Projeto e Análise de Algoritmos

Flávio L. C. de Moura

02 de maio de 2023

Nas últimas aulas, resolvemos algumas relações de recorrência que expressam o comportamento de algoritmo recursivos. O primeiro método utilizado consiste no *método da substituição* que é composto de duas etapas:

1. Adivinhar/chutar uma solução;

2. Usar indução matemática para provar que o chute está correto.

A primeira etapa pode passar a ideia de que o chute é feito de forma arbitrária e sem critérios, mas isto obviamente não faz sentido. Observe que não precisamos encontrar uma solução exata, mas apenas uma cota para a recorrência em questão, o que em geral é mais fácil. Por exemplo, considere a seguinte relação de recorrência

$$\begin{aligned} T(n) &= 2.T(n-1) + c \\ T(1) &= 1 \end{aligned} \tag{1}$$

onde c é uma constante. Inicialmente, podemos *iterar* a recorrência até que possamos visualizar a forma (ou padrão) da solução:

$$\begin{aligned}
T(n) &= 2.\underline{T(n-1)} + c \\
&= 2.\overline{(2.T(n-2) + c)} + c \\
&= 2^2.T(n-2) + 2.c + c \\
&= 2^2.\underline{T(n-2)} + 2.c + c \\
&= 2^2.\overline{(2.T(n-3) + c)} + 2.c + c \\
&= 2^3.T(n-3) + 2^2.c + 2.c + c \\
&= \ldots \text{(aqui podemos inferir um padrão)} \\
&= 2^{n-1}.T(n-(n-1)) + 2^{n-2}.c + \ldots + 2^2.c + 2.c + c \\
&= 2^{n-1}.T(1) + \sum_{i=0}^{n-2} 2^i.c \\
&= 2^{n-1} + c.\sum_{i=0}^{n-2} 2^i \\
&= 2^{n-1} + c.(2^{n-1} - 1) \\
&= (c+1).2^{n-1} - c
\end{aligned}$$

Agora podemos utilizar indução para verificar que a solução encontrada é correta. A indução é sobre $n \geq 1$. A base da indução ($n = 1$ nos dá que $T(1) = (c+1).2^{1-1} - c = c + 1 - c = 1$ que coincide com a condição de contorno dada em (1). No passo indutivo ($n > 1$), temos $T(n) = 2.T(n-1) + c \stackrel{h.i.}{=} 2.((c+1).2^{n-2} - c) + c = 2.(c+1).2^{n-2} - 2.c + c = (c+1).2^{n-1} - c$ como queríamos concluir. Esta solução nos permite concluir que $T(n) = \Theta(2^n)$.

Uma questão que surgiu na aula passada foi sobre a recorrência $T(1) = \Theta(1), T(n) = 3T(n/3+5)+n/2$, e o que argumentamos é o *aumento* em 5 não deveria alterar o comportamento assintótico da recorrência dada em relação à recorrência $T(1) = \Theta(1), T(n) = 3T(n/3) + n/2$. Podemos verificar esta afirmação da seguinte forma: podemos utilizar o Teorema Mestre para obtermos uma solução da recorrência $T(1) = \Theta(1), T(n) =$

$3T(n/3)+n/2$, e utilizar esta solução como chute para $T(1)=\Theta(1), T(n)=3T(n/3+5)+n/2$ via o método da substituição.

Vimos duas versões do Teorema Mestre. A primeira pode ser encontrada em [3]:

Teorema 1. *Seja $T(n)$ uma função eventualmente não-decrescente que satisfaz a recorrência*

$T(n)=a.T(n/b)+f(n), \quad$ *para* $n=b^k, k=1,2,3,\ldots$

$T(1)=c$

onde $a\geq 1, b\geq 2$ e $c\geq 0$. Se $f(n)=\Theta(n^d)$, onde $d\geq 0$, então

$$T(n)=\begin{cases}\Theta(n^{\log_b a}), & se\ a>b^d\\ \Theta(n^d.\lg n), & se\ a=b^d\\ \Theta(n^d), & se\ a<b^d\end{cases}$$

