógico é sempre falso. (Um exemplo de uma fbf deste tipo é $A \wedge A'$.) Mais uma vez, vamos supor que estamos tentando provar $P \rightarrow Q$. Construindo uma tabela-verdade, vemos que

$(P \wedge Q' \rightarrow 0) \rightarrow (P \rightarrow Q)$

é uma tautologia, logo, para provar o teorema $P \rightarrow Q$, basta provar que $P \wedge Q' \rightarrow 0$. Portanto, em uma demonstração por absurdo, supomos que a hipótese e a negação da conclusão são ambas verdadeiras e tentamos deduzir uma contradição dessas proposições.

EXEMPLO 10

Vamos demonstrar por absurdo a proposição "Se um número somado a ele mesmo é igual a ele mesmo, então este número é 0". Vamos representar por x um número qualquer. A hipótese é que $x + x = x$ e a conclusão é que $x = 0$. Para demonstrar por absurdo, suponha que $x + x = x$ e $x \neq 0$. Então, $2x = x$ e $x \neq 0$. Como $x \neq 0$, podemos dividir ambos os lados da equação $2x = x$ por x, obtendo $2 = 1$, uma contradição. Logo, $(x + x = x) \rightarrow (x = 0)$. ●

LEMBRETE:
Para provar que alguma coisa não é verdadeira, tente uma demonstração por absurdo.

Apesar do Exemplo 10, pensamos imediatamente em uma demonstração por absurdo quando queremos provar que alguma coisa *não é* verdade. É difícil provar que alguma coisa *não é verdade*; é muito mais fácil *supor que é verdade* e obter uma contradição.

EXEMPLO 11

Uma demonstração por absurdo bem conhecida é a de que $\sqrt{2}$ não é um número racional. Lembre-se de que um **número racional** é um número que pode ser colocado na forma p/q, onde p e q são inteiros, $q \neq 0$ e p e q não têm fatores comuns (exceto $\pm$ 1).

Vamos supor que $\sqrt{2}$ é racional. Então $\sqrt{2} = p/q$ e $2 = p^2/q^2$, ou seja, $2q^2 = p^2$. Então, 2 divide p^2, logo — como o próprio 2 é indivisível — 2 tem que dividir p. Isto significa que 2 é um fator de p, donde 4 é um fator de p^2 e a equação $2q^2 = p^2$ pode ser escrita como $2q^2 = 4x$ ou $q^2 = 2x$. Vemos desta equação que 2 divide q^2, logo 2 divide q. Mas, então, 2 é um fator de q e de p, o que contradiz a declaração de que p e q não têm fatores comuns. Portanto, $\sqrt{2}$ não é racional. ●

A demonstração do Exemplo 11 envolve mais do que simples manipulações algébricas. Muitas vezes é necessário usar muitas palavras em uma demonstração.

PROBLEMA PRÁTICO 6

Demonstre por absurdo que o produto de inteiros ímpares não é par. (Fizemos uma demonstração direta de uma proposição equivalente no Exemplo 9.) ●

A demonstração por absurdo pode ser uma técnica muito útil, mas é fácil pensar que fizemos tal demonstração quando, de fato, não fizemos. Por exemplo, suponha que admitimos $P \wedge Q'$ e somos capazes de deduzir Q sem usar Q'. Dizemos, então que $Q \wedge Q'$ é uma contradição. Mas o que realmente aconteceu aqui foi uma demonstração direta do fato $P \rightarrow Q$, e a demonstração deveria ser reescrita nesta forma. Assim, no Exemplo 10, poderíamos supor, como antes, que $x + x = x$ e que $x \neq 0$. Poderíamos argumentar, então, que de $x + x = x$ obtemos $2x = x$ e, subtraindo x de ambos os lados, $x = 0$. Temos, então, $x = 0$ e $x \neq 0$, uma contradição. No entanto, nunca utilizamos a hipótese $x \neq 0$ neste argumento; de fato, provamos diretamente que $x + x = x$ implica $x = 0$.

Uma outra afirmação enganadora de demonstração por absurdo ocorre quando supomos $P \wedge Q'$ e deduzimos P' sem utilizar a hipótese P. Afirmamos, então, que $P \wedge P'$ é uma contradição. O que realmente aconteceu aqui foi uma demonstração direta da contrapositiva, $Q' \rightarrow P'$, e construímos uma demonstração por contraposição e não por absurdo. Tanto neste caso como no anterior, não é que as demonstrações estejam erradas, é que simplesmente não são demonstrações por absurdo.

A Tabela 2.2 resume as técnicas de demonstração úteis que discutimos até agora.

Técnica de Demonstração	Abordagem para Provar $P \rightarrow Q$	Observações
Demonstração por Exaustão	Demonstre $P \rightarrow Q$ para todos os casos possíveis.	Pode ser usada apenas para demonstrar um número finito de casos.
Demonstração Direta	Suponha P, deduza Q.	Abordagem padrão — o que se deve tentar, em geral.
Demonstração por Contraposição	Suponha Q', deduza P'.	Use esta técnica se Q' parecer dar mais munição do que P.
Demonstração por Absurdo	Suponha $P \wedge Q'$, deduza uma contradição.	Use esta técnica quando Q disser que alguma coisa não é verdade.

Tabela 2.2

Acidentes Felizes

Algumas vezes temos sorte e fazemos, acidentalmente, descobertas inesperadas. Embora isso não seja, de fato, uma técnica geral de demonstração, algumas das demonstrações mais interessantes são geradas por observações engenhosas que podemos admirar, mesmo que nunca fôssemos capazes de fazer tal demonstração por conta própria. Vamos considerar duas dessas demonstrações, só para nos divertir.

EXEMPLO 12

Um torneio de tênis tem 342 jogadores. Uma única partida envolve dois jogadores. O vencedor de uma partida vai jogar com o vencedor de uma outra partida na próxima rodada, enquanto os perdedores são eliminados do torneio. Os 2 jogadores que venceram todas as partidas nas rodadas anteriores vão jogar na final e o vencedor ganha o torneio. Prove que o número total de partidas que serão jogadas é 341.

A maneira trabalhosa de provar esse resultado é calcular $342/2 = 171$ para obter o número de partidas na primeira rodada, resultando em 171 vencedores que vão para a próxima rodada. Para a segunda rodada, temos $171/2 = 85$ mais 1 que sobra; teremos 85 partidas e 85 vencedores, mais o que sobrou, para a terceira rodada. A terceira rodada tem $86/2 = 43$ partidas e assim por diante. O número total de partidas é a soma $171 + 85 + 43 + \ldots$.

A observação engenhosa é notar que cada partida resulta em exatamente 1 perdedor, de modo que o número de partidas é igual ao número de perdedores no torneio. Como existe apenas 1 ganhador, existem 341 perdedores, e, portanto, são jogadas 341 partidas. ●

EXEMPLO 13

Um tabuleiro de damas ou de xadrez padrão consiste em 8 fileiras de 8 quadrados cada. Quadrados adjacentes têm cores alternadas, branco e preto (ou vermelho e preto). Um conjunto de 32 ladrilhos 1×2, cada um cobrindo 2 quadrados, cobrem o tabuleiro completamente (4 ladrilhos por fileira, 8 fileiras). Prove que, se os quadrados nos cantos diagonalmente opostos do tabuleiro forem removidos, o que resta do tabuleiro não pode ser coberto com 31 ladrilhos.

A maneira trabalhosa de provar esse resultado é tentar todas as possibilidades com 31 ladrilhos e verificar que todas falham. A observação engenhosa é notar que os cantos opostos têm a mesma cor, de modo que o tabuleiro com os cantos removidos tem dois quadrados a menos da mesma cor. Cada ladrilho cobre um quadrado de cada cor, de modo que qualquer conjunto de ladrilhos tem que cobrir um número igual de quadrados de cada cor e não pode cobrir o tabuleiro com os cantos faltando. ●

Seção 2.1 Revisão

Técnicas

- Procura por um contra-exemplo.
- Construção de demonstrações diretas, por contraposição e por absurdo.

Idéias Principais

O raciocínio indutivo é usado para formular uma conjectura baseada na experiência.

O raciocínio dedutivo é usado para refutar uma conjectura, através de um contra-exemplo, ou para prová-la.

Ao provar uma conjectura sobre algum assunto, pode-se usar fatos sobre o assunto.

Em condições adequadas, uma demonstração por contraposição ou por absurdo pode funcionar melhor do que uma demonstração direta.

Exercícios 2.1

As definições a seguir podem ser úteis em alguns dos exercícios.

- Um *quadrado perfeito* é um inteiro n da forma $n = k^2$ para algum inteiro k.
- Um *número primo* é um inteiro $n > 1$ que não é divisível por nenhum inteiro positivo diferente de 1 e de n.
- Dados dois números x e y, $x < y$ significa $y - x > 0$.
- O *valor absoluto* ou *módulo* de um número x, $|x|$, é igual a x se $x \geq 0$ e é igual a $-x$ se $x < 0$.

★**1.** Escreva a recíproca e a contrapositiva de cada proposição no Exercício 4 da Seção 1.1.

2. Dê contra-exemplos para as proposições a seguir.
- **a.** Toda figura geométrica com quatro ângulos retos é um quadrado.
- **b.** Se um número real não é positivo, então tem que ser negativo.
- **c.** Todas as pessoas com cabelo ruivo têm olhos verdes ou são altas.
- **d.** Todas as pessoas com cabelo ruivo têm olhos verdes e são altas.

Nos Exercícios 3 a 28, prove a proposição dada.

★**3.** Se $n = 25$, 100 ou 169, então n é um quadrado perfeito e é uma soma de dois quadrados perfeitos.

4. Se n é um inteiro par, $4 \leq n \leq 12$, então n é uma soma de dois números primos.

5. Para qualquer inteiro positivo n menor ou igual a 3, $n! < 2^n$.

6. Para $2 \leq n \leq 4$, $n^2 \geq 2^n$.

7. A soma de dois inteiros pares é par (faça uma demonstração direta).

8. A soma de dois inteiros pares é par (faça uma demonstração por absurdo).

★**9.** A soma de dois inteiros ímpares é par.

10. A soma de um inteiro par com um inteiro ímpar é ímpar.

11. O produto de quaisquer dois inteiros consecutivos é par.

12. A soma de um inteiro com seu quadrado é par.

★**13.** O quadrado de um número par é divisível por 4.

14. Para todo inteiro n, o número

$$3(n^2 + 2n + 3) - 2n^2$$

é um quadrado perfeito.

15. Se um número x é positivo, $x + 1$ também é (faça uma demonstração por contraposição).

16. O número n é um inteiro ímpar se, e somente se, $3n + 5$ é um inteiro par.

★**17.** Para x e y números positivos, $x < y$ se, e somente se, $x^2 < y^2$.

18. Se $x^2 + 2x - 3 = 0$, então $x \neq 2$.

19. Se x é um número primo par, então $x = 2$.

★**20.** Se dois inteiros são divisíveis por algum inteiro n, então sua soma é divisível por n.

21. Se o produto de dois inteiros não é divisível por um inteiro n, então nenhum dos inteiros é divisível por n.

22. A soma de três inteiros consecutivos é divisível por 3.

★**23.** O quadrado de um inteiro ímpar é igual a $8k + 1$ para algum inteiro k.

24. A diferença entre dois cubos consecutivos é ímpar.

25. A soma dos quadrados de dois inteiros ímpares não pode ser um quadrado perfeito. (*Dica*: Use o Exercício 23.)

★26. O produto dos quadrados de dois inteiros é um quadrado perfeito.

27. Dados dois números quaisquer x e y, $|xy| = |x||y|$.

28. Dados dois números quaisquer x e y, $|x + y| \leq |x| + |y|$.

29. O valor A é a média dos n números $x_1, x_2, \ldots, x_n$. Prove que pelo menos um dos $x_1, x_2, \ldots, x_n$ é maior ou igual a A.

30. Suponha que você tentou usar os passos do Exemplo 11 para tentar provar que $\sqrt{4}$ não é um número racional. Em que lugar sua demonstração falharia?

31. Prove que $\sqrt{3}$ não é um número racional.

32. Prove que $\sqrt{5}$ não é um número racional.

33. Prove que $\sqrt[3]{2}$ não é um número racional.

Nos Exercícios 34 a 46, prove a proposição dada ou prove que ela é falsa.

★34. O produto de quaisquer três inteiros consecutivos é par.

35. A soma de quaisquer três inteiros consecutivos é par.

36. O produto de um inteiro e seu quadrado é par.

★37. A soma de um inteiro e seu cubo é par.

38. Qualquer inteiro positivo pode ser escrito como a soma dos quadrados de dois inteiros.

39. Para um inteiro positivo x, $x + \dfrac{1}{x} \geq 2$.

40. Para todo número primo n, $n + 4$ é primo.

41. Para todo inteiro positivo n, $2^n + 1$ é primo.

42. Para n um inteiro par, $n > 2$, $2^n - 1$ não é primo.

43. O produto de dois números racionais é racional.

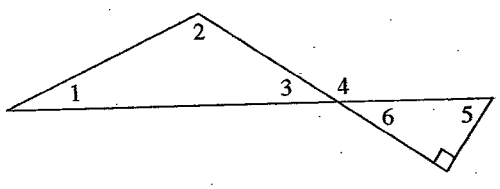
★44. A soma de dois números racionais é racional.

45. O produto de dois números irracionais é irracional.

46. A soma de um número racional com um número irracional é irracional.

Para os Exercícios 47 a 49, use a figura abaixo e os seguintes fatos de geometria:

- A soma dos ângulos internos de um triângulo é 180°.
- Ângulos opostos formados pela interseção de duas retas são iguais.
- Um ângulo onde as duas semi-retas que o formam estão contidas na mesma reta, mas têm sentidos opostos, mede 180°.
- Um ângulo reto mede 90°.



47. Prove que a medida do ângulo 4 é a soma das medidas dos ângulos 1 e 2.

★48. Prove que a medida do ângulo 5 mais a medida do ângulo 3 é 90°.

49. Se os ângulos 1 e 5 são iguais, então o ângulo 2 é reto.

50. Prove que a soma dos inteiros de 1 a 100 é 5050. (*Dica*: Ao invés de somar todos esses números, tente fazer a mesma observação engenhosa que o matemático alemão Karl Frederick Gauss (1777-1855) fez quando criança na escola: agrupe os números em pares, usando 1 e 100, 2 e 99 etc.)

Seção 2.2 Indução

Primeiro Princípio de Indução

Existe uma última técnica de demonstração particularmente útil em ciência da computação. Para ilustrar como ela funciona, imagine que você está subindo uma escada infinitamente alta. Como você sabe se será capaz de chegar a um degrau arbitrariamente alto? Suponha que você faça as seguintes hipóteses sobre sua capacidade de subir:

1. Você consegue alcançar o primeiro degrau.
2. Uma vez chegando a um degrau, você sempre é capaz de chegar ao próximo. (Note que esta asserção é um condicional.)

Se a proposição 1 e o condicional 2 são ambos verdadeiros, então, pela proposição 1, você consegue chegar no primeiro degrau e, portanto, pela proposição 2, consegue chegar no segundo; novamente pela proposição 2, você consegue chegar no terceiro degrau; mais uma vez pela proposição 2, você consegue chegar no quarto degrau; e assim por diante. Você pode subir tão alto quanto quiser. Ambas as hipóteses são necessárias. Se apenas a primeira proposição fosse verdadeira, você não teria nenhuma garantia de passar do primeiro degrau e, se apenas a segunda fosse verdadeira, você poderia não ser capaz de começar nunca. Vamos supor que os degraus da escada estejam numerados pelos inteiros positivos — 1, 2, 3 etc. Agora pense sobre uma propriedade específica que um número possa ter. Em vez de "chegar a um degrau arbitrariamente alto", podemos falar sobre um inteiro positivo arbitrário tendo essa propriedade. Vamos usar a notação $P(n)$ para dizer que o inteiro positivo n tem a propriedade P. Como usar a mesma técnica que usamos para subir a escada para provar que, qualquer que seja o inteiro positivo n, temos $P(n)$? As duas asserções que precisamos provar são

1. $P(1)$ (1 tem a propriedade P)
2. Para qualquer inteiro positivo k, $P(k) \rightarrow P(k+1)$ (Se qualquer número tem a propriedade P, o próximo também tem.)

Se pudermos provar ambas as proposições 1 e 2, então $P(n)$ é válida para qualquer inteiro positivo n, da mesma forma que você poderia subir até um degrau arbitrário da escada.

Primeiro Princípio de Indução Matemática

1. $P(1)$ é verdade
2. $(\forall k)[P(k)$ verdade $\rightarrow P(k+1)$ verdade$]$

$\rightarrow P(n)$ verdade para todo inteiro positivo n

O primeiro princípio de indução matemática é um condicional. A conclusão é uma proposição da forma "$P(n)$ é verdadeiro para todo inteiro positivo n". Portanto, sempre que quisermos provar que alguma coisa é verdade para todo inteiro positivo n, é bastante provável que a indução matemática seja uma técnica apropriada.

LEMBRETE: Para provar que alguma coisa é verdadeira para todo inteiro $n \geq$ que algum valor, pense em indução.

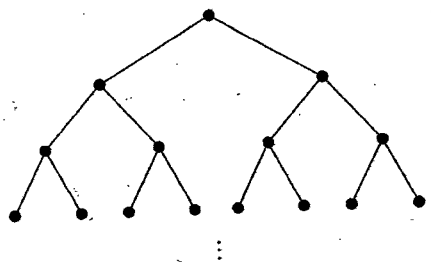
Para mostrar que a conclusão deste condicional é verdadeira, precisamos provar que as duas hipóteses, 1 e 2, são verdadeiras. Para provar a proposição 1, basta mostrar que o número 1 tem a propriedade P, geralmente uma tarefa trivial. A proposição 2 é um condicional que tem que ser válido para todo k. Para provar este condicional, suponha que $P(k)$ é verdadeiro para um inteiro positivo arbitrário k e mostre, baseado nesta hipótese, que $P(k+1)$ é verdadeiro. Você deve se convencer de que supor que o número k tem a propriedade P não é a mesma coisa que supor o que queremos provar (esta é uma confusão comum na primeira vez que se encontra uma demonstração deste tipo). Esta é, simplesmente, a maneira de proceder para obter uma demonstração direta do condicional $P(k) \rightarrow P(k+1)$.

Ao fazer uma demonstração por indução, o estabelecimento da veracidade da proposição 1 é chamado de **base da indução** ou **passo básico** da demonstração por indução. O estabelecimento da veracidade de $P(k) \rightarrow P(k+1)$ é o **passo indutivo**. Quando supomos que $P(k)$ é verdade para provar o passo indutivo, $P(k)$ é chamada de **hipótese de indução**.

Todos os métodos de demonstração de que falamos neste capítulo são técnicas para o raciocínio dedutivo — maneiras de provar uma conjectura que talvez tenha sido formulada por um raciocínio indutivo. A indução matemática também é uma técnica *dedutiva* e não um método de raciocínio indutivo (não se confunda com a terminologia). Nas outras técnicas de demonstração, podemos começar com uma hipótese e juntar diversos fatos até que "tropeçamos" na conclusão. De fato, mesmo que a nossa conjectura esteja ligeiramente incorreta, podemos ver qual deve ser a conclusão correta ao fazer a demonstração. Na indução matemática, no entanto, precisamos saber, desde o início, qual é a forma exata da propriedade $P(n)$ que queremos estabelecer. A indução matemática, portanto, não é uma técnica de demonstração exploratória — pode apenas confirmar uma conjectura correta.

Demonstrações por Indução Matemática

Suponha que um ancestral Silva casou-se e teve dois filhos. Vamos chamar estes dois filhos de geração 1. Suponha, agora, que cada um destes filhos tenha dois filhos; então, a geração 2 contém quatro descendentes. Isto continua de geração em geração. A árvore genealógica da família Silva, portanto, tem a forma ilustrada na Fig. 2.2. (Ela é exatamente igual à Fig. 1.1, onde obtivemos todos os valores lógicos possíveis para n letras de proposição.)



Geração	Descendentes
1	$2 = 2^1$
2	$4 = 2^2$
3	$8 = 2^3$
⋮	⋮

Fig. 2.2

Parece que a geração n contém 2^n descendentes. Mais formalmente, se denotarmos por $P(n)$ o número de descendentes em cada geração, nossa conjectura é que

$$P(n) = 2^n$$

Podemos usar indução para *provar* que nossa conjectura para $P(n)$ está correta.

O passo básico é estabelecer $P(1)$, que é a equação

$$P(1) = 2^1 = 2$$

Isto é verdade, pois nos foi dito que Silva teve dois filhos. Vamos supor agora que nossa conjectura está correta para uma geração arbitrária k, $k \geq 1$, isto é, vamos supor que

$$P(k) = 2^k$$

e tentar mostrar que

$$P(k + 1) = 2^{k+1}$$

Nesta família, cada descendente tem dois filhos, de modo que o número de descendentes na geração $k + 1$ será o dobro do número de descendentes na geração k, ou seja, $P(k + 1) = 2P(k)$. Pela hipótese de indução, $P(k) = 2^k$, logo

$$P(k + 1) = 2P(k) = 2(2^k) = 2^{k+1}$$

e, de fato,

$$P(k + 1) = 2^{k+1}$$

Isto completa nossa demonstração. Agora que estamos tranqüilos sobre o clã dos Silva, podemos aplicar o método de demonstração por indução a problemas menos óbvios.

