Capítulo 3

Crescimento de funções

A ordem de crescimento do tempo de execução de um algoritmo, definida no Capítulo 2, fornece uma caracterização simples da eficiência do algoritmo, e também nos permite comparar o desempenho relativo de algoritmos alternativos. Uma vez que o tamanho da entrada n se torna grande o suficiente, a ordenação por intercalação, com seu tempo de execução do pior caso $\Theta(n \lg n)$, vence a ordenação por inserção, cujo tempo de execução do pior caso é $\Theta(n^2)$. Embora às vezes seja possível determinar o tempo exato de execução de um algoritmo, como fizemos no caso da ordenação por inserção no Capítulo 2, a precisão extra em geral não vale o esforço de calculá-la. Para entradas grandes o bastante, as constantes multiplicativas e os termos de mais baixa ordem de um tempo de execução exato são dominados pelos efeitos do próprio tamanho da entrada.

Quando observamos tamanhos de entrada grandes o suficiente para tornar relevante apenas a ordem de crescimento do tempo de execução, estamos estudando a eficiência ***assintótica*** dos algoritmos. Ou seja, estamos preocupados com a maneira como o tempo de execução de um algoritmo aumenta com o tamanho da entrada *no limite*, à medida que o tamanho da entrada aumenta indefinidamente (sem limitação). Em geral, um algoritmo que é assintoticamente mais eficiente será a melhor escolha para todas as entradas, exceto as muito pequenas.

Este capítulo oferece vários métodos padrão para simplificar a análise assintótica de algoritmos. A próxima seção começa definindo diversos tipos de "notação assintótica", da qual já vimos um exemplo na notação Θ. Várias convenções de notação usadas em todo este livro serão então apresentadas, e finalmente faremos uma revisão do comportamento de funções que surgem comumente na análise de algoritmos.

3.1 Notação assintótica

As notações que usamos para descrever o tempo de execução assintótica de um algoritmo são definidas em termos de funções cujos domínios são o conjunto dos números naturais $\mathbf{N} = \{0, 1, 2, \ldots\}$. Tais notações são convenientes para descrever a função do tempo de execução do pior caso $T(n)$, que em geral é definida somente sobre tamanhos de entrada inteiros. Contudo, às vezes é conveniente *abusar* da notação assintótica de várias maneiras. Por exemplo, a notação é estendida com facilidade ao domínio dos números reais ou, de modo alternativo, limitado a um subconjunto dos números naturais. Porém, é importante entender o significado preciso da notação para que, quando houver um abuso em seu uso, ela não seja *mal utilizada*. Esta seção define as notações assintóticas básicas e também apresenta alguns abusos comuns.

Notação Θ

No Capítulo 2, descobrimos que o tempo de execução do pior caso da ordenação por inserção é $T(n) = \Theta(n^2)$. Vamos definir o que significa essa notação. Para uma dada função $g(n)$, denotamos por $\Theta(g(n))$ o *conjunto de funções*

$$\Theta(g(n)) = \{f(n) : \text{existem constantes positivas } c_1, c_2 \text{ e } n_0 \text{ tais que } 0 \le c_1 g(n) \le f(n) \le c_2 g(n) \text{ para todo } n \ge n_0\}.^1$$

Uma função $f(n)$ pertence ao conjunto $\Theta(g(n))$ se existem constantes positivas c_1 e c_2 tais que ela possa ser "imprensada" entre $c_1 g(n)$ e $c_2 g(n)$, para um valor de n suficientemente grande. Como $\Theta(g(n))$ é um conjunto, poderíamos escrever "$f(n) \in \Theta(g(n))$" para indicar que $f(n)$ é um membro de (ou pertence a) $\Theta(g(n))$. Em vez disso, em geral escreveremos "$f(n) = \Theta(g(n))$" para expressar a mesma noção. Esse abuso da igualdade para denotar a condição de membro de um conjunto (pertinência) pode a princípio parecer confuso, mas veremos adiante nesta seção que ele tem suas vantagens.

A Figura 3.1(a) apresenta um quadro intuitivo das funções $f(n)$ e $g(n)$, onde $f(n) = \Theta(g(n))$. Para todos os valores de n à direita de n_0, o valor de $f(n)$ reside em $c_1 g(n)$ ou acima dele, e em $c_2 g(n)$ ou abaixo desse valor. Em outras palavras, para todo $n \ge n_0$, a função $f(n)$ é igual a $g(n)$ dentro de um fator constante. Dizemos que $g(n)$ é um ***limite assintoticamente restrito*** para $f(n)$.

A definição de $\Theta(g(n))$ exige que todo membro $f(n) \in \Theta(g(n))$ seja ***assintoticamente não negativo***, isto é, que $f(n)$ seja não negativo sempre que n for suficientemente grande. (Uma função ***assintoticamente positiva*** é uma função positiva para todo n suficientemente grande.) Em conseqüência disso, a própria função $g(n)$ deve ser assintoticamente não negativa, ou então o conjunto $\Theta(g(n))$ é vazio. Por essa razão, vamos supor que toda função usada dentro da notação Θ é assintoticamente não negativa. Essa premissa também se mantém para as outras notações assintóticas definidas neste capítulo.

No Capítulo 2, introduzimos uma noção informal da notação Θ que consistia em descartar os termos de mais baixa ordem e ignorar o coeficiente inicial do termo de mais alta ordem. Vamos justificar brevemente essa intuição, usando a definição formal para mostrar que $\frac{1}{2}n^2 - 3n = \Theta(n^2)$. Para isso, devemos definir constantes positivas c_1, c_2 e n_0 tais que

$$c_1 n^2 \le \frac{1}{2}n^2 - 3n \le c_2 n^2$$

para todo $n \ge n_0$. A divisão por n^2 produz

$$c_1 \le \frac{1}{2} - \frac{3}{n} \le c_2.$$

A desigualdade do lado direito pode ser considerada válida para qualquer valor de $n \ge 1$, escolhendo-se $c_2 \ge 1/2$. Do mesmo modo, a desigualdade da esquerda pode ser considerada válida para qualquer valor de $n \ge 7$, escolhendo-se $c_1 \le 1/14$. Assim, escolhendo $c_1 = 1/14$, $c_2 = 1/2$ e $n_0 = 7$, podemos verificar que $\frac{1}{2}n^2 - 3n = \Theta(n^2)$. Certamente, existem outras opções para as constantes, mas o fato importante é que existe *alguma* opção. Observe que essas constantes dependem da função $\frac{1}{2}n^2 - 3n$; uma função diferente pertencente a $\Theta(n^2)$ normalmente exigiria constantes distintas.

[1] Na notação de conjuntos, um sinal de dois-pontos deve ser lido como "tal que".

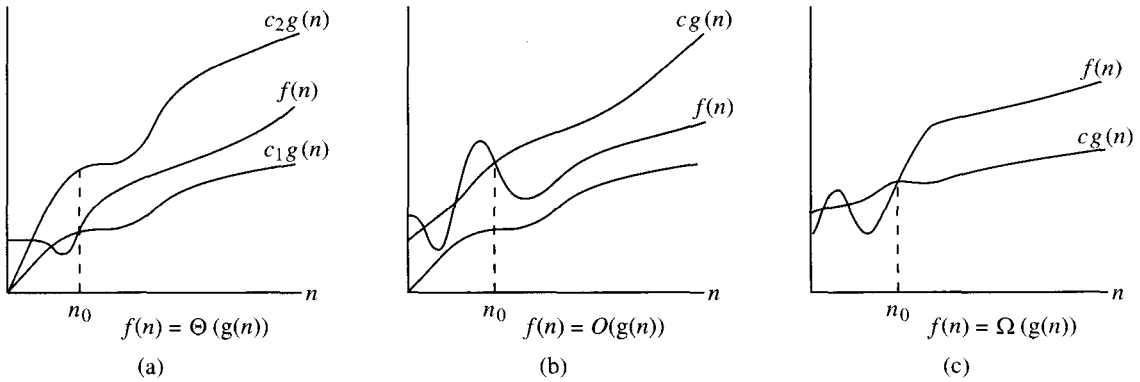

FIGURA 3.1 Exemplos gráficos das notações Θ, O e Ω. Em cada parte, o valor de n_0 mostrado é o valor mínimo possível; qualquer valor maior também funcionaria. **(a)** A notação Θ limita uma função ao intervalo entre fatores constantes. Escrevemos $f(n) = \Theta(g(n))$ se existem constantes positivas n_0, c_1 e c_2 tais que, à direita de n_0, o valor de $f(n)$ sempre reside entre $c_1g(n)$ e $c_2g(n)$ inclusive. **(b)** A notação O dá um limite superior para uma função dentro de um fator constante. Escrevemos $f(n) = O(g(n))$ se existem constantes positivas n_0 e c tais que, à direita de n_0, o valor de $f(n)$ sempre reside em ou abaixo de $cg(n)$. **(c)** A notação Ω dá um limite inferior para uma função dentro de um fator constante. Escrevemos $f(n) = \Omega(g(n))$ se existem constantes positivas $n0$ e c tais que, à direita de n_0, o valor de $f(n)$ sempre reside em ou acima de $cg(n)$

Também podemos usar a definição formal para verificar que $6n^3 \neq \Theta(n^2)$. Vamos supor, a título de contradição, que existam c_2 e n_0 tais que $6n^3 \leq c_2n_2$ para todo $n \geq n_0$. Mas então $n \leq c_2/6$, o que não pode ser válido para um valor de n arbitrariamente grande, pois c_2 é constante.

Intuitivamente, os termos de mais baixa ordem de uma função assintoticamente positiva podem ser ignorados na determinação de limites assintoticamente restritos, porque eles são insignificantes para grandes valores de n. Uma minúscula fração do termo de mais alta ordem é suficiente para dominar os termos de mais baixa ordem. Desse modo, a definição de c_1 com um valor ligeiramente menor que o coeficiente do termo de mais alta ordem e a definição de c_2 com um valor ligeiramente maior permite que as desigualdades na definição da notação Θ sejam satisfeitas. O coeficiente do termo de mais alta ordem pode do mesmo modo ser ignorado, pois ele só muda $c1$ e c_2 por um fator constante igual ao coeficiente.

Como exemplo, considere qualquer função quadrática $f(n) = an^2 + bn + c$, onde a, b e c são constantes e $a > 0$. Descartando os termos de mais baixa ordem e ignorando a constante, produzimos $f(n) = \Theta(n^2)$. Formalmente, para mostrar a mesma coisa, tomamos as constantes $c_1 = a/4$, $c_2 = 7a/4$ e $n_0 = 2.\max((|b|/a), \sqrt{(|c|/a)})$. O leitor poderá verificar que $0 \leq c_1n^2 \leq an^2 + bn + c \leq c_2n^2$ para todo $n \geq n_0$. Em geral, para qualquer polinômio $p(n) = \sum_{i=0}^{d} a_i n^i$, onde a_i são constantes e $a_d > 0$, temos $p(n) = \Theta(n^d)$ (ver Problema 3-1).

Tendo em vista que qualquer constante é um polinômio de grau 0, podemos expressar qualquer função constante como $\Theta(n^0)$, ou $\Theta(1)$. Porém, essa última notação é um abuso secundário, porque não está claro qual variável está tendendo a infinito.[2] Usaremos com freqüência a notação $\Theta(1)$ para indicar uma constante ou uma função constante em relação a alguma variável.

[2] O problema real é que nossa notação comum para funções não distingue funções de valores. No cálculo de λ, os parâmetros para uma função estão claramente especificados: a função n^2 poderia ser escrita como $\lambda n.n^2$, ou até mesmo $\lambda r.r^2$. Porém, a adoção de uma notação mais rigorosa complicaria manipulações algébricas, e assim optamos por tolerar o abuso.

Notação *O*

A notação Θ limita assintoticamente uma função acima e abaixo. Quando temos apenas um ***limite assintótico superior***, usamos a notação O. Para uma dada função $g(n)$, denotamos por $O(g(n))$ (lê-se "O maiúsculo de g de n" ou, às vezes, apenas "o de g de n") o conjunto de funções

$$O(g(n)) = \{f(n) : \text{existem constantes positivas } c \text{ e } n_0 \text{ tais que } 0 \le f(n) \le cg(n) \text{ para todo } n \ge n_0\}.$$

Usamos a notação O para dar um limite superior sobre uma função, dentro de um fator constante. A Figura 3.1(b) mostra a intuição por trás da notação O. Para todos os valores n à direita de n_0, o valor da função $f(n)$ está em ou abaixo de $g(n)$.

Para indicar que uma função $f(n)$ é um membro de $O(g(n))$, escrevemos $f(n) = O(g(n))$. Observe que $f(n) = \Theta(g(n))$ implica $f(n) = O(g(n))$, pois a notação Θ é uma noção mais forte que a notação O. Em termos da teoria de conjuntos, temos $\Theta(g(n)) \subseteq O(g(n))$. Desse modo, nossa prova de que qualquer função quadrática $an^2 + bn + c$, onde $a > 0$, está em $\Theta(n^2)$ também mostra que qualquer função quadrática está em $O(n^2)$. O que pode ser mais surpreendente é o fato de que qualquer função *linear* $an + b$ está em $O(n^2)$, o que é facilmente verificado fazendo-se $c = a + |b|$ e $n_0 = 1$.

Alguns leitores que viram antes a notação O podem achar estranho que devamos escrever, por exemplo, $n = O(n^2)$. Na literatura, a notação O é usada às vezes de modo informal para descrever limites assintoticamente restritos, ou seja, o que definimos usando a notação Θ. Contudo, neste livro, quando escrevermos $f(n) = O(g(n))$, estamos simplesmente afirmando que algum múltiplo constante de $g(n)$ é um limite assintótico superior sobre $f(n)$, sem qualquer menção sobre o quanto um limite superior é restrito. A distinção entre limites assintóticos superiores e limites assintoticamente restritos agora se tornou padrão na literatura de algoritmos.

Usando a notação O, podemos descrever freqüentemente o tempo de execução de um algoritmo apenas inspecionando a estrutura global do algoritmo. Por exemplo, a estrutura de loop duplamente aninhado do algoritmo de ordenação por inserção vista no Capítulo 2 produz imediatamente um limite superior $O(n^2)$ sobre o tempo de execução do pior caso: o custo do loop interno é limitado na parte superior por $O(1)$ (constante), os índices i e j são ambos no máximo n, e o loop interno é executado no máximo uma vez para cada um dos n^2 pares de valores correspondentes a i e j.

Tendo em vista que a notação O descreve um limite superior, quando a empregamos para limitar o tempo de execução do pior caso de um algoritmo, temos um limite sobre o tempo de execução do algoritmo em cada entrada. Desse modo, o limite $O(n^2)$ no tempo de execução do pior caso da ordenação por inserção também se aplica a seu tempo de execução sobre toda entrada. Porém, o limite $\Theta(n^2)$ no tempo de execução do pior caso da ordenação por inserção não implica um limite $\Theta(n^2)$ no tempo de execução da ordenação por inserção em *toda* entrada. Por exemplo, vimos no Capítulo 2 que, quando a entrada já está ordenada, a ordenação por inserção funciona no tempo $\Theta(n)$.

Tecnicamente, é um abuso dizer que o tempo de execução da ordenação por inserção é $O(n^2)$, pois, para um dado n, o tempo de execução real varia, dependendo da entrada específica de tamanho n. Quando afirmamos que "o tempo de execução é $O(n^2)$", queremos dizer que existe uma função $f(n)$ que é $O(n^2)$ tal que, para qualquer valor de n, não importando que entrada específica de tamanho n seja escolhida, o tempo de execução sobre essa entrada tem um limite superior determinado pelo valor $f(n)$. De modo equivalente, dizemos que o tempo de execução do pior caso é $O(n^2)$.

