3 CRESCIMENTO DE FUNÇÕES

A ordem de crescimento do tempo de execução de um algoritmo, definida no Capítulo 2, dá uma caracterização simples da eficiência do algoritmo e também nos permite comparar o desempenho relativo de algoritmos alternativos. Tão logo o tamanho da entrada n se torne suficientemente grande, 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, o que ganhamos em precisão em geral não vale o esforço do cálculo. Para entradas suficientemente grandes, as constantes multiplicativas e os termos de ordem mais baixa de um tempo de execução exato são dominados pelos efeitos do próprio tamanho da entrada. Quando observamos tamanhos de entrada suficientemente grandes para tornar relevante apenas a ordem de crescimento do tempo de execução, estamos estudando a eficiência ***assintótica*** dos algoritmos. Isto é, estamos preocupados com o modo como o tempo de execução de um algoritmo aumenta com o tamanho da entrada *no limite*, à medida que o tamanho da entrada aumenta 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 padrões 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. Então, apresentaremos várias convenções de notação usadas em todo este livro e, por fim, 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 assintótico de um algoritmo são definidas em termos de funções cujos domínios são o conjunto dos números naturais $= \{0, 1, 2, ...\}$. Tais notações são convenientes para descrever a função $T(n)$ do tempo de execução do pior caso, que em geral é definida somente para tamanhos de entrada inteiros. Contudo, às vezes, consideramos que é conveniente *abusar* da notação assintótica de vários modos. Por exemplo, poderíamos estender a notação ao domínio dos números reais ou, como alternativa, restringi-la a um subconjunto dos números naturais. Porém, é importante entender o significado preciso da notação para que, quando abusarmos, ela não seja *mal utilizada*. Esta seção define as notações assintóticas básicas e também apresenta alguns abusos comuns.

Notação assintótica, funções e tempos de execução

Usaremos a notação assintótica primariamente para descrever o tempo de execução de algoritmos, como fizemos quando escrevemos que o tempo de execução do pior caso para a ordenação por inserção é $\Theta(n_2)$. Todavia, na realidade a notação assintótica aplica-se a funções. Lembre-se de que caracterizamos o tempo de execução do pior caso da ordenação por inserção como $an_2 + bn + c$, para algumas constantes a, b e c. Quando afirmamos que o tempo de execução da ordenação por inserção é $\Theta(n_2)$, abstraímos alguns detalhes dessa função. Como a notação assintótica aplica-se a funções, o que quisemos dizer é que $\Theta(n_2)$ era a função $an_2 + bn + c$ que, aqui, por acaso caracteriza o tempo de execução do pior caso da ordenação por inserção. Neste livro, as funções às quais aplicamos a notação assintótica, normalmente caracterizarão os tempos de execução de algoritmos. Porém, a notação assintótica pode se aplicar a funções que caracterizam algum outro aspecto dos algoritmos (a quantidade de espaço que eles usam, por exemplo) ou até mesmo a funções que absolutamente nada têm a ver com algoritmos.

Mesmo quando utilizamos a notação assintótica para o tempo de execução de um algoritmo, precisamos entender *a qual* tempo de execução estamos nos referindo. Às vezes, estamos interessados no tempo de execução do pior caso. Porém, frequentemente queremos caracterizar o tempo de execução, seja qual for a entrada. Em outras palavras, muitas vezes desejamos propor um enunciado abrangente que se aplique a todas as entradas, e não apenas ao pior caso. Veremos que as notações assintóticas prestam-se bem à caracterização de tempos de execução, não importando qual seja a entrada.

Notação Θ

No Capítulo 2, vimos 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*

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

Uma função $f(n)$ pertence ao conjunto $\Theta(g(n))$ se existirem constantes positivas c_1 e c_2 tais que ela possa ser "encaixada" 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 parecer confuso, mas veremos mais adiante nesta seção que ele tem suas vantagens.

A Figura 3.1(a) apresenta um quadro intuitivo de funções $f(n)$ e $g(n)$, onde $f(n) = \Theta(g(n))$. Para todos os valores de n em n_0 ou à direita de n_0, o valor de $f(n)$ encontra-se em $c_1\, g(n)$ ou acima dele e em $c_2\, g(n)$ ou abaixo desse valor. Em outras palavras, para todo $n \geq n_0$, 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 negativa sempre que n for suficientemente grande. (Uma função ***assintoticamente positiva*** é uma função positiva para todo n suficientemente grande.) Por consequência, a própria função $g(n)$ deve ser assintoticamente não negativa, senão o conjunto $\Theta(g(n))$ é vazio. Por isso, consideraremos 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 ordem mais baixa e ignorar o coeficiente inicial do termo de ordem mais alta. 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 , c e n tais que