EXEMPLO 14

Prove que a equação

$$1 + 3 + 5 + \cdots + (2n - 1) = n^2 \tag{1}$$

é verdadeira para qualquer inteiro positivo n. A propriedade $P(n)$ aqui é que a equação (1) acima é válida. O termo à esquerda do sinal de igualdade é a soma de todos os inteiros ímpares de 1 até $2n - 1$. Embora possamos verificar a veracidade dessa equação para qualquer valor particular de n substituindo este valor na equação, não podemos substituir n por todos os inteiros positivos que existem. Assim, uma demonstração por exaustão não funciona. Uma demonstração por indução é apropriada.

O passo básico é estabelecer $P(1)$, que é a equação (1) quando n tem o valor 1, ou seja,

$$P(1): \quad 1 = 1^2$$

Isto é certamente verdade. Para a hipótese de indução, vamos supor $P(k)$ para um inteiro positivo arbitrário k, que é a equação (1) quando n tem o valor k, isto é,

$$P(k): \quad 1 + 3 + 5 + \cdots + (2k - 1) = k^2 \tag{2}$$

(Note que $P(k)$ *não* é a equação $(2k - 1) = k^2$, que só é verdade para $k = 1$.) Usando a hipótese de indução, queremos mostrar $P(k + 1)$, que é a equação (1) quando n assume o valor $k + 1$, ou seja,

$$P(k+1): \quad 1 + 3 + 5 + \cdots + [2(k+1) - 1] \stackrel{?}{=} (k+1)^2 \tag{3}$$

(O ponto de interrogação em cima do sinal de igualdade é para nos lembrar de que é este fato que queremos provar, ao invés de ser alguma coisa que já sabemos.)

A chave de uma demonstração por indução é encontrar um modo de relacionar o que queremos saber — $P(k + 1)$, equação (3) — e o que supusemos — $P(k)$, equação (2). O lado esquerdo de $P(k + 1)$ pode ser reescrito mostrando-se a penúltima parcela:

$$1 + 3 + 5 + \cdots + (2k - 1) + [2(k + 1) - 1]$$

Esta expressão contém o termo à esquerda do sinal de igualdade na equação (2). Como estamos supondo que $P(k)$ é válida, podemos substituir este termo pelo termo à direita do sinal de igualdade na equação (2). Obtemos, então,

$$\begin{aligned} &1 + 3 + 5 + \cdots + [2(k+1) - 1] \\ &\quad = 1 + 3 + 5 + \cdots + (2k - 1) + [2(k+1) - 1] \\ &\quad = k^2 + [2(k+1) - 1] \\ &\quad = k^2 + [2k + 2 - 1] \\ &\quad = k^2 + 2k + 1 \\ &\quad = (k+1)^2 \end{aligned}$$

Portanto,

$$1 + 3 + 5 + \cdots + [2(k+1) - 1] = (k+1)^2$$

o que mostra a validade de $P(k + 1)$, provando, assim, que a equação (1) é verdadeira para qualquer inteiro positivo n.

A Tabela 2.3 resume os três passos necessários para uma demonstração usando o primeiro princípio de indução.

Demonstração usando o primeiro princípio de indução	
Passo 1	Prove a base da indução.
Passo 2	Suponha $P(k)$.
Passo 3	Prove $P(k + 1)$.

Tabela 2.3

EXEMPLO 15

Prove que

$$1 + 2 + 2^2 + \cdots + 2^n = 2^{n+1} - 1$$

para todo $n \geq 1$.

Novamente, indução é apropriada. $P(1)$ é a equação

$$1 + 2 = 2^{1+1} - 1 \quad \text{ou} \quad 3 = 2^2 - 1$$

que é verdadeira. Vamos considerar $P(k)$

$$1 + 2 + 2^2 + \cdots + 2^k = 2^{k+1} - 1$$

como a hipótese de indução e tentar estabelecer $P(k + 1)$:

$$1 + 2 + 2^2 + \cdots + 2^{k+1} \stackrel{?}{=} 2^{k+1+1} - 1$$

Novamente, reescrevendo a soma à esquerda do sinal de igualdade de $P(k + 1)$, vemos como a hipótese de indução pode ser usada:

$$\begin{aligned} &1 + 2 + 2^2 + \cdots + 2^{k+1} \\ &\quad = 1 + 2 + 2^2 + \cdots + 2^k + 2^{k+1} \\ &\quad = 2^{k+1} - 1 + 2^{k+1} \qquad \text{(da hipótese de indução } P(k)\text{)} \\ &\quad = 2(2^{k+1}) - 1 \\ &\quad = 2^{k+1+1} - 1 \end{aligned}$$

LEMBRETE:
Para provar que $P(k) \rightarrow P(k + 1)$, você tem que descobrir o caso $P(k)$ dentro do caso $P(k + 1)$.

Portanto,

$$1 + 2 + 2^2 + \cdots + 2^{k+1} = 2^{k+1+1} - 1$$

o que mostra $P(k + 1)$, concluindo a demonstração.

PROBLEMA PRÁTICO 7

Prove que, para qualquer inteiro positivo n,

$$1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}$$

Nem todas as demonstrações por indução envolvem somas. Outras identidades algébricas sobre os inteiros positivos podem ser demonstradas por indução, assim como proposições não-algébricas, como o número de descendentes na geração n da família Silva.

EXEMPLO 16

Prove que, para qualquer inteiro positivo n, $2^n > n$.

$P(1)$ é a proposição $2^1 > 1$, que certamente é verdade. Vamos supor agora $P(k)$, $2^k > k$, e tentar concluir $P(k + 1)$, $2^{k+1} > k + 1$. Começando com a expressão em $P(k + 1)$ à esquerda do sinal de desigualdade, observamos que $2^{k+1} = 2^k \cdot 2$. Usando a hipótese de indução $2^k > k$ e multiplicando os dois membros desta desigualdade por 2, obtemos $2^k \cdot 2 > k \cdot 2$. Completando o argumento,

$$2^{k+1} = 2^k \cdot 2 > k \cdot 2 = k + k \geq k + 1$$

ou seja,

$$2^{k+1} > k + 1$$

EXEMPLO 17

Prove que para qualquer inteiro positivo n, o número $2^{2n} - 1$ é divisível por 3.

O passo básico é mostrar $P(1)$, isto é, que $2^{2(1)} - 1 = 4 - 1 = 3$ é divisível por 3. Isto é evidente.

Vamos supor que $2^{2k} - 1$ é divisível por 3, o que significa que $2^{2k} - 1 = 3m$ para algum inteiro m, ou seja, $2^{2k} = 3m + 1$. Queremos mostrar que $2^{2(k+1)} - 1$ é divisível por 3.

$$\begin{aligned} 2^{2(k+1)} - 1 &= 2^{2k+2} - 1 \\ &= 2^2 \cdot 2^{2k} - 1 \\ &= 2^2(3m + 1) - 1 \qquad \text{(pela hipótese de indução)} \\ &= 12m + 4 - 1 \\ &= 12m + 3 \\ &= 3(4m + 1) \qquad \text{onde } 4m + 1 \text{ é inteiro} \end{aligned}$$

Portanto, $2^{2(k+1)} - 1$ é divisível por 3.

Para o primeiro passo em uma demonstração por indução, pode ser apropriado começar em 0, ou em 2 ou 3, em vez de 1. O mesmo princípio se aplica, independentemente do degrau onde você começa a subir na escada.

EXEMPLO 18

Prove que $n^2 > 3n$ para $n \geq 4$.

Devemos usar indução aqui, começando com a base da indução em $P(4)$. (Testando os valores $n = 1, 2$ e 3, pode-se mostrar que a desigualdade não é válida para esses valores.) $P(4)$ é a desigualdade $4^2 > 3(4)$, ou seja, $16 > 12$, que é verdadeira. A hipótese de indução é que $k^2 > 3k$, onde $k \geq 4$, e queremos mostrar que $(k + 1)^2 > 3(k + 1)$.

$$\begin{aligned} (k + 1)^2 &= k^2 + 2k + 1 \\ &> 3k + 2k + 1 \qquad \text{(pela hipótese de indução)} \\ &\geq 3k + 8 + 1 \qquad \text{(pois } k \geq 4\text{)} \\ &> 3k + 3 \\ &= 3(k + 1) \end{aligned}$$

PROBLEMA PRÁTICO 8

Prove que $2^{n+1} < 3^n$ para todo $n > 1$.

Demonstrações supostamente por indução, mas que não o são de fato, também são possíveis. Quando demonstramos $P(k + 1)$ sem usar $P(k)$, fizemos uma demonstração direta de $P(k + 1)$, onde $k + 1$ é arbitrário. Não é que a demonstração esteja errada, apenas ela deve ser reescrita como uma demonstração direta de $P(n)$ para qualquer n e não como uma demonstração por indução.

Pode ser conveniente uma demonstração por indução mesmo em aplicações não tão óbvias como nos exemplos anteriores. Isto acontece, geralmente, quando alguma quantidade na proposição a ser demonstrada toma valores inteiros não-negativos arbitrários.

EXEMPLO 19

Uma linguagem de programação pode ser projetada com as seguintes convenções em relação à multiplicação: um único fator não necessita de parênteses, mas o produto "a vezes b" precisa ser escrito na forma $(a)b$. Assim, o produto

$$a \cdot b \cdot c \cdot d \cdot e \cdot f \cdot g$$

poderia ser escrito nesta linguagem como

$$((((((a)b)c)d)e)f)g$$

ou como

$$((a)b)(((c)d)(e)f)g$$

dependendo da ordem em que se vai executar o produto. O resultado é o mesmo em qualquer dos casos.

Queremos mostrar que qualquer produto pode ser escrito com um número par de parênteses. A demonstração é por indução no número de fatores. Para um único fator, temos 0 parênteses, um número par. Suponha que, para qualquer produto com k fatores, temos um número par de parênteses. Agora considere um produto P com $k + 1$ fatores. P pode ser considerado um produto de r vezes s, onde r tem k fatores e s é um único fator. Pela hipótese de indução, r tem um número par de parênteses. Podemos, então, escrever r vezes s como $(r)s$. Isto adiciona mais 2 parênteses ao número par de parênteses em r, dando a P um número par de parênteses. ●

EXEMPLO 20

Um problema de "ladrilhagem" fornece uma boa ilustração de indução em um contexto geométrico. Um *ângulo de ferro* é uma peça em forma de L cobrindo três quadrados em um tabuleiro quadriculado, como o de xadrez (veja a Fig. 2.3a). O problema é mostrar que, para qualquer inteiro positivo n, se removermos um quadrado de um tabuleiro originalmente com $2^n \times 2^n$ quadrados, ele pode ser ladrilhado — completamente recoberto — com ângulos de ferro.

A base da indução é $n = 1$, o que nos dá um tabuleiro com 2×2 quadrados. A Fig. 2.3b mostra a solução, neste caso se for removido o canto direito superior. A remoção de qualquer dos outros três quadrados funciona da mesma maneira. Suponha agora que qualquer tabuleiro $2^k \times 2^k$ com um quadrado removido pode ser ladrilhado usando-se ângulos de ferro e vamos considerar um tabuleiro $2^{k+1} \times 2^{k+1}$. Precisamos mostrar que ele pode ser ladrilhado quando se remove um quadrado. Para relacionar o caso $k + 1$ com a hipótese de indução, divida o tabuleiro $2^{k+1} \times 2^{k+1}$ em quatro partes iguais. Cada parte é um tabuleiro $2^k \times 2^k$ e um deles vai ter um quadrado faltando (Fig. 2.3c). Pela hipótese de indução, este tabuleiro pode ser ladrilhado. Retire um canto de cada uma das outras três partes como na Fig. 2.3d. Pela hipótese de indução, estas três partes com os quadrados removidos podem ser ladrilhadas. Como um único ângulo de ferro pode ser usado para cobrir estes três quadrados retirados, temos que o tabuleiro original $2^{k+1} \times 2^{k+1}$ com um dos quadrados removidos pode ser ladrilhado. ●

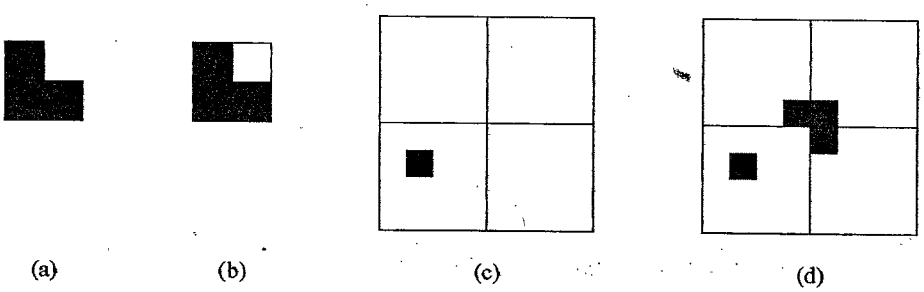


Fig. 2.3

Segundo Princípio de Indução

Além do primeiro princípio de indução, que temos usado,

1. $P(1)$ é verdade
2. $(\forall k)[P(k)$ verdade $\rightarrow P(k + 1)$ verdade$]$

$\rightarrow P(n)$ verdade para todo inteiro positivo n

existe um segundo princípio de indução.

Segundo Princípio de Indução Matemática

1'. $P(1)$ é verdade
2'. $(\forall k)[P(r)$ verdade para todo r, $1 \leq r \leq k \rightarrow P(k + 1)$ verdade$]$

$\rightarrow P(n)$ verdade para todo inteiro positivo n

Estes dois princípios de indução diferem na proposição 2 e 2′. Na proposição 2, precisamos ser capazes de provar, para um inteiro positivo arbitrário k, que $P(k + 1)$ é verdadeira baseado apenas na hipótese de que $P(k)$ é verdadeira. Na proposição 2, podemos supor que $P(r)$ é verdadeira para todos os inteiros r entre 1 e um inteiro positivo arbitrário k para provar $P(k + 1)$. Isto parece nos dar muito mais "munição", de modo que pode acontecer, algumas vezes, de sermos capazes de provar o condicional em 2′ sem conseguirmos provar o condicional em 2.

O que nos permite deduzir $(\forall n)P(n)$ em cada caso? Veremos que os dois princípios, ou seja, os dois métodos de demonstração, são equivalentes. Em outras palavras, se aceitamos como válido o primeiro princípio, então o segundo também é válido, e reciprocamente. Para provar a equivalência entre os dois princípios, vamos considerar um outro princípio, que parece tão óbvio que não necessita de discussão.

Princípio da Boa Ordenação

Toda coleção de inteiros positivos que contém algum elemento tem um menor elemento.

Veremos, também, que os seguintes condicionais são verdadeiros:

segundo princípio de indução → primeiro princípio de indução
primeiro princípio de indução → princípio da boa ordenação
princípio da boa ordenação → segundo princípio de indução

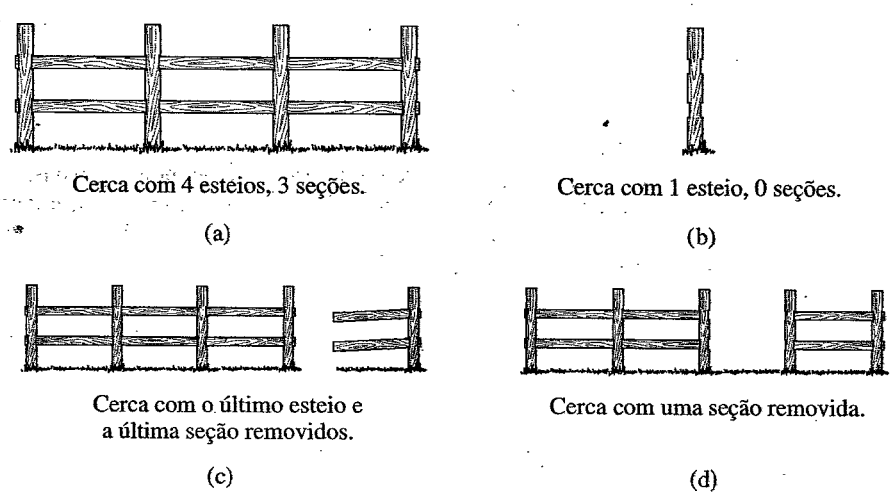
Como conseqüência, todos os três princípios são equivalentes, e aceitar qualquer um deles como verdadeiro significa aceitar os outros dois também.

Para provar que o segundo princípio de indução implica o primeiro, suponha que aceitamos o segundo princípio como sendo um argumento válido. Podemos, então, mostrar que o primeiro princípio é válido, isto é, que podemos concluir $P(n)$ para todo n das proposições 1 e 2. Se a proposição 1 é verdadeira, a proposição 1′ também o é. Se a proposição 2 é verdadeira, a proposição 2′ também o é, pois podemos dizer que concluímos $P(k + 1)$ de $P(r)$ para todo r entre 1 e k, embora tenhamos usado apenas a condição $P(k)$. (De maneira mais precisa, a proposição 2′ necessita que provemos que $P(1) \wedge P(2) \wedge \ldots \wedge P(k) \rightarrow P(k + 1)$; mas $P(1) \wedge P(2) \wedge \ldots \wedge P(k) \rightarrow P(k)$ e, pela proposição 2, $P(k) \rightarrow P(k + 1)$, logo $P(1) \wedge P(2) \wedge \ldots \wedge P(k) \rightarrow P(k + 1)$.) Pelo segundo princípio de indução, podemos concluir $P(n)$ para todo n. As demonstrações de que o primeiro princípio de indução implica o princípio da boa ordenação, que por sua vez implica o segundo princípio de indução, são deixadas como exercício na Seção 3.1.

Para distinguir entre uma demonstração por indução usando o primeiro ou o segundo princípio, vamos considerar um exemplo um tanto pitoresco que pode ser provado das duas maneiras.

EXEMPLO 21

Prove que uma cerca reta com n esteios tem $n - 1$ seções para qualquer $n \geq 1$ (veja a Fig. 2.4a).



Cerca com 4 esteios, 3 seções. (a)

Cerca com 1 esteio, 0 seções. (b)

Cerca com o último esteio e a última seção removidos. (c)

Cerca com uma seção removida. (d)

Fig. 2.4

Seja $P(n)$ a proposição que uma cerca com n esteios tem $n - 1$ seções; vamos provar que $P(n)$ é verdadeira para todo $n \geq 1$.

Vamos começar com o primeiro princípio de indução. Para o passo básico, $P(1)$ diz que uma cerca com apenas 1 esteio tem 0 seções, o que é claramente verdade (veja a Fig. 2.4b). Suponha que $P(k)$ é verdadeira:

uma cerca com k esteios tem $k - 1$ seções

e tente provar $P(k + 1)$:

(?) uma cerca com $k + 1$ esteios tem k seções.

Dada uma cerca com $k + 1$ esteios, como podemos relacioná-la a uma cerca com k esteios de modo a usar a hipótese de indução? Podemos cortar fora o último esteio e a última seção (Fig. 2.4c). A cerca resultante tem k esteios e, pela hipótese de indução, tem $k - 1$ seções. Portanto, a cerca original tinha k seções.

Vamos agora provar o mesmo resultado usando o segundo princípio de indução. O passo básico é igual ao do caso anterior. Para a hipótese de indução, supomos que

para todo r, $1 \leq r \leq k$, uma cerca com r esteios tem $r - 1$ seções

e tentar provar $P(k + 1)$:

(?) uma cerca com $k + 1$ esteios tem k seções

Para uma cerca com $k + 1$ esteios, divida a cerca em duas partes removendo uma seção (Fig. 2.4d). As duas partes da cerca têm r_1 e r_2 esteios, onde $1 \leq r_1 \leq k$, $1 \leq r_2 \leq k$ e $r_1 + r_2 = k + 1$. Pela hipótese de indução, as duas partes têm, respectivamente, $r_1 - 1$ e $r_2 - 1$ seções, logo a cerca original tinha

$(r_1 - 1) + (r_2 - 1) + 1$ seções

(O 1 extra é pela seção que foi removida.) A aritmética nos diz, então, que tínhamos

$r_1 + r_2 - 1 = (k + 1) - 1 = k$ seções

Isto prova que uma cerca com $k + 1$ esteios tem k seções, o que verifica a veracidade de $P(k + 1)$, completando a demonstração pelo segundo princípio de indução. ●

O Exemplo 21 permite usar qualquer uma das formas de uma demonstração por indução, já que podemos diminuir a cerca retirando-se uma extremidade ou uma seção central. O problema no Exemplo 19 é semelhante.

EXEMPLO 22

Vamos mostrar, de novo, que qualquer produto de fatores pode ser escrito nesta linguagem de programação com um número par de parênteses, desta vez usando o segundo princípio de indução. O passo básico é o mesmo que o do Exemplo 19: um único fator tem 0 parênteses, um número par. Suponha que qualquer produto de r fatores, $1 \leq r \leq k$, pode ser escrito com um número par de parênteses. Considere agora um produto P com $k + 1$ fatores. Então P pode ser escrito como um produto $(S)T$ de dois fatores, S e T, onde S tem r_1 fatores e T tem r_2 fatores. Temos $1 \leq r_1 \leq k$, $1 \leq r_2 \leq k$ e $r_1 + r_2 = k + 1$. Pela hipótese de indução, S e T têm, cada um, um número par de parênteses e, portanto, $P = (S)T$ também tem um número par de parênteses. ●

A maior parte dos problemas não funciona bem com uma das formas de indução: os problemas da cerca e da linguagem de programação são um tanto ou quanto artificiais. Em geral, usamos a segunda forma quando o problema se divide, naturalmente, no meio em vez de crescer em uma das pontas.