Notação Ω

Da mesma maneira que a notação O fornece um limite assintótico *superior* sobre uma função, a notação Ω fornece um ***limite assintótico inferior***. Para uma determinada função $g(n)$, denotamos por $\Omega(g(n))$ (lê-se "ômega maiúsculo de g de n" ou, às vezes, "ômega de g de n") o conjunto de funções

$$\Omega(g(n)) = \{f(n) : \text{existem constantes positivas } c \text{ e } n_0 \text{ tais que } 0 \leq cg(n) \leq f(n) \text{ para todo } n \geq n_0\}.$$

A intuição por trás da notação Ω é mostrada na Figura 3.1(c). Para todos os valores n à direita de n_0, o valor de $f(n)$ está em ou acima de $g(n)$.

A partir das definições das notações assintóticas que vimos até agora, é fácil demonstrar o importante teorema a seguir (ver Exercício 3.1-5).

Teorema 3.1

Para duas funções quaisquer $f(n)$ e $g(n)$, temos $f(n) = \Theta(g(n))$ se e somente se $f(n) = O(g(n))$ e $f(n) = \Omega(g(n))$. ■

Como exemplo de aplicação desse teorema, nossa demonstração de que $an^2 + bn + c = \Theta(n^2)$ para quaisquer constantes a, b e c, onde $a > 0$, implica imediatamente que $an^2 + bn + c = \Omega(n^2)$ e $an^2 + bn + c = O(n^2)$. Na prática, em lugar de usar o Teorema 3.1 para obter limites assintóticos superiores e inferiores a partir de limites assintoticamente restritos, como fizemos nesse exemplo, nós o utilizamos normalmente para demonstrar limites assintoticamente restritos a partir de limites assintóticos superiores e inferiores.

Considerando-se que a notação Ω descreve um limite inferior, quando a usamos para limitar o tempo de execução do melhor caso de um algoritmo, por implicação também limitamos o tempo de execução do algoritmo sobre entradas arbitrárias. Por exemplo, o tempo de execução no melhor caso da ordenação por inserção é $\Omega(n)$, o que implica que o tempo de execução da ordenação por inserção é $\Omega(n)$.

Assim, o tempo de execução da ordenação por inserção recai entre $\Omega(n)$ e $O(n^2)$, pois ele fica em qualquer lugar entre uma função linear de n e uma função quadrática de n. Além disso, esses limites são assintoticamente tão restritos quanto possível: por exemplo, o tempo de execução da ordenação por inserção não é $\Omega(n^2)$, pois existe uma entrada para a qual a ordenação por inserção é executada no tempo $\Theta(n)$ (por exemplo, quando a entrada já está ordenada). Contudo, não é contraditório dizer que o tempo de execução do *pior caso* da ordenação por inserção é $\Omega(n^2)$, tendo em vista que existe uma entrada que faz o algoritmo demorar o tempo $\Omega(n^2)$. Quando afirmamos que o *tempo de execução* (sem modificador) de um algoritmo é $\Omega(g(n))$, queremos dizer que, *independentemente da entrada específica de tamanho n escolhida para cada valor de n*, o tempo de execução sobre essa entrada é pelo menos uma constante vezes $g(n)$, para um valor de n suficientemente grande.

Notação assintótica em equações e desigualdades

Já vimos como a notação assintótica pode ser usada dentro de fórmulas matemáticas. Por exemplo, na introdução da notação O, escrevemos "$n = O(n^2)$".Também poderíamos escrever $2n^2 + 3n + 1 = 2n^2 + \Theta(n)$. Como interpretaremos tais fórmulas?

Quando a notação assintótica está sozinha no lado direito de uma equação (ou desigualdade), como em $n = O(n^2)$, já definimos o sinal de igualdade como símbolo de pertinência a um conjunto: $n \in O(n^2)$. Porém, em geral, quando a notação assintótica aparecer em uma fórmula, nós a interpretaremos com o significado de alguma função anônima que não nos preocupare-

mos em nomear. Por exemplo, a fórmula $2n^2 + 3n + 1 = 2n^2 + \Theta(n)$ significa que $2n^2 + 3n + 1 = 2n^2 + f(n)$, onde $f(n)$ é alguma função no conjunto $\Theta(n)$. Nesse caso, $f(n) = 3n + 1$, que de fato está em $\Theta(n)$.

O uso da notação assintótica dessa maneira pode ajudar a eliminar detalhes não essenciais e a desordem em uma equação. Por exemplo, expressamos no Capítulo 2 o tempo de execução do pior caso da ordenação por intercalação como a recorrência

$$T(n) = 2T(n/2) + \Theta(n).$$

Se estivermos interessados apenas no comportamento assintótico de $T(n)$, não haverá sentido em especificar exatamente todos os termos de mais baixa ordem; todos eles serão considerados incluídos na função anônima denotada pelo termo $\Theta(n)$.

O número de funções anônimas em uma expressão é entendido como igual ao número de vezes que a notação assintótica aparece. Por exemplo, na expressão

$$\sum_{i=1}^{n} O(i).$$

existe apenas uma única função anônima (uma função de i). Portanto, essa expressão *não* é o mesmo que $O(1) + O(2) + \cdots + O(n)$ que, na realidade, não tem uma interpretação clara.

Em alguns casos, a notação assintótica aparece no lado esquerdo de uma equação, como em

$$2n^2 + \Theta(n) = \Theta(n^2) .$$

Interpretamos tais equações usando a seguinte regra: *Independentemente de como as funções anônimas são escolhidas no lado esquerdo do sinal de igualdade, existe um modo de escolher as funções anônimas no lado direito do sinal de igualdade para tornar a equação válida*. Desse modo, o significado de nosso exemplo é que, para *qualquer* função $f(n) \in \Theta(n)$, existe *alguma* função $g(n) \in \Theta(n^2)$, tal que $2n^2 + f(n) = g(n)$ para todo n. Em outras palavras, o lado direito de uma equação fornece um nível mais grosseiro de detalhes que o lado esquerdo.

Vários desses relacionamentos podem ser encadeados, como em

$$\begin{aligned} 2n^2 + 3n + 1 &= 2n^2 + \Theta(n) \\ &= \Theta(n^2) . \end{aligned}$$

Podemos interpretar cada equação separadamente pela regra anterior. A primeira equação diz que existe *alguma* função $f(n) \in \Theta(n)$ tal que $2n^2 + 3n + 1 = 2n^2 + f(n)$ para todo n. A segunda equação afirma que, para *qualquer* função $g(n) \in \Theta(n)$ (como a função $f(n)$ que acabamos de mencionar), existe *alguma* função $h(n) \in \Theta(n^2)$ tal que $2n^2 + g(n) = h(n)$ para todo n. Observe que essa interpretação implica que $2n^2 + 3n + 1 = \Theta(n^2)$, que é aquilo que o encadeamento de equações nos fornece intuitivamente.

Notação *o*

O limite assintótico superior fornecido pela notação O pode ser ou não assintoticamente restrito. O limite $2n^2 = O(n^2)$ é assintoticamente restrito, mas o limite $2n = O(n^2)$ não o é. Usamos a notação o para denotar um limite superior que não é assintoticamente restrito. Definimos formalmente $o(g(n))$ (lê-se "o minúsculo de g de n") como o conjunto

$o(g(n)) = \{f(n)$: para qualquer constante positiva $c > 0$, existe uma constante $n_0 > 0$ tal que $0 \leq f(n) < cg(n)$ para todo $n \geq n_0\}$.

Por exemplo, $2n = o(n^2)$, mas $2n^2 \neq o(n^2)$.

As definições da notação O e da notação o são semelhantes. A principal diferença é que em $f(n) = O(g(n))$, o limite $0 \leq f(n) \leq cg(n)$ se mantém válido para *alguma* constante $c > 0$ mas, em $f(n) = o(g(n))$, o limite $0 \leq f(n) < cg(n)$ é válido para *todas* as constantes $c > 0$. Intuitivamente, na notação o, a função $f(n)$ se torna insignificante em relação a $g(n)$ à medida que n se aproxima do infinito; isto é,

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 . \tag{3.1}$$

Alguns autores usam esse limite como uma definição da notação o; a definição neste livro também restringe as funções anônimas a serem assintoticamente não negativas.

Notação ω

Por analogia, a notação ω está para a notação Ω como a notação o está para a notação O. Usamos a notação ω para denotar um limite inferior que não é assintoticamente restrito. Um modo de defini-la é

$f(n) \in \omega(g(n))$ se e somente se $g(n) \in o(f(n))$.

Porém, formalmente, definimos $\omega(g(n))$ (lê-se "ômega minúsculo de g de n") como o conjunto

$\omega(g(n)) = \{f(n)$: para qualquer constante positiva $c > 0$, existe uma constante $n_0 > 0$ tal que $0 \leq cg(n) < f(n)$ para todo $n \geq n_0\}$.

Por exemplo, $n^2/2 = \omega(n)$, mas $n^2/2 \neq \omega(n^2)$. A relação $f(n) = \omega(g(n))$ implica que

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty ,$$

se o limite existe. Isto é, $f(n)$ se torna arbitrariamente grande em relação a $g(n)$ à medida que n se aproxima do infinito.

Comparação de funções

Muitas das propriedades relacionais de números reais também se aplicam a comparações assintóticas. No caso das propriedades seguintes, suponha que $f(n)$ e $g(n)$ sejam assintoticamente positivas.

Transitividade:

$f(n) = \Theta(g(n))$ e $g(n) = \Theta(h(n))$ implicam $f(n) = \Theta(h(n))$,
$f(n) = O(g(n))$ e $g(n) = O(h(n))$ implicam $f(n) = O(h(n))$,
$f(n) = \Omega(g(n))$ e $g(n) = \Omega(h(n))$ implicam $f(n) = \Omega(h(n))$,
$f(n) = o(g(n))$ e $g(n) = o(h(n))$ implicam $f(n) = o(h(n))$,
$f(n) = \omega(g(n))$ e $g(n) = \omega(h(n))$ implicam $f(n) = \omega(h(n))$.

Reflexividade:

$f(n) = \Theta(f(n))$
$f(n) = O(f(n))$
$f(n) = \Omega(f(n))$

Simetria:

$f(n) = \Theta(g(n))$ se e somente se $g(n) = \Theta(f(n))$

Simetria de transposição:

$f(n) = O(g(n))$ se e somente se $g(n) = \Omega(f(n))$
$f(n) = o(g(n))$ se e somente se $g(n) = \omega(f(n))$

Pelo fato dessas propriedades se manterem válidas para notações assintóticas, é possível traçar uma analogia entre a comparação assintótica de duas funções f e g e a comparação de dois números reais a e b:

$$\begin{aligned} f(n) &= O(g(n)) \quad \approx \quad a \leq b, \\ f(n) &= \Omega(g(n)) \quad \approx \quad a \geq b, \\ f(n) &= \Theta(g(n)) \quad \approx \quad a = b, \\ f(n) &= o(g(n)) \quad \approx \quad a < b, \\ f(n) &= \omega(g(n)) \quad \approx \quad a > b. \end{aligned}$$

Dizemos que $f(n)$ é ***assintoticamente menor*** que $g(n)$ se $f(n) = o(g(n))$, e que $f(n)$ é ***assintoticamente maior*** que $g(n)$ se $f(n) = \omega(g(n))$.

Contudo, uma propriedade de números reais não é transportada para a notação assintótica:

Tricotomia: Para dois números reais quaisquer a e b, exatamente uma das propriedades a seguir deve ser válida: $a < b$, $a = b$ ou $a > b$.

Embora dois números reais quaisquer possam ser comparados, nem todas as funções são assintoticamente comparáveis. Ou seja, para duas funções $f(n)$ e $g(n)$, pode acontecer que nem $f(n) = O(g(n))$, nem $f(n) = \Omega(g(n))$ seja válida. Por exemplo, as funções n e $n^{1 + \operatorname{sen} n}$ não podem ser comparadas utilizando-se a notação assintótica, pois o valor do expoente em $n^{1 + \operatorname{sen} n}$ oscila entre 0 e 2, assumindo todos os valores intermediários.

Exercícios

3.1-1

Sejam $f(n)$ e $g(n)$ funções assintoticamente não negativas. Usando a definição básica da notação Θ, prove que $\max(f(n), g(n)) = \Theta(f(n) + g(n))$.

3.1-2

Mostre que, para quaisquer constantes reais a e b, onde $b > 0$,

$$(n + a)^b = \Theta(n^b) . \tag{3.2}$$

3.1-3

Explique por que a declaração "O tempo de execução no algoritmo A é no mínimo $O(n^2)$" é isenta de significado.

3.1-4
É verdade que $2^{n+1} = O(2^n)$? É verdade que $2^{2n} = O(2^n)$?

3.1-5
Demonstre o Teorema 3.1.

3.1-6
Prove que o tempo de execução de um algoritmo é $\Theta(g(n))$ se e somente se seu tempo de execução do pior caso é $O(g(n))$ e seu tempo de execução do melhor caso é $\Omega(g(n))$.

3.1-7
Prove que $o(g(n)) \cap \omega(g(n))$ é o conjunto vazio.

3.1-8
Podemos estender nossa notação ao caso de dois parâmetros n e m que podem tender a infinito independentemente a taxas distintas. Para uma dada função $g(n, m)$, denotamos por $O(g(n, m))$ o conjunto de funções

$$O(g(n, m)) = \{f(n, m) : \text{existem constantes positivas } c, n_0 \text{ e } m_0 \text{ tais que } 0 \le f(n, m) \le cg(n, m) \text{ para todo } n \ge n_0 \text{ e } m \ge m_0\} .$$

Forneça definições correspondentes para $\Omega(g(n, m))$ e $\Theta(g(n, m))$.

3.2 Notações padrão e funções comuns

Esta seção revê algumas funções e notações matemáticas padrão e explora os relacionamentos entre elas. A seção também ilustra o uso das notações assintóticas.

Monotonicidade

Uma função $f(n)$ é ***monotonicamente crescente*** (ou ***monotonamente crescente***) se $m \le n$ implica $f(m) \le f(n)$. De modo semelhante, ela é ***monotonicamente decrescente*** (ou ***monotonamente decrescente***) se $m \le n$ implica $f(m) \ge f(n)$. Uma função $f(n)$ é ***estritamente crescente*** se $m < n$ implica $f(m) < f(n)$ e ***estritamente decrescente*** se $m < n$ implica $f(m) > f(n)$.

Pisos e tetos

Para qualquer número real x, denotamos o maior inteiro menor que ou igual a x por $\lfloor x \rfloor$ (lê-se "o piso de x") e o menor inteiro maior que ou igual a x por $\lceil x \rceil$ (lê-se "o teto de x"). Para todo x real,

$$x - 1 < \lfloor x \rfloor \le x \le \lceil x \rceil < x + 1 . \tag{3.3}$$

Para qualquer inteiro n,

$$\lceil n/2 \rceil + \lfloor n/2 \rfloor = n$$

e, para qualquer número real $n \ge 0$ e inteiros $a, b > 0$,

$$\lceil \lceil n/a \rceil / b \rceil = \lceil n/ab \rceil , \tag{3.4}$$

$$\lfloor \lfloor n/a \rfloor / b \rfloor = \lfloor n/ab \rfloor , \tag{3.5}$$

$$\lceil a/b \rceil \leq (a + (b - 1))/b, \tag{3.6}$$

$$\lfloor a/b \rfloor \leq (a - (b - 1))/b. \tag{3.7}$$

A função piso $f(x) = \lfloor x \rfloor$ é monotonicamente crescente, como também a função teto $f(x) = \lceil x \rceil$.