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

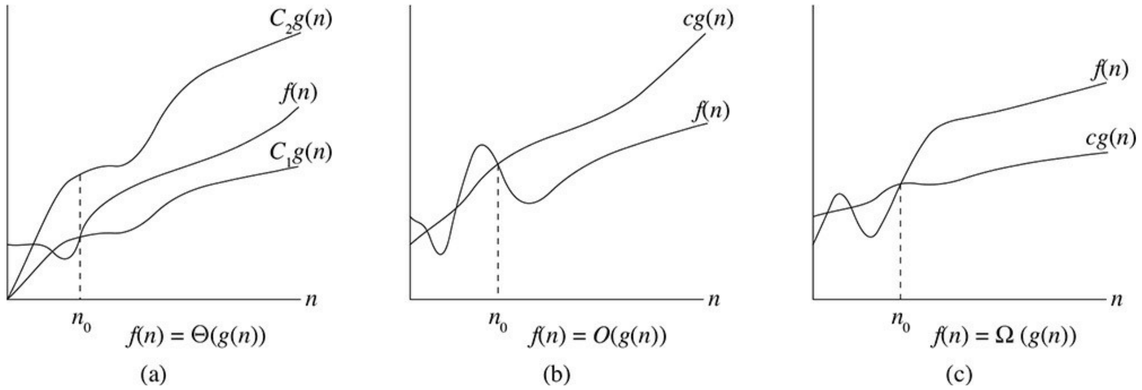

Figura 3.1 Exemplos gráficos das notações Θ , O e Ω . Em cada parte, o valor de n_0 mostrado é o mínimo valor possível; qualquer valor maior também funcionaria. **(a)** A notação Θ limita uma função entre fatores constantes. Escrevemos $f(n) = \Theta(g(n))$ se existirem constantes positivas $n_{0,}$ c_1 e c_2 tais que, em n_0 e à direita de n_0 , o valor de $f(n)$ sempre encontrar-se entre $c_1\ g(n)$ e $c_2\ g(n)$ inclusive. **(b)** A notação O dá um limite superior para uma função dentro de um fator constante. Escrevemos $f(n) = O(g(n))$ se existirem constantes positivas n_0 e c tais que, em n_0 e à direita de n_0 , o valor de $f(n)$ sempre estiver 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 existirem constantes positivas n_0 e c tais que, em n_0 e à direita de n_0 , o valor de $f(n)$ sempre estiver acima de $cg(n)$.

para todo $n \geq n_0$. Dividindo por n^2 temos

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

A desigualdade do lado direito pode ser considerada válida para qualquer valor de $n \geq 1$, se escolhermos qualquer constante $c_2 \geq 1/2$. Do mesmo modo, a desigualdade da esquerda pode ser considerada válida para qualquer valor de $n \geq 7$, se escolhermos qualquer constante $c_1 \leq 1/14$. Assim, escolhendo $c_1 = 1/14$, $c_2 = 1/2$ e $n_0 = 7$, podemos verificar que $\frac{1}{2}$ $n2 - 3n = \Theta(n2\ )$. Certamente, existem outras opções para as constantes, mas o importante é que existe *alguma* opção. Observe que essas constantes dependem da função $\frac{1}{2}$ $n2\ -3n$; uma função diferente pertencente a $\Theta(n_2)$ normalmente exigiria constantes diferentes.

Também podemos usar a definição formal para verificar que $6n_3 \neq \Theta(n_2)$. A título de contradição, suponha que existam c_2 e n_0 tais que $6n^3 \leq c_2n^2$ para todo $n \geq n_0$. Mas, então, divisão por n^2 dá $n \leq c_2/6$, o que não pode ser válido para um valor de n arbitrariamente grande, já que c_2 é constante.

Intuitivamente, os termos de ordem mais baixa de uma função assintoticamente positiva podem ser ignorados na determinação de limites assintoticamente restritos porque eles são insignificantes para grandes valores de n. Quando n é grande, até uma minúscula fração do termo de ordem mais alta é suficiente para dominar os termos de ordem mais baixa. Desse modo, definir c_1 como um valor ligeiramente menor que o coeficiente do termo de ordem mais alta e definir c_2 como um valor ligeiramente maior permite que as desigualdades na definição da notação Θ sejam satisfeitas. Da mesma maneira, o coeficiente do termo de ordem mais alta pode ser ignorado, já que ele só muda c_1 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 ordem mais baixa 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 $n_0 = 2 \cdot \max(|b|/a, \sqrt{|c|/a})$. . O leitor poderá verificar que $0 \le c_1n^2 \le an^2 + bn + c \le c_2n^2$ para todo $n \ge n_0$. 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 $p(n) = \Theta(n^d)$ (veja 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, esta última notação é um pequeno abuso porque a expressão não indica qual variável está tendendo a infinito.[2] Usaremos com frequência a notação $\Theta(1)$ para indicar uma constante ou uma função constante em relação a alguma variável.