EXEMPLO 23

Prove que, para todo $n \geq 2$, n é um número primo ou é um produto de números primos.

Vamos adiar a decisão sobre se usamos o primeiro ou o segundo princípio de indução; o passo básico é o mesmo nos dois casos e não precisamos começar com 1. É claro que devemos começar aqui com 2. $P(2)$ é a proposição que 2 é um número primo ou um produto de primos. Como $P(2)$ é primo, a proposição é verdadeira. Pulando adiante, para qualquer dos dois princípios precisaremos analisar o caso $k + 1$. Se $k + 1$ for primo, estamos feitos. Se $k + 1$ não for primo, então é um número composto e pode ser escrito na forma $k + 1 = ab$. Dividimos $k + 1$ em dois fatores e talvez nenhum deles tenha o valor k, de modo que uma hipótese apenas sobre $P(k)$ não é suficiente. Usaremos, então, o segundo princípio de indução.

LEMBRETE:
Use o segundo princípio de indução quando o caso $k + 1$ depende de resultados anteriores a k.

Vamos começar de novo e supor que, para todo r, $2 \leq r \leq k$, $P(k)$ é verdadeira — r é primo ou um produto de primos. Considere agora o número $k + 1$. Se $k + 1$ for primo, terminamos. Se $k + 1$ não for primo, então é um número composto e pode ser escrito na forma $k + 1 = ab$, onde $1 < a < k + 1$ e $1 < b < k + 1$. (Esta é uma fatoração não-trivial, de modo que nenhum fator pode ser igual a 1 ou a $k + 1$.) Portanto, $2 \leq a \leq k$ e $2 \leq b \leq k$. A hipótese de indução pode ser aplicada a a e a b, logo cada um deles ou é um primo, ou é um produto de primos. Portanto, $k + 1$ é um produto de primos. Isto verifica $P(k + 1)$ e completa a demonstração pelo segundo princípio de indução. ●

EXEMPLO 24

Prove que qualquer quantia para franquia postal maior ou igual a 8 centavos pode ser conseguida usando-se apenas selos de 3 e 5 centavos.

A proposição $P(n)$ agora é que precisamos apenas de selos de 3 e 5 centavos para obter n centavos em selos e queremos provar que $P(n)$ é verdadeira para todo $n \geq 8$. A base da indução é estabelecer $P(8)$, o que

é feito pela equação

$8 = 3 + 5$

Por motivos que ficarão claros em alguns instantes, vamos estabelecer, também, dois casos adicionais, $P(9)$ e $P(10)$, pelas equações

$9 = 3 + 3 + 3$

$10 = 5 + 5$

Vamos supor, agora, que $P(r)$ é verdadeira para qualquer r, $8 \leq r \leq k$, e considerar $P(k + 1)$. Podemos supor que $k + 1$ é pelo menos 11, já que provamos $P(r)$ para $r = 8$, 9 e 10. Se $k + 1 \geq 11$, então $(k + 1) - 3 = k - 2 \geq 8$ e, pela hipótese de indução, $P(k - 2)$ é verdade. Portanto, $k - 2$ pode ser escrito como uma soma de números iguais a 3 e a 5; adicionando-se mais um 3, obtemos $k + 1$ como uma soma de números iguais a 3 e a 5. Isto prova a veracidade de $P(k + 1)$ e completa a demonstração. ●

PROBLEMA PRÁTICO 9

a. Por que os casos adicionais $P(9)$ e $P(10)$ foram demonstrados separadamente no Exemplo 24?
b. Por que não poderíamos usar o primeiro princípio de indução na demonstração do Exemplo 24? ●

Seção 2.2 Revisão

Técnicas

- Utilização do primeiro princípio de indução em demonstrações.
- Utilização do segundo princípio de indução em demonstrações.

Idéias Principais

A indução matemática é uma técnica para provar propriedades dos inteiros positivos.

Uma demonstração por indução não precisa começar com 1.

A indução pode ser usada para provar proposições sobre quantidades cujos valores são inteiros não-negativos arbitrários.

O primeiro e o segundo princípios de indução provam a mesma conclusão, mas uma das abordagens pode ser mais fácil de usar em uma determinada situação.

Exercícios 2.2

Nos Exercícios de 1 a 20, use indução matemática para provar que as proposições dadas são verdadeiras para todo inteiro positivo n.

★**1.** $2 + 6 + 10 + \cdots + (4n - 2) = 2n^2$

2. $2 + 4 + 6 + \cdots + 2n = n(n + 1)$

★**3.** $1 + 5 + 9 + \cdots + (4n - 3) = n(2n - 1)$

4. $1 + 3 + 6 + \cdots + \dfrac{n(n+1)}{2} = \dfrac{n(n+1)(n+2)}{6}$

★**5.** $4 + 10 + 16 + \cdots + (6n - 2) = n(3n + 1)$

6. $5 + 10 + 15 + \cdots + 5n = \dfrac{5n(n+1)}{2}$

7. $1^2 + 2^2 + \cdots + n^2 = \dfrac{n(n+1)(2n+1)}{6}$

8. $1^3 + 2^3 + \cdots + n^3 = \dfrac{n^2(n+1)^2}{4}$

★**9.** $1^2 + 3^2 + \cdots + (2n - 1)^2 = \dfrac{n(2n-1)(2n+1)}{3}$

10. $1^4 + 2^4 + \cdots + n^4 = \dfrac{n(n+1)(2n+1)(3n^2+3n-1)}{30}$

11. $1 \cdot 3 + 2 \cdot 4 + 3 \cdot 5 + \cdots + n(n + 2) = \dfrac{n(n+1)(2n+7)}{6}$

12. $1 + a + a^2 + \cdots + a^{n-1} = \dfrac{a^n - 1}{a - 1}$ para $a \neq 0, a \neq 1$

★13. $\dfrac{1}{1 \cdot 2} + \dfrac{1}{2 \cdot 3} + \dfrac{1}{3 \cdot 4} + \cdots + \dfrac{1}{n(n+1)} = \dfrac{n}{n+1}$

14. $\dfrac{1}{1 \cdot 3} + \dfrac{1}{3 \cdot 5} + \dfrac{1}{5 \cdot 7} + \cdots + \dfrac{1}{(2n-1)(2n+1)} = \dfrac{n}{2n+1}$

★15. $1^2 - 2^2 + 3^2 - 4^2 + \cdots + (-1)^{n+1} n^2 = \dfrac{(-1)^{n+1}(n)(n+1)}{2}$

16. $2 + 6 + 18 + \cdots + 2 \cdot 3^{n-1} = 3^n - 1$

17. $2^2 + 4^2 + \cdots + (2n)^2 = \dfrac{2n(n+1)(2n+1)}{3}$

18. $1 \cdot 2 + 2 \cdot 3 + 3 \cdot 4 + \cdots + n(n+1) = \dfrac{n(n+1)(n+2)}{3}$

19. $\dfrac{1}{1 \cdot 4} + \dfrac{1}{4 \cdot 7} + \dfrac{1}{7 \cdot 10} + \cdots + \dfrac{1}{(3n-2)(3n+1)} = \dfrac{n}{(3n+1)}$

20. $1 \cdot 1! + 2 \cdot 2! + 3 \cdot 3! + \ldots + n \cdot n! = (n+1)! - 1$, onde $n!$ é o produto dos inteiros positivos de 1 a n.

★21. Uma *progressão geométrica* é uma seqüência de termos com um termo inicial a tal que cada termo a seguir é obtido multiplicando-se o termo anterior por uma *mesma razão r*. Prove a fórmula para a soma dos n primeiros termos de uma progressão geométrica ($n \geq 1$):

$$a + ar + ar^2 + \cdots + ar^{n-1} = \frac{a - ar^n}{1 - r}$$

22. Uma *progressão aritmética* é uma seqüência de termos com um termo inicial a tal que cada termo a seguir é obtido somando-se uma *mesma parcela d* ao termo anterior. Prove a fórmula para a soma dos n primeiros termos de uma progressão aritmética ($n \geq 1$):

$$a + (a + d) + (a + 2d) + \cdots + [a + (n-1)d] =$$
$$= \frac{n}{2}[2a + (n-1)d]$$

23. Prove que

$$(-2)^0 + (-2)^1 + (-2)^2 + \cdots + (-2)^n = \frac{1 - 2^{n+1}}{3}$$

para todo inteiro positivo ímpar n.

24. Prove que $n^2 \geq 2n + 3$ para $n \geq 3$.

★25. Prove que $n^2 > n + 1$ para $n \geq 2$.

26. Prove que $n^2 > 5n + 10$ para $n > 6$.

27. Prove que $2^n > n^2$ para $n \geq 5$.

28. Prove que $n! > n^2$ para $n \geq 4$, onde $n!$ é o produto dos inteiros positivos de 1 a n.

★29. Prove que $2^n < n!$ para $n \geq 4$.

30. Prove que $2^{n-1} \leq n!$ para $n \geq 1$.

31. Prove que $n! < n^n$ para $n \geq 2$.

32. Prove que $(1 + x)^n > 1 + x^n$ para $n > 1, x > 0$.

33. Prove que $(a/b)^{n+1} < (a/b)^n$ para $n \geq 1$ e $0 < a < b$.

★34. Prove que $1 + 2 + \ldots + n < n^2$ para $n > 1$.

35. **a.** Tente usar indução para provar que

$$1 + \frac{1}{2} + \frac{1}{4} + \cdots + \frac{1}{2^n} < 2 \text{ para } n \geq 1$$

O que não funciona?

b. Prove que

$$1 + \frac{1}{2} + \frac{1}{4} + \cdots + \frac{1}{2^n} = 2 - \frac{1}{2^n} \text{ para } n \geq 1$$

mostrando, assim, que

$$1 + \frac{1}{2} + \frac{1}{4} + \cdots + \frac{1}{2^n} < 2 \text{ para } n \geq 1$$

Nos Exercícios de 36 a 47, prove que a proposição é verdadeira para todo inteiro positivo n.

★**36.** $2^{3n} - 1$ é divisível por 7.

37. $3^{2n} + 7$ é divisível por 8.

38. $7^n - 2^n$ é divisível por 5.

39. $13^n - 6^n$ é divisível por 7.

★**40.** $2^n + (-1)^{n+1}$ é divisível por 3.

41. $2^{5n+1} + 5^{n+2}$ é divisível por 27.

42. $3^{4n+2} + 5^{2n+1}$ é divisível por 14.

43. $7^{2n} + 16n - 1$ é divisível por 64.

★**44.** $10^n + 3 \cdot 4^{n+2} + 5$ é divisível por 9.

45. $n^3 - n$ é divisível por 3.

46. $n^3 + 2n$ é divisível por 3.

47. $x^n - 1$ é divisível por $x - 1$ para $x \neq 1$.

★**48.** Prove o *Teorema de DeMoivre:*

$$(\cos \theta + i \operatorname{sen} \theta)^n = \cos n\theta + i \operatorname{sen} n\theta$$

para todo $n \geq 1$. *Dica*: Lembre-se das fórmulas para a soma de ângulos da trigonometria:

$$\cos(\alpha + \beta) = \cos \alpha \cos \beta - \operatorname{sen} \alpha \operatorname{sen} \beta$$
$$\operatorname{sen}(\alpha + \beta) = \operatorname{sen} \alpha \cos \beta + \cos \alpha \operatorname{sen} \beta$$

49. Prove que

$$\operatorname{sen} \theta + \operatorname{sen} 3\theta + \cdots + \operatorname{sen}(2n - 1)\theta = \frac{\operatorname{sen}^2 n\theta}{\operatorname{sen} \theta}$$

para todo $n \geq 1$ e para todo θ tal que $\operatorname{sen} \theta \neq 0$.

★**50.** Use indução para provar que o produto de três inteiros positivos consecutivos quaisquer é divisível por 3.

51. Suponha que a exponenciação é definida pela equação

$$x^j \cdot x = x^{j+1}$$

para qualquer $j \geq 1$. Use indução para provar que $x^n \cdot x^m = x^{n+m}$, para $n \geq 1$ e $m \geq 1$. (*Dica*: Faça a indução para m fixo e n arbitrário.)

52. De acordo com o Exemplo 20, é possível usar ângulos de ferro para ladrilhar um tabuleiro 4×4 com o canto direito superior removido. Desenhe tal ladrilhagem.

53. O Exemplo 20 não cobre o caso de tabuleiros com tamanhos diferentes de potências de 2. Verifique se é possível ladrilhar um tabuleiro 3×3.

54. Considere uma coleção de n retas infinitas tal que duas retas nesta coleção nunca são paralelas e três retas nunca têm um ponto comum de interseção. Mostre que, para $n \geq 1$, as retas dividem o plano em $(n^2 + n + 2)/2$ regiões separadas.

55. Uma cadeia formada por algarismos binários, 0 e 1, vai ser convertida em uma cadeia de paridade par adicionando-se um bit[1] de paridade ao final da cadeia. O bit de paridade é inicialmente 0. Quando um 0 é processado, o bit de paridade não se altera. Quando um 1 é processado, o bit de paridade muda de 0 para 1 ou de 1 para 0. Prove que o número de algarismos iguais a 1 na cadeia final, incluindo o bit de paridade, é sempre par. (*Dica*: Considere vários casos.)

[1]Utilizaremos, como é usual, a nomenclatura inglesa *bit* para denotar um algarismo binário. (A palavra *bit* é uma abreviação para *binary digit*, que significa *algarismo binário*.) (*N.T.*)

★**56.** O que está errado na seguinte "demonstração" por indução matemática? Vamos provar que, para qualquer inteiro positivo n, n é igual a 1 mais n. Suponha que $P(k)$ é verdadeira.

$k = k + 1$

Somando 1 aos dois lados desta equação, obtemos

$k + 1 = k + 2$

Portanto,

$P(k + 1)$ é verdadeira

57. O que está errado na seguinte "demonstração" por indução matemática? Vamos provar que todos os computadores são montados pelo mesmo fabricante. Em particular, vamos provar que em qualquer coleção de n computadores, onde n é um inteiro positivo, todos os computadores foram montados pelo mesmo fabricante. Vamos primeiro provar $P(1)$, um processo trivial, já que em qualquer coleção com apenas um computador existe apenas um fabricante. Vamos supor $P(k)$, isto é, em qualquer coleção de k computadores, todos os computadores foram montados pelo mesmo fabricante. Para provar $P(k + 1)$, vamos considerar uma coleção qualquer de $k + 1$ computadores. Vamos retirar um desses $k + 1$ computadores (vamos chamá-lo de HAL) da coleção. Pela hipótese, todos os k computadores restantes foram montados pelo mesmo fabricante. Vamos trocar o HAL de lugar com um destes computadores. No novo grupo de k computadores, todos têm o mesmo fabricante. Assim, o fabricante de HAL é o mesmo que produziu todos os outros computadores e todos os $k + 1$ computadores têm o mesmo fabricante.

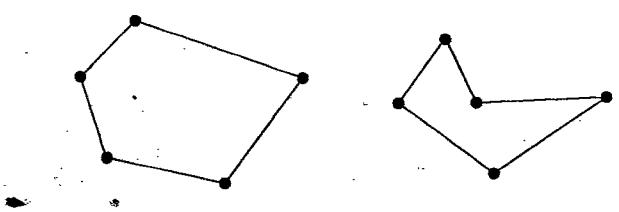
58. Uma tribo obscura tem uma linguagem com apenas três palavras básicas, *mono*, *nono* e *sono*. Novas palavras são compostas agregando-se estas palavras em qualquer ordem, como em *sononomonononono*. Qualquer justaposição deste tipo é válida.

a. Use o primeiro princípio de indução (sobre o número de palavras básicas em cada palavra composta) para provar que qualquer palavra nesta linguagem tem um número par de letras iguais a *o*.

b. Use o segundo princípio de indução (sobre o número de palavras básicas em cada palavra composta) para provar que qualquer palavra nesta linguagem tem um número par de letras iguais a *o*.

★**59.** Considere fbfs proposicionais contendo apenas os conectivos $\wedge$, $\vee$ e $\rightarrow$ (sem a negação) e tais que, ao usar um conectivo para juntar duas fbfs, temos que usar parênteses. Conte cada letra de proposição, conectivo ou parênteses como um símbolo. Por exemplo, $((A) \wedge (B)) \vee ((C) \wedge (D))$ é uma fbf deste tipo, com 19 símbolos. Prove que qualquer fbf deste tipo tem um número ímpar de símbolos.

60. Um *polígono simples fechado* consiste em n pontos no plano unidos em pares por n segmentos de reta; cada ponto é a extremidade de exatamente dois segmentos de reta (veja os exemplos na figura a seguir). Prove que a soma dos ângulos internos de um polígono simples fechado com n lados é $(n - 2)180°$ para todo $n \geq 3$. (*Dica*: Para o caso $k + 1$, divida o polígono em duas partes.)



(a) (b)

★**61.** Prove que qualquer quantia em selos maior ou igual a 2 centavos pode ser obtida usando-se apenas selos de 2 e 3 centavos.

62. Prove que qualquer quantia em selos maior ou igual a 12 centavos pode ser obtida usando-se apenas selos de 4 e 5 centavos.

63. Prove que qualquer quantia em selos maior ou igual a 14 centavos pode ser obtida usando-se apenas selos de 3 e 8 centavos.

★**64.** Prove que qualquer quantia em selos maior ou igual a 64 centavos pode ser obtida usando-se apenas selos de 5 e 17 centavos.

65. Em qualquer grupo de k pessoas, $k \geq 1$, cada pessoa cumprimenta, com aperto de mão, todas as outras pessoas. Encontre uma fórmula para o número de apertos de mão e prove-a usando indução.

Seção 2.3 Mais sobre Demonstração de Correção

Explicamos, na Seção 1.6, como usar um sistema de lógica formal para demonstrar matematicamente a correção de um programa. Asserções ou predicados envolvendo as variáveis do programa são inseridos no início, no final e em pontos intermediários entre as proposições do programa. Então, demonstrar a correção de qualquer proposição particular s_i envolve provar que o condicional representado pela tripla de Hoare

$$\{Q\}\ s_i\ \{R\} \tag{1}$$

é verdadeiro. Aqui, Q e R são asserções conhecidas como a precondição e a pós-condição, respectivamente, para a proposição. Podemos demonstrar a correção do programa se todos estes condicionais forem verdadeiros.

Discutimos, no Cap. 1, regras de inferência que fornecem condições sob as quais o condicional (1) é verdadeiro quando s_i é uma atribuição e quando s_i é um condicional. Vamos agora usar regras de inferência que nos dão condições sob as quais (1) é verdadeiro quando s_i é um laço. Adiamos a análise de proposições em laço até agora porque usa-se indução matemática na aplicação desta regra de inferência.

A Regra do Laço

Suponha que s_i é uma proposição com laço da forma

enquanto condição B **faça**
  P
fim do enquanto

onde B é uma condição que pode ser falsa ou verdadeira e P é um segmento de programa. Quando esta proposição é executada, verifica-se a condição B. Caso B seja verdadeira, o segmento de programa P é executado e B é reavaliada. Se B continua sendo verdadeira, o segmento de programa é executado novamente, B é reavaliada de novo, e assim por diante. Se a condição B torna-se falsa em algum momento, o laço termina.

A forma condicional (1) que pode ser usada quando s_i é uma proposição com laço impõe (como o axioma de atribuição o fez) uma relação entre a precondição e a pós-condição. A precondição Q é válida antes de se entrar no laço; parece estranho mas uma das condições é que Q continue válido depois que o laço terminar (o que significa que devemos procurar um Q que queremos que seja verdadeiro quando o laço terminar). Além disso, B' — a condição para o laço parar — também tem que ser verdadeira. Assim, (1) vai ter a forma

$$\{Q\}\ s_i\ \{Q \wedge B'\} \tag{2}$$

EXEMPLO 25

Considere a função em pseudocódigo a seguir, que se supõe dar como resposta o valor $x * y$ para inteiros não-negativos x e y.

```
Produto(inteiro não-negativo x; inteiro não-negativo y)
Variáveis locais:
inteiros i, j
   i = 0
   j = 0
   enquanto i ≠ x faça
      j = j + y
      i = i + 1
   fim do enquanto
   //j agora tem o valor x * y
   retorne j
fim da função Produto
```

Esta função contém um laço; a condição B para que o laço continue a ser executado é que $i \neq x$. A condição B' que pára a execução do laço é que $i = x$. Ao final do laço, afirma-se, no comentário, que j tem o valor $x * y$. Dado que $i = x$ quando o laço termina, a asserção $j = i * y$ também teria que ser verdadeira. Assim, ao término do laço, se a função faz o que se afirma, a asserção

$$j = i * y \wedge i = x$$

é verdadeira. Isto é da forma $Q \wedge B'$, se considerarmos a asserção

$$j = i * y$$

como Q. Para colocar na mesma forma que (2), a asserção $j = i * y$ teria que ser verdadeira antes do laço. Isto ocorre, de fato, já que imediatamente antes do laço temos que $i = j = 0$.