Aritmética modular

Para qualquer inteiro a e qualquer inteiro positivo n, o valor $a \bmod n$ é o ***resto*** (ou ***resíduo***) do quociente a/n:

$$a \bmod n = a - \lfloor a/n \rfloor n. \tag{3.8}$$

Dada uma noção bem definida do resto da divisão de um inteiro por outro, é conveniente fornecer uma notação especial para indicar a igualdade de restos. Se $(a \bmod n) = (b \bmod n)$, escrevemos $a \equiv b \pmod n$ e dizemos que a é ***equivalente*** a b, módulo n. Em outras palavras, $a \equiv b \pmod n$ se a e b têm o mesmo resto quando divididos por n. De modo equivalente, $a \equiv b \pmod n$ se e somente se n é um divisor de $b - a$. Escrevemos $a \not\equiv b \pmod n$ se a não é equivalente a b, módulo n.

Polinômios

Dado um inteiro não negativo d, um ***polinômio em n de grau d*** é uma função $p(n)$ da forma

$$p(n) = \sum_{i=0}^{d} a_i n^i ,$$

onde as constantes $a_0, a_1, \ldots, a_d$ são os ***coeficientes*** do polinômio e $a_d \neq 0$. Um polinômio é assintoticamente positivo se e somente se $a_d > 0$. No caso de um polinômio assintoticamente positivo $p(n)$ de grau d, temos $p(n) = \Theta(n^d)$. Para qualquer constante real $a \geq 0$, a função n^a é monotonicamente crescente, e para qualquer constante real $a \leq 0$, a função n^a é monotonicamente decrescente. Dizemos que uma função $f(n)$ é ***polinomialmente limitada*** se $f(n) = O(n^k)$ para alguma constante k.

Exponenciais

Para todos os valores $a \neq 0$, m e n reais, temos as seguintes identidades:

$$\begin{aligned} a^0 &= 1, \\ a^1 &= a, \\ a^{-1} &= 1/a, \\ (a^m)^n &= a^{mn}, \\ (a^m)^n &= (a^n)^m, \\ a^m a^n &= a^{m+n}. \end{aligned}$$

Para todo n e $a \geq 1$, a função a^n é monotonicamente crescente em n. Quando conveniente, consideraremos $0^0 = 1$.

As taxas de crescimento de polinômios e exponenciais podem ser relacionadas pelo fato a seguir. Para todas as constantes reais a e b tais que $a > 1$,

$$\lim_{n \to \infty} \frac{n^b}{a^n} = 0 , \tag{3.9}$$

da qual podemos concluir que

$$n^b = o(a^n) .$$

Portanto, qualquer função exponencial com uma base estritamente maior que 1 cresce mais rapidamente que qualquer função polinomial.

Usando e para denotar 2,71828 ..., a base da função logaritmo natural, temos para todo x real,

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots = \sum_{i=0}^{\infty} \frac{x^i}{i!} , \tag{3.10}$$

onde "!" denota a função fatorial, definida mais adiante nesta seção. Para todo x real, temos a desigualdade

$$e^x \geq 1 + x , \tag{3.11}$$

onde a igualdade se mantém válida somente quando $x = 0$. Quando $|x| \leq 1$, temos a aproximação

$$1 + x \leq e^x \leq 1 + x + x^2 . \tag{3.12}$$

Quando $x \to 0$, a aproximação de e^x por $1 + x$ é bastante boa:

$$e^x = 1 + x + \Theta(x^2) .$$

(Nessa equação, a notação assintótica é usada para descrever o comportamento de limite como $x \to 0$ em lugar de $x \to \infty$.) Temos, para todo x,

$$\lim_{n \to \infty} \left(1 + \frac{x}{n}\right)^n = e^x . \tag{3.13}$$

Logaritmos

Utilizaremos as seguintes notações:

$$\begin{aligned}
\lg n &= \log_2 n && \text{(logaritmo binário),} \\
\ln n &= \log_e n && \text{(logaritmo natural),} \\
\lg^k n &= (\lg n)^k && \text{(exponenciação),} \\
\lg \lg n &= \lg(\lg n) && \text{(composição).}
\end{aligned}$$

Uma importante convenção notacional que adotaremos é que as *funções logarítmicas se aplicarão apenas ao próximo termo na fórmula*; assim, $\lg n + k$ significará $(\lg n) + k$ e não $\lg(n + k)$. Se mantivermos $b > 1$ constante, então para $n > 0$, a função $\log_b n$ será estritamente crescente.

Para todo $a > 0$, $b > 0$, $c > 0$ e n real,

$$a = b^{\log_b a}, \tag{3.14}$$

$$\log_c(ab) = \log_c a + \log_c b,$$

$$\log_b a^n = n \log_b a,$$

$$\log_b a = \frac{\log_c a}{\log_c b},$$

$$\log_b(1/a) = -\log_b a, \tag{3.15}$$

$$\log_b a = \frac{1}{\log_a b}$$

$$a^{\log_b c} = c^{\log_b a},$$

onde, em cada equação anterior, as bases de logaritmos não são iguais a 1.

Pela equação (3.14), a mudança da base de um logaritmo de uma constante para outra só altera o valor do logaritmo por um fator constante, e assim usaremos com freqüência a notação "$\lg n$" quando não nos importarmos com fatores constantes, como na notação O. Os cientistas da computação consideram 2 a base mais natural para logaritmos, porque muitos algoritmos e estruturas de dados envolvem a divisão de um problema em duas partes.

Existe uma expansão de série simples para $\ln(1 + x)$ quando $|x| < 1$:

$$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \frac{x^5}{5} - \cdots.$$

Também temos as seguintes desigualdades para $x > -1$:

$$\frac{x}{1+x} \le \ln(1+x) \le x, \tag{3.16}$$

onde a igualdade é válida somente para $x = 0$.

Dizemos que uma função $f(n)$ é ***polilogaritmicamente limitada*** se $f(n) = O(\lg^k n)$ para alguma constante k. Podemos relacionar o crescimento de polinômios e polilogaritmos substituindo n por $\lg n$ e a por 2^a na equação (3.9), resultando em:

$$\lim_{n\to\infty} \frac{\lg^b n}{(2^a)^{\lg n}} = \lim_{n\to\infty} \frac{\lg^b n}{n^a} = 0.$$

A partir desse limite, podemos concluir que

$$\lg^b n = o(n^a)$$

para qualquer constante $a > 0$. Desse modo, qualquer função polinomial positiva cresce mais rapidamente que qualquer função polilogarítmica.

Fatoriais

A notação $n!$ (lê-se "n fatorial") é definida para inteiros $n \geq 0$ como

$$n! = \begin{cases} 1 & \text{se } n = 0, \\ n \cdot (n-1)! & \text{se } n > 0. \end{cases}$$

Então, $n! = 1 \cdot 2 \cdot 3 \cdots n$.

Um limite superior fraco na função fatorial é $n! \leq n^n$, pois cada um dos n termos no produto do fatorial é no máximo n. A ***aproximação de Stirling***,

$$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n\left(1 + \Theta\left(\frac{1}{n}\right)\right), \tag{3.17}$$

onde e é a base do logaritmo natural, nos dá um limite superior mais restrito, e um limite inferior também. Pode-se demonstrar que (consulte o Exercício 3.2-3)

$$\begin{aligned} n! &= o(n^n), \\ n! &= \omega(2^n), \\ \lg(n!) &= \Theta(n \lg n), \end{aligned} \tag{3.18}$$

onde a aproximação de Stirling é útil na demonstração da equação (3.18). A equação a seguir também é válida para todo $n \geq 1$:

$$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{\alpha_n} \tag{3.19}$$

onde

$$\frac{1}{12n+1} < \alpha_n < \frac{1}{12n}. \tag{3.20}$$

Iteração funcional

Usamos a notação $f^{(i)}(n)$ para denotar a função $f(n)$ aplicada iterativamente i vezes a um valor inicial n. Formalmente, seja $f(n)$ uma função sobre os reais. Para inteiros não negativos i, definimos recursivamente:

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

Por exemplo, se $f(n) = 2n$, então $f^{(i)}(n) = 2^i n$.

A função logaritmo repetido

Usamos a notação $\lg^* n$ (lê-se "log asterisco de n") para denotar o logaritmo repetido, que é definido como a seguir. Seja $\lg^{(i)} n$ definida da maneira anterior, com $f(n) \lg n$. Como o logaritmo de um número não positivo é indefinido, $\lg^{(i)} n$ só é definido se $\lg^{(i-1)} n > 0$. Certifique-se de distinguir $\lg^{(i)} n$ (a função logaritmo aplicada i vezes em sucessão, começando com o argumento n) de $\lg^i n$ (o logaritmo de n elevado à i-ésima potência). A função logaritmo repetido é definida como

$$\lg^* n = \min \{i \geq 0 : \lg^{(i)} n \leq 1\}.$$

O logaritmo repetido é uma função que cresce *muito* lentamente:

$$\begin{aligned} \lg^* 2 &= 1 , \\ \lg^* 4 &= 2 , \\ \lg^* 16 &= 3 , \\ \lg^* 65536 &= 4 , \\ \lg^* (2^{65536}) &= 5 . \end{aligned}$$

Tendo em vista que o número de átomos no universo visível é estimado em cerca 10^{80}, que é muito menor que 2^{65536}, raramente encontraremos uma entrada de tamanho n tal que $\lg^* n > 5$.

Números de Fibonacci

Os ***números de Fibonacci*** são definidos pela seguinte recorrência:

$$\begin{aligned} F_0 &= 0 , \\ F_1 &= 1 , \\ F_i &= F_{i-1} + F_{i-2} \quad \text{para } i \geq 2. \end{aligned} \tag{3.21}$$

Portanto, cada número de Fibonacci é a soma dos dois números anteriores, produzindo a seqüência

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...

Os números de Fibonacci estão relacionados com a ***razão áurea*** ϕ e a seu conjugado $\hat{\phi}$, que são dados pelas seguintes fórmulas:

$$\begin{aligned} \phi &= \frac{1+\sqrt{5}}{2} \\ &= 1{,}61803\ldots , \\ \hat{\phi} &= \frac{1-\sqrt{5}}{2} \\ &= -0{,}61803\ldots . \end{aligned} \tag{3.22}$$

Especificamente, temos

$$F_i = \frac{\phi^i - \hat{\phi}^i}{\sqrt{5}} \tag{3.23}$$

que pode ser demonstrada por indução (Exercício 3.2-6). Como $|\hat{\phi}| < 1$, temos $|\hat{\phi}| / \sqrt{5} < 1 / \sqrt{5} < 1/2$, de modo que o i-ésimo número de Fibonacci F_i é igual a $\phi^i / \sqrt{5}$ arredondado para o inteiro mais próximo. Desse modo, os números de Fibonacci crescem exponencialmente.

Exercícios

3.2-1
Mostre que, se $f(n)$ e $g(n)$ são funções monotonicamente crescentes, então também o são as funções $f(n) + g(n)$ e $f(g(n))$, e se $f(n)$ e $g(n)$ são além disso não negativas, então $f(n) \cdot g(n)$ é monotonicamente crescente.

3.2-2
Prove a equação (3.15).

3.2-3
Prove a equação (3.18). Prove também que $n! = \omega(2^n)$ e que $n! = o(n^n)$.

3.2-4 ★
A função $\lceil \lg n \rceil !$ é polinomialmente limitada? A função $\lceil \lg \lg n \rceil !$ é polinomialmente limitada?

3.2-5 ★
Qual é assintoticamente maior: $\lg(\lg^* n)$ ou $\lg^* (\lg n)$?

3.2-6
Prove por indução que o i-ésimo número de Fibonacci satisfaz à igualdade

$$F_i = \frac{\phi^i - \hat{\phi}^i}{\sqrt{5}},$$

onde ϕ é a razão áurea e $\hat{\phi}$ é seu conjugado.

3.2-7
Prove que, para $i \geq 0$, o $(i + 2)$-ésimo número de Fibonacci satisfaz a $F_{i+2} \geq \phi^i$.

Problemas

3-1 Comportamento assintótico de polinômios
Seja

$$p(n) = \sum_{i=0}^{d} a_i n^i ,$$

onde $a_d > 0$, um polinômio de grau d em n, e seja k uma constante. Use as definições das notações assintóticas para provar as propriedades a seguir.

a. Se $k \geq d$, então $p(n) = O(n^k)$.

b. Se $k \leq d$, então $p(n) = \Omega(n^k)$.

c. Se $k = d$, então $p(n) = \Theta(n^k)$.

d. Se $k > d$, então $p(n) = o(n^k)$.

e. Se $k < d$, então $p(n) = \omega(n^k)$.

3-2 Crescimentos assintóticos relativos
Indique, para cada par de expressões (A, B) na tabela a seguir, se A é O, o, Ω, ω ou Θ de B. Suponha que $k \geq 1$, $\epsilon > 0$ e $c > 1$ são constantes. Sua resposta deve estar na forma da tabela, com "sim" ou "não" escrito em cada retângulo.

	A	B	O	o	Ω	ω	Θ
a.	$\lg^k n$	n^ϵ					
b.	n^k	c^n					
c.	$\sqrt{n}$	$n^{\sin n}$					
d.	2^n	$2^{n/2}$					
e.	$n^{\lg c}$	$c^{\lg n}$					
f.	$\lg(n!)$	$\lg(n^n)$					

3-3 Ordenação por taxas de crescimento assintóticas

a. Ordene as funções a seguir por ordem de crescimento; ou seja, encontre um arranjo $g_1, g_2, \ldots, g_{30}$ das funções que satisfazem a $g_1 = \Omega(g_2), g_2 = \Omega(g_3), \ldots, g_{29} = \Omega(g_{30})$. Particione sua lista em classes de equivalência tais que $f(n)$ e $g(n)$ estejam na mesma classe se e somente se $f(n) = \Theta(g(n))$.

$\lg(\lg^* n)$	$2^{\lg^* n}$	$(\sqrt{2})^{\lg n}$	n^2	$n!$	$(\lg n)!$
$(\frac{3}{2})^n$	n^3	$\lg^2 n$	$\lg(n!)$	2^{2^n}	$n^{1/\lg n}$
$\ln \ln n$	$\lg^* n$	$n \cdot 2^n$	$n^{\lg \lg n}$	$\ln n$	1
$2^{\lg n}$	$(\lg n)^{\lg n}$	e^n	$4^{\lg n}$	$(n+1)!$	$\sqrt{\lg n}$
$\lg^*(\lg n)$	$2^{\sqrt{2 \lg n}}$	n	$2n$	$n \lg n$	2^{2n+1}

b. Dê um exemplo de uma única função não negativa $f(n)$ tal que, para todas as funções $g_i(n)$ da parte (a), $f(n)$ não seja nem $O(g_i(n))$, nem $\Omega(g_i(n))$.

3-4 Propriedades da notação assintótica

Sejam $f(n)$ e $g(n)$ funções assintoticamente positivas. Prove ou conteste cada uma das seguintes conjecturas.

a. $f(n) = O(g(n))$ implica $g(n) = O(f(n))$.

b. $f(n) + g(n) = \Theta(\min(f(n), g(n)))$.

c. $f(n) = O(g(n))$ implica $\lg(f(n)) = O(\lg(g(n)))$, onde $\lg(g(n)) \geq 1$ e $f(n) \geq 1$ para todo n suficientemente grande.

d. $f(n) = O(g(n))$ implica $2^{f(n)} = O(2^{g(n)})$.

e. $f(n) = O((f(n))^2)$.

f. $f(n) = O(g(n))$ implica $g(n) = \Omega(f(n))$.

g. $f(n) = \Theta(f(n/2))$.

h. $f(n) + o(f(n)) = \Theta(f(n))$.