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 "*Ó* grande de *g* de *n*" ou, às vezes, apenas "*ó* 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 a 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 em n_0 ou à direita de n_0, o valor da função $f(n)$ está abaixo de $cg(n)$.

Escrevemos $f(n) = O(g(n))$ para indicar que uma função $f(n)$ é um membro do conjunto $O(g(n))$. Observe que $f(n) = \Theta(g(n))$ implica $f(n) = O(g(n))$, já que a notação Θ é uma noção mais forte que a notação O. Em termos da teoria de conjuntos, escrevemos $\Theta(g(n)) \subseteq O(g(n))$. Assim, 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 desse tipo está em $O(n_2)$. O que pode ser mais surpreendente é que, quando $a > 0$, qualquer função *linear* $an + b$ está em $O(n_2)$, o que é facilmente verificado fazendo $c = a + |b|$ e $n_0 = \max(1, -b/a)$.

Se você já viu a notação O antes, poderá achar estranho que escrevamos, por exemplo, $n = O(n_2)$. Na literatura, verificamos que, às vezes, a notação O é utilizada informalmente para descrever limites assintoticamente justos, isto é, aquilo que definimos usando a notação Θ. Contudo, neste livro, quando escrevermos $f(n) = O(g(n))$, estaremos simplesmente afirmando que algum múltiplo constante de $g(n)$ é um limite assintótico superior para $f(n)$, sem qualquer menção de precisão. A distinção entre limites assintóticos superiores e limites assintoticamente justos é padrão na literatura de algoritmos.

Usando a notação O, podemos descrever frequentemente 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 $O(n_2)$ para o tempo de execução do pior caso: o custo de cada iteração do loop interno é limitado na parte superior por $O(1)$ (constante), os índices i e j são no máximo n, e o loop interno é executado no máximo uma vez para cada um dos n_2 pares de valores para 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 para o tempo de execução do algoritmo em cada entrada — o enunciado abrangente do qual falamos anteriormente. Desse modo, o limite $O(n_2)$ para o tempo de execução do pior caso da ordenação por inserção também se aplica a seu tempo de execução para toda entrada. Porém, o limite $\Theta(n_2)$ para o tempo de execução do pior caso da ordenação por inserção não implica um limite $\Theta(n_2)$ para o 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 $O(n_2)$, visto que, 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 qual entrada específica de tamanho n seja escolhida, o tempo de execução para 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* para uma função, a notação Ω nos dá um ***limite assintótico inferior***. Para uma determinada função $g(n)$, denotamos por $\Omega(g(n))$ (lê-se "ômega grande 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 Figura 3.1(c) mostra a intuição por trás da notação Ω. Para todos os valores n em n_0 ou à direita de n_0, o valor de $f(n)$ encontra-se em $g(n)$ ou acima de $g(n)$.

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

Teorema 3.1

Para quaisquer duas funções $f(n)$ 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 vez de usar o Teorema 3.1 para obter limites assintóticos superiores e inferiores a partir de limites assintoticamente precisos, como fizemos nesse exemplo, nós o utilizamos normalmente para demonstrar limites assintoticamente precisos a partir de limites assintóticos superiores e inferiores.

Quando dizemos que o *tempo de execução* (sem modificador) de um algoritmo é $\Omega(g(n))$, queremos dizer que, *não importando qual entrada específica de tamanho n seja escolhida para cada valor de n*, o tempo de execução para essa entrada é no mínimo uma constante vezes $g(n)$, para n suficientemente grande. De modo equivalente, estamos dando um limite inferior para o tempo de execução do melhor caso de um algoritmo. Por exemplo, o tempo de execução para o melhor caso da ordenação por inserção é $\Omega(n)$, o que implica que o tempo de execução da ordenação por inserção é $\Omega(n)$.

Portanto, o tempo de execução da ordenação por inserção pertence às funções $\Omega(n)$ e $O(n_2)$, já que ele se encontra em qualquer lugar entre uma função linear de n e uma função quadrática de n. Além disso, esses limites são tão justos assintoticamente quanto possível: por exemplo, o tempo de execução da ordenação por inserção não é $\Omega(n_2)$, visto que existe uma entrada para a qual a ordenação por inserção é executada no tempo $\Theta(n)$ (por exemplo, quando a entrada já está ordenada). Entretanto, não é contraditório dizer que o tempo de execução do *pior caso* da ordenação por inserção é $\Omega(n_2)$, visto que existe uma entrada que faz o algoritmo demorar o tempo $\Omega(n_2)$.