Parece que temos um candidato para Q, neste exemplo, para o condicional (2), mas ainda não temos a regra de inferência que nos permite dizer quando (2) é um condicional verdadeiro. (Lembre-se de que descobrimos Q supondo que o código para a função funciona corretamente.)

A asserção Q tem que ser verdadeira antes de começar o laço. Se o condicional (2) é verdadeiro, Q tem que ser verdadeira após o término do laço. Já que podemos não saber exatamente quando o laço vai terminar, Q precisa permanecer verdadeira a cada iteração, o que inclui a iteração final. Q representa um predicado, ou uma relação, entre os valores das variáveis do programa. Se esta relação entre os valores das variáveis do programa é válida antes da execução de uma iteração do laço e após a execução da iteração, então a *relação* entre estas variáveis não é afetada pela ação da iteração do laço, embora os valores propriamente ditos possam ser modificados. Tal relação é chamada de um **invariante do laço**.

A **regra de inferência para laços** permite que a veracidade de (2) seja inferida de um condicional dizendo que Q é um invariante do laço, isto é, que se Q é verdadeira e se a condição B é verdadeira, de modo que é executada outra iteração do laço, então Q permanece verdadeira após esta iteração. A regra está formalmente enunciada na Tabela 2.4.

De	Podemos Deduzir	Nome da Regra	Restrições sobre o Uso
$\{Q \wedge B\}\ P\ \{Q\}$	$\{Q\}\ s_i\ \{Q \wedge B'\}$	laço	s_i tem a forma **enquanto** condição B **faça** P **fim do enquanto**

Tabela 2.4

Para usar esta regra de inferência, precisamos encontrar um invariante do laço Q que seja *útil* — um que afirme o que queremos e o que esperamos que aconteça — e depois provar o condicional

$$\{Q \wedge B\}\ P\ \{Q\}$$

E é aqui que a indução entra em cena. Vamos denotar por $Q(n)$ a proposição que um invariante do laço Q proposto seja verdadeiro após n iterações do laço. Como não sabemos quantas iterações serão executadas no laço (isto é, por quanto tempo a condição B permanece verdadeira), queremos mostrar que $Q(n)$ é verdadeira para todo $n \geq 0$. (O valor $n = 0$ corresponde à asserção antes do início do laço, antes de qualquer iteração.)

EXEMPLO 26

Considere, novamente, a função em pseudocódigo do Exemplo 25. Neste exemplo, conjecturamos que Q é a relação

$$j = i * y$$

Para usar a regra de inferência do laço, precisamos provar que Q é um invariante do laço.

As quantidades x e y permanecem constantes durante todo o processo, mas os valores de i e j variam dentro do laço. Vamos denotar por i_n e j_n, respectivamente, os valores de i e j após n iterações do laço. Então, $Q(n)$ é a proposição $j_n = i_n * y$. Vamos provar por indução que $Q(n)$ é válida para todo $n \geq 0$. $Q(0)$ é a proposição

$$j_0 = i_0 * y$$

que, como observamos no Exemplo 25, é verdadeira, pois antes de qualquer iteração, ao chegar pela primeira vez no laço, são atribuídos a ambos, i e j, o valor 0. (Formalmente, o axioma de atribuição poderia ser usado para provar que estas condições sobre i e j são válidas neste instante.)

Suponha $Q(k)$: $j_k = i_k * y$.
Mostre $Q(k+1)$: $j_{k+1} = i_{k+1} * y$

Entre o instante em que j e i têm os valores j_k e i_k e o instante em que j e i têm os valores j_{k+1} e i_{k+1}, ocorre uma iteração do laço. Nesta iteração, j muda adicionando-se y a seu valor anterior e i muda adicionando-se 1. Ou seja,

$$j_{k+1} = j_k + y \tag{3}$$
$$i_{k+1} = i_k + 1 \tag{4}$$

Então

$$\begin{aligned} j_{k+1} &= j_k + y && \text{(de (3))} \\ &= i_k * y + y && \text{(pela hipótese de indução)} \\ &= (i_k + 1)y \\ &= i_{k+1} * y && \text{(de (4))} \end{aligned}$$

Acabamos de provar que Q é um invariante do laço.

A regra de inferência do laço nos permite inferir que, ao sair do laço, a condição $Q \wedge B'$ é válida, o que, neste caso, se torna

$$j = i * y \wedge i = x$$

Portanto, neste instante, a proposição

$$j = x * y$$

é verdadeira, o que é exatamente o que a função é suposta de calcular. ●

O Exemplo 26 ilustra o fato de que invariantes do laço dizem algo mais forte sobre o programa do que o que queremos, de fato, mostrar; o que queremos mostrar é o caso particular do invariante ao final do laço. Encontrar o invariante do laço apropriado necessita de uma análise de trás para a frente, partindo da conclusão desejada, como no Exemplo 25.

Não provamos, na realidade, que o laço neste exemplo termina. O que provamos foi uma **correção parcial** — o programa produzirá a resposta correta se terminar. Como x é um inteiro não-negativo e i é um inteiro que começa em 0 e vai aumentando de 1 em 1 a cada iteração, sabemos que $i = x$ vai ter que ser verdade em algum instante.

PROBLEMA PRÁTICO 10

Mostre que a função a seguir dá como resposta o valor $x + y$ para inteiros não-negativos x e y, provando o invariante do laço Q: $j = x + i$ e calculando Q ao final do laço.

```
Soma(inteiro não-negativo x; inteiro não-negativo y)
Variáveis locais:
inteiros i, j
  i = 0
  j = 0
  enquanto i ≠ y faça
    j = j + 1
    i = i + 1
  fim do enquanto
  //agora j tem o valor x + y
  retorne j
fim da função Soma
```

●

As duas funções, a do Exemplo 25 e a do Problema Prático 10, não são muito realistas; afinal de contas, se quiséssemos calcular $x * y$ ou $x + y$ poderíamos, sem dúvida, usar um único comando. No entanto, as mesmas técnicas podem ser aplicadas a cálculos mais significativos, como o algoritmo de Euclides.

O Algoritmo de Euclides

O **algoritmo de Euclides** foi elaborado pelo matemático grego Euclides há mais de 2300 anos, o que o torna um dos algoritmos mais antigos conhecidos. Este algoritmo encontra o maior divisor comum entre dois inteiros não-negativos a e b, onde ambos não são nulos ao mesmo tempo. O **máximo divisor comum** de a e b, denotado por mdc(a, b), é o maior inteiro que divide (sem resto) tanto a quanto b. Por exemplo, mdc(12, 18) é 6, mdc(420, 66) = 6 e mdc(18, 0) = 18 (este último porque qualquer número diferente de 0 divide 0).

O algoritmo de Euclides funciona por meio de uma sucessão de divisões. Para encontrar mdc(a, b), supondo que $a \geq b$, divida, primeiro, a por b, obtendo um quociente e um resto. Formalmente, neste instante temos $a = q_1 b + r_1$, onde $0 \leq r_1 < b$. A seguir, divida o divisor, b, pelo resto r_1, obtendo $b = q_2 r_1 + r_2$, onde $0 \leq r_2 < r_1$. Novamente, divida o divisor, r_1, pelo resto r_2, obtendo $r_1 = q_3 r_2 + r_3$, onde $0 \leq r_3 < r_2$. É claro que temos aqui um processo em laço. Este processo termina quando encontramos um resto 0; o máximo divisor comum é o último divisor utilizado.

EXEMPLO 27

Para encontrar mdc(420, 66) são efetuadas as seguintes divisões:

$$\begin{array}{r|l} 420 & 66 \\ \underline{396} & 6 \\ 24 & \end{array} \qquad \begin{array}{r|l} 66 & 24 \\ \underline{48} & 2 \\ 18 & \end{array} \qquad \begin{array}{r|l} 24 & 18 \\ \underline{18} & 1 \\ 6 & \end{array} \qquad \begin{array}{r|l} 18 & 6 \\ \underline{18} & 3 \\ 0 & \end{array}$$

A resposta é 6, o divisor utilizado quando temos resto 0. ●

A seguir apresentamos uma versão em pseudocódigo do algoritmo na forma de uma função que retorna o valor mdc(a, b).

ALGORITMO Algoritmo de Euclides

MDC(inteiro não-negativo a; inteiro não-negativo b)
//$a \geq b$, um dos dois diferentes de zero
Variáveis locais:
inteiros i, j
  $i = a$
  $j = b$
  enquanto $j \neq 0$ **faça**
    calcule $i = qj + r, 0 \leq r < j$
    $i = j$
    $j = r$
  fim do enquanto
  //agora i tem o valor do mdc(a, b)
  retorne i
fim da função MDC

Queremos mostrar que esta função está correta, mas precisamos, primeiro, de um fato adicional, a saber,

$$(\forall \text{ inteiros } a, b, q, r)[(a = qb + r) \rightarrow (\text{mdc}(a, b) = \text{mdc}(b, r))] \tag{5}$$

Para provar (5), vamos supor que $a = qb + r$ e que c divide tanto a quanto b, de modo que $a = q_1c$ e $b = q_2c$. Então

$$r = a - qb = q_1c - qq_2c = c(q_1 - qq_2)$$

de modo que c também divide r. Portanto, qualquer inteiro que divida a e b também divide b e r. Suponha, agora, que d divide tanto b quanto r, de modo que $b = q_3d$ e $r = q_4d$. Então

$$a = qb + r = qq_3d + q_4d = d(qq_3 + q_4)$$

logo d também divide a. Portanto, qualquer inteiro que divida b e r também divide a e b. Como (a, b) e (b, r) têm precisamente os mesmos divisores, eles têm que ter o mesmo máximo divisor comum.

EXEMPLO 28

Prove que o algoritmo de Euclides está correto.

Usando a função MDC, vamos provar o invariante de laço Q: mdc(i, j) = mdc(a, b) e calcular Q quando o laço terminar. Usaremos indução para provar $Q(n)$: mdc(i_n, j_n) = mdc(a, b) para todo $n \geq 0$. $Q(0)$ é a proposição

$$\text{mdc}(i_0, j_0) = \text{mdc}(a, b)$$

que é verdadeira, já que quando chegamos no laço pela primeira vez i e j têm os valores a e b, respectivamente.

Suponha $Q(k)$: mdc(i_k, j_k) = mdc(a, b)
Mostre $Q(k + 1)$: mdc(i_{k+1}, j_{k+1}) = mdc(a, b)

Pelos comandos de atribuição dentro do laço, sabemos que

$$i_{k+1} = j_k$$
$$j_{k+1} = r_k$$

Então,

$$\begin{aligned} \text{mdc}(i_{k+1}, j_{k+1}) &= \text{mdc}(j_k, r_k) \\ &= \text{mdc}(i_k, j_k) \quad \text{de (5)} \\ &= \text{mdc}(a, b) \quad \text{pela hipótese de indução} \end{aligned}$$

e, portanto, Q é um invariante do laço. Quando o laço termina, mdc(i, j) = mdc(a, b) e $j = 0$, logo mdc$(i, 0)$ = mdc(a, b). Mas mdc$(i, 0) = i$, de modo que i = mdc(a, b). Portanto a função MDC está correta. ●

Seção 2.3 Revisão

Técnicas

- Verificação da correção de um segmento de programa incluindo uma proposição com laço.
- Cálculo do mdc(a, b) usando o algoritmo de Euclides.

Idéias Principais

Podemos usar um invariante do laço, demonstrando por indução no número de iterações, para provar a correção de um programa.

Pode-se demonstrar que o algoritmo de Euclides clássico está correto.

Exercícios 2.3

Nos Exercícios de 1 a 4, prove que o pseudocódigo do segmento de programa está correto demonstrando o invariante do laço Q e calculando Q depois do laço terminar.

1. Função que retorna o valor de x^2 para $x \geq 1$

```
Quadrado(inteiro positivo x)
Variáveis locais:
inteiros i, j
    i = 1
    j = 1
    enquanto i ≠ x faça
        j = j + 2i + 1
        i = i + 1
    fim do enquanto
    //agora j tem o valor x²
    retorne j
fim da função Quadrado
    Q: j = i²
```

★2. Função que retorna o valor de $x!$ para $x \geq 1$

```
Fatorial(inteiro positivo x)
Variáveis locais:
inteiros i, j
    i = 2
    j = 1
    enquanto i ≠ x + 1 faça
        j = j * i
        i = i + 1
    fim do enquanto
    //agora j tem o valor x!
    retorne j
fim da função Fatorial
    Q: j = (i − 1)!
```

3. Função que retorna o valor de x^y para $x, y \geq 1$

```
Potência(inteiro positivo x; inteiro positivo y)
Variáveis locais:
inteiros i, j
    i = 1
    j = x
    enquanto i ≠ y faça
        j = j * x
        i = i + 1
    fim do enquanto
    //agora j tem o valor x^y
    retorne j
fim da função Potência
    Q: j = x^i
```

4. Função para calcular e escrever o quociente q e o resto r quando x é dividido por y, $x \geq 0, y \geq 1$

```
Divide(inteiro não-negativo x; inteiro positivo y)
Variáveis locais:
inteiros não-negativos q, r
    q = 0
    r = x
    enquanto r ≥ y faça
```

```
        q = q + 1
        r = r − y
    fim do enquanto
    //agora q e r são o quociente e o resto
    escreva("O quociente é" q "e o resto é" r)
fim da função Divide
    Q: x = q * y + r
```

Para os Exercícios de 5 a 8, use o algoritmo de Euclides para encontrar o máximo divisor comum dos números dados.

5. (2420, 70)
★6. (735, 90)
7. (1326, 252)
8. (1018215, 2695)

Nos Exercícios de 9 a 13, prove que o segmento de programa está correto encontrando e demonstrando o invariante do laço apropriado Q e calculando Q depois de o laço terminar.

9. Função que retorna o valor de $x - y$ para $x, y \geq 0$

```
Diferença(inteiro não-negativo x; inteiro não-negativo y)
Variáveis locais:
inteiros i, j
    i = 0
    j = x
    enquanto i ≠ y faça
        j = j − 1
        i = i + 1
    fim do enquanto
    //agora j tem o valor x − y
    retorne j
fim da função Diferença
```

★10. Função que retorna o valor de $x * y^n$ para $n \geq 0$

```
Cálculo(inteiro x; inteiro y; inteiro não-negativo n)
Variáveis locais:
inteiros i, j
    i = 0
    j = x
    enquanto i ≠ n faça
        j = j * y
        i = i + 1
    fim do enquanto
    //agora j tem o valor x * y^n
    retorne j
fim da função Cálculo
```

11. Função que retorna o valor de 2^n para $n \geq 1$

```
PotênciasDe2(inteiro positivo n)
Variáveis locais:
inteiros i, j
    i = 1
    j = 2
    enquanto i ≠ n faça
        j = j * 2
        i = i + 1
    fim do enquanto
    //agora j tem o valor 2^n
    retorne j
fim da função PotênciasDe2
```

★12. Função que retorna o valor de $x * n!$ para $n \geq 1$

```
Outra(inteiro x; inteiro positivo n)
Variáveis locais:
inteiros i, j
    i = 1
```

```
    j = x
    enquanto i ≠ n faça
      j = j * (i + 1)
      i = i + 1
    fim do enquanto
    //agora j tem o valor x * n!
    retorne j
  fim da função Outra
```

13. Função que retorna o valor do polinômio

$$a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0$$

```
para um valor dado de x
  Polinômio(real a_n; ...; real a_0; real x)
  Variáveis locais:
  inteiros i, j
    i = n
    j = a
    enquanto i ≠ 0 faça
      j = j * x + a_{i-1}
      i = i - 1
    fim do enquanto
    //agora j tem o valor do polinômio em x
    retorne j
  fim da função Polinômio
```

Seção 2.4 Recursão e Relações de Recorrência

Definições Recorrentes

Uma definição onde o item sendo definido aparece como parte da definição é chamada de uma **definição recorrente** ou **definição por recorrência** ou ainda **definição por indução**. A princípio isto não parece fazer sentido — como podemos definir alguma coisa em termos de si mesma? Isto funciona porque uma definição recorrente tem duas partes:

1. Uma base, ou condição básica, onde alguns casos simples (pelo menos um) do item que está sendo definido é dado explicitamente.
2. Um passo de indução ou recorrência, onde novos casos do item que está sendo definido são dados em função de casos anteriores.

A parte 1 nos dá um lugar para começar, fornecendo alguns casos simples e concretos; a parte 2 nos permite construir novos casos, a partir destes simples, e depois construir ainda outros casos a partir destes novos, e assim por diante. (O nome "definição por indução" é devido à analogia com demonstrações por indução matemática. Em uma demonstração por indução existe uma base da indução, a saber, mostrar que $P(1)$ — ou P em algum outro valor inicial — é verdadeira, e existe um passo indutivo, onde a veracidade de $P(k+1)$ é estabelecida a partir da veracidade de P em valores anteriores.)

Recorrência é uma idéia importante que pode ser usada para definir seqüências de objetos, coleções mais gerais de objetos e operações com objetos. (O predicado Prolog *na-cadeia-alimentar* da Seção 1.5 foi definido de forma recorrente.) Até algoritmos podem ser recorrentes.

Seqüências Definidas por Recorrência

Uma **seqüência** S é uma lista de objetos que são numerados em determinada ordem; existe um primeiro objeto, um segundo, e assim por diante. $S(k)$ denota o k-ésimo objeto na seqüência. Uma seqüência é definida por recorrência nomeando-se, explicitamente, o primeiro valor (ou alguns poucos primeiros valores) na seqüência e depois definindo valores subseqüentes na seqüência em termos de valores anteriores.

EXEMPLO 29 A seqüência S é definida por recorrência por

1. $S(1) = 2$
2. $S(n) = 2S(n-1)$ para $n \geq 2$

Pela proposição 1, $S(1)$, o primeiro objeto em S, é 2. Depois, pela proposição 2, o segundo objeto em S é $S(2) = 2S(1) = 2(2) = 4$. Novamente pela proposição 2, $S(3) = 2S(2) = 2(4) = 8$. Continuando deste modo, vemos que a seqüência S é

2, 4, 8, 16, 32, ...

Uma regra como a da proposição 2 no Exemplo 29, que define um valor de uma seqüência em termos de um ou mais valores anteriores, é chamada uma **relação de recorrência**.

PROBLEMA PRÁTICO 11

A seqüência T é definida por recorrência por:

1. $T(1) = 1$
2. $T(n) = T(n - 1) + 3$ para $n \geq 2$

Escreva os cinco primeiros valores da seqüência T.

EXEMPLO 30

A famosa **seqüência de Fibonacci**, introduzida no século XIII por um comerciante e matemático italiano, é definida por recorrência por

$F(1) = 1$
$F(2) = 1$
$F(n) = F(n - 2) + F(n - 1)$ para $n > 2$

Aqui são dados os dois primeiros valores da seqüência e a relação de recorrência define o n-ésimo valor em termos dos dois valores precedentes. É melhor pensar na relação de recorrência em sua forma mais geral, que diz que F em qualquer valor — exceto em 1 e 2 — é a soma de F em seus dois valores anteriores.

PROBLEMA PRÁTICO 12

Escreva os oito primeiros valores da seqüência de Fibonacci.

EXEMPLO 31

Prove que, na seqüência de Fibonacci,

$F(n + 4) = 3F(n + 2) - F(n)$ para todo $n \geq 1$

Como queremos provar que alguma coisa é verdadeira para todo $n \geq 1$, é natural pensar em uma demonstração por indução. E, como o valor de $F(n)$ depende de $F(n - 1)$ e de $F(n - 2)$, deve-se usar o segundo princípio de indução. Para a base da indução, vamos provar dois casos, $n = 1$ e $n = 2$. Para $n = 1$, obtemos

$F(5) = 3F(3) - F(1)$

ou (usando os valores calculados no Problema Prático 12)

$5 = 3(2) - 1$

que é verdade. Para $n = 2$,

$F(6) = 3F(4) - F(2)$

ou

$8 = 3(3) - 1$

que também é verdade. Suponha que, para todo r, $1 \leq r \leq k$,

$F(r + 4) = 3F(r + 2) - F(r)$

Vamos mostrar o caso $k + 1$, onde $k + 1 \geq 3$. Queremos mostrar, então, que

$F(k + 1 + 4) \stackrel{?}{=} 3F(k + 1 + 2) - F(k + 1)$

ou

$F(k + 5) \stackrel{?}{=} 3F(k + 3) - F(k + 1)$

Da relação de recorrência para a seqüência de Fibonacci, temos

$F(k + 5) = F(k + 3) + F(k + 4)$ (F em qualquer valor é a soma de F nos dois valores anteriores)

e, pela hipótese de indução, com $r = k - 1$ e $r = k$, respectivamente, temos

$F(k + 3) = 3F(k + 1) - F(k - 1)$

e

$F(k + 4) = 3F(k + 2) - F(k)$

Portanto,

$$\begin{aligned} F(k+5) &= F(k+3) + F(k+4) \\ &= [3F(k+1) - F(k-1)] + [3F(k+2) - F(k)] \\ &= 3[F(k+1) + F(k+2)] - [F(k-1) + F(k)] \\ &= 3F(k+3) - F(k+1) \qquad \text{(usando novamente a relação de recorrência)} \end{aligned}$$

Isto completa a demonstração por indução. ●

EXEMPLO 32

A fórmula

$$F(n+4) = 3F(n+2) - F(n) \text{ para todo } n \geq 1$$

do Exemplo 31, também pode ser provada diretamente, sem indução, usando apenas a relação de recorrência na definição dos números de Fibonacci. A relação de recorrência

$$F(n+2) = F(n) + F(n+1)$$

pode ser reescrita na forma

$$F(n+1) = F(n+2) - F(n) \qquad (1)$$

Logo,

$$\begin{aligned} F(n+4) &= F(n+3) + F(n+2) \\ &= F(n+2) + F(n+1) + F(n+2) && \text{reescrevendo } F(n+3) \\ &= F(n+2) + [F(n+2) - F(n)] + F(n+2) && \text{reescrevendo } F(n+1) \text{ usando (1)} \\ &= 3F(n+2) - F(n) \end{aligned}$$

●

PROBLEMA PRÁTICO 13

Na demonstração por indução do Exemplo 31, por que é necessário provar o caso $n = 2$ como um caso particular? ●

Conjuntos Definidos por Recorrência

Os objetos em uma seqüência são ordenados — existe um primeiro objeto, um segundo e assim por diante. Um conjunto de objetos é uma coleção na qual não há nenhuma ordem imposta. Alguns conjuntos podem ser definidos por recorrência.