3-5 Variações sobre O e Ω

Alguns autores definem Ω de um modo ligeiramente diferente do modo como nós definimos; vamos usar $\overset{\infty}{\Omega}$ (lê-se "ômega infinito") para essa definição alternativa. Dizemos que $f(n) = \overset{\infty}{\Omega}(g(n))$ se existe uma constante positiva c tal que $f(n) \geq cg(n) \geq 0$ para infinitamente muitos inteiros n.

a. Mostre que, para duas funções quaisquer $f(n)$ e $g(n)$ que são assintoticamente não negativas, $f(n) = O(g(n))$ ou $f(n) = \overset{\infty}{\Omega}(g(n))$ ou ambas, enquanto isso não é verdade se usamos Ω em lugar de $\overset{\infty}{\Omega}$.

b. Descreva as vantagens e as desvantagens potenciais de se usar $\overset{\infty}{\Omega}$ em vez de Ω para caracterizar os tempos de execução de programas.

Alguns autores também definem O de um modo ligeiramente diferente; vamos usar O' para a definição alternativa. Dizemos que $f(n) = O'(g(n))$ se e somente se $|f(n)| = O(g(n))$.

c. O que acontece para cada sentido de "se e somente se" no Teorema 3.1 se substituirmos O por O', mas ainda usarmos Ω?

Alguns autores definem $\widetilde{O}$ (lê-se "'o' suave") para indicar O com fatores logarítmicos ignorados:

$$\widetilde{O}(g(n)) = \{f(n) : \text{ existem constantes positivas } c, k \text{ e } n_0 \text{ tais que } 0 \leq f(n) \leq cg(n) \lg^k(n) \text{ para todo } n \geq n_0\} .$$

d. Defina $\widetilde{\Omega}$ e $\widetilde{\Theta}$ de maneira semelhante. Prove a analogia correspondente ao Teorema 3.1.

3-6 Funções repetidas

O operador de iteração * usado na função lg* pode ser aplicado a qualquer função monotonicamente crescente $f(n)$ sobre os reais. Para uma dada constante $c \in \mathbf{R}$, definimos a função repetida (ou iterada) f_c^* por

$$f_c^*(n) = \min \{i \geq 0 : f^{(i)}(n) \leq c\} ,$$

que não necessita ser bem definida em todos os casos. Em outras palavras, a quantidade $f_c^*(n)$ é o número de aplicações repetidas da função f necessárias para reduzir seu argumento a c ou menos.

Para cada uma das funções $f(n)$ e constantes c a seguir, forneça um limite tão restrito quanto possível sobre $f_c^*(n)$.

	$f(n)$	c	$f_c^*(n)$
a.	$n - 1$	0	
b.	$\lg n$	1	
c.	$n/2$	1	
d.	$n/2$	2	
e.	$\sqrt{n}$	2	
f.	$\sqrt{n}$	1	
g.	$n^{1/3}$	2	
h.	$n/\lg n$	2	

Notas do capítulo

Knuth [182] traça a origem da notação O até um texto de teoria dos números escrito por P. Bachmann em 1892. A notação o foi criada por E. Landau em 1909, para sua descrição da distribuição de números primos. As notações Ω e Θ foram defendidas por Knuth [186] para corrigir a prática popular, mas tecnicamente ruim, de se usar na literatura a notação O para limites superiores e inferiores. Muitas pessoas continuam a usar a notação O onde a notação Θ é mais precisa tecnicamente. Uma discussão adicional da história e do desenvolvimento de notações assintóticas pode ser encontrada em Knuth [182, 186] e em Brassard e Bratley [46].

Nem todos os autores definem as notações assintóticas do mesmo modo, embora as várias definições concordem na maioria das situações comuns. Algumas das definições alternativas envolvem funções que não são assintoticamente não negativas, desde que seus valores absolutos sejam adequadamente limitados.

A equação (3.19) se deve a Robbins [260]. Outras propriedades de funções matemáticas elementares podem ser encontradas em qualquer bom manual de referência de matemática, como Abramowitz e Stegun [1] ou Zwillinger [320], ou em um livro de cálculo, como Apostol [18] ou Thomas e Finney [296]. Knuth [182] e também Graham, Knuth e Patashnik [132] contêm uma grande quantidade de material sobre matemática discreta, como a que se utiliza em ciência da computação.

Capítulo 4

Recorrências

Como observamos na Seção 2.3.2, quando um algoritmo contém uma chamada recursiva a ele próprio, seu tempo de execução pode freqüentemente ser descrito por uma recorrência. Uma ***recorrência*** é uma equação ou desigualdade que descreve uma função em termos de seu valor em entradas menores. Por exemplo, vimos na Seção 2.3.2 que o tempo de execução do pior caso $T(n)$ do procedimento MERGE-SORT poderia ser descrito pela recorrência

$$T(n) = \begin{cases} \Theta(1) & \text{se } n = 1, \\ 2T(n/2) + \Theta(n) & \text{se } n > 1. \end{cases} \tag{4.1}$$

cuja solução se afirmava ser $T(n) = \Theta(n \lg n)$.

Este capítulo oferece três métodos para resolver recorrências – ou seja, para obter limites assintóticos "Θ" ou "O" sobre a solução. No ***método de substituição***, supomos um limite hipotético, e depois usamos a indução matemática para provar que nossa suposição era correta. O ***método de árvore de recursão*** converte a recorrência em uma árvore cujos nós representam os custos envolvidos em diversos níveis da recursão; usamos técnicas para limitar somatórios com a finalidade de solucionar a recorrência. O ***método mestre*** fornece limites para recorrências da forma

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

onde $a \geq 1$, $b > 1$ e $f(n)$ é uma função dada; o método requer memorização de três casos mas, depois que você o fizer, será fácil descobrir limites assintóticos para muitas recorrências simples.

Detalhes técnicos

Na prática, negligenciamos certos detalhes técnicos quando enunciamos e resolvemos recorrências. Um bom exemplo de um detalhe que freqüentemente é ignorado é a suposição de argumentos inteiros para funções. Normalmente, o tempo de execução $T(n)$ de um algoritmo só é definido quando n é um inteiro, pois, para a maior parte dos algoritmos, o tamanho da entrada é sempre um inteiro. Por exemplo, a recorrência que descreve o tempo de execução do pior caso de MERGE-SORT é na realidade

$$T(n) = \begin{cases} \Theta(1) & \text{se } n = 1, \\ T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \Theta(n) & \text{se } n > 1. \end{cases} \tag{4.2}$$

As condições limite representam outra classe de detalhes que em geral ignoramos. Tendo em vista que o tempo de execução de um algoritmo sobre uma entrada de dimensões constantes é uma constante, as recorrências que surgem dos tempos de execução de algoritmos geralmente têm $T(n) = \Theta(1)$ para n suficientemente pequeno. Em conseqüência disso, por conveniência, em geral omitiremos declarações sobre as condições limite de recorrências e iremos supor que $T(n)$ é constante para n pequeno. Por exemplo, normalmente enunciamos a recorrência (4.1) como

$$T(n) = 2T(n/2) + \Theta(n) , \tag{4.3}$$

sem fornecer explicitamente valores para n pequeno. A razão é que, embora a mudança no valor de $T(1)$ altere a solução para a recorrência, normalmente a solução não muda por mais de um fator constante, e assim a ordem de crescimento não é alterada.

Quando enunciamos e resolvemos recorrências, com freqüência omitimos pisos, tetos e condições limite. Seguimos em frente sem esses detalhes, e mais tarde definimos se eles têm importância ou não. Em geral, eles não têm importância, mas é importante saber quando têm. A experiência ajuda, e também alguns teoremas declarando que esses detalhes não afetam os limites assintóticos de muitas recorrências encontradas na análise de algoritmos (ver Teorema 4.1). Porém, neste capítulo, examinaremos alguns desses detalhes para mostrar os ótimos métodos de solução de pontos de recorrência.

4.1 O método de substituição

O método de substituição para resolver recorrências envolve duas etapas:

1. Pressupor a forma da solução.
2. Usar a indução matemática para encontrar as constantes e mostrar que a solução funciona.

O nome vem da substituição da resposta pressuposta para a função quando a hipótese indutiva é aplicada a valores menores. Esse método é eficiente, mas obviamente só pode ser aplicado em casos nos quais é fácil pressupor a forma da resposta.

O método de substituição pode ser usado para estabelecer limites superiores ou inferiores sobre uma recorrência. Como exemplo, vamos determinar um limite superior sobre a recorrência

$$T(n) = 2T(\lfloor n/2 \rfloor) + n , \tag{4.4}$$

que é semelhante às recorrências (4.2) e (4.3). Supomos que a solução é $T(n) = O(n \lg n)$. Nosso método é provar que $T(n) \leq cn \lg n$ para uma escolha apropriada da constante $c > 0$. Começamos supondo que esse limite se mantém válido para $\lfloor n/2 \rfloor$, ou seja, que $T(\lfloor n/2 \rfloor) \leq c \lfloor n/2 \rfloor \lg(\lfloor n/2 \rfloor)$. A substituição na recorrência produz

$$\begin{aligned} T(n) &\leq 2(c \lfloor n/2 \rfloor \lg(\lfloor n/2 \rfloor)) + n \\ &\leq cn \lg(n/2) + n \\ &= cn \lg n - cn \lg 2 + n \\ &= cn \lg n - cn + n \\ &\leq cn \lg n , \end{aligned}$$

onde o último passo é válido desde que $c \geq 1$.

Agora, a indução matemática exige que mostremos que nossa solução se mantém válida para as condições limite. Normalmente, fazemos isso mostrando que as condições limite são adequadas para casos básicos da prova indutiva. No caso da recorrência (4.4), devemos mostrar que podemos escolher uma constante c grande o suficiente para que o limite $T(n) \leq cn \lg n$ também funcione para as condições limite. Às vezes, essa exigência pode levar a problemas. Vamos supor, como argumento, que $T(1) = 1$ seja a única condição limite da recorrência. Então, para $n = 1$, o limite $T(n) \leq cn \lg n$ produz $T(1) \leq c\ 1 \lg 1 = 0$, o que está em desacordo com $T(1) = 1$. Conseqüentemente, o caso básico de nossa prova indutiva deixa de ser válido.

Essa dificuldade para provar uma hipótese indutiva para uma condição limite específica pode ser facilmente contornada. Por exemplo, na recorrência (4.4), tiramos proveito do fato de que a notação assintótica só exige que demonstremos que $T(n) \leq cn \lg n$ para $n \geq n_0$, onde n_0 é uma constante de nossa escolha. A idéia é remover a difícil condição limite $T(1) = 1$ da consideração na prova indutiva. Observe que, para $n > 3$, a recorrência não depende diretamente de $T(1)$. Desse modo, podemos substituir $T(1)$ por $T(2)$ e $T(3)$ como os casos básicos na prova indutiva, deixando $n_0 = 2$. Observe que fazemos uma distinção entre o caso básico da recorrência ($n = 1$) e os casos básicos da prova indutiva ($n = 2$ e $n = 3$). Da recorrência, derivamos $T(2) = 4$ e $T(3) = 5$. A prova indutiva de que $T(n) \leq cn \lg n$ para alguma constante $c \geq 1$ pode agora ser completada escolhendo-se um valor de c grande o bastante para que $T(2) \leq c\ 2 \lg 2$ e $T(3) \leq c\ 3 \lg 3$. Como observamos, qualquer valor de $c \geq 2$ é suficiente para que os casos básicos de $n = 2$ e $n = 3$ sejam válidos. Para a maioria das recorrências que examinaremos, é simples estender as condições limite para fazer a hipótese indutiva funcionar para n pequeno.

Como fazer um bom palpite

Infelizmente, não há nenhum modo geral de adivinhar as soluções corretas para recorrências. Pressupor uma solução exige experiência e, ocasionalmente, criatividade. Entretanto, por sorte, existem algumas heurísticas que podem ajudá-lo a se tornar um bom criador de suposições. Você também poderá usar árvores de recursão, que veremos na Seção 4.2, para gerar boas hipóteses.

Se uma recorrência for semelhante a uma que você já tenha visto antes, então será razoável supor uma solução semelhante. Como exemplo, considere a recorrência

$$T(n) = 2T(\lfloor n/2 \rfloor + 17) + n\ ,$$

que parece difícil devido ao "17" acrescentado ao argumento de T no lado direito. Intuitivamente, porém, esse termo adicional não pode afetar de maneira substancial a solução para a recorrência. Quando n é grande, a diferença entre $T(\lfloor n/2 \rfloor)$ e $T(\lfloor n/2 \rfloor + 17)$ não é tão grande: ambos os termos cortam n quase uniformemente pela metade. Em conseqüência disso, fazemos a suposição de que $T(n) = O(n \lg n)$, que você pode verificar como correto usando o método de substituição (ver Exercício 4.1-5).

Outro caminho para fazer uma boa suposição é provar informalmente limites superiores e inferiores sobre a recorrência, e então reduzir o intervalo de incerteza. Por exemplo, podemos começar com o limite inferior $T(n) = \Omega(n)$ para a recorrência (4.4), pois temos o termo n na recorrência, e podemos demonstrar um limite superior inicial $T(n) = O(n^2)$. Então, podemos diminuir gradualmente o limite superior e elevar o limite inferior, até convergirmos sobre a solução correta, assintoticamente restrita, $T(n) = \Theta(n \lg n)$.

Sutilezas

Há momentos em que é possível fazer uma suposição correta em um limite assintótico sobre a solução de uma recorrência, mas, de algum modo, a matemática não parece funcionar na indu-

ção. Em geral, o problema é que a hipótese indutiva não é forte o bastante para provar o limite detalhado. Quando você chegar a um nó como esse, revisar a suposição subtraindo um termo de mais baixa ordem freqüentemente permitirá que a matemática funcione.

Considere a recorrência

$$T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1 .$$

Supomos por hipótese que a solução seja $O(n)$ e tentamos mostrar que $T(n) \leq cn$ para uma escolha apropriada da constante c. Substituindo nossa suposição na recorrência, obtemos

$$\begin{aligned} T(n) &\leq c\lfloor n/2 \rfloor + c\lceil n/2 \rceil + 1 \\ &= cn + 1 , \end{aligned}$$

o que não implica $T(n) \leq cn$ para qualquer escolha de c. É uma tentação experimentar um palpite maior, digamos $T(n) = O(n^2)$, o que poderia funcionar; porém, de fato, nossa suposição de que a solução é $T(n) = O(n)$ é correta. No entanto, para mostrar isso, devemos criar uma hipótese indutiva mais forte.

Intuitivamente, nossa suposição é quase correta: a única diferença é a constante 1, um termo de mais baixa ordem. Apesar disso, a indução matemática não funciona, a menos que provemos a forma exata da hipótese indutiva. Superamos nossa dificuldade *subtraindo* um termo de mais baixa ordem da nossa suposição anterior. Nossa nova suposição é $T(n) \leq cn - b$, onde $b \geq 0$ é constante. Agora temos

$$\begin{aligned} T(n) &\leq (c\lfloor n/2 \rfloor - b) + (c\lceil n/2 \rceil - b) + 1 \\ &= cn - 2b + 1 \\ &\leq cn - b , \end{aligned}$$

desde que $b \geq 1$. Como antes, a constante c deve ser escolhida com um valor grande o suficiente para tratar as condições limite.

A maioria das pessoas acha antiintuitiva a idéia de subtrair um termo de mais baixa ordem. Afinal, se a matemática não funciona, não deveríamos estar aumentando nossa suposição? A chave para entender esse passo é lembrar que estamos usando a indução matemática: podemos provar algo mais forte para um determinado valor, supondo algo mais forte para valores menores.

Como evitar armadilhas

É fácil errar na utilização da notação assintótica. Por exemplo, na recorrência (4.4), podemos "provar" falsamente que $T(n) = O(n)$, supondo $T(n) \leq cn$, e então argumentando que

$$\begin{aligned} T(n) &\leq 2(c\lfloor n/2 \rfloor) + n \\ &\leq cn + n \\ &= O(n) , \qquad \Leftarrow \textit{errado!!} \end{aligned}$$

pois c é uma constante. O erro é que não provamos a *forma exata* da hipótese indutiva, ou seja, que $T(n) \leq cn$.