E a segunda versão, um pouco mais geral, está disponível em [1]:

Teorema 2. *Sejam $a\geq 1$ e $b>1$ constantes, $f(n)$ uma função assintoticamente positiva, e $T(n)$ definida nos inteiros não-negativos pela recorrência $T(n)=a.T(n/b)+f(n)$, onde n/b deve ser interpretado como $\lfloor n/b\rfloor$ ou $\lceil n/b\rceil$. Então $T(n)$ tem as seguintes cotas assintóticas:*

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 $a.f(n/b)\leq c.f(n)$ para alguma constante $c<1$, então para todo n suficientemente grande, temos que $T(n)=\Theta(f(n))$.*

Em ambos os casos, obtemos $T(n)=\Theta(n.\lg n)$ como solução da recorrência $T(1)=\Theta(1)$, $T(n)=3T(n/3)+n/2$. Utilizaremos esta solução para encontrarmos um chute e, via o método da substituição, mostraremos que este chute é solução da recorrência $T(1)=\Theta(1), T(n)=3T(n/3+5)+n/2$. Para que possamos utilizar indução, precisamos remover a notação assintótica da solução. Um outro aspecto importante é que o método da substituição é bastante eficiente para estabelecer tanto cota superior quanto inferior de uma recorrência, mas normalmente não é uma boa ideia estabelecer ambas as cotas simultaneamente.

Exercício 3. *Mostre que a recorrência $T(1)=\Theta(1)$, $T(n)=3T(n/3+5)+n/2$ tem solução $T(n)=O(n.\lg n)$.*

Assim, o método da substituição pode ser visto como uma forma para provar cotas assintóticas de uma recorrência por indução. A escolha de um chute, quando não for obtido iterativamente como no primeiro exemplo, exige diversos cuidados. Para maiores detalhes, recomendamos a leitura da Seção 4.3 de [1] e/ou da nova edição [2].

Uma outra questão interessante que surgiu na aula passada inclui alguns casos de recorrências onde o Teorema Mestre não se aplica. A recorrência $T(n)=2T(n/2)+n.\lg n$ aparentemente cai no caso 3 do Teorema Mestre porque $n.\lg n$ é assintoticamente limitada inferiormente por $n=n^{\lg 2}$. Mas o problema é que ela não polinomialmente maior: de fato, para que o caso 3 aplicasse precisaríamos que $n.\lg n=\Omega(n^{1+\epsilon})$ para algum $\epsilon>0$. Ou seja, $n.\lg n\geq c.n^{1+\epsilon} \implies \lg n\geq c.n^{\epsilon}$ mas esta última desigualdade não vale para nenhum $\epsilon>0$. Para mostrar este fato, prove o teorema a seguir (disponibilizado como exercícios no pdf das aulas 5 e 6 (`http://flaviomoura.info/files/paa-2023-1-aulas05e06.pdf`)):

Teorema 4. $\lg n=o(n^{\alpha}),\forall\alpha>0$. *Ou seja, a função logaritmo cresce mais lentamente do que qualquer potência de n (incluindo potências fracionárias)*

onde

Definição 5. *Seja $g(n)$ uma função dos inteiros não-negativos nos reais positivos. Definimos, $o(g(n))=\{f(n):$ para qualquer constante positiva c, existe uma constante positiva n_0 tal que $0\leq f(n)<c\cdot g(n),\forall n\geq n_0.\}$*

Outro exercício interessante consiste em mostrar que qualquer polinômio cresce mais lentamente do que qualquer função exponencial:

Teorema 6. $n^k = o(2^n), \forall k > 0$. *Ou seja, potências de* n *crescem mais lentamente que a função exponencial* 2^n. *Mais ainda, potências de* n *crescem mais lentamente do que qualquer função exponencial* $c^n, c > 1$.

Referências

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

[2] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, Cambridge, MA, USA, fourth edition, April 2022.

[3] A. V. Levitin. *Introduction to the Design and Analysis of Algorithms, Third Edition*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2012.