EXEMPLO 33

Na Seção 1.1, notamos que certas cadeias de letras de proposição, conectivos lógicos e parênteses, tais como $(A \vee B)' \wedge C$, são consideradas legítimas, enquanto outras, como $\wedge \wedge A''B$, não o são. A sintaxe para arrumar tais símbolos constitui a definição do conjunto de fórmulas proposicionais bem-formuladas e é uma definição por recorrência.

1. Qualquer letra de proposição é uma fbf.
2. Se P e Q são fbfs, então $(P \wedge Q)$, $(P \vee Q)$, $(P \rightarrow Q)$, (P') e $(P \leftrightarrow Q)$ também o são.

Muitas vezes omitimos os parênteses quando isto não causa confusão; assim, podemos escrever $(P \vee Q)$ como $P \vee Q$, ou (P') como P'. Começando com letras de proposição e usando, repetidamente, a regra 2, podemos construir todas as fbfs proposicionais. Por exemplo, A, B e C são todas as fbfs pela regra 1. Pela regra 2,

$$(A \wedge B) \quad \text{e} \quad (C')$$

são, ambas, fbfs. Novamente pela regra 2,

$$((A \wedge B) \rightarrow (C'))$$

é uma fbf. Aplicando a regra 2 mais uma vez, obtemos a fbf

$$(((A \wedge B) \rightarrow (C'))')$$

Eliminando alguns parênteses, podemos escrever esta fbf como

$$((A \wedge B) \rightarrow C')'$$

●

PROBLEMA PRÁTICO 14

Mostre como construir a fbf $((A \vee (B') \rightarrow C)$ da definição no Exemplo 33. ●

PROBLEMA PRÁTICO 15

Uma definição por recorrência para um conjunto de pessoas que são ancestrais de João poderia ter a seguinte base:

Os pais de João são seus ancestrais.

Dê o passo indutivo. ●

Cadeias de símbolos retiradas de um "alfabeto" finito são objetos encontrados com freqüência em ciência da computação. Computadores guardam os dados como **cadeias binárias**, cadeias do alfabeto que consiste apenas em 0s e 1s; compiladores vêem proposições ou comandos em programas como cadeias de *marcas* ou *sinais*, tais como palavras-chave e identificadores. A coleção de todas as cadeias de comprimento finito formada por símbolos de um alfabeto, chamadas de cadeias *de* um alfabeto, podem ser definidas de forma recorrente (veja o Exemplo 34). Muitos conjuntos de cadeias com propriedades particulares também têm definições recorrentes.

EXEMPLO 34

O conjunto de todas as cadeias (de comprimento finito) de símbolos de um alfabeto A é denotado por $A*$. A definição recorrente de $A*$ é

1. A **cadeia vazia** λ (a cadeia sem nenhum símbolo) pertence a $A*$.
2. Um único elemento qualquer de A pertence a $A*$.
3. Se x e y são cadeias em $A*$, então a **concatenação** xy de x e y também pertence a $A*$.

As partes 1 e 2 constituem a base e a parte 3 é o passo indutivo desta definição. Note que, para qualquer cadeia x, $x\lambda = \lambda x = x$. ●

Se $x = 1011$ e $y = 001$, escreva as cadeias xy, yx e $yx\lambda x$. ● PROBLEMA PRÁTICO 16

Dê uma definição recorrente para o conjunto de todas as cadeias binárias que são **palíndromos**, cadeias que são iguais se lidas normalmente e de trás para a frente. ● PROBLEMA PRÁTICO 17

EXEMPLO 35

Suponha que, em determinada linguagem de programação, os identificadores podem ser cadeias alfanuméricas de comprimento arbitrário, mas têm que começar com uma letra. Uma definição recorrente para o conjunto destas cadeias é

1. Uma única letra é um identificador.
2. Se A é um identificador, a concatenação de A e qualquer letra ou dígito também o é.

Uma notação mais simbólica para descrever conjuntos de cadeias definidas por recorrência é chamada de **forma de Backus Naur**, ou **FBN**, desenvolvida originalmente para definir a linguagem de programação ALGOL. Em notação FBN, os itens que são definidos em termos de outros itens são envolvidos pelos símbolos de menor e maior (< >), enquanto itens específicos que não podem ser divididos não aparecem desta forma. Um segmento vertical | denota uma escolha e tem o mesmo significado que a palavra *ou*. A definição em FBN de um identificador é

<identificador> ::= <letra> | <identificador> <letra> | <identificador> <dígito>
<letra> ::= *a* | *b* | *c* | ... | *z*
<dígito> ::= 1 | 2 | ... | 9

Assim, o identificador *me*2 pode ser obtido da definição por uma seqüência de escolhas como

<identificador>	pode ser	<identificador> <dígito>
	que pode ser	<identificador>2
	que pode ser	<identificador> <letra>2
	que pode ser	<identificador>*e*2
	que pode ser	<letra>*e*2
	que pode ser	*me*2 ●

Operações Definidas por Recorrência

Certas operações em objetos podem ser definidas de forma recorrente, como nos Exemplos 36 e 37.

EXEMPLO 36

Uma definição recorrente da operação de exponenciação a^n de um número real não-nulo a, onde n é um inteiro não-negativo, é

1. $a^0 = 1$
2. $a^n = (a^{n-1})a$ para $n \geq 1$ ●

EXEMPLO 37

Uma definição recorrente para a multiplicação de dois inteiros positivos m e n é

1. $m(1) = m$
2. $m(n) = m(n-1) + m$ para $n \geq 2$ ●

Seja x uma cadeia de um determinado alfabeto. Dê uma definição recorrente da operação x^n (concatenação de x consigo mesmo n vezes) para $n \geq 1$. ● PROBLEMA PRÁTICO 18

Na Seção 1.1 definimos a operação de disjunção lógica de duas letras de proposição. Isto pode servir como base para uma definição recorrente da disjunção de n letras de proposição, $n \geq 2$:

1. $A_1 \vee A_2$ definida como na Seção 1.1
2. $A_1 \vee \cdots \vee A_n = (A_1 \vee \cdots \vee A_{n-1}) \vee A_n$ para $n > 2$ (2)

Usando esta definição, podemos generalizar a associatividade da disjunção (equivalência tautológica 2a) dizendo que, em uma disjunção de n letras de proposição, o agrupamento entre parênteses é desnecessário porque todos estes agrupamentos são equivalentes à expressão geral de n letras de proposição. Em forma simbólica, para qualquer n com $n \geq 3$ e qualquer p com $1 \leq p \leq n - 1$,

$$(A_1 \vee \cdots \vee A_p) \vee (A_{p+1} \vee \cdots \vee A_n) \Leftrightarrow A_1 \vee \cdots \vee A_n$$

Esta equivalência pode ser demonstrada por indução em n. Para $n = 3$,

$$A_1 \vee (A_2 \vee A_3) \Leftrightarrow (A_1 \vee A_2) \vee A_3 \quad \text{(pela equivalência 2a)}$$
$$= A_1 \vee A_2 \vee A_3 \quad \text{(pela equação (2))}$$

Suponha que, para $n = k$ e $1 \leq p \leq k - 1$,

$$(A_1 \vee \cdots \vee A_p) \vee (A_{p+1} \vee \cdots \vee A_k) \Leftrightarrow A_1 \vee \cdots \vee A_k$$

Então, para $n = k + 1$ e $1 \leq p \leq k$,

$$(A_1 \vee \cdots \vee A_p) \vee (A_{p+1} \vee \cdots \vee A_{k+1})$$
$$= (A_1 \vee \cdots \vee A_p) \vee [(A_{p+1} \vee \cdots \vee A_k) \vee A_{k+1}] \quad \text{(pela equação (2))}$$
$$\Leftrightarrow [(A_1 \vee \cdots \vee A_p) \vee (A_{p+1} \vee \cdots \vee A_k)] \vee A_{k+1} \quad \text{(pela equivalência 2a)}$$
$$\Leftrightarrow (A_1 \vee \cdots \vee A_k) \vee A_{k+1} \quad \text{(pela hipótese de indução)}$$
$$= A_1 \vee \cdots \vee A_{k+1} \quad \text{(pela equação (2))}$$

Algoritmos Definidos por Recorrência

O Exemplo 29 dá uma definição recorrente para uma seqüência S. Suponha que queremos escrever um programa de computador para calcular $S(n)$ para algum inteiro positivo n. Podemos usar uma entre duas abordagens. Se queremos encontrar $S(12)$, por exemplo, podemos começar com $S(1) = 2$ e depois calcular $S(2)$, $S(3)$, e assim por diante, como fizemos no Exemplo 29, até chegar, finalmente, em $S(12)$. Sem dúvida, esta abordagem envolve iteração em alguma espécie de laço. A seguir, vamos dar uma função em pseudocódigo S que usa este algoritmo iterativo. A base, com $n = 1$, é obtida na primeira cláusula da proposição **se**; o valor 2 é retornado. A cláusula **senão**, para $n > 1$, tem uma atribuição inicial e entra em um laço **enquanto**, que calcula valores maiores da seqüência até atingir o limite superior correto. Você pode seguir a execução deste algoritmo para alguns valores de n para se convencer de que ele funciona.

ALGORITMO

```
S(inteiro n)
//função que calcula iterativamente o valor S(n)
//para a seqüência S do Exemplo 29
Variáveis locais:
inteiro i              //índice do laço
ValorCorrente          //valor corrente da função S
   se n = 1 então
     retorne 2
   senão
     i = 2
     ValorCorrente = 2
     enquanto i <= n faça
       ValorCorrente = 2 * ValorCorrente
       i = i + 1
     fim do enquanto
     //agora o ValorCorrente tem o valor S(n)
     retorne ValorCorrente
   fim do se
fim da função S
```

A segunda abordagem para calcular $S(n)$ usa diretamente a definição recorrente de S. A versão a seguir é de um *algoritmo recorrente*, escrito, novamente, como uma função em pseudocódigo.

ALGORITMO

```
S(inteiro n)
//função que calcula o valor S(n) de forma recorrente
//para a seqüência S do Exemplo 29
   se n = 1 então
     retorne 2
   senão
     retorne 2 * S(n − 1)
   fim do se
fim da função S
```

O corpo dessa função consiste em uma única proposição do tipo **se-então-senão**. Para compreender como essa função funciona, vamos seguir a execução para calcular o valor de $S(3)$. Chamamos a função com um valor de entrada $n = 3$. Como n não é 1, a execução é direcionada para a cláusula **senão**. Neste instante, a atividade de calcular $S(3)$ tem que ser suspensa até se conhecer o valor de $S(2)$. Qualquer informação conhecida, relevante para o cálculo de $S(3)$, é armazenada na memória do computador em uma pilha, que será recuperada quando o cálculo for completado. (Uma pilha é uma coleção de dados onde qualquer novo item vai para o topo da pilha e, em qualquer instante, apenas o item do topo é acessível ou pode ser removido da pilha. Portanto, uma pilha é uma estrutura LIFO — do inglês *last in, first out*, isto é, o último a entrar é o primeiro a sair.) A função é chamada novamente com um valor de entrada $n = 2$. Mais uma vez, a cláusula **senão** é executada e o cálculo de $S(2)$ é suspenso, com as informações relevantes armazenadas na pilha, enquanto a função é chamada novamente com $n = 1$.

Desta vez é executada a primeira cláusula da proposição **se** e o valor da função, 2, pode ser calculado diretamente. Esta chamada final da função está completa, e o valor 2 é usado na penúltima chamada da função, que remove agora da pilha qualquer informação relevante ao caso $n = 2$, calcula $S(2)$ e usa este valor na invocação prévia (inicial) da função. Finalmente, esta chamada original de S é capaz de esvaziar a pilha e completar seu cálculo, retornando o valor de $S(3)$.

Quais são as vantagens relativas dos algoritmos iterativo e recorrente ao executar a mesma tarefa? Neste exemplo, a versão recorrente é certamente mais curta, já que não precisa gerenciar um cálculo em laço. A descrição da execução do algoritmo recorrente parece soar mais complicada do que a do algoritmo iterativo, mas todos os passos são executados automaticamente. Não precisamos estar conscientes do que está acontecendo internamente, exceto para observar que uma série longa de chamadas recorrentes pode usar muita memória ao armazenar na pilha as informações relevantes para as invocações prévias. Se a utilização da memória for excessiva, pode acontecer um "transbordamento" (*overflow*) da pilha. Além de usar mais memória, algoritmos recorrentes podem necessitar de mais cálculos e executar mais lentamente do que os não-recorrentes (veja o Exercício 7 no segmento No Computador, ao final deste capítulo).

De qualquer forma, a recorrência fornece um modo natural de pensar em muitas situações, algumas das quais necessitariam soluções não-recorrentes muito complexas. Uma solução recorrente é bem adequada para o problema de calcular os valores de uma seqüência definida de maneira recorrente. Muitas linguagens de programação aceitam recorrência.

PROBLEMA PRÁTICO 19

Escreva o corpo de uma função recorrente para calcular $T(n)$ para a seqüência T definida no Problema Prático 11. ●

EXEMPLO 38

No Exemplo 37, foi dada uma definição recorrente para a multiplicação de dois inteiros positivos m e n. A seguir temos uma função em pseudocódigo para a multiplicação baseada nessa definição. ●

ALGORITMO

```
Produto(inteiro m; inteiro n)
//função que calcula de forma recorrente o produto de m e n
   se n = 1 então
     retorne m
   senão
     retorne Produto(m, n − 1) + m
   fim do se
fim da função Produto
```

LEMBRETE: Pense em um algoritmo recorrente sempre que você puder resolver o problema a partir de soluções de versões menores do problema.

Um algoritmo recorrente chama a si mesmo com valores de entrada "menores". Suponha que um problema pode ser resolvido encontrando-se soluções para as versões menores do mesmo problema e que as versões menores acabam se tornando casos triviais, facilmente solucionados. Então, um algoritmo recorrente pode ser útil, mesmo que o problema original não tenha sido enunciado de forma recorrente.

Para nos convencermos de que um determinado algoritmo recorrente funciona, não precisamos começar com um dado particular de entrada, ir diminuindo, tratando de casos cada vez menores, e depois ir voltando, percorrendo todo o caminho inverso. Fizemos isto ao discutir o cálculo de $S(3)$, mas foi só para ilustrar a mecânica de um cálculo recorrente. Em vez disso, podemos verificar o caso trivial (como demonstrar a base em uma demonstração por indução) e verificar que, se o algoritmo funciona corretamente ao ser chamado com valores de entrada menores, então ele resolve, de fato, o problema para o valor de entrada original (o que é semelhante a provar $P(k + 1)$ da hipótese $P(k)$ em uma demonstração por indução).

EXEMPLO 39

Uma das tarefas mais comuns em processamento de dados é colocar uma lista L de n itens em ordem numérica ou alfabética, em ordem crescente ou decrescente. (Esta lista pode conter nomes de clientes, por exemplo, e, em ordem alfabética, "Vargas, Joana" deve vir depois de "Teixeira, José".) O algoritmo de **ordenação por seleção** — um algoritmo simples mas particularmente eficaz — é descrito em pseudocódigo mais adiante.

Esta função ordena os j primeiros itens em L em ordem crescente; quando a função é chamada pela primeira vez, j tem o valor n (de modo que a primeira chamada ordena toda a lista). A parte recorrente do algoritmo está dentro da cláusula **senão**; o algoritmo examina a seção da lista sob consideração e encontra o valor de i para o qual $L(i)$ tem o valor máximo. Ele, então, permuta $L(i)$ e $L(j)$, depois que o valor máximo ocorre na posição j, a última posição na parte da lista que está sendo considerada. $L(j)$ está correto agora e não deve mais ser modificado, de modo que este processo é repetido na lista de $L(1)$ a $L(j - 1)$. Se esta parte da lista for ordenada corretamente, então a lista inteira será ordenada corretamente. Quando j tem o valor 1, a parte da lista que está sendo considerada tem apenas um elemento, que tem que estar no lugar certo. Neste instante, a lista toda está ordenada.

ALGORITMO OrdenaçãoPorSeleção

```
OrdenaçãoPorSeleção(lista L; inteiro j)
//algoritmo recorrente para ordenar os itens de 1 a j em uma lista L em ordem crescente
   se j = 1 então
     a ordenação está completa, escreva a lista ordenada
   senão
     encontre o índice i do maior item em L entre 1 e j
     permute L(i) e L(j)
     OrdenaçãoPorSeleção(L, j − 1)
   fim do se
fim da função OrdenaçãoPorSeleção
```

EXEMPLO 40

Agora que ordenamos nossa lista, uma outra tarefa comum é procurar um item particular na lista. (Joana Vargas já é uma cliente?) Uma técnica eficiente de busca em uma lista ordenada é o **algoritmo** recorrente de **busca binária**, descrito em pseudocódigo a seguir.

ALGORITMO BuscaBinária

```
BuscaBinária(lista L; inteiro i; inteiro j; tipo item x)
//procura na lista ordenada L, de L(i) a L(j), pelo item x
   se i > j então
     escreva("não-encontrado")
   senão
     encontre o índice k do item do meio na lista L(i) − L(j)
     se x = item do meio então
       escreva("encontrado")
     senão
       se x < item do meio então
         BuscaBinária(L, i, k − 1, x)
       senão
         BuscaBinária(L, k + 1, j, x)
       fim do se
     fim do se
   fim do se
fim da função BuscaBinária
```

Este algoritmo procura na seção da lista entre $L(i)$ e $L(j)$ por um item x; inicialmente, i e j têm os valores 1 e n, respectivamente. A primeira cláusula do primeiro **se** é o passo básico que diz que x não pode ser encontrado em uma lista vazia, onde o primeiro índice é maior do que o último. Na cláusula **senão** principal, o item do meio em uma seção da lista tem que ser encontrado. (Se a seção tem um número ímpar de itens, existe, de fato, um item do meio; se a seção contém um número par de itens, basta escolher como "item do meio" o que fica no final da primeira metade da seção da lista.) Comparando x com o item do meio, localizamos a metade da lista onde devemos procurar a seguir. ●

EXEMPLO 41

Vamos aplicar o algoritmo de busca binária à lista

3, 7, 8, 10, 14, 18, 22, 34

onde o item a ser encontrado x é o número 25. A lista inicial não é vazia, logo o item do meio é localizado e encontra-se o valor 10. Então, x é comparado com o item do meio. Como $x > 10$, a busca é feita na segunda metade da lista, a saber, entre os itens

14, 18, 22, 34

Novamente, esta lista não é vazia e o item do meio é 18. Como $x > 18$, procura-se na segunda metade da lista, isto é, entre os itens

22, 34

Nesta lista não-vazia, o item do meio é 22. Como $x > 22$, a busca continua na segunda metade da lista, a saber,

34

Esta é uma lista de apenas um elemento, com o item do meio sendo este único elemento. Como $x < 34$, começa uma busca na "primeira metade" da lista; mas a primeira metade é vazia. O algoritmo termina aqui com a informação de que x não está na lista.

Essa execução necessita de quatro comparações ao todo; x é comparado com 10, 18, 22 e 34. ●

PROBLEMA PRÁTICO 20

Em uma busca binária da lista no Exemplo 41, nomeie os elementos que são comparados com x, se x tem o valor 8. ●

Vimos uma série de definições recorrentes. A Tabela 2.5 resume suas características.

Definições Recorrentes	
O que Está Sendo Definido	Características
Seqüência Recorrente	O primeiro ou os dois primeiros valores da seqüência são conhecidos; os outros elementos da seqüência são definidos em termos dos anteriores.
Conjunto Recorrente	Alguns elementos específicos do conjunto são conhecidos; outros elementos no conjunto são construídos a partir dos elementos que já sabemos que pertencem ao conjunto.
Operação Recorrente	Um caso "pequeno" da operação fornece um valor específico; outros casos são definidos a partir de casos menores.
Algoritmo Recorrente	Para o menor valor do(s) argumentos(s), o comportamento do algoritmo é conhecido; para valores maiores, o algoritmo chama a si mesmo com argumento(s) menor(es).