Como trocar variáveis

Às vezes, um pouco de manipulação algébrica pode tornar uma recorrência desconhecida semelhante a uma que você já viu antes. Como exemplo, considere a recorrência

$$T(n) = 2T(\lfloor\sqrt{n}\rfloor) + \lg n ,$$

que parece difícil. Entretanto, podemos simplificar essa recorrência com uma troca de variáveis. Por conveniência, não nos preocuparemos em arredondar valores, como $\sqrt{n}$, para inteiros. Renomear $m = \lg n$ produz

$$T(2^m) = 2T(2^{m/2}) + m .$$

Agora podemos renomear $S(m) = T(2^m)$ para produzir a nova recorrência

$$S(m) = 2S(m/2) + m ,$$

que é muito semelhante à recorrência (4.4). De fato, essa nova recorrência tem a mesma solução: $S(m) = O(m \lg m)$. Trocando de volta de $S(m)$ para $T(n)$, obtemos $T(n) = T(2^m) = S(m) = O(m \lg m) = O(\lg n \lg \lg n)$.

Exercícios

4.1-1
Mostre que a solução de $T(n) = T(\lceil n/2 \rceil) + 1$ é $O(\lg n)$.

4.1-2
Vimos que a solução de $T(n) = 2T(\lfloor n/2 \rfloor) + n$ é $O(n \lg n)$. Mostre que a solução dessa recorrência também é $\Omega(n \lg n)$. Conclua que a solução é $\Theta(n \lg n)$.

4.1-3
Mostre que, supondo uma hipótese indutiva diferente, podemos superar a dificuldade com a condição limite $T(1) = 1$ para a recorrência (4.4), sem ajustar as condições limite para a prova indutiva.

4.1-4
Mostre que $\Theta(n \lg n)$ é a solução para a recorrência "exata" (4.2) para a ordenação por intercalação.

4.1-5
Mostre que a solução para $T(n) = 2T(\lfloor n/2 \rfloor + 17) + n$ é $O(n \lg n)$.

4.1-6
Resolva a recorrência $T(n) = 2T(\sqrt{n})$, fazendo uma troca de variáveis. Sua solução deve ser assintoticamente restrita. Não se preocupe em saber se os valores são integrais.

4.2 O método de árvore de recursão

Embora o método de substituição possa fornecer uma prova sucinta de que uma solução para uma recorrência é correta, às vezes é difícil apresentar uma boa suposição. Traçar uma árvore de recursão, como fizemos em nossa análise da recorrência de ordenação por intercalação na Seção 2.3.2, é um caminho direto para se criar uma boa suposição. Em uma ***árvore de recursão***, cada nó representa o custo de um único subproblema em algum lugar no conjunto de invocações de funções recursivas. Somamos os custos dentro de cada nível da árvore para obter um conjunto de custos por nível, e então somamos todos os custos por nível para determinar o custo total de todos os níveis da recursão. As árvores de recursão são particularmente úteis quando a recorrência descreve o tempo de execução de um algoritmo de dividir e conquistar.

Uma árvore de recursão é mais bem usada para gerar uma boa suposição, que é então verificada pelo método de substituição. Quando utiliza uma árvore de recursão para gerar uma boa suposição, você freqüentemente pode tolerar uma pequena quantidade de "sujeira", pois irá verificar sua suposição mais tarde. Porém, se for muito cuidadoso ao criar uma árvore de recursão e somar os custos, você poderá usar a árvore de recursão como uma prova direta de uma solução para uma recorrência. Nesta seção, usaremos árvores de recursão para gerar boas suposições e, na Seção 4.4, utilizaremos árvores de recursão diretamente para provar o teorema que forma a base do método mestre.

Por exemplo, vamos ver como uma árvore de recursão forneceria uma boa suposição para a recorrência $T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2)$. Começamos concentrando nossa atenção em encontrar um limite superior para a solução. Considerando que sabemos que pisos e tetos são normalmente insatisfatórios para resolver recorrências (aqui está um exemplo de sujeira que podemos tolerar), criamos uma árvore de recursão para a recorrência $T(n) = 3T(n/4) + cn^2$, tendo estabelecido o coeficiente constante implícito $c > 0$.

A Figura 4.1 mostra a derivação da árvore de recursão para $T(n) = 3T(n/4) + cn^2$. Por conveniência, supomos que n é uma potência exata de 4 (outro exemplo de sujeira tolerável). A parte (a) da figura mostra $T(n)$ que, na parte (b), foi expandida até uma árvore equivalente representando a recorrência. O termo cn^2 na raiz representa o custo no nível de recursão superior, e as três subárvores da raiz representam os custos resultantes dos subproblemas de tamanho $n/4$. A parte (c) mostra esse processo levado um passo adiante pela expansão de cada nó com custo $T(n/4)$ da parte (b). O custo para cada um dos três filhos da raiz é $c(n/4)^2$. Continuamos a expandir cada nó na árvore, desmembrando-o em suas partes constituintes conforme determinado pela recorrência.

Tendo em vista que os tamanhos de subproblemas diminuem à medida que nos afastamos da raiz, eventualmente temos de alcançar uma condição limite. A que distância da raiz nós a encontramos? O tamanho do subproblema para um nó na profundidade i é $n/4^i$. Desse modo, o tamanho do subproblema chega a $n = 1$ quando $n/4^i = 1$ ou, de modo equivalente, quando $i = \log_4 n$. Assim, a árvore tem $\log_4 n + 1$ níveis $(0, 1, 2, \ldots, \log_4 n)$.

Em seguida, determinamos o custo em cada nível da árvore. Cada nível tem três vezes mais nós que o nível acima dele, e assim o número de nós na profundidade i é 3^i. Como os tamanhos de subproblemas se reduzem por um fator de 4 para cada nível que descemos a partir da raiz, cada nó na profundidade i, para $i = 0, 1, 2, \ldots, \log_4 n - 1$, tem o custo de $c(n/4^i)^2$. Multiplicando, vemos que o custo total sobre todos os nós na profundidade i, para $i = 0, 1, 2, \ldots, \log_4 n - 1$, é $3^i c(n/4^i)^2 = (3/16)^i cn^2$. O último nível, na profundidade $\log_4 n$, tem $3^{\log_4 n} = n^{\log_4 3}$ nós, cada qual contribuindo com o custo $T(1)$, para um custo total de $n^{\log_4 3} T(1)$, que é $\Theta(n^{\log_4 3})$.

Agora somamos os custos sobre todos os níveis para determinar o custo correspondente à árvore inteira:

$$T(n) = cn^2 + \frac{3}{16}cn^2 + \left(\frac{3}{16}\right)^2 cn^2 + \cdots + \left(\frac{3}{16}\right)^{\log_4 n - 1} cn^2 + \Theta(n^{\log_4 3})$$

$$= \sum_{i=0}^{\log_4 n - 1} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3})$$

$$= \frac{(3/16)^{\log_4 n} - 1}{(3/16) - 1} cn^2 + \Theta(n^{\log_4 3})$$

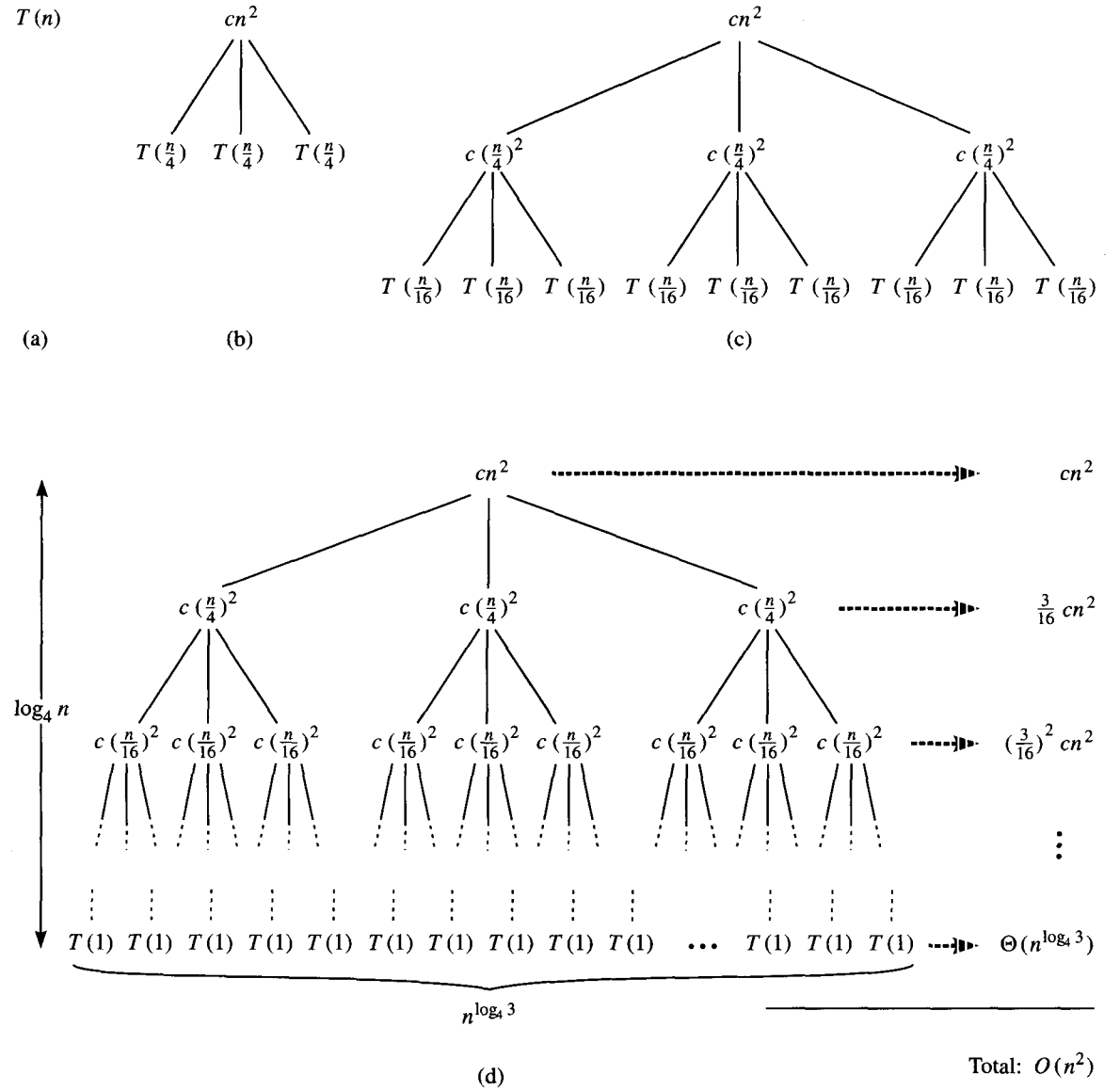

FIGURA 4.1 A construção de uma árvore de recursão para a recorrência $T(n) = 3T(n/4) + cn^2$. A parte **(a)** mostra $T(n)$, que é progressivamente expandido nas partes b a d para formar a árvore de recursão. A árvore completamente expandida da parte (d) tem altura $\log_4 n$ (ela tem $\log_4 n + 1$ níveis)

Essa última fórmula parece um pouco confusa até percebermos que é possível mais uma vez tirar proveito de pequenas porções de sujeira e usar uma série geométrica decrescente infinita como limite superior. Voltando um passo atrás e aplicando a equação (A.6), temos

$$
\begin{aligned}
T(n) &= \sum_{i=0}^{\log_4 n - 1} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\
&< \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\
&= \frac{1}{1-(3/16)} cn^2 + \Theta(n^{\log_4 3}) \\
&= \frac{16}{13} cn^2 + \Theta(n^{\log_4 3}) \\
&= O(n^2) .
\end{aligned}
$$

Desse modo, derivamos uma suposição de $T(n) = O(n^2)$ para nossa recorrência original $T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2)$. Nesse exemplo, os coeficientes de cn^2 formam uma série geométrica decrescente e, pela equação (A.6), a soma desses coeficientes é limitada na parte superior pela constante 16/13. Tendo em vista que a contribuição da raiz para o custo total é cn^2, a raiz contribui com uma fração constante do custo total. Em outras palavras, o custo total da árvore é dominado pelo custo da raiz.

De fato, se $O(n^2)$ é realmente um limite superior para a recorrência (como verificaremos em breve), então ele deve ser um limite restrito. Por quê? A primeira chamada recursiva contribui com o custo $\Theta(n^2)$, e então $\Omega(n^2)$ deve ser um limite inferior para a recorrência.

Agora podemos usar o método de substituição para verificar que nossa suposição era correta, isto é, $T(n) = O(n^2)$ é um limite superior para a recorrência $T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2)$. Queremos mostrar que $T(n) \leq dn^2$ para alguma constante $d > 0$. Usando a mesma constante $c > 0$ de antes, temos

$$\begin{aligned} T(n) &\leq 3T(\lfloor n/4 \rfloor) + cn^2 \\ &\leq 3d \lfloor n/4 \rfloor^2 + cn^2 \\ &\leq 3d\,(n/4)^2 + cn^2 \\ &= \frac{3}{16} dn^2 + cn^2 \\ &\leq dn^2 , \end{aligned}$$

onde a última etapa é válida desde que $d \geq (16/13)c$.

Como outro exemplo mais complicado, a Figura 4.2 mostra a árvore de recursão para

$$T(n) = T(n/3) + T(2n/3) + O(n) .$$

(Novamente, omitimos as funções piso e teto por simplicidade.) Como antes, c representa o fator constante no termo $O(n)$. Quando somamos os valores em todos os níveis da árvore de recursão, obtemos um valor cn para cada nível. O caminho mais longo da raiz até uma folha é $n \to (2/3)n \to (2/3)^2 n \to \cdots \to 1$. Tendo em vista que $(2/3)^k n = 1$ quando $k = \log_{3/2} n$, a altura da árvore é $\log_{3/2} n$.

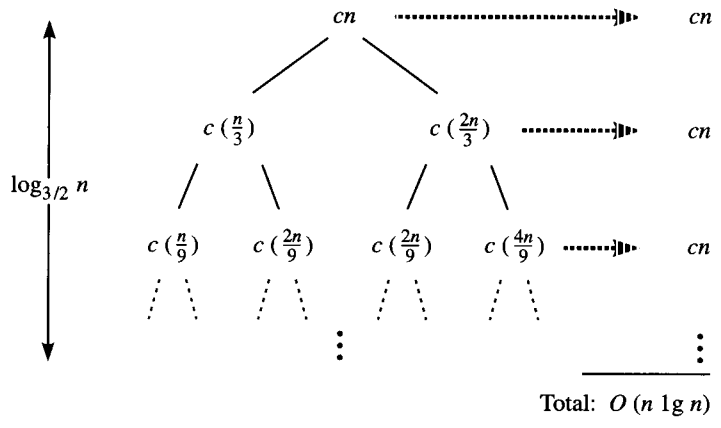

FIGURA 4.2 Uma árvore de recursão para a recorrência $T(n) = T(n/3) + T(2n/3) + cn$

Intuitivamente, esperamos que a solução para a recorrência seja no máximo o número de níveis vezes o custo de cada nível, ou $O(cn \log_{3/2} n) = O(n \lg n)$. O custo total é distribuído de

modo uniforme ao longo dos níveis da árvore de recursão. Existe uma complicação aqui: ainda temos de considerar o custo das folhas. Se essa árvore de recursão fosse uma árvore binária completa de altura $\log_{3/2} n$, haveria $2^{\log_{3/2} n} = n^{\log_{3/2} 2}$ folhas. Como o custo de cada folha é uma constante, o custo total de todas as folhas seria então $\Theta(n^{\log_{3/2} 2})$, que é $\omega(n \lg n)$. Contudo, essa árvore de recursão não é uma árvore binária completa, e assim ela tem menos de $n^{\log_{3/2} 2}$ folhas. Além disso, à medida que descemos a partir da raiz, mais e mais nós internos estão ausentes. Conseqüentemente, nem todos os níveis contribuem com um custo exatamente igual a cn; os níveis em direção à parte inferior contribuem menos. Poderíamos desenvolver uma contabilidade precisa de todos os custos, mas lembre-se de que estamos apenas tentando apresentar uma suposição para usar no método de substituição. Vamos tolerar a sujeira e tentar mostrar que uma suposição $O(n \lg n)$ para o limite superior é correta.