Notação assintótica em equações e desigualdades

Já vimos como a notação assintótica pode ser usada dentro de fórmulas matemáticas. Por exemplo, quando apresentamos a 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 (isto é, não está dentro de uma fórmula maior) no lado direito de uma equação (ou desigualdade), como em $n = O(n_2)$, já definimos que o sinal de igualdade significa pertinência a um conjunto: $n \in O(n_2)$. Porém, em geral, quando a notação assintótica aparece em uma fórmula, interpretamos que ela representa alguma função anônima que não nos preocupamos 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 vale $f(n) = 3n + 1$, que de fato está em $\Theta(n)$.

Utilizar a notação assintótica dessa maneira pode ajudar a eliminar detalhes não essenciais e confusos em uma equação. Por exemplo, no Capítulo 2 expressamos 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 há sentido em especificar exatamente todos os termos de ordem mais baixa; entendemos que todos eles estão incluídos na função anônima denotada pelo termo $\Theta(n)$.

Entendemos também que o número de funções anônimas em uma expressão é igual ao número de vezes que a notação assintótica aparece. Por exemplo, na expressão

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

há apenas uma função anônima (uma função de i). Portanto, essa expressão *não* é o mesmo que $O(1) + O(2) + ... + O(n)$ 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*. Assim, nosso exemplo significa 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.

Podemos encadear várias dessas relações, como em

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

Podemos interpretar cada equação separadamente pelas regras citadas. 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 dá intuitivamente.

Notação *O*

O limite assintótico superior fornecido pela notação O pode ser ou não assintoticamente justo. O limite $2n_2 = O(n_2)$ é assintoticamente justo, mas o limite $2n = O(n_2)$ não é. Usamos a notação o para denotar um limite superior que não é assintoticamente justo. Definimos formalmente $o(g(n))$ (lê-se "*ó* pequeno de g de n") como o conjunto

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

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 $f(n)$ torna-se 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 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 preciso. Um modo de defini-lo 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 pequeno de g de n") como o conjunto $\omega\,(\,g(n)) = \{\,f(n)$: para qualquer constante positiva $c > 0$, existe uma constante

$$n_0 > 0 \text{ tal que } 0 \leq cg(n) < f(n) \text{ para todo } n \geq n_0\}.$$

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 existir. 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 às comparações assintóticas. No caso das propriedades seguintes, considere que $f(n)$ e $g(n)$ são 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	$f(n) = o(h(n))$,
$f(n) = (\,g(n))$	e	$g(n) = (h(n))$	implicam	$f(n) = (h(n))$.

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 $g(n) = \omega(f(n))$.

Como essas propriedades se mantêm válidas para notações assintóticas, podemos traçar uma analogia entre a comparação assintótica de duas funções f e g e a comparação de dois números reais a e b:

$f(n) = O(g(n))$	é como	$a \leq b$,
$f(n) = \Omega(g(n))$	é como	$a \geq b$,
$f(n) = \Theta(g(n))$	é como	$a = b$,
$f(n) = o(g(n))$	é como	$a < b$,
$f(n) = (g(n))$	é como	$a > b$.

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 das propriedades de números reais não é transportada para a notação assintótica:

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

Embora quaisquer dois números reais possam ser comparados, nem todas as funções são assintoticamente comparáveis. Isto é, para duas funções $f(n)$ e $g(n)$, pode acontecer que nem $f(n) = O(g(n))$ nem $f(n) = \Omega(g(n))$ sejam válidas. Por exemplo, não podemos comparar as funções n e $n_1 + {}^{sen\ n}$ utilizando a notação assintótica, visto que o valor do expoente em $n_1 + {}^{sen\ n}$ 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 $O(n_2)$" não tem sentido.

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

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

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

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 padrões e explora as relações entre elas. A seção também ilustra o uso das notações assintóticas.

Monotonicidade

Uma função $f(n)$ é ***monotonicamente crescente*** se $m \le n$ implica $f(m) \le f(n)$. De modo semelhante, ela é ***monotonicamente 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 ou igual a x por x (lê-se "o piso de x") e o menor inteiro maior ou igual a x por x (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,
$n/2 + n/2 = n$
e, para qualquer número real $n \ge 0$ e inteiros a, $b > 0$,

$$\left\lceil \frac{\lceil x/a \rceil}{b} \right\rceil = \left\lceil \frac{x}{ab} \right\rceil, \tag{3.4}$$

$$\left\lfloor \frac{\lfloor x/a \rfloor}{b} \right\rfloor = \left\lfloor \frac{x}{ab} \right\rfloor, \tag{3.5}$$

$$\left\lceil \frac{a}{b} \right\rceil \le \frac{a + (b-1)}{b}, \tag{3.6}$$

$$\left\lfloor \frac{a}{b} \right\rfloor \ge \frac{a - (b-1)}{b}. \tag{3.7}$$

A função piso $f(x) = x$ é monotonicamente crescente, como também a função teto $f(x) = x$.

Aritmética modular