Tabela 2.5

Resolvendo Relações de Recorrência

Desenvolvemos dois algoritmos, um iterativo, o outro recorrente, para calcular um valor $S(n)$ para a seqüência S do Exemplo 29. No entanto, existe uma maneira ainda mais fácil para calcular $S(n)$. Lembre-se de que

$$S(1) = 2 \quad (1)$$
$$S(n) = 2S(n-1) \text{ para } n \geq 2 \quad (2)$$

Como

$$S(1) = 2 = 2^1$$
$$S(2) = 4 = 2^2$$
$$S(3) = 8 = 2^3$$
$$S(4) = 16 = 2^4$$

e assim por diante, vemos que

$$S(n) = 2^n \tag{3}$$

Usando a equação (3), podemos substituir um valor para n e calcular diretamente $S(n)$ sem ter que calcular — explicitamente ou por recorrência — todos os valores menores que S antes. Uma equação como (3), onde podemos substituir um valor e obter diretamente o que queremos, é chamada uma **solução em forma fechada** para a relação de recorrência (2) sujeita à condição básica (1). Quando encontramos uma solução em forma fechada, dizemos que **resolvemos** a relação de recorrência. É claro que sempre que possível seria bom encontrar uma solução em forma fechada.

Uma técnica para resolver relações de recorrência é uma abordagem do tipo "expandir, conjecturar e verificar", que usa repetidamente a relação de recorrência para expandir a expressão para o n-ésimo termo até podermos ter uma idéia da forma geral. Finalmente esta conjectura é verificada por indução matemática.

EXEMPLO 42

Considere novamente a condição básica e a relação de recorrência para a seqüência S do Exemplo 29:

$$S(1) = 2 \tag{4}$$
$$S(n) = 2S(n-1) \text{ para } n \geq 2 \tag{5}$$

Vamos fingir que não sabemos a solução em forma fechada e usar a abordagem de expandir, conjecturar e verificar. Começando com $S(n)$, expandimos usando, repetidamente, a relação de recorrência. Lembre-se sempre de que a relação de recorrência é uma receita que diz que qualquer elemento de S pode ser substituído por duas vezes o elemento anterior. Aplicamos essa receita a S para n, $n-1$, $n-2$, e assim por diante:

$$\begin{aligned} S(n) &= 2S(n-1) \\ &= 2[2S(n-2)] = 2^2S(n-2) \\ &= 2^2[2S(n-3)] = 2^3S(n-3) \end{aligned}$$

Olhando o padrão que está se desenvolvendo, conjecturamos que, após k expansões, a equação tem a forma

$$S(n) = 2^kS(n-k)$$

Esta expansão dos elementos de S em função de elementos anteriores tem que parar quando $k = n-1$. Neste ponto, temos

$$\begin{aligned} S(n) &= 2^{n-1}S[n-(n-1)] \\ &= 2^{n-1}S(1) = 2^{n-1}(2) = 2^n \end{aligned}$$

que expressa a solução em forma fechada.

Ainda não terminamos, no entanto, pois conjecturamos qual deveria ser a forma geral. Precisamos confirmar nossa solução em forma fechada por indução no valor de n. A proposição que queremos provar, portanto, é que $S(n) = 2^n$ para $n \geq 1$.

Para a base da indução, $S(1) = 2^1$. Isto é verdadeiro pela equação (4). Vamos supor que $S(k) = 2^k$. Então

$$\begin{aligned} S(k+1) &= 2S(k) && \text{(pela equação (5))} \\ &= 2(2^k) && \text{(pela hipótese de indução)} \\ &= 2^{k+1} \end{aligned}$$

Isso prova que nossa solução em forma fechada está correta. ●

PROBLEMA PRÁTICO 21

Encontre uma solução em forma fechada para a relação de recorrência, sujeita à condição básica, para a seqüência T.

1. $T(1) = 1$
2. $T(n) = T(n-1) + 3$ para $n \geq 2$

(*Dica*: Expanda, conjecture e verifique.) ●

Métodos para resolver relações de recorrência são um tanto como os usados para resolver equações diferenciais. Em particular, tanto as relações de recorrência quanto as equações diferenciais são classificadas em tipos, muitos dos quais têm fórmulas conhecidas para suas soluções.

Uma relação de recorrência para uma seqüência $S(n)$ é dita **linear** se os valores anteriores de S que aparecem na definição aparecem apenas na primeira potência. A relação de recorrência linear mais geral tem a forma

$$S(n) = f_1(n)S(n-1) + f_2(n)S(n-2) + \cdots + f_k(n)S(n-k) + g(n)$$

onde os coeficientes f_i e g podem ser expressões envolvendo n. A relação de recorrência tem **coeficientes constantes** se todos os f_i forem constantes. Ela é de **primeira ordem** se o n-ésimo termo depende apenas

do termo $n - 1$. Relações de recorrência lineares de primeira ordem com coeficientes constantes têm, portanto, a forma

$$S(n) = cS(n - 1) + g(n) \tag{6}$$

Finalmente, a relação de recorrência é **homogênea** se $g(n) = 0$ para todo n.

Vamos encontrar a fórmula para a solução da equação (6), a relação de recorrência linear de primeira ordem genérica com coeficientes constantes, sujeita à condição básica de que $S(1)$ seja conhecida. Vamos usar a abordagem de expandir, conjecturar e verificar. O que vamos fazer é uma generalização do que foi feito no Exemplo 42. Usando a equação (6) repetidamente e simplificando, obtemos

$$\begin{aligned} S(n) &= cS(n - 1) + g(n) \\ &= c[cS(n - 2) + g(n - 1)] + g(n) \\ &= c^2S(n - 2) + cg(n - 1) + g(n) \\ &= c^2\,[cS(n - 3) + g(n - 2)] + cg(n - 1) + g(n) \\ &= c^3S(n - 3) + c^2\,g(n - 2) + cg(n - 1) + g(n) \\ &\vdots \end{aligned}$$

Após k expansões, a forma geral parece ser

$$S(n) = c^kS(n - k) + c^{k-1}g(n - (k - 1)) + \cdots + cg(n - 1) + g(n)$$

Se o primeiro termo da seqüência é conhecido, então a expansão termina quando $n - k = 1$, ou $k = n - 1$, quando

$$\begin{aligned} S(n) &= c^{n-1}S(1) + c^{n-2}g(2) + \cdots + cg(n - 1) + g(n) \\ &= c^{n-1}S(1) + c^{n-2}g(2) + \cdots + c^1g(n - 1) + c^0g(n) \end{aligned} \tag{7}$$

Podemos usar a **notação de somatório** para escrever parte desta expressão de forma mais compacta. A letra grega maiúscula sigma, Σ, denota uma soma. A notação

$$\sum_{i=p}^{q} (\text{expressão})$$

diz para substituir os valores sucessivos de i, o **índice do somatório**, do limite inferior p até o limite superior q, e depois somar os resultados. (Veja o Apêndice A para uma discussão da notação de somatório.) Assim, por exemplo,

$$\sum_{i=1}^{n} (2i - 1) = 1 + 3 + 5 + \cdots + (2n - 1)$$

No Exemplo 14, Seção 2.2, provamos por indução que o valor desta soma é n^2.

Com a notação de somatório, a equação (7) fica

$$S(n) = c^{n-1}S(1) + \sum_{i=2}^{n} c^{n-i}g(i)$$

Pode-se usar indução, como no Exemplo 42, para verificar que esta fórmula é a solução da relação de recorrência (6) (veja o Exercício 82).

Portanto, a solução da relação de recorrência (6) é

$$S(n) = c^{n-1}S(1) + \sum_{i=2}^{n} c^{n-i}g(i) \tag{8}$$

Esta ainda não é, no entanto, uma solução em forma fechada, porque precisamos encontrar uma expressão para o somatório. Em geral, ou é trivial encontrar a soma ou já encontramos seu valor na Seção 2.2 usando indução matemática.

Encontramos, portanto, uma solução geral de uma vez por todas para qualquer relação de recorrência da forma (6); este processo *não precisa ser repetido*. Tudo o que é necessário é colocar seu problema na forma (6) para encontrar o valor de c e a fórmula para $g(n)$, e depois substituir estes valores na expressão em (8).

EXEMPLO 43

A seqüência $S(n)$ do Exemplo 42,

$$S(1) = 2$$

$$S(n) = 2S(n - 1) \text{ para } n \geq 2$$

é uma relação de recorrência linear homogênea de primeira ordem com coeficientes constantes. Em outras palavras, ela coincide com a equação (6) com $c = 2$ e $g(n) = 0$. Da fórmula (8), a solução em forma

fechada é

$$S(n) = 2^{n-1}(2) + \sum_{i=2}^{n} 0 = 2^n$$

o que está de acordo com nosso resultado anterior.

PROBLEMA PRÁTICO 22 Refaça o Problema Prático 21 usando a equação (8).

Temos agora duas maneiras alternativas para resolver uma relação de recorrência linear de primeira ordem com coeficientes constantes. A Tabela 2.6 resume estas abordagens.

Para Resolver Relações de Recorrência da Forma $S(n) = cS(n-1) + g(n)$ Sujeitas à Condição Básica $S(1)$	
Método	**Passos**
Expandir, conjecturar, verificar	1. Use a relação de recorrência repetidamente até poder adivinhar o padrão. 2. Decida qual será o padrão quando $n - k = 1$. 3. Verifique a fórmula resultante por indução.
Fórmula da solução	1. Coloque sua relação de recorrência na forma $S(n) = cS(n-1) + g(n)$ para encontrar c e $g(n)$. 2. Use c, $g(n)$ e $S(1)$ na fórmula $S(n) = c^{n-1}S(1) + \sum_{i=2}^{n} c^{n-i}\, g(i)$ 3. Calcule o somatório resultante para obter a expressão final.

Tabela 2.6

EXEMPLO 44 Encontre uma solução em forma fechada para a relação de recorrência

$$S(n) = 2S(n-1) + 3 \text{ para } n \geq 2$$

sujeita à condição básica

$$S(1) = 4$$

Usaremos o método da fórmula da solução. Comparando nossa relação de recorrência

$$S(n) = 2S(n-1) + 3$$

com a forma geral $S(n) = cS(n-1) + g(n)$, vemos que

$$c = 2 \qquad g(n) = 3$$

O fato de que $g(n) = 3$ diz que g tem um valor constante de 3 independentemente do valor de n. Substituindo na forma geral da solução $S(n) = c^{n-1}S(1) + \sum_{i=2}^{n} c^{n-i}g(i)$, obtemos

$$\begin{aligned} S(n) &= 2^{n-1}(4) + \sum_{i=2}^{n} 2^{n-i}(3) \\ &= 2^{n-1}(2^2) + 3\sum_{i=2}^{n} 2^{n-i} \\ &= 2^{n+1} + 3[2^{n-2} + 2^{n-3} + \cdots + 2^1 + 2^0] \\ &= 2^{n+1} + 3[2^{n-1} - 1] \end{aligned}$$

(do Exemplo 15)

Logo, o valor de $S(5)$, por exemplo, é $2^6 + 3(2^4 - 1) = 64 + 3(15) = 109$.

LEMBRETE: Ao expandir, certifique-se de que colocou todas as partes da receita da relação de recorrência, como o +3 neste exemplo.

De maneira alternativa, usando o método de expandir, conjecturar e verificar, expandimos

$$\begin{aligned} S(n) &= 2S(n-1) + 3 \\ &= 2[2S(n-2) + 3] + 3 = 2^2S(n-2) + 2 \cdot 3 + 3 \\ &= 2^2[2S(n-3) + 3] + 2 \cdot 3 + 3 = 2^3S(n-3) + 2^2 \cdot 3 + 2 \cdot 3 + 3 \\ &\vdots \end{aligned}$$

Esta forma geral parece ser

$$S(n) = 2^k S(n-k) + 2^{k-1} \cdot 3 + 2^{k-2} \cdot 3 + \cdots + 2^2 \cdot 3 + 2 \cdot 3 + 3$$

que, quando $n - k = 1$ ou $k = n - 1$, fica

$$\begin{aligned} S(n) &= 2^{n-1}S(1) + 2^{n-2} \cdot 3 + 2^{n-3} \cdot 3 + \cdots + 2^2 \cdot 3 + 2 \cdot 3 + 3 \\ &= 2^{n-1}(4) + 3[2^{n-2} + 2^{n-3} + \cdots + 2^2 + 2 + 1] \\ &= 2^{n+1} + 3[2^{n-1} - 1] \qquad \text{(do Exemplo 15)} \end{aligned}$$

Finalmente, precisamos provar por indução que $S(n) = 2^{n+1} + 3[2^{n-1} - 1]$.

Base: $n = 1$: $S(1) = 4 = 2^2 + 3[2^0 - 1]$, verdadeiro
Suponha que $S(k) = 2^{k+1} + 3[2^{k-1} - 1]$
Mostre que $S(k+1) = 2^{k+2} + 3[2^k - 1]$

$$\begin{aligned} S(k+1) &= 2S(k) + 3 && \text{pela relação de recorrência} \\ &= 2(2^{k+1} + 3[2^{k-1} - 1]) + 3 && \text{pela hipótese de indução} \\ &= 2^{k+2} + 3 \cdot 2^k - 6 + 3 && \text{colocando o 3 em evidência} \\ &= 2^{k+2} + 3[2^k - 1] \end{aligned}$$

Seção 2.4 Revisão

Técnicas

- Geração de valores em uma seqüência definida por recorrência.
- Demonstração de propriedades da seqüência de Fibonacci.
- Reconhecimento de objetos em uma coleção definida por recorrência.
- Obtenção de definições recorrentes para um conjunto particular de objetos.
- Obtenção de definições recorrentes para determinadas operações em objetos.
- Obtenção de algoritmos recorrentes para gerar seqüências definidas por recorrência.
- Resolução de relações de recorrência lineares de primeira ordem com coeficientes constantes através da fórmula da solução.
- Resolução de relações de recorrência pelo método de expandir, conjecturar e verificar.

Idéias Principais

Podem ser dadas definições recorrentes para seqüências de objetos, conjuntos de objetos e operações em objetos, onde a condição básica é conhecida e novas informações dependem de informações já conhecidas.

Algoritmos recorrentes fornecem um modo natural de resolver determinados problemas chamando a mesma tarefa para versões menores do problema.

Certas relações de recorrência têm solução em forma fechada.

Exercícios 2.4

Para os Exercícios de 1 a 10, escreva os cinco primeiros valores da seqüência.

★**1.** $S(1) = 10$
$S(n) = S(n-1) + 10$ para $n \geq 2$

2. $A(1) = 2$
$A(n) = \dfrac{1}{A(n-1)}$ para $n \geq 2$

3. $B(1) = 1$
$B(n) = B(n-1) + n^2$ para $n \geq 2$

★**4.** $S(1) = 1$
$S(n) = S(n-1) + \dfrac{1}{n}$ para $n \geq 2$

5. $T(1) = 1$
$T(n) = nT(n-1)$ para $n \geq 2$

6. $P(1) = 1$
$P(n) = n^2P(n-1) + (n-1)$ para $n \geq 2$

★7. $M(1) = 2$
$M(2) = 2$
$M(n) = 2M(n-1) + M(n-2)$ para $n > 2$

8. $D(1) = 3$
$D(2) = 5$
$D(n) = (n-1)D(n-1) + (n-2)D(n-2)$ para $n > 2$

9. $W(1) = 2$
$W(2) = 3$
$W(n) = W(n-1)W(n-2)$ para $n > 2$

10. $T(1) = 1$
$T(2) = 2$
$T(3) = 3$
$T(n) = T(n-1) + 2T(n-2) + 3T(n-3)$ para $n > 3$

Nos Exercícios de 11 a 15, prove a propriedade dada dos números de Fibonacci diretamente da definição.

★11. $F(n+1) + F(n-2) = 2F(n)$ para $n \geq 3$

12. $F(n) = 5F(n-4) + 3F(n-5)$ para $n \geq 6$

13. $[F(n+1)]^2 = [F(n)]^2 + F(n-1)F(n+2)$ para $n \geq 2$

14. $F(n+3) = 2F(n+1) + F(n)$ para $n \geq 1$

15. $F(n+6) = 4F(n+3) + F(n)$ para $n \geq 1$

Nos Exercícios de 16 a 19, prove a propriedade dada dos números de Fibonacci para todo $n \geq 1$. (*Dica*: O primeiro princípio de indução vai funcionar.)

★16. $F(1) + F(2) + \cdots + F(n) = F(n+2) - 1$

17. $F(2) + F(4) + \cdots + F(2n) = F(2n+1) - 1$

18. $F(1) + F(3) + \cdots + F(2n-1) = F(2n)$

19. $[F(1)]^2 + [F(2)]^2 + \cdots + [F(n)]^2 = F(n)F(n+1)$

Nos Exercícios de 20 a 23, prove a propriedade dada dos números de Fibonacci usando o segundo princípio de indução.

★20. Exercício 14

21. Exercício 15

22. $F(n) < 2^n$ para $n \geq 1$

23. $F(n) > \left(\frac{3}{2}\right)^{n-1}$ para $n \geq 6$

24. Os valores p e q são definidos da seguinte maneira:

$$p = \frac{1+\sqrt{5}}{2} \quad \text{e} \quad q = \frac{1-\sqrt{5}}{2}$$

a. Prove que $1 + p = p^2$ e $1 + q = q^2$.
b. Prove que

$$F(n) = \frac{p^n - q^n}{p - q}$$

c. Use a parte (b) para provar que

$$F(n) = \frac{\sqrt{5}}{5}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{\sqrt{5}}{5}\left(\frac{1-\sqrt{5}}{2}\right)^n$$

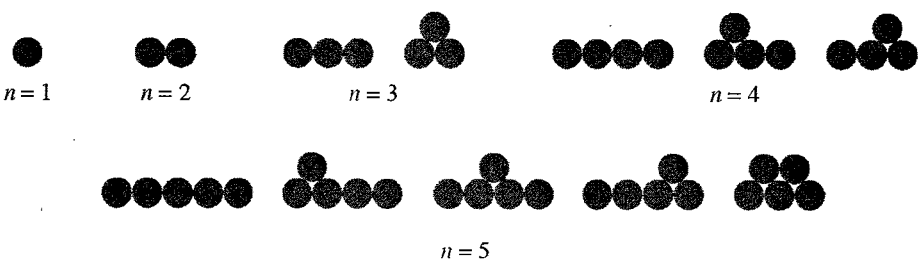
é uma solução em forma fechada para a seqüência de Fibonacci.

Para os Exercícios de 25 a 28, decida se as seqüências descritas são subseqüências da seqüência de Fibonacci, isto é, se seus elementos são alguns, ou todos, os elementos, na ordem correta, da seqüência de Fibonacci.[2]

25. A seqüência $A(n)$, onde $A(n) = (n - 1)2^{n-2} + 1$, $n \geq 1$. Os quatro primeiros valores são 1, 2, 5 e 13, os quais — até agora — formam uma subseqüência da seqüência de Fibonacci.

★26. A seqüência $B(n)$, onde $B(n) = 1 +$ (a soma dos n primeiros elementos da seqüência de Fibonacci), $n \geq 1$. Os quatro primeiros valores são 2, 3, 5 e 8, os quais — até agora — formam uma subseqüência da seqüência de Fibonacci.

27. A seqüência $C(n)$, onde $C(n)$ é o número de maneiras diferentes que podemos arrumar n moedas em fileiras horizontais de modo que todas as moedas em cada fileira fiquem encostadas e toda moeda acima da fileira mais embaixo fique encostada em duas moedas na fileira logo abaixo, $n \geq 1$. Os cinco primeiros valores são 1, 1, 2, 3 e 5, os quais — até agora — formam uma subseqüência da seqüência de Fibonacci.



28. A seqüência $D(n)$, onde $D(n)$ descreve o número de maneiras diferentes de pintar o chão de um prédio com n andares de modo que o chão de cada andar seja pintado de amarelo ou verde e que dois andares adjacentes não podem, ambos, ter o chão verde (embora dois andares adjacentes possam ter o chão amarelo), $n \geq 1$. Os quatro primeiros valores são 2, 3, 5 e 8, os quais — até agora — formam uma subseqüência da seqüência de Fibonacci. Por exemplo, $D(3) = 5$, já que um prédio de 3 andares pode ter o chão pintado de

A	A	A	V	V
A	A	V	A	A
A	V	A	V	A

(*Dica*: Pense em uma expressão recorrente para $D(n + 1)$.)

29. O problema original colocado por Fibonacci tratava de pares de coelhos. Dois coelhos não cruzam até terem dois meses de idade. Depois disto, cada par de coelhos produz um novo par a cada mês. Suponha que nenhum coelho morre. Denote por $C(n)$ o número de pares de coelhos ao final de n meses se você começa com um único par de coelhos. Mostre que $C(n)$ é a seqüência de Fibonacci.

★30. Escreva uma definição recorrente para uma progressão geométrica com termo inicial a e razão r (veja o Exercício 21, Seção 2.2).

31. Escreva uma definição recorrente para uma progressão aritmética com termo inicial a e parcela a ser somada d (veja o Exercício 22, Seção 2.2).

★32. Em um experimento, uma determinada colônia de bactérias tem uma população inicial de 50.000. A população é contada a cada 2 horas, e, ao final de cada intervalo de 2 horas, a população triplica.
a. Escreva uma definição recorrente para a seqüência $A(n)$, o número de bactérias presentes no início do n-ésimo período de tempo.
b. No início de que intervalo existirão 1.350.000 bactérias presentes?