Na verdade, podemos usar o método de substituição para verificar aquele $O(n \lg n)$ é um limite superior para a solução da recorrência. Mostramos que $T(n) \leq dn \lg n$, onde d é uma constante positiva apropriada. Temos

$$
\begin{aligned}
T(n) &\leq T(n/3) + T(2n/3) + cn \\
&\leq d(n/3) \lg(n/3) + d(2n/3) \lg(2n/3) + cn \\
&= (d(n/3) \lg n - d(n/3) \lg 3) \\
&\qquad + (d(2n/3) \lg n = d(2n/3) \lg(3/2) + cn \\
&= dn \lg n - d(n/3) \lg 3 + (2n/3) \lg(3/2)) + cn \\
&= dn \lg n - d(n/3) \lg 3 + (2n/3) \lg 3 - (2n/3) \lg 2) + cn \\
&= dn \lg n - dn(\lg 3 - 2/3) + cn \\
&\leq dn \lg n ,
\end{aligned}
$$

desde que $d \geq c/(\lg 3 - (2/3))$. Desse modo, não tivemos de executar uma contabilidade de custos mais precisa na árvore de recursão.

Exercícios

4.2-1

Use uma árvore de recursão para determinar um bom limite superior assintótico na recorrência $T(n) = 3T(\lfloor n/2 \rfloor) + n$. Use o método de substituição para verificar sua resposta.

4.2-2

Demonstre que a solução para a recorrência $T(n) = T(n/3) + T(2n/3) + cn$, onde c é uma constante, é $\Omega(n \lg n)$, apelando para uma árvore de recursão.

4.2-3

Trace a árvore de recursão para $T(n) = 4T(\lfloor n/2 \rfloor) + cn$, onde c é uma constante, e forneça um limite assintótico restrito sobre sua solução. Verifique o limite pelo método de substituição.

4.2-4

Use uma árvore de recursão com o objetivo de fornecer uma solução assintoticamente restrita para a recorrência $T(n) = T(n - a) + T(a) + cn$, onde $a \geq 1$ e $c > 0$ são constantes.

4.2-5

Use uma árvore de recursão para fornecer uma solução assintoticamente restrita para a recorrência $T(n) = T(\alpha n) + T((1 - \alpha)n) + cn$, onde α é uma constante no intervalo $0 < \alpha < 1$ e $c > 0$ também é uma constante.

4.3 O método mestre

O método mestre fornece um processo de "livro de receitas" para resolver recorrências da forma

$$T(n) = aT(n/b) + f(n) , \tag{4.5}$$

onde $a \geq 1$ e $b > 1$ são constantes e $f(n)$ é uma função assintoticamente positiva. O método mestre exige a memorização de três casos, mas, daí em diante, a solução de muitas recorrências pode ser descoberta com grande facilidade, freqüentemente sem lápis e papel.

A recorrência (4.5) descreve o tempo de execução de um algoritmo que divide um problema de tamanho n em a subproblemas, cada um do tamanho n/b, onde a e b são constantes positivas. Os a subproblemas são resolvidos recursivamente, cada um no tempo $T(n/b)$. O custo de dividir o problema e combinar os resultados dos subproblemas é descrito pela função $f(n)$. (Ou seja, usando a notação da Seção 2.3.2, $f(n) = D(n) + C(n)$.) Por exemplo, a recorrência que surge do procedimento MERGE-SORT tem $a = 2$, $b = 2$ e $f(n) = \Theta(n)$.

Como uma questão de correção técnica, na realidade a recorrência não está bem definida, porque n/b não poderia ser um inteiro. Porém, a substituição de cada um dos a termos $T(n/b)$ por $T(\lfloor n/b \rfloor)$ ou $T(\lceil n/b \rceil)$ não afeta o comportamento assintótico da recorrência. (Provaremos isso na próxima seção.) Por essa razão, em geral, consideramos conveniente omitir as funções piso e teto ao se escreverem recorrências de dividir e conquistar dessa forma.

O teorema mestre

O método mestre depende do teorema a seguir.

Teorema 4.1 (Teorema mestre)
Sejam $a \geq 1$ e $b > 1$ constantes, seja $f(n)$ uma função e seja $T(n)$ definida sobre os inteiros não negativos pela recorrência

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

onde interpretamos n/b com o significado de $\lfloor n/b \rfloor$ ou $\lceil n/b \rceil$. Então, $T(n)$ pode ser limitado assintoticamente como a seguir.

1. Se $f(n) = O(n^{\log_b a - \epsilon})$ para alguma constante $\epsilon > 0$, então $T(n) = \Theta(n^{\log_b a})$.
2. Se $f(n) = \Theta(n^{\log_b a})$, então $T(n) = \Theta(n^{\log_b a} \lg n)$.
3. Se $f(n) = \Omega(n^{\log_b a + \epsilon})$ para alguma constante $\epsilon > 0$, e se $af(n/b) \leq cf(n)$ para alguma constante $c < 1$ e para todo n suficientemente grande, então $T(n) = \Theta(f(n))$. ■

Antes de aplicar o teorema mestre a alguns exemplos, vamos passar algum tempo tentando entender o que ele significa. Em cada um dos três casos, estamos comparando a função $f(n)$ com a função $n^{\log_b a}$. Intuitivamente, a solução para a recorrência é determinada pela maior das duas funções. Se, como no caso 1, a função $n^{\log_b a}$ for a maior, então a solução será $T(n) = \Theta(n^{\log_b a})$. Se, como no caso 3, a função $f(n)$ for a maior, então a solução será $T(n) = \Theta(f(n))$. Se, como no caso 2, as duas funções tiverem o mesmo tamanho, faremos a multiplicação por um fator logarítmico e a solução será $T(n) = \Theta(n^{\log_b a} \lg n) = \Theta(f(n) \lg n)$.

Além dessa intuição, existem alguns detalhes técnicos que devem ser entendidos. No primeiro caso, $f(n)$ não só deve ser menor que $n^{\log_b a}$, mas tem de ser *polinomialmente* menor. Ou seja, $f(n)$ deve ser assintoticamente menor que $n^{\log_b a}$ por um fator n^{ϵ} para alguma constante $\epsilon > 0$. No terceiro caso, $f(n)$ não apenas deve ser maior que $n^{\log_b a}$; ela tem de ser polinomialmente maior e, além disso, satisfazer à condição de "regularidade" de que $af(n/b) \leq cf(n)$. Essa condição é satisfeita pela maioria das funções polinomialmente limitadas que encontraremos.

É importante perceber que os três casos não abrangem todas as possibilidades para $f(n)$. Existe uma lacuna entre os casos 1 e 2 quando $f(n)$ é menor que $n^{\log_b a}$, mas não polinomialmente menor. De modo semelhante, há uma lacuna entre os casos 2 e 3 quando $f(n)$ é maior que $n^{\log_b a}$, mas não polinomialmente maior. Se a função $f(n)$ recair em uma dessas lacunas, ou se a condição de regularidade no caso 3 deixar de ser válida, o método mestre não poderá ser usado para resolver a recorrência.

Como usar o método mestre

Para usar o método mestre, simplesmente determinamos qual caso (se houver algum) do teorema mestre se aplica e anotamos a resposta.

Como primeiro exemplo, considere

$$T(n) = 9T(n/3) + n$$

Para essa recorrência, temos $a = 9, b = 3, f(n) = n$ e, portanto, temos $n^{\log_b a} = n^{\log_3 9} = \Theta(n^2)$. Como $f(n) = O(n^{\log_3 9 - \epsilon})$, onde $\epsilon = 1$, podemos aplicar o caso 1 do teorema mestre e concluir que a solução é $T(n) = \Theta(n^2)$.

Agora, considere

$$T(n) = T(2n/3) + 1,$$

na qual $a = 1, b = 3/2, f(n) = 1$ e $n^{\log_b a} = n^{\log_{3/2} 1} = n^0 = 1$. Aplica-se o caso 2, pois $f(n) = \Theta(n^{\log_b a}) = \Theta(1)$, e portanto a solução para a recorrência é $T(n) = \Theta(\lg n)$.

Para a recorrência

$$T(n) = 3T(n/4) + n \lg n,$$

temos $a = 3, b = 4, f(n) = n \lg n$ e $n^{\log_b a} = n^{\log_4 3} = O(n^{0,793})$. Como $f(n) = \Omega(n^{\log_4 3 + \epsilon})$, onde $\epsilon \approx 0{,}2$, aplica-se o caso 3 se podemos mostrar que a condição de regularidade é válida para $f(n)$. Para n suficientemente grande, $af(n/b) = 3(n/4) \lg(n/4) \leq (3/4)n \lg n = cf(n)$ para $c = 3/4$. Conseqüentemente, de acordo com o caso 3, a solução para a recorrência é $T(n) = \Theta(n \lg n)$.

O método mestre não se aplica à recorrência

$$T(n) = 2T(n/2) + n \lg n,$$

mesmo que ela tenha a forma apropriada: $a = 2, b = 2, f(n) = n \lg n$ e $n^{\log_b a} = n$. Parece que o caso 3 deve se aplicar, pois $f(n) = n \lg n$ é assintoticamente maior que $n^{\log_b a} = n$. O problema é que ela não é *polinomialmente* maior. A razão $f(n)/n^{\log_b a} = (n \lg n)/n = \lg n$ é assintoticamente menor que n^{ϵ} para qualquer constante positiva ϵ. Conseqüentemente, a recorrência recai na lacuna entre o caso 2 e o caso 3. (Veja no Exercício 4.4-2 uma solução.)

Exercícios

4.3-1

Use o método mestre para fornecer limites assintóticos restritos para as recorrências a seguir.

a. $T(n) = 4T(n/2) + n$.

b. $T(n) = 4T(n/2) + n^2$.

c. $T(n) = 4T(n/2) + n^3$.

4.3-2
O tempo de execução de um algoritmo A é descrito pela recorrência $T(n) = 7T(n/2) + n^2$. Um algoritmo concorrente A' tem um tempo de execução $T'(n) = aT'(n/4) + n^2$. Qual é o maior valor inteiro para a tal que A' seja assintoticamente mais rápido que A?

4.3-3
Use o método mestre para mostrar que a solução para a recorrência de pesquisa binária $T(n) = T(n/2) + \Theta(1)$ é $T(n) = \Theta(\lg n)$. (Veja no Exercício 2.3-5 uma descrição da pesquisa binária.)

4.3-4
O método mestre pode ser aplicado à recorrência $T(n) = 4T(n/2) + n^2 \lg n$? Por que ou por que não? Forneça um limite superior assintótico para essa recorrência.

4.3-5 ★
Considere a condição de regularidade $af(n/b) \leq cf(n)$ para alguma constante $c < 1$, que faz parte do caso 3 do teorema mestre. Dê um exemplo de uma função simples $f(n)$ que satisfaça a todas as condições no caso 3 do teorema mestre, exceto à condição de regularidade.

★ 4.4 Prova do teorema mestre

Esta seção contém uma prova do teorema mestre (Teorema 4.1). A prova não precisa ser entendida para se aplicar o teorema.

A prova tem duas partes. A primeira parte analisa a recorrência "mestre" (4.5), sob a hipótese simplificadora de que $T(n)$ é definida apenas sobre potências exatas de $b > 1$; ou seja, para $n = 1, b, b^2, \ldots$ Essa parte fornece toda a intuição necessária para se entender por que o teorema mestre é verdadeiro. A segunda parte mostra como a análise pode ser estendida a todos os inteiros positivos n e é apenas a técnica matemática aplicada ao problema do tratamento de pisos e tetos.

Nesta seção, algumas vezes abusaremos ligeiramente de nossa notação assintótica, usando-a para descrever o comportamento de funções que só são definidas sobre potências exatas de b. Lembre-se de que as definições de notações assintóticas exigem que os limites sejam provados para todos os números grandes o suficiente, não apenas para aqueles que são potências de b. Tendo em vista que poderíamos produzir novas notações assintóticas que se aplicassem ao conjunto $\{b^i : i = 0, 1, \ldots\}$ em lugar dos inteiros negativos, esse abuso é secundário.

Apesar disso, sempre deveremos nos resguardar quando estivermos usando a notação assintótica sobre um domínio limitado, a fim de não chegarmos a conclusões inadequadas. Por exemplo, provar que $T(n) = O(n)$ quando n é uma potência exata de 2 não garante que $T(n) = O(n)$. A função $T(n)$ poderia ser definida como

$$T(n) = \begin{cases} n & \text{se } n = 1, 2, 4, 8, \ldots, \\ n^2 & \text{em caso contrário,} \end{cases}$$

e, nesse caso, o melhor limite superior que pode ser provado é $T(n) = O(n^2)$. Devido a esse tipo de conseqüência drástica, nunca empregaremos a notação assintótica sobre um domínio limitado sem tornar absolutamente claro a partir do contexto que estamos fazendo isso.

4.4.1 A prova para potências exatas

A primeira parte da prova do teorema mestre analisa a recorrência (4.5),

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

para o método mestre, sob a hipótese de que n é uma potência exata de $b > 1$, onde b não precisa ser um inteiro. A análise está dividida em três lemas. O primeiro reduz o problema de resolver a recorrência mestre ao problema de avaliar uma expressão que contém um somatório. O segundo determina limites sobre esse somatório. O terceiro lema reúne os dois primeiros para provar uma versão do teorema mestre correspondente ao caso em que n é uma potência exata de b.