33. Uma quantia de \$500,00 é investida em uma aplicação que paga juros de 10% capitalizados anualmente.
a. Escreva uma definição recorrente para $Q(n)$, a quantia total na aplicação no início do n-ésimo ano.
b. Depois de quantos anos de aplicação teremos um saldo maior do que \$700,00?

34. Uma coleção T de números é definida por recorrência por:
1. 2 pertence a T.
2. Se X pertence a T, então $X + 3$ e $2 * X$ também pertencem.

[2] Os Exercícios de 25 a 28 foram tirados da seção *Mathematical Recreations*, de Ian Stewart, na edição de maio de 1995 da revista *Scientific American*.

Quais dos números a seguir pertencem a T?

a. 6 **b.** 7 **c.** 19 **d.** 12

35. Uma coleção M de números é definida por recorrência por:
1. 2 e 3 pertencem a M.
2. Se X e Y pertencem a M, então $X * Y$ também pertence.
Quais dos números a seguir pertencem a M?

a. 6 **b.** 9 **c.** 16 **d.** 21 **e.** 26 **f.** 54 **g.** 72 **h.** 218

★36. Uma coleção S de cadeia de caracteres é definida por recorrência por:
1. a e b pertencem a S.
2. Se X pertence a S, então Xb também pertence.
Quais das cadeias a seguir pertencem a S?

a. a **b.** ab **c.** aba **d.** $aaab$ **e.** $bbbbb$

37. Uma coleção W de cadeias de símbolos é definida por recorrência por:
1. a, b e c pertencem a W.
2. Se X pertence a W, então $a(X)c$ também pertence.
Quais das cadeias a seguir pertencem a W?

a. $a(b)c$ **b.** $a(a(b)c)c$ **c.** $a(abc)c$ **d.** $a(a(a(a)c)c)c$
e. $a(aacc)c$

★38. Dê uma definição recorrente para o conjunto de todas as fbfs predicadas unárias em x.

39. Dê uma definição recorrente para o conjunto de todas as fórmulas bem-formuladas de aritmética inteira, envolvendo números inteiros e as operações aritméticas $+$, $-$, $*$ e $/$.

40. Dê uma definição recorrente para o conjunto de todas as cadeias de parênteses bem-balanceadas.

41. Dê uma definição recorrente para o conjunto de todas as cadeias binárias contendo um número ímpar de elementos iguais a 0.

★42. Dê uma definição recorrente para x^R, a cadeia em ordem inversa da cadeia x.

43. Dê uma definição recorrente para $|x|$, o comprimento da cadeia x.

★44. Use a notação FBN para definir o conjunto dos inteiros positivos.

45. Use a notação FBN para definir o conjunto dos números decimais, que consistem em um sinal opcional ($+$ ou $-$) seguido de um ou mais dígitos, seguido de uma vírgula, seguida de zeros ou mais dígitos.

★46. Dê uma definição recorrente para a operação fatorial $n!$ para $n \geq 1$.

47. Dê uma definição recorrente para a adição de dois inteiros não-negativos m e n.

48. **a.** Dê uma definição recorrente para a operação de escolher o máximo entre n inteiros $a_1, \ldots, a_n$, $n \geq 2$.
b. Dê uma definição recorrente para a operação de escolher o mínimo entre n inteiros $a_1, \ldots, a_n$, $n \geq 2$.

49. **a.** Dê uma definição recorrente para a conjunção de n letras de proposição em lógica proposicional, $n \geq 2$.
b. Escreva uma generalização da associatividade para a conjunção (equivalência tautológica 2b da Seção 1.1) e use indução para prová-la.

★50. Sejam A e $B_1, B_2, \ldots, B_n$ letras de proposição. Prove a extensão finita para as equivalências da distributividade na lógica proposicional:

$$A \vee (B_1 \wedge B_2 \wedge \cdots \wedge B_n) \Leftrightarrow (A \vee B_1) \wedge (A \vee B_2) \wedge \cdots \wedge (A \vee B_n)$$

e

$$A \wedge (B_1 \vee B_2 \vee \cdots \vee B_n) \Leftrightarrow (A \wedge B_1) \vee (A \wedge B_2) \vee \cdots \vee (A \wedge B_n)$$

para $n \geq 2$.

51. Sejam $B_1, B_2, \ldots, B_n$ letras de proposição. Prove a extensão finita para as leis de De Morgan:

$$(B_1 \vee B_2 \vee \cdots \vee B_n)' \Leftrightarrow B_1' \wedge B_2' \wedge \cdots \wedge B_n'$$

e

$$(B_1 \wedge B_2 \wedge \cdots \wedge B_n)' \Leftrightarrow B_1' \vee B_2' \vee \cdots \vee B_n'$$

para $n \geq 2$.

Nos Exercícios de 52 a 57, escreva o corpo de uma função recorrente para calcular $S(n)$, onde S é a seqüência dada.

52. 1, 3, 9, 27, 51, ...

53. 2, 1, 1/2, 1/4, 1/8, ...

★54. 1, 2, 4, 7, 11, 16, 22, ...

55. 2, 4, 16, 256, ...

56. $a, b, a + b, a + 2b, 2a + 3b, 3a + 5b, \ldots$

★57. $p, p - q, p + q, p - 2q, p + 2q, p - 3q, \ldots$

58. Qual o valor retornado pela função recorrente Mistério para um valor de entrada n?

```
Mistério (inteiro n)
  se n = 1 então
    retorne 1
  senão
    retorne Mistério(n − 1) + 1
  fim do se
fim da função Mistério
```

59. A função recorrente a seguir é chamada inicialmente com um valor de i igual a 1. L é uma lista (*array*) de 10 inteiros. O que a função faz?

```
g(lista L; inteiro i; inteiro x)
  se i > 10 então
    retorne 0
  senão
    se L(i) = x então
      retorne 10
    senão
      retorne g(L, i + 1, x)
    fim do se
  fim do se
fim da função g
```

★60. Descreva informalmente um algoritmo recorrente que inverte a ordem dos elementos em uma lista.

61. Descreva informalmente um algoritmo recorrente para calcular a soma dos dígitos de um inteiro positivo.

62. Descreva informalmente um algoritmo recorrente para calcular o máximo divisor comum de dois inteiros positivos a e b, onde $a \geq b$. (*Dica*: A solução baseia-se no algoritmo de Euclides, discutido na Seção 2.3. Em particular, use a expressão (5) do Exemplo 27, deste capítulo.)

★63. Simule a execução do algoritmo *OrdenaçãoPorSeleção* na lista L a seguir; escreva a lista após cada troca que muda a lista.

4, 10, −6, 2, 5

64. Simule a execução do algoritmo *OrdenaçãoPorSeleção* na lista L a seguir; escreva a lista após cada troca que muda a lista.

9, 0, 2, 6, 4

65. O algoritmo de busca binária é usado com a lista a seguir; x tem o valor "Curitiba". Diga com quais elementos x é comparado.

Brasília, Campos, Itapemirim, Nova Friburgo, Petrópolis, São Paulo, Varginha

66. O algoritmo de busca binária é usado com a lista a seguir; x tem o valor "farinha". Diga com quais elementos x é comparado.

açúcar, chocolate, farinha, manteiga, óleo, ovos

67. Demonstre a correção da função iterativa dada nesta seção para calcular $S(n)$ do Exemplo 29, onde $S(n) = 2^n$.

Nos Exercícios de 68 a 73, resolva a relação de recorrência sujeita à condição básica.

★**68.** $S(1) = 5$
$S(n) = S(n - 1) + 5$ para $n \geq 2$

69. $F(1) = 2$
$F(n) = 2F(n - 1) + 2^n$ para $n \geq 2$

70. $T(1) = 1$
$T(n) = 2T(n - 1) + 1$ para $n \geq 2$
(*Dica*: Veja o Exemplo 15.)

★**71.** $A(1) = 1$
$A(n) = A(n - 1) + n$ para $n \geq 2$
(*Dica*: Veja o Problema Prático 7.)

72. $S(1) = 1$
$S(n) = S(n - 1) + 2n - 1$ para $n \geq 2$
(*Dica*: Veja o Exemplo 14.)

73. $P(1) = 2$
$P(n) = 2P(n - 1) + n2^n$ para $n \geq 2$
(*Dica*: Veja o Problema Prático 7.)

Para os Exercícios 74 e 75, resolva a relação de recorrência sujeita à condição básica usando a abordagem de expandir, conjecturar e verificar.

★**74.** $F(1) = 1$
$F(n) = nF(n - 1)$ para $n \geq 2$

75. $S(1) = 1$
$S(n) = nS(n - 1) + n!$

76. Uma colônia de morcegos é contada a cada dois meses. As quatro primeiras contagens são 1.200, 1.800, 2.700 e 4.050. Se esta taxa de crescimento continuar, qual será a 12.ª contagem? (*Dica*: Escreva e resolva uma relação de recorrência.)

★**77.** No início deste capítulo, o empreiteiro afirmou:

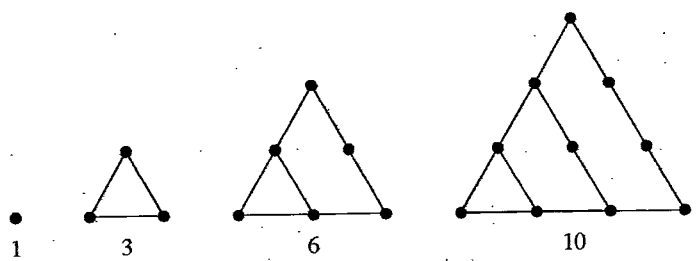
O material a ser estocado decai a uma taxa de 5% ao ano. Portanto, ao final de 20 anos, restará apenas aproximadamente um terço do material ativo original.

Escreva e resolva uma relação de recorrência para verificar a afirmação do empreiteiro; note que ao final de 20 anos começa o vigésimo primeiro ano.

78. Investe-se \$1.000,00 em uma aplicação que paga 8% de juros ao ano. Ao final de cada ano, aplica-se mais \$100,00. Qual o total da aplicação ao final de 7 anos (isto é, no início do oitavo ano)? (*Dica*: Escreva e resolva uma relação de recorrência. Consulte, também, o Exercício 21 da Seção 2.2 para a fórmula da soma de uma progressão geométrica.)

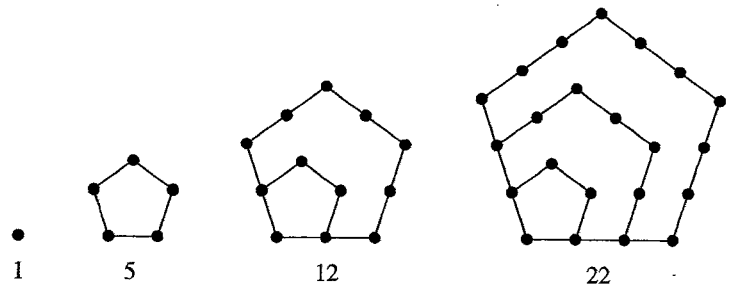
79. Estima-se que a população de mariscos em uma baía é de aproximadamente 1.000.000. Estudos mostram que a poluição reduz esta população em torno de 2% ao ano, enquanto outros fatores parecem reduzir esta população em 10.000 por ano. Após 9 anos, isto é, no início do décimo ano, qual a população aproximada de mariscos? (*Dica*: Escreva e resolva uma relação de recorrência. Consulte, também, o Exercício 21 da Seção 2.2 para a fórmula da soma de uma progressão geométrica.)

★**80.** Os primeiros membros da Associação de Pitágoras definiram *números poligonais* como sendo o número de pontos em determinadas configurações geométricas. Os primeiros *números triangulares* são 1, 3, 6 e 10:



Encontre uma fórmula para o n-ésimo número triangular. (*Dica*: Veja o Problema Prático 7.)

81. Os primeiros *números pentagonais* (veja o Exercício 80) são 1, 5, 12 e 22:



Encontre uma fórmula para o n-ésimo número triangular. (*Dica*: Veja o Exercício 22 da Seção 2.2 para a fórmula da soma de uma progressão aritmética.)

82. Use indução para verificar que a equação (8) desta seção é a solução da relação de recorrência (6) sujeita à condição básica de que $S(1)$ é conhecido.

Seção 2.5 Análise de Algoritmos

Muitas vezes existe mais de um algoritmo para executar a mesma tarefa. Ao comparar algoritmos, pode-se usar diversos critérios para se decidir qual o "melhor". Poderíamos perguntar, por exemplo, qual o mais fácil de entender ou qual o mais eficiente. Uma maneira de julgar a eficiência de um algoritmo é estimar o número de operações que ele tem que executar. Contamos apenas as operações básicas para a tarefa em questão, e não operações de "manutenção", que contribuem pouco para o "trabalho" total necessário.

Suponha, por exemplo, que a tarefa é procurar, em uma lista ordenada contendo n itens, um item particular x. Como qualquer algoritmo possível parece necessitar da comparação de elementos na lista com x até se encontrar um elemento igual, a operação básica a ser contada é a comparação.

O **algoritmo de busca seqüencial** compara, simplesmente, com cada elemento da lista até se encontrar x ou acabar a lista. A seguir, fornecemos uma descrição em pseudocódigo para o algoritmo de busca seqüencial. O número de operações executadas por este algoritmo vai ser máximo quando x é o último elemento da lista ou quando x não pertence à lista. Em qualquer destes casos, todos os elementos são comparados com x, logo fazem-se n comparações.

ALGORITMO BuscaSeqüencial

```
BuscaSeqüencial(lista L; inteiro n; tipo item x)
//procura em uma lista L com n itens pelo item x

Variável local:
inteiro i                //marca a posição na lista
  i = 1
  enquanto L(i) ≠ x e i ≤ n faça
    i = i + 1
  fim do enquanto
  se L(i) = x então
    escreva("Encontrado")
  senão
    escreva("Não-encontrado")
  fim do se
fim da função BuscaSeqüencial
```

É claro que x pode ser o primeiro elemento da lista, quando se faz apenas uma comparação; também pode acontecer de x estar no meio da lista, necessitando de aproximadamente $n/2$ comparações. Obviamente existem muitas possibilidades, e ajudaria se pudéssemos ter alguma idéia da quantidade média de operações necessárias. Isto necessitaria de alguma descrição da lista média e da relação média entre x e os itens daquela lista. Mas comportamento médio é, em geral, muito difícil de se determinar, não apenas para este, mas para a maioria dos algoritmos. Para comparar a eficiência dos algoritmos, portanto, vamos nos contentar, freqüentemente, com a contagem do número de operações no pior caso possível.

O estudo da eficiência dos algoritmos, isto é, o número de operações que eles executam, é chamado de **análise de algoritmos**. Foram desenvolvidas diversas técnicas para a análise de algoritmos. Algumas vezes pode-se fazer uma análise razoavelmente direta apenas inspecionando-se o algoritmo.

EXEMPLO 45

A seguir vem um algoritmo para escrever a soma de m notas de testes para cada aluno em uma lista, depois de se retirar a menor nota do aluno. O laço exterior passa por cada um dos n alunos; o laço interior percorre as notas do aluno corrente. O algoritmo atribui valores a variáveis e faz comparações (para encontrar a menor nota de cada aluno). Também executa somas e subtrações, e contaremos o número destas operações aritméticas.

```
para i = 1 até n faça
   menor = aluno(i).teste(1)
   soma = aluno(i).teste(1)
   para j = 2 até m faça
      soma = soma + aluno(i).teste(j)        //A
      se aluno(i).teste(j) < menor então
         menor = aluno(i).teste(j)
      fim do se
   fim do para
   soma = soma − menor;        //S
   escreva("Total para o aluno", i, "é", soma)
fim do para
```

A subtração ocorre na linha marcada com //S, que é executada uma vez em cada laço externo (para cada aluno), em um total de n vezes. A adição, no entanto, ocorre na linha marcada //A, no laço interno, que é executado $m - 1$ vezes para cada aluno, ou seja, para cada uma das n passagens do laço externo. O número total de adições, portanto, é $n(m - 1)$. O número total de operações aritméticas é $n + n(m - 1)$. O mesmo número de operações aritméticas é sempre feito; não existe caso melhor, nem pior, nem médio. ●

Vamos analisar nesta seção algoritmos recorrentes. Como a maior parte da atividade de um algoritmo recorrente acontece "fora das vistas", nas diversas chamadas que podem ocorrer, uma análise usando uma técnica de contagem direta como no Exemplo 45 não vai funcionar. A análise de algoritmos recorrentes envolve, muitas vezes, a resolução de uma relação de recorrência.

Análise Usando Relações de Recorrência (Busca Binária)

Como exemplo, vamos considerar um outro algoritmo (além da busca seqüencial) para se procurar em uma lista ordenada, o algoritmo de busca binária da Seção 2.4. Quantas comparações são necessárias, no pior caso, para o algoritmo de busca binária? Uma comparação é feita com o valor do meio da lista, depois o processo é repetido em metade da lista. Se a lista original tem n elementos, então a metade da lista tem, no máximo, $n/2$ elementos. (No Exemplo 41, por exemplo, onde 10 é o valor do meio, a "metade" da direita da lista tem 4 elementos, mas a "metade" da esquerda tem apenas 3.) Se vamos continuar cortando a lista pela metade, é conveniente considerar apenas o caso quando temos um número inteiro de elementos cada vez que dividimos ao meio a lista, de modo que vamos supor que $n = 2^m$ para algum $m \geq 0$. Se $C(n)$ denota o número máximo de comparações necessárias para uma lista de n elementos, então

$$C(n) = 1 + C\left(\frac{n}{2}\right)$$

Isto reflete uma comparação com o valor do meio seguida de quantas comparações forem necessárias para a metade da lista. Não sabemos, de fato, quantas comparações serão necessárias para metade da lista, assim como não sabemos quantas são necessárias para toda a lista, mas já temos uma notação para expressar este valor simbolicamente. Acabamos de escrever uma relação de recorrência. A condição básica é que

$$C(1) = 1$$

já que é necessária apenas uma comparação em uma lista com um só elemento.

Como esta relação de recorrência não é de primeira ordem, para resolvê-la vamos começar do início com o método de expandir, conjecturar, verificar. Além disto, a solução vai envolver a função logaritmo; para uma revisão desta função e de suas propriedades, veja o Apêndice B.

EXEMPLO 46

Resolva a relação de recorrência

$$C(n) = 1 + C\left(\frac{n}{2}\right) \text{ para } n \geq 2,\ n = 2^m$$

sujeita à condição básica

$C(1) = 1$

Expandindo, obtemos

$$C(n) = 1 + C\left(\frac{n}{2}\right)$$
$$= 1 + \left(1 + C\left(\frac{n}{4}\right)\right)$$
$$= 1 + 1 + \left(1 + C\left(\frac{n}{8}\right)\right)$$

e o termo geral parece ser

$$C(n) = k + C\left(\frac{n}{2^k}\right)$$

O processo pára quando $2^k = n$ ou $k = \log_2 n$. (Vamos omitir a base 2 a partir de agora — $\log n$ vai simbolizar $\log_2 n$.) Então

$C(n) = \log n + C(1) = 1 + \log n$

Vamos agora usar indução para mostrar que $C(n) = 1 + \log n$ para todo $n \geq 1$, $n = 2^m$. Esta é uma forma um pouco diferente de indução, já que todos os valores de interesse são potências de 2. Vamos considerar 1 a base da indução, mas depois provaremos que, se a afirmação é verdadeira para k, então ela é verdadeira para $2k$. A proposição, então, será verdadeira para 1, 2, 4, 8, ..., isto é, para todas as potências inteiras não-negativas de 2, que é o que queremos.

$C(1) = 1 + \log 1 = 1 + 0 = 1$, verdadeira

Suponha que $C(k) = 1 + \log k$. Então

$C(2k) = 1 + C(k)$	(pela relação de recorrência)
$= 1 + 1 + \log k$	(pela hipótese de indução)
$= 1 + \log 2 + \log k$	($\log 2 = 1$)
$= 1 + \log 2k$	(propriedade de logaritmos)

Isto completa a demonstração por indução. ●

Pelo Exemplo 46, o número máximo de comparações necessárias em uma busca binária de uma lista ordenada com n elementos, com $n = 2^m$, é $1 + \log n$. No Exemplo 41, n era 8, e foram necessárias quatro comparações ($1 + \log 8$) no pior caso (x não-pertencente à lista). Uma busca seqüencial necessitaria de oito comparações. Como

$1 + \log n < n$ para $n = 2^m$, $n \geq 4$

a busca binária é, quase sempre, mais eficiente do que a busca seqüencial. No entanto, o algoritmo de busca seqüencial tem uma grande vantagem — se a lista em questão *não está ordenada*, o algoritmo de busca seqüencial funciona, mas o de busca binária não. Se vamos primeiro ordenar a lista e depois usar o algoritmo de busca binária, precisamos, então, considerar o número de operações necessárias para ordenar a lista por diversos algoritmos diferentes.

O algoritmo de busca binária é recorrente, o que significa que o algoritmo chama a si mesmo com valores de entrada menores. Neste caso a versão menor é muito menor — aproximadamente a metade do problema original. Isto fica claro na relação de recorrência, onde $C(n)$ depende de $C(n/2)$ e não de $C(n - 1)$. Algoritmos com relações de recorrência desta forma, onde o problema é decomposto em subproblemas muito menores, são chamados, algumas vezes, de algoritmos do tipo **dividir para conquistar**.

A relação de recorrência para a busca binária é um caso particular da forma geral

$$S(n) = cS\left(\frac{n}{2}\right) + g(n) \text{ para } n \geq 2, n = 2^m \tag{1}$$

onde c é uma constante e g pode ser uma expressão envolvendo n. Gostaríamos de encontrar uma solução em forma fechada para (1) sujeita à condição básica de que $S(1)$ é conhecido. Então, poderíamos resolver qualquer relação de recorrência da forma (1) substituindo na fórmula da solução, como fizemos na última seção para relações de recorrência de primeira ordem. Observe que (1) não é uma relação de recorrência de primeira ordem, já que o valor em n não depende do valor em $n - 1$. Poderíamos usar a abordagem de expandir, conjecturar e verificar, mas, em vez disto, vamos fazer algumas transformações em (1) para convertê-la em uma relação de recorrência de primeira ordem com coeficientes constantes e depois usar o que já sabemos.

A equação (1) supõe que $n = 2^m$ com $n \geq 2$. Disso segue que $m = \log n$ e $m \geq 1$. Substituindo n por 2^m na equação (1) resulta em

$$S(2^m) = cS(2^{m-1}) + g(2^m) \tag{2}$$

Vamos agora representar $S(2^m)$ por $T(m)$ na equação (2),

$$T(m) = cT(m-1) + g(2^m) \text{ para } m \geq 1 \tag{3}$$

A equação (3) é uma equação linear de primeira ordem com coeficientes constantes; da equação (8) na Seção 2.4 obtemos a solução

$$T(m) = c^{m-1}T(1) + \sum_{i=2}^{m} c^{m-i}g(2^i) \tag{4}$$

sujeita à condição básica de que $T(1)$ é conhecido. Como a equação (3) é válida para $m = 1$, sabemos que

$$T(1) = cT(0) + g(2)$$

Substituindo em (4), obtemos

$$T(m) = c^m T(0) + \sum_{i=1}^{m} c^{m-i}g(2^i) \tag{5}$$

Substituindo $T(m)$ por $S(2^m)$, (5) torna-se

$$S(2^m) = c^m S(2^0) + \sum_{i=1}^{m} c^{m-i}g(2^i)$$

LEMBRETE: Na parte de somatório da fórmula geral da solução, c está elevado à potência $(\log n) - i$, e não à potência $(\log n) - 1$.

Finalmente, fazendo $2^m = n$, ou $m = \log n$, obtemos

$$S(n) = c^{\log n}S(1) + \sum_{i=1}^{\log n} c^{(\log n)-i}g(2^i) \tag{6}$$

A equação (6) é, então, a solução da relação de recorrência (1). Como anteriormente, para usar esta solução geral, basta colocar esta relação de recorrência na forma (1) para determinar c e $g(n)$, depois substituir na equação (6). Se você puder calcular o somatório, o resultado será uma solução em forma fechada.

EXEMPLO 47

A relação de recorrência para o algoritmo de busca binária é

$$C(1) = 1$$

$$C(n) = 1 + C\left(\frac{n}{2}\right) \text{ para } n \geq 2,\ n = 2^m$$

que é da forma da equação (1), com $c = 1$ e $g(n) = 1$. A solução, de acordo com a fórmula (6), é

$$C(n) = 1^{\log n}C(1) + \sum_{i=1}^{\log n} 1^{(\log n)-i}(1)$$

$$= 1 + (\log n)(1) = 1 + \log n$$

o que está de acordo com nosso resultado anterior. ●

PROBLEMA PRÁTICO 23

Mostre que a solução da relação de recorrência

$$S(1) = 1$$

$$S(n) = 2S\left(\frac{n}{2}\right) + 1 \text{ para } n \geq 2,\ n = 2^m$$

é $2n - 1$. (*Dica*: Veja o Exemplo 15 e note que $2^{\log n} = n$.) ●

Cota Superior (Algoritmo de Euclides)

O algoritmo de Euclides, como apresentado na Seção 2.3, usa um laço de enquanto para efetuar divisões sucessivas de modo a encontrar o mdc(a, b) para inteiros não-negativos a e b, $a \geq b$. Para analisar o algoritmo de Euclides, precisamos decidir primeiro que operações contaremos. Como este algoritmo efetua repetidas divisões, vamos considerar a operação de divisão como nossa unidade básica. Dados a e b, suponha que $b \leq a = n$, de modo que n é uma medida do tamanho dos valores de entrada. Queremos encontrar $E(n)$, que denota a quantidade necessária de operações (o número de divisões) para encontrar o mdc(a, b) no pior caso.

Uma versão recorrente do algoritmo de Euclides também pode ser escrita (veja o Exercício 62, Seção 2.4); a chave para a versão recorrente é reconhecer que o cálculo do mdc(a, b) envolve encontrar o mdc(b, r), onde r é o resto da divisão de a por b. Acabamos de ver um caso em que as operações de um algoritmo

recorrente (a busca binária) podiam ser expressas convenientemente como uma relação de recorrência em que o tamanho dos valores de entrada é reduzido pela metade após cada operação. Uma relação de recorrência expressaria $E(n)$ em termos de E com valores menores. Mas quais são estes valores menores? Para encontrar o mdc(a, b), encontramos o mdc(b, r), de modo que é claro que os valores de entrada estão ficando menores, mas de que maneira? Considere o Exemplo 27, onde, para encontrar o mdc(420, 66), foram efetuadas as seguintes divisões:

$$\begin{array}{r|l} 420 & 66 \\ -396 & 6 \\ \hline 24 & \end{array} \qquad \begin{array}{r|l} 66 & 24 \\ -48 & 2 \\ \hline 18 & \end{array} \qquad \begin{array}{r|l} 24 & 18 \\ -18 & 1 \\ \hline 6 & \end{array} \qquad \begin{array}{r|l} 18 & 6 \\ -18 & 3 \\ \hline 0 & \end{array}$$

Aqui os valores que são divididos sucessivamente são 420, 66, 24 e 18. A mudança de 420 para 66 é muito maior do que dividir pela metade, enquanto a mudança de 24 para 18 é menor.

De fato, não encontraremos uma relação de recorrência ou uma expressão exata para $E(n)$. Mas podemos, pelo menos, encontrar uma *cota superior* para $E(n)$. Uma **cota superior** é um valor superior para a quantidade de operações efetuadas por um algoritmo; o algoritmo *não* pode precisar de *mais operações* do que a cota superior, mas pode não precisar de tantas.

Para encontrar essa cota superior, vamos mostrar que, se $i > j$ e se i for dividido por j com resto r, então $r < i/2$. Existem dois casos:

1. Se $j \leq i/2$, então $r < i/2$, pois $r < j$.
2. Se $j > i/2$, então $i = 1 * j + (i - j)$; em outras palavras, o quociente é 1 e r é $i - j$, que é $< i/2$.

No algoritmo de Euclides, o resto r em qualquer etapa torna-se o dividendo (o número que está sendo dividido) dois passos adiante. Logo, os dividendos sucessivos são, pelo menos, divididos por dois a cada duas divisões. O valor n pode ser dividido por dois $\log n$ vezes; portanto, são feitas, no máximo, $2 \log n$ divisões. Assim,

$$E(n) \leq 2 \log n \tag{7}$$

O valor de $2 \log n$ para $n = 420$ é quase 18, ao passo que foram necessárias apenas 4 divisões para se obter o mdc(420, 66). É claro que esta cota superior é bastante fraca, como dizer que todos os alunos em sala têm menos de três metros e meio. Uma cota superior mais fina (isto é, mais baixa) é obtida nos Exercícios de 16 a 18 ao final desta seção.

Seção 2.5 Revisão

Técnicas

- Resolução de relações de recorrência obtidas de algoritmos do tipo dividir para conquistar usando-se uma fórmula para a solução.

Idéias Principais

A análise de um algoritmo estima o número de operações básicas que o algoritmo efetua.

A análise de algoritmos recorrentes leva, com freqüência, a relações de recorrência.

Na falta de uma expressão exata para o número de operações efetuadas por um algoritmo, pode ser possível encontrar uma cota superior.

Exercícios 2.5

1. Para o algoritmo do Exemplo 45, conte o número total de atribuições e comparações feitas no melhor dos casos (número mínimo de operações) e no pior dos casos (número máximo de operações); descreva cada um destes casos.
★2. Descreva uma versão recorrente do algoritmo de busca seqüencial para encontrar o item x em uma lista contendo n itens.
3. Usando a versão recorrente do Exercício 2, escreva uma relação de recorrência para o número de comparações de x com os elementos da lista feitas pelo algoritmo de busca seqüencial no pior caso e resolva esta relação de recorrência. (Como antes, a resposta deve ser n.)

Nos Exercícios de 4 a 7, resolva a relação de recorrência dada sujeita à condição básica. (*Dica*: Veja o Exemplo 15 e observe que $2^{\log n} = n$.)

4. $T(1) = 3$

$$T(n) = T\left(\frac{n}{2}\right) + n \text{ para } n \geq 2, n = 2^m$$

★5. $P(1) = 1$

$$P(n) = 2P\left(\frac{n}{2}\right) + 3$$

6. $S(1) = 1$

$$S(n) = 2S\left(\frac{n}{2}\right) + n$$

7. $P(1) = 1$

$$P(n) = 2P\left(\frac{n}{2}\right) + n^2$$

Os Exercícios de 8 a 10 se referem ao algoritmo *OrdenaçãoPorSeleção* da Seção 2.4.

★8. Em uma parte do algoritmo *OrdenaçãoPorSeleção*, é preciso encontrar o índice do maior item em uma lista. Em uma lista (não-ordenada) com n elementos, quantas comparações são necessárias, no pior caso, para encontrar o maior elemento? Quantas comparações são necessárias em média?

9. Definindo a operação básica como sendo a comparação dos elementos na lista e ignorando as operações de permutação dos elementos na lista, escreva uma relação de recorrência para a quantidade de operações executadas pela ordenação por seleção em uma lista com n elementos. (*Dica*: Use o resultado do Exercício 8.)

10. Resolva a relação de recorrência do Exercício 9.

Os Exercícios de 11 a 15 estão relacionados com um algoritmo de ordenação chamado *OrdenaçãoPorFusão*, que pode ser descrito da seguinte maneira: uma lista com um elemento já está ordenada, não há necessidade de fazer nada; se a lista tiver mais de um elemento, divida-a pela metade, ordene cada metade e depois combine as duas listas em uma única lista ordenada.

★11. A parte de fusão do algoritmo *OrdenaçãoPorFusão* precisa que se compare os elementos de cada uma das listas ordenadas para ver qual o próximo elemento na lista combinada e ordenada. Quando acabam os elementos de uma das listas, os elementos restantes na outra podem ser adicionados sem outras comparações. Dados os pares de listas a seguir, combine-os e conte o número de comparações feitas para fundi-las em uma única lista ordenada.
a. 6, 8, 9 e 1, 4, 5. **b.** 1, 5, 8 e 2, 3, 4.
c. 0, 2, 3, 4, 7, 10 e 1, 8, 9.

12. Sob que circunstâncias teremos que executar o número máximo de comparações ao se fundir duas listas ordenadas? Se as duas listas têm comprimento r e s, qual é o número máximo de comparações?

★13. Escreva uma relação de recorrência para o número de comparações entre os elementos das listas efetuadas pelo algoritmo *OrdenaçãoPorFusão* no pior caso. Suponha que $n = 2^m$.

14. Resolva a relação de recorrência do Exercício 13.

15. Compare o comportamento, no pior caso, dos algoritmos *OrdenaçãoPorSeleção* e *OrdenaçãoPorFusão* para $n = 4$, 8, 16 e 32 (use uma calculadora).

Os Exercícios de 16 a 19 procuram uma cota superior melhor para o número de divisões necessárias para que o algoritmo de Euclides encontre o mdc(a, b). Suponha que a e b são inteiros não-negativos e que $a > b$. (Se $a = b$, é necessária apenas uma divisão, de modo que este caso corresponde ao menor número de operações; queremos uma cota superior para o número máximo de operações.)

16. Suponha que são necessárias m divisões para encontrar o mdc(a, b). Prove, por indução, que, para $m \geq 1$, $a \geq F(m + 2)$ e $b \geq F(m + 1)$, onde $F(n)$ é a seqüência de Fibonacci. (*Dica*: Para encontrar o mdc(a, b), o algoritmo calcula o mdc(b, r) depois da primeira divisão.)

★17. Suponha que são necessárias m divisões para encontrar o mdc(a, b), com $m \geq 4$, e que $a = n$. Prove que

$$\left(\frac{3}{2}\right)^{m+1} < F(m + 2) \leq n$$

(*Dica*: Use o resultado do Exercício 16 desta seção e o do Exercício 23 da Seção 2.4.)

18. Suponha que são necessárias m divisões para encontrar o mdc(a, b), com $m \geq 4$, e que $a = n$. Prove que

$$m < (\log_{1,5} n) - 1$$

(*Dica*: Use o resultado do Exercício 17.)

19. a. Calcule o mdc(89, 55) e conte o número de divisões necessárias.

b. Calcule uma cota superior para o número de divisões necessárias para calcular o mdc(89, 55) usando a equação (7).

c. Calcule uma cota superior para o número de divisões necessárias para calcular o mdc(89, 55) usando o resultado do Exercício 18.

Revisão do Capítulo 2

Terminologia

algoritmo de busca binária
algoritmo de busca seqüencial
algoritmo de Euclides
algoritmo de ordenação por seleção
algoritmo do tipo dividir para conquistar
análise de algoritmos
base da indução
cadeia binária
cadeia vazia
concatenação
contra-exemplo
contrapositiva
correção parcial
cota superior
definição por indução
definição por recorrência
demonstração direta
demonstração por absurdo
demonstração por casos
demonstração por contraposição
demonstração por exaustão
forma de Backus-Naur (FBN)
hipótese de indução
índice do somatório
invariante do laço
máximo divisor comum
n fatorial
notação de somatório
número racional
palíndromo
passo indutivo
primeiro princípio de indução matemática
princípio da boa ordenação
raciocínio dedutivo
raciocínio indutivo
recíproca
regra de inferência para laços
relação de recorrência
relação de recorrência com coeficientes constantes
relação de recorrência de primeira ordem
relação de recorrência homogênea
relação de recorrência linear
resolução de uma relação de recorrência
segundo princípio de indução matemática
seqüência
seqüência de Fibonacci
solução em forma fechada

Autoteste Responda se as afirmações a seguir são verdadeiras ou falsas sem recorrer ao capítulo.

Seção 2.1

1. Uma conjectura nunca pode ser demonstrada provando-se, simplesmente, um número finito de casos.
2. Uma demonstração por absurdo de $P \rightarrow Q$ começa supondo P e Q'.
3. No enunciado do teorema "o dobro de um inteiro ímpar é par", está subentendido um quantificador existencial.
4. Para provar a conjectura "se Macapá é a capital, Amapá é o estado", é suficiente provar que "se Amapá é o estado, Macapá é a capital".
5. Para provar "A se, e somente se, B", precisamos provar que $A \rightarrow B$ e que $B \rightarrow A$.

Seção 2.2

6. Indução é uma técnica de demonstração apropriada para se provar uma proposição sobre todos os inteiros positivos.
7. O passo básico de uma demonstração por indução corresponde à demonstração de que uma propriedade é verdadeira para $n = 1$.
8. Se a veracidade de $P(k + 1)$ depende da veracidade de outros valores anteriores, além de $P(k)$, então deve-se usar o segundo princípio da indução.
9. A chave de uma demonstração pelo primeiro princípio de indução é entender como a veracidade de P em $k + 1$ depende da veracidade de P em k.
10. Em uma demonstração por indução da proposição

$$1^3 + 2^2 + \cdots + n^3 = \frac{n^2(n+1)^2}{4}$$

a hipótese de indução é

$$k^3 = \frac{k^2(k+1)^2}{4}$$

Seção 2.3

11. Um invariante do laço permanece válido até a saída do laço, quando se torna falso.

12. A correção parcial de uma proposição em laço significa que o laço se comporta corretamente para alguns valores de entrada e não para outros.

13. O segundo princípio de indução é usado para se provar que uma proposição é um invariante do laço porque o laço pode ser executado um número arbitrário de vezes.

14. Se uma proposição em laço é da forma

enquanto (condição B)
P
fim do enquanto

então o invariante de laço Q vai ser B'.

15. Ao calcular o mdc(42, 30) pelo algoritmo de Euclides, efetua-se a divisão de 30 por 12.

Seção 2.4

16. Uma seqüência definida por

$S(1) = 7$

$S(n) = 3S(n - 1) + 2$ para $n \geq 2$

contém o número 215.

17. Algoritmos recorrentes são valiosos principalmente porque são mais eficientes do que algoritmos iterativos.

18. Ao se aplicar o algoritmo de busca binária à lista

2, 5, 7, 10, 14, 20

onde $x = 8$ é o item procurado, x nunca é comparado a 5.

19. Uma solução em forma fechada de uma relação de recorrência é obtida aplicando-se indução matemática à relação de recorrência.

20. $S(n) = 2S(n - 1) + 3S(n - 2) = 5n$ é uma relação de recorrência linear de primeira ordem com coeficientes constantes.

Seção 2.5

21. A análise de um algoritmo encontra, em geral, a quantidade de operações efetuadas no pior caso porque é muito difícil analisar um caso médio.

22. Algoritmos do tipo dividir para conquistar levam a relações de recorrência que não são de primeira ordem.

23. A busca binária é mais eficiente do que a busca seqüencial em uma lista ordenada com mais de três elementos.

24. Se uma busca seqüencial fosse reescrita como um algoritmo recorrente, seria um algoritmo do tipo dividir para conquistar.

25. Uma cota superior para o algoritmo de Euclides é um valor superior para a quantidade de divisões efetuadas para calcular o mdc(a, b).

No Computador

Para os Exercícios de 1 a 10, escreva um programa que produza a saída desejada a partir dos dados de entrada fornecidos.

1. *Entrada*: Número n de termos de uma progressão geométrica (veja o Exercício 21, Seção 2.2), o termo inicial a e a razão r

Saída: Soma dos n primeiros termos usando

a. iteração
b. a fórmula do Exercício 21, Seção 2.2

2. *Entrada*: Número n de termos de uma progressão aritmética (veja o Exercício 22, Seção 2.2), o termo inicial a e a parcela d

Saída: Soma dos n primeiros termos usando

a. iteração
b. a fórmula do Exercício 22, Seção 2.2

3. *Entrada*: Número n

Saída: Soma dos n primeiros cubos usando

a. iteração, usando apenas multiplicação e adição; explicite na saída o número de multiplicações e adições efetuadas
b. a fórmula do Exercício 8, Seção 2.2, usando apenas multiplicação, adição e divisão; explicite na saída o número de multiplicações, adições e divisões efetuadas

4. *Entrada*: Nenhuma

Saída: Tabela mostrando todos os inteiros n, $8 \leq n \leq 100$, como uma soma de números iguais a 3 e a 5 (veja o Exemplo 24)

5. *Entrada*: Cadeia binária

Saída: Mensagem indicando se a cadeia de entrada é um palíndromo (veja o Problema Prático 17)
Algoritmo: Use recorrência.

6. *Entrada*: Cadeia de caracteres x e um inteiro positivo n

Saída: Concatenação de n cópias de x
Algoritmo: Use recorrência.
(Algumas linguagens de programação já vêm com capacidade de manipulação de cadeias, como a concatenação.)

7. *Entrada*: Inteiro positivo n

Saída: O n-ésimo valor em uma seqüência de Fibonacci usando

a. iteração
b. recorrência

Insira agora um contador em cada versão para indicar o número total de adições efetuadas. Execute cada versão para diversos valores de n e coloque em um único gráfico o número de adições em função de n para cada versão.

8. *Entrada*: Dois inteiros positivos m e n

Saída: mdc(m, n) usando

a. a versão iterativa do algoritmo de Euclides
b. a versão recorrente do algoritmo de Euclides

9. *Entrada*: Lista não-ordenada de 10 inteiros

Saída: A lista de entrada ordenada em ordem crescente
Algoritmo: Use a ordenação por seleção recorrente do Exemplo 39.

10. *Entrada*: Lista ordenada de 10 inteiros e um inteiro x

Saída: Mensagem indicando se x pertence à lista
Algoritmo: Use o algoritmo de busca binária do Exemplo 40.

11. A fórmula $3^n < n!$ é verdadeira para todo $n \geq N$. Escreva um programa para determinar N e depois prove o resultado por indução.

12. A fórmula $2^n > n^3$ é verdadeira para todo $n \geq N$. Escreva um programa para determinar N e depois prove o resultado por indução.

13. O valor $(1 + \sqrt{5})/2$, conhecido como *razão áurea*, está relacionado à seqüência de Fibonacci por

$$\lim_{n \to \infty} \frac{F(n+1)}{F(n)} = \frac{1+\sqrt{5}}{2}$$

Verifique este limite calculando $F(n + 1)/F(n)$ para $n = 10, 15, 25, 50$ e 100, e comparando os resultados com a razão áurea.

14. Compare o número de operações efetuadas pelos algoritmos de busca seqüencial e busca binária em uma lista ordenada com n elementos calculando n e $1 + \log n$ para valores de n de 1 a 100. Apresente os resultados em forma gráfica.