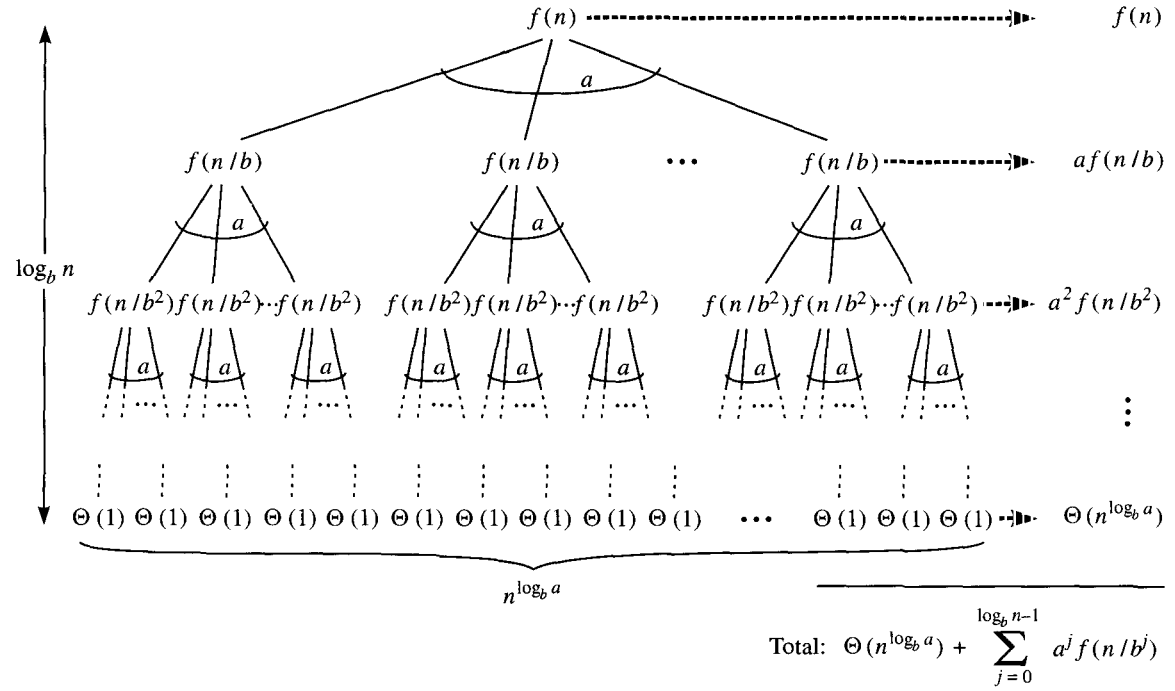

FIGURA 4.3 A árvore de recursão gerada por $T(n) = aT(n/b) + f(n)$. A árvore é uma árvore a-ária completa com $n^{\log_b a}$ folhas e altura $\log_b n$. O custo de cada nível é mostrado à direita, e sua soma é dada na equação (4.6)

Lema 4.2

Sejam $a \geq 1$ e $b > 1$ constantes, e seja $f(n)$ uma função não negativa definida sobre potências exatas de b. Defina $T(n)$ sobre potências exatas de b pela recorrência

$$T(n) = \begin{cases} \Theta(1) & \text{se } n = 1 , \\ aT(n/b) + f(n) & \text{se } n = b^i , \end{cases}$$

onde i é um inteiro positivo. Então

$$T(n) = \Theta(n^{\log_b a}) + \sum_{j=0}^{\log_b n - 1} a^j f(n/b^j) . \tag{4.6}$$

Prova Usamos a árvore de recursão da Figura 4.3. A raiz da árvore tem custo $f(n)$, e ela tem a filhas, cada uma com o custo $f(n/b)$. (É conveniente imaginar a como sendo um inteiro, especialmente ao se visualizar a árvore de recursão, mas a matemática não o exige.) Cada uma dessas filhas tem a filhas com o custo $f(n/b^2)$ e, desse modo, existem a^2 nós que estão à distância 2 da raiz. Em geral, há a^j nós à distância j da raiz, e cada um tem o custo $f(n/b^j)$. O custo de cada folha é $T(1) = \Theta(1)$, e cada folha está a uma distância $\log_b n$ da raiz, pois $n/b^{\log_b n} = 1$. Existem $a^{\log_b n} = n^{\log_b a}$ folhas na árvore.

Podemos obter a equação (4.6) fazendo o somatório dos custos de cada nível da árvore, como mostra a figura. O custo para um nível j de nós internos é $a^j f(n/b^j)$, e assim o total de todos os níveis de nós internos é

$$\sum_{j=0}^{\log_b n-1} a^j (n/b^j) \ .$$

No algoritmo básico de dividir e conquistar, essa soma representa os custos de dividir problemas em subproblemas, e depois combinar novamente os subproblemas. O custo de todas as folhas, que é o custo de efetuar todos os $n^{\log_b a}$ subproblemas de tamanho 1, é $\Theta(n^{\log_b a})$. ■

Em termos da árvore de recursão, os três casos do teorema mestre correspondem a casos nos quais o custo total da árvore é (1) dominado pelos custos nas folhas, (2) distribuído uniformemente entre os níveis da árvore ou (3) dominado pelo custo da raiz.

O somatório na equação (4.6) descreve o custo dos passos de divisão e combinação no algoritmo básico de dividir e conquistar. O lema seguinte fornece limites assintóticos sobre o crescimento do somatório.

Lema 4.3

Sejam $a \geq 1$ e $b > 1$ constantes, e seja $f(n)$ uma função não negativa definida sobre potências exatas de b. Uma função $g(n)$ definida sobre potências exatas de b por

$$g(n) = \sum_{j=0}^{\log_b n-1} a^j f(n/b^j) \tag{4.7}$$

pode então ser limitada assintoticamente para potências exatas de b como a seguir.

1. Se $f(n) = O\,O(n^{\log_b a-\epsilon})$ para alguma constante $\epsilon > 0$, então $g(n) = O(n^{\log_b a})$.
2. Se $f(n) = \Theta(n^{\log_b a})$, então $g(n) = \Theta(n^{\log_b a} \lg n)$.
3. Se $af(n/b) \leq cf(n)$ para alguma constante $c < 1$ e para todo $n \geq b$, então $g(n) = \Theta(f(n))$.

Prova Para o caso 1, temos $f(n) = O(n^{\log_b a-\epsilon})$, implicando que $f(n/b^j) = O((n/b^j)^{\log_b a-\epsilon})$. A substituição na equação (4.7) resulta em

$$g(n) = O\left(\sum_{j=0}^{\log_b n-1} a^j \left(\frac{n}{b^j}\right)^{\log_b a-\epsilon}\right). \tag{4.8}$$

Limitamos o somatório dentro da notação O pela fatoração dos termos e simplificação, o que resulta em uma série geométrica crescente:

$$\begin{aligned}
\sum_{j=0}^{\log_b n-1} a^j \left(\frac{n}{b^j}\right)^{\log_b a-\epsilon} &= n^{\log_b a-\epsilon} \sum_{j=0}^{\log_b n-1} \left(\frac{ab^\epsilon}{b^{\log_b a}}\right)^j \\
&= n^{\log_b a-\epsilon} \sum_{j=0}^{\log_b n-1} (b^\epsilon)^j \\
&= n^{\log_b a-\epsilon} \left(\frac{b^{\epsilon \log_b n}-1}{b^\epsilon-1}\right) \\
&= n^{\log_b a-\epsilon} \left(\frac{n^\epsilon-1}{n^\epsilon-1}\right).
\end{aligned}$$

Tendo em vista que b e ϵ são constantes, podemos reescrever a última expressão como $n^{\log_b a-\epsilon}O(n^{\epsilon}) = O(n^{\log_b a})$. Substituindo por essa expressão o somatório na equação (4.8), temos

$$g(n) = O(n^{\log_b a}),$$

e o caso 1 fica provado.

Sob a hipótese de que $f(n) = \Theta(n^{\log_b a})$ para o caso 2, temos que $f(n/b^j) = \Theta((n/b^j)^{\log_b a-\epsilon})$. A substituição na equação (4.7) produz

$$g(n) = \Theta\left(\sum_{j=0}^{\log_b n-1} a^j\left(\frac{n}{b^j}\right)^{\log_b a}\right). \tag{4.9}$$

Limitamos o somatório dentro de Θ como no caso 1 mas, dessa vez, não obtemos uma série geométrica. Em vez disso, descobrimos que todo termo do somatório é o mesmo:

$$\begin{aligned}\sum_{j=0}^{\log_b n-1} a^j\left(\frac{n}{b^j}\right)^{\log_b a} &= n^{\log_b a}\sum_{j=0}^{\log_b n-1}\left(\frac{a}{b^{\log_b a}}\right)^j \\ &= n^{\log_b a}\sum_{j=0}^{\log_b n-1} 1 \\ &= n^{\log_b a}\log_b n .\end{aligned}$$

Substituindo por essa expressão o somatório da equação (4.9), obtemos

$$\begin{aligned}g(n) &= \Theta(n^{\log_b a}\log_b n) \\ &= \Theta(n^{\log_b a}\lg n),\end{aligned}$$

e caso 2 fica provado.

O caso 3 é provado de forma semelhante. Como $f(n)$ aparece na definição (4.7) de $g(n)$ e todas os termos de $g(n)$ são não negativos, podemos concluir que $g(n) = \Omega(f(n))$ para potências exatas de b. Sob a hipótese de que $af(n/b) \le cf(n)$ para alguma constante $c < 1$ e todo $n \ge b$, temos $f(n/b) \le (c/a)f(n)$. Iteragindo j vezes, temos $f(n/b^j) \le (c/a)^j f(n)$ ou, de modo equivalente, $a^j f(n/b^j) \le c^j f(n)$. A substituição na equação (4.7) e a simplificação produzem uma série geométrica mas, diferente da série no caso 1, esta tem termos decrescentes:

$$\begin{aligned}g(n) &= \sum_{j=0}^{\log_b n-1} a^j f(n/b^j) \\ &\le \sum_{j=0}^{\log_b n-1} c^j f(n) \\ &\le f(n)\sum_{j=0}^{\infty} c^j \\ &= f(n)\left(\frac{1}{1-c}\right) \\ &= O(f(n)),\end{aligned}$$

pois c é constante. Desse modo, podemos concluir que $g(n) = \Theta(f(n))$ para potências exatas de b. O caso 3 fica provado, o que completa a prova do lema. ■

Agora podemos provar uma versão do teorema mestre para o caso em que n é uma potência exata de b.

Lema 4.4
Sejam $a \geq 1$ e $b > 1$ constantes, e seja $f(n)$ uma função não negativa definida sobre potências exatas de b. Defina $T(n)$ sobre potências exatas de b pela recorrência

$$T(n) = \begin{cases} \Theta(1) & \text{se } n = 1 , \\ aT(n/b) + f(n) & \text{se } n = b^i , \end{cases}$$

onde i é um inteiro positivo. Então, $T(n)$ pode ser limitado assintoticamente para potências exatas de b como a seguir.

1. Se $f(n) = O(n^{\log_b a - \epsilon})$ para alguma constante $\epsilon > 0$, então $T(n) = \Theta(n^{\log_b a})$.
2. Se $f(n) = \Theta(n^{\log_b a})$, então $T(n) = \Theta(n^{\log_b a} \lg n)$.
3. Se $f(n) = \Omega(n^{\log_b a + \epsilon})$ para alguma constante $\epsilon > 0$, e se $af(n/b) \leq cf(n)$ para alguma constante $c < 1$ e todo n suficientemente grande, então $T(n) = \Theta(f(n))$.

Prova Empregamos os limites do Lema 4.3 para avaliar o somatório (4.6) a partir do Lema 4.2. Para o caso 1, temos

$$\begin{aligned} T(n) &= \Theta(n^{\log_b a}) + O(n^{\log_b a}) \\ &= \Theta(n^{\log_b a}) , \end{aligned}$$

e, para o caso 2,

$$\begin{aligned} T(n) &= \Theta(n^{\log_b a}) + \Theta(n^{\log_b a} \lg n) \\ &= \Theta(n^{\log_b a} \lg n) . \end{aligned}$$

Para o caso 3,

$$\begin{aligned} T(n) &= \Theta(n^{\log_b a}) + \Theta(f(n)) \\ &= \Theta(f(n)) , \end{aligned}$$

porque $f(n) = \Omega(n^{\log_b a + \epsilon})$. ■

4.4.2 Pisos e tetos

Para completar a prova do teorema mestre, devemos agora estender nossa análise à situação em que pisos e tetos são usados na recorrência mestre, de forma que a recorrência seja definida para todos os inteiros, não apenas para potências exatas de b. Obter um limite inferior sobre

$$T(n) = aT(\lceil n/b \rceil) + f(n) \tag{4.10}$$

e um limite superior sobre

$$T(n) = aT(\lfloor n/b \rfloor) + f(n) \tag{4.11}$$

é rotina, pois o limite $\lceil n/b \rceil \geq n/b$ pode ser forçado no primeiro caso para produzir o resultado desejado, e o limite $\lfloor n/b \rfloor \leq n/b$ pode ser forçado no segundo caso. A imposição de um limite inferior sobre a recorrência (4.11) exige quase a mesma técnica que a imposição de um limite superior sobre a recorrência (4.10); assim, apresentaremos somente esse último limite.

Modificamos a árvore de recursão da Figura 4.3 para produzir a árvore de recursão da Figura 4.4. À medida que nos aprofundamos na árvore de recursão, obtemos uma seqüência de invocações recursivas sobre os argumentos

n ,

$\lceil n/b \rceil$,

$\lceil \lceil n/b \rceil /b \rceil$,

$\lceil \lceil \lceil n/b \rceil /b \rceil /b \rceil$,

$\vdots$

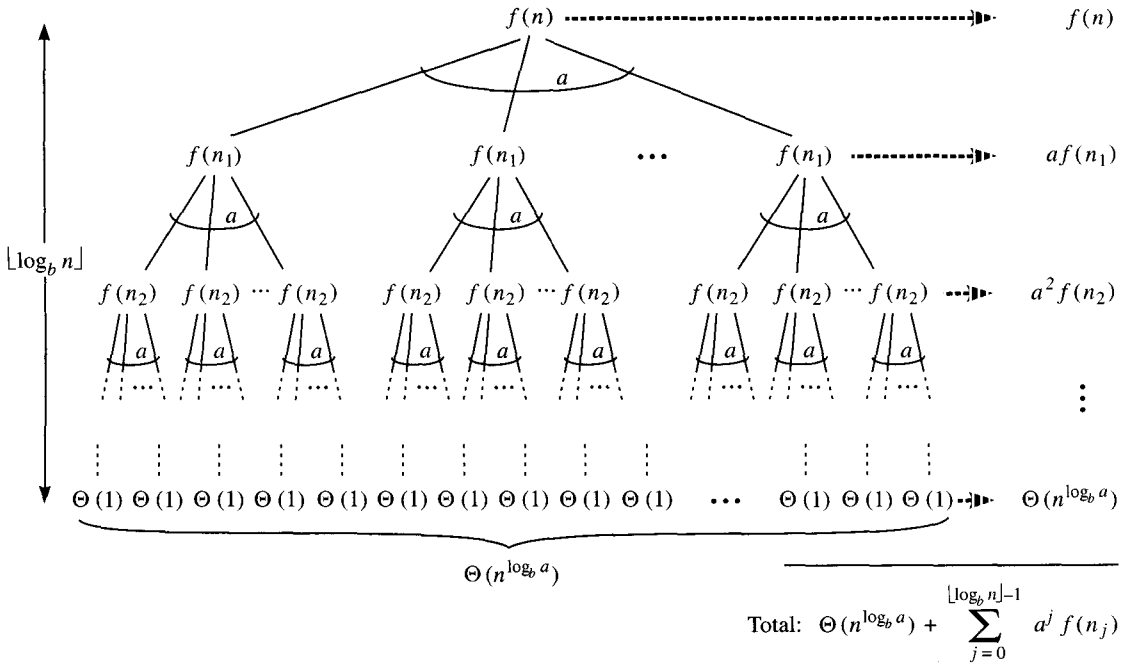

FIGURA 4.4 A árvore de recursão gerada por $T(n) = aT(\lceil n/b \rceil) + f(n)$. O argumento recursivo n_j é dado pela equação (4.12)

Vamos denotar o j-ésimo elemento na seqüência por n_j, onde

$$n_j = \begin{cases} n & \text{se } j = 0 , \\ \lceil n_{j-1}/b \rceil & \text{se } j > 0 . \end{cases} \tag{4.12}$$

Nossa primeira meta é determinar a profundidade k tal que n_k é uma constante. Usando a desigualdade $\lceil x \rceil \leq x + 1$, obtemos

$$\begin{aligned} n_0 &\leq n , \\ n_1 &\leq \frac{n}{b} + 1 , \\ n_2 &\leq \frac{n}{b^2} + \frac{1}{b} + 1 , \\ n_3 &\leq \frac{n}{b^3} + \frac{n}{b^2} , \frac{1}{b} + 1 , \\ &\vdots \end{aligned}$$

Em geral,

$$\begin{aligned} n_j &\le \frac{n}{b^j} + \sum_{i=0}^{j-1} \frac{1}{b^i} \\ &< \frac{n}{b^j} + \sum_{i=0}^{\infty} \frac{1}{b^i} \\ &= \frac{n}{b^j} + \frac{b}{b-1} . \end{aligned}$$

Fazendo $j = \lfloor \log_b n \rfloor$, obtemos

$$\begin{aligned} n_{\lfloor \log_b n \rfloor} &< \frac{n}{b^{\lceil \log_b n \rceil}} + \frac{b}{b-1} \\ &\le \frac{n}{b^{\log_b n - 1}} + \frac{b}{b-1} \\ &= \frac{n}{n/b} + \frac{b}{b-1} \\ &= b + \frac{b}{b-1} \\ &= O(1) , \end{aligned}$$

e desse modo vemos que, à profundidade $\lfloor \log_b n \rfloor$, o tamanho do problema é no máximo uma constante.

Da Figura 4.4, observamos que

$$T(n) = \Theta(n^{\log_b a}) + \sum_{j=0}^{\lfloor \log_b n \rfloor - 1} a^j f(n_j) , \tag{4.13}$$

que é quase igual à equação (4.6), a não ser pelo fato de n ser um inteiro arbitrário e não se restringir a ser uma potência exata de b.

Agora podemos avaliar o somatório

$$g(n) = \sum_{j=0}^{\lfloor \log_b n \rfloor - 1} a^j f(n_j) \tag{4.14}$$

a partir de (4.13) de maneira análoga à prova do Lema 4.3. Começando com o caso 3, se $af(\lceil n/b \rceil) \le cf(n)$ para $n > b + b/(b-1)$, onde $c < 1$ é uma constante, segue-se que $a^j f(n_j) \le c^j f(n)$. Portanto, a soma na equação (4.14) pode ser avaliada da mesma maneira que no Lema 4.3. Para o caso 2, temos $f(n) = \Theta(n^{\log_b a})$. Se pudermos mostrar que $f(n_j) = O(n^{\log_b a} / a^j) = O((n/b^j)^{\log_b a})$, então a prova para caso 2 do Lema 4.3 passará. Observe que $j \le \lfloor \log_b n \rfloor$ implica $b^j/n \le 1$. O limite $f(n) = O(n^{\log_b a})$ implica que existe uma constante $c > 0$ tal que, para n_j suficientemente grande,

$$\begin{aligned} f(n_j) &\le c\left(\frac{n}{b^j} + \frac{b}{b-1} \right)^{\log_b a} \\ &= c\left(\frac{n}{b^j} \left(1 + \frac{b^j}{n} \cdot \frac{b}{b-1} \right) \right)^{\log_b a} \end{aligned}$$

$$= c\left(\frac{n^{\log_b a}}{a^j}\right)\left(1+\left(\frac{b^j}{n}\cdot\frac{b}{b-1}\right)\right)^{\log_b a}$$

$$\le c\left(\frac{n^{\log_b a}}{a^j}\right)\left(1+\frac{b}{b-1}\right)^{\log_b a}$$

$$= O\left(\frac{n^{\log_b a}}{a^j}\right).$$

pois $c(1+b/(b-1))^{\log_b a}$ é uma constante. Portanto, o caso 2 fica provado. A prova do caso 1 é quase idêntica. A chave é provar o limite $f(n_j) = O(n^{\log_b a-\epsilon})$, semelhante à prova correspondente do caso 2, embora a álgebra seja mais complicada.

Agora provamos os limites superiores no teorema mestre para todos os inteiros n. A prova dos limites inferiores é semelhante.

Exercícios

4.4-1 ⋆

Forneça uma expressão simples e exata para n_i na equação (4.12) para o caso em que b é um inteiro positivo, em vez de um número real arbitrário.

4.4-2 ⋆

Mostre que, se $f(n) = \Theta(n^{\log_b a} \lg^k n)$, onde $k \ge 0$, a recorrência mestre tem a solução $T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$. Por simplicidade, restrinja sua análise a potências exatas de b.

4.4-3 ⋆

Mostre que o caso 3 do teorema mestre é exagerado, no sentido de que a condição de regularidade $af(n/b) \le cf(n)$ para alguma constante $c < 1$ implica que existe uma constante $\epsilon > 0$ tal que $f(n) = \Omega(n^{\log_b a+\epsilon})$.

Problemas

4-1 Exemplos de recorrência

Forneça limites assintóticos superiores e inferiores para $T(n)$ em cada uma das recorrências a seguir. Suponha que $T(n)$ seja constante para $n \le 2$. Torne seus limites tão restritos quanto possível e justifique suas respostas.

a. $T(n) = 2T(n/2) + n^3$.

b. $T(n) = T(9n/10) + n$.

c. $T(n) = 16T(n/4) + n^2$.

d. $T(n) = 7T(n/3) + n^2$.

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

f. $T(n) = 2T(n/4) + \sqrt{n}$.

g. $T(n) = T(n-1) + n$.

h. $T(n) = T(\sqrt{n}) + 1$.

4-2 Como localizar o inteiro perdido

Um arranjo $A[1..n]$ contém todos os inteiros de 0 a n, exceto um. Seria fácil descobrir o inteiro que falta no tempo $O(n)$, usando um arranjo auxiliar $B[0..n]$ para registrar os números que aparecem em A. Porém, neste problema, não podemos ter acesso a todo um inteiro em A com uma única operação. Os elementos de A são representados em binário, e a única operação que pode-

mos usar para obter acesso a eles é "buscar o j-ésimo bit de $A[i]$", que demora um tempo constante.

Mostre que, se usarmos apenas essa operação, ainda poderemos descobrir o inteiro que falta no tempo $O(n)$.

4-3 Custos de passagem de parâmetros

Em todo este livro, partimos da hipótese de que a passagem de parâmetros durante as chamadas de procedimentos demora um tempo constante, mesmo que um arranjo de N elementos esteja sendo passado. Essa hipótese é válida na maioria dos sistemas, porque é passado um ponteiro para o arranjo, e não o próprio arranjo. Este problema examina as implicações de três estratégias de passagem de parâmetros:

1. Um arranjo é passado por ponteiro. Tempo = $\Theta(1)$.
2. Um arranjo é passado por cópia. Tempo = $\Theta(N)$, onde N é o tamanho do arranjo.
3. Um arranjo é passado por cópia somente do subintervalo ao qual o procedimento chamado poderia ter acesso. Tempo = $\Theta(p - q + 1)$ se o subarranjo $A[p \,..\, q]$ for passado.

a. Considere o algoritmo de pesquisa binária recursiva para localizar um número em um arranjo ordenado (ver Exercício 2.3-5). Forneça recorrências para os tempos de execução do pior caso de pesquisa binária quando os arranjos são passados com o uso de cada um dos três métodos anteriores, e forneça bons limites superiores nas soluções das recorrências. Seja N o tamanho do problema original e n o tamanho de um subproblema.

b. Repita a parte (a) para o algoritmo MERGE-SORT da Seção 2.3.1.

4-4 *Outros exemplos de recorrência*

Forneça limites assintóticos superiores e inferiores para $T(n)$ em cada uma das recorrências a seguir. Suponha que $T(n)$ seja constante para n suficientemente pequeno. Torne seus limites tão restritos quanto possível e justifique suas respostas.

a. $T(n) = 3T(n/2) + n \lg n$.

b. $T(n) = 5T(n/5) + n/\lg n$.

c. $T(n) = 4T(n/2) + n^2\sqrt{n}$.

d. $T(n) = 3T(n/3 + 5) + n/2$.

e. $T(n) = 2T(n/2) + n/\lg n$.

f. $T(n) = T(n/2) + T(n/4) + T(n/8) + n$.

g. $T(n) = T(n - 1) + 1/n$.

h. $T(n) = T(n - 1) + \lg n$.

i. $T(n) = T(n - 2) + 2 \lg n$.

j. $T(n) = \sqrt{n}T(\sqrt{n}) + n$.

4-5 Números de Fibonacci

Este problema desenvolve propriedades dos números de Fibonacci, que são definidos pela recorrência (3.21). Usaremos a técnica de gerar funções para resolver a recorrência de Fibonacci. Defina a ***função geradora*** (ou ***série de potências formais***) $\mathcal{F}$ como]

$$\mathcal{F}(z) = \sum_{i=0}^{\infty} F_i z^i$$

$$= 0 + z + z^2 + 2z^3 + 3z^4 + 5z^5 + 8z^6 + 13z^7 + 21z^8 + \ldots,$$

onde F_i é o i-ésimo número de Fibonacci.

a. Mostre que $\mathcal{F}(z) = z + z\mathcal{F}(z) + z^2\mathcal{F}(z)$..

b. Mostre que

$$\mathcal{F}(z) = \frac{z}{1 - z - z^2}$$

$$= \frac{z}{(1 - \phi z)(1 - \hat{\phi} z)}$$

$$= \frac{1}{\sqrt{5}}\left(\frac{1}{1 - \phi z} - \frac{1}{1 - \hat{\phi} z}\right),$$

onde

$$\phi = \frac{1 + \sqrt{5}}{2} = 1{,}61803\ldots$$

e

$$\hat{\phi} = \frac{1 - \sqrt{5}}{2} = -0{,}61803\ldots .$$

c. Mostre que

$$\mathcal{F}(z) = \sum_{i=0}^{\infty} \frac{1}{\sqrt{5}}(\phi^i - \hat{\phi}^i) z^i .$$

d. Prove que $F_i = \phi^i / \sqrt{5}$ para $i > 0$, arredondado para o inteiro mais próximo. (*Sugestão:* Observe que $|\hat{\phi}| < 1$.)

e. Prove que $F_{i+2} \geq \phi^i$ para $i \geq 0$.

4-6 Testes de chips VLSI

O professor Diógenes tem n chips VLSI[1] supostamente idênticos que, em princípio, são capazes de testar uns aos outros. O aparelho de teste do professor acomoda dois chips de cada vez. Quando o aparelho está carregado, cada chip testa o outro e informa se ele está bom ou ruim. Um chip bom sempre informa com precisão se o outro chip está bom ou ruim, mas a resposta de um chip ruim não é confiável. Portanto, os quatro resultados possíveis de um teste são:

Chip *A* informa	Chip *B* informa	Conclusão
B está bom	*A* está bom	Ambos estão bons ou ambos ruins
B está bom	*A* está ruim	Pelo menos um está ruim
B está ruim	*A* está bom	Pelo menos um está ruim
B está ruim	*A* está ruim	Pelo menos um está ruim

[1]VLSI significa *very large scale integration*, ou integração em escala muito grande, que é a tecnologia de chips de circuitos integrados usada para fabricar a maioria dos microprocessadores de hoje.

a. Mostre que, se mais de $n/2$ chips estão ruins, o professor não pode necessariamente descobrir quais chips estão bons usando qualquer estratégia baseada nessa espécie de teste aos pares. Suponha que os chips ruins possam conspirar para enganar o professor.

b. Considere o problema de descobrir um único chip bom entre n chips, supondo que mais de $n/2$ dos chips estejam bons. Mostre que $\lfloor n/2 \rfloor$ testes de pares sejam suficientes para reduzir o problema a outro com praticamente metade do tamanho.

c. Mostre que os chips bons podem ser identificados com $\Theta(n)$ testes de pares, supondo-se que mais de $n/2$ dos chips estejam bons. Forneça e resolva a recorrência que descreve o número de testes.

4-7 Arranjos de Monge

Um arranjo $m \times n$ A de números reais é um ***arranjo de Monge*** se, para todo i, j, k e l tais que $1 \leq i < k \leq m$ e $1 \leq j < l \leq n$, temos

$$A[i, j] + A[k, l] \leq A[i, l] + A[k, j].$$

Em outras palavras, sempre que escolhemos duas linhas e duas colunas de um arranjo de Monge e consideramos os quatro elementos nas interseções das linhas e das colunas, a soma dos elementos superior esquerdo e inferior direito é menor ou igual à soma dos elementos inferior esquerdo e superior direito. Por exemplo, o arranjo a seguir é um arranjo de Monge:

10 17 13 28 23
17 22 16 29 23
24 28 22 34 24
11 13 6 17 7
45 44 32 37 23
36 33 19 21 6
75 66 51 53 34

a. Prove que um arranjo é de Monge se e somente se para todo $i = 1, 2, \ldots, m - 1$ e $j = 1, 2, \ldots, n - 1$, temos

$$A[i, j] + A[i + 1, j + 1] \leq A[i, j + 1] + A[i + 1, j].$$

(*Sugestão:* Para a parte "somente se", use a indução separadamente sobre linhas e colunas.)

b. O arranjo a seguir não é de Monge. Troque um elemento para transformá-lo em um arranjo de Monge. (*Sugestão:* Use a parte (a).)

37 23 22 32
21 6 7 10
53 34 30 31
32 13 9 6
43 21 15 8

c. Seja $f(i)$ o índice da coluna que contém o elemento mínimo mais à esquerda da linha i. Prove que $f(1) \leq f(2) \leq \ldots \leq f(m)$ para qualquer arranjo de Monge $m \times n$.

d. Aqui está uma descrição de um algoritmo de dividir e conquistar que calcula o elemento mínimo mais à esquerda em cada linha de um arranjo de Monge $m \times n$ A:

> Construa uma submatriz A' de A consistindo nas linhas de numeração par de A. Determine recursivamente o mínimo mais à esquerda para cada linha de A'. Em seguida, calcule o mínimo mais à esquerda nas linhas de numeração ímpar de A.

Explique como calcular o mínimo mais à esquerda nas linhas de numeração ímpar de A (dado que o mínimo mais à esquerda das linhas de numeração seja conhecido) no tempo $O(m + n)$.

e. Escreva a recorrência que descreve o tempo de execução do algoritmo descrito na parte (d). Mostre que sua solução é $O(m + n \log m)$.

Notas do capítulo

As recorrências foram estudadas desde 1202 aproximadamente, por L. Fibonacci, e em sua homenagem os números de Fibonacci receberam essa denominação. A. De Moivre introduziu o método de funções geradoras (ver Problema 4-5) para resolver recorrências. O método mestre foi adaptado de Bentley, Haken e Saxe [41], que fornecem o método estendido justificado pelo Exercício 4.4-2. Knuth [182] e Liu [205] mostram como resolver recorrências lineares usando o método de funções geradoras. Purdom e Brown [252] e Graham, Knuth e Patashnik [132] contêm discussões extensas da resolução de recorrências.

Vários pesquisadores, inclusive Akra e Bazzi [13], Roura [262] e Verma [306], forneceram métodos para resolução de recorrências de dividir e conquistar mais gerais do que as que são resolvidas pelo método mestre. Descrevemos aqui o resultado de Akra e Bazzi, que funciona para recorrências da forma

$$T(n) = \sum_{i=1}^{k} a_i T(\lfloor n/b_i \rfloor) + f(n) , \tag{4.15}$$

onde $k \geq 1$; todos os coeficientes a_i são positivos e somam pelo menos 1; todos os valores b_i são maiores que 1; $f(n)$ é limitada, positiva e não decrescente; e, para todas as constantes $c > 1$, existem constantes $n_0, d > 0$ tais que $f(n/c) \geq df(n)$ para todo $n \geq n_0$. Esse método funcionaria sobre uma recorrência como $T(n) = T(\lfloor n/3 \rfloor) + T(\lfloor 2n/3 \rfloor) + O(n)$, para a qual o método mestre não se aplica. Para resolver a recorrência (4.15), primeiro encontramos o valor de p tal que $\sum_{i=1}^{k} a_i b_i^{-p} = 1$. (Tal p sempre existe, ele é único e positivo.) A solução para a recorrência é então

$$T(n) = \Theta(n^p) + \Theta\left(n^p \int_{n'}^{n} \frac{f(x)}{x^{p+1}} dx \right) ,$$

para uma constante n' suficientemente grande. O método de Akra-Bazzi pode ser um tanto difícil de usar, mas ele serve para resolver recorrências que modelam a divisão do problema em subproblemas de tamanhos substancialmente desiguais. O método mestre é mais simples para usar, mas só se aplica quando os tamanhos dos subproblemas são iguais.