Árvore de recursão para resolver recorrências: passo a passo

Abrantes Araújo Silva Filho

2020-05-27

1 Árvore de recursão para resolver recorrências

Ao descrever o tempo de execução de alguns algoritmos precisaremos utilizar e resolver **recorrências**, que são equações ou desigualdades que descrevem uma função em termos de seu valor para entradas menores. O exemplo padrão é o *merge sort*, cujo tempo de execução $T(n)$ é dado pela recorrência:

$$T(n) = \begin{cases} \Theta(1) & \text{se } n = 1 \\ T\left(\left\lceil \frac{n}{2} \right\rceil\right) + T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + \Theta(n) & \text{se } n > 1 \end{cases} \tag{1}$$

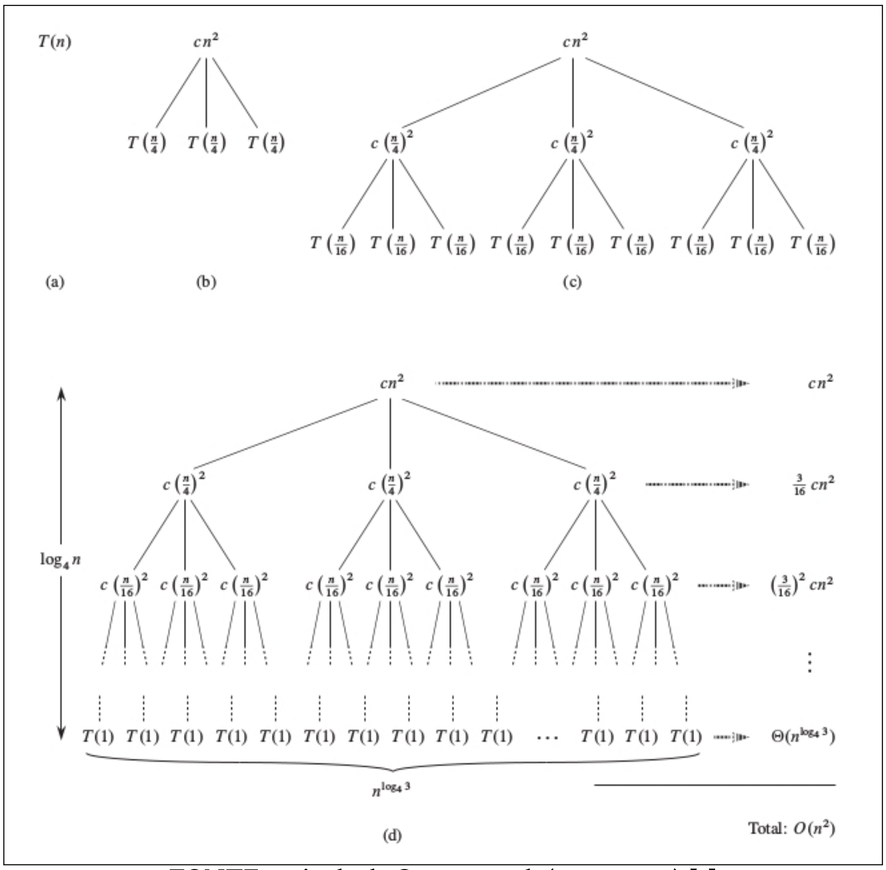
Apesar da recorrência exibida na Equação 1 ser matematicamente correta, na prática não precisamos trabalhar com todo esse rigor e podemos simplficar um pouco as coisas. Em primeiro lugar podemos ignorar a situação na qual $n = 1$, pois estamos interessados no comportamento da função para grandes valores de n. E, em segundo lugar, a diferença entre $\lceil n/2 \rceil$ e $\lfloor n/2 \rfloor$ não é grande e pode ser omitida. Desse modo podemos simplificar a recorrência do *merge sort* para:

$$T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n) \tag{2}$$

Para resolver a recorrência da Equação 2 poderíamos arriscar um palpite para a solução, por exemplo $T(n) = O(n \log_2(n))$, e usar o **método da substituição** e indução matemática para verificar se o palpite está correto. O problema desse método de resolução é determinar qual o palpite correto: a princípio não temos nenhuma idéia de qual palpite utilizar. Qual seria o palpite correto para uma recorrência mais complicada, como $T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2)$ ou como $T(n) = 3T(\lfloor n/2 \rfloor) + \Theta(n)$?

Para resolver o problema de identificar um bom palpite utilizamos o **método da árvore de recursão**. A idéia básica é a seguinte: expandimos a recorrência criando uma árvore de nós onde cada nó representa o custo de um único subproblema; no final, se somarmos os custos de todos os nós, teremos um bom palpite para a solução da recorrência, e esse palpite pode ser verificado pelo método da substituição. Por exemplo, para a recorrência $T(n) = 3T\left(\lfloor n/4 \rfloor\right) + \Theta(n^2)$ teríamos a seguinte árvore de recursão:

Figura 1: Árvore de recursão para $T(n) = 3T\left(\lfloor n/4 \rfloor\right) + \Theta(n^2)$



FONTE: retirado de Cormen et al. (2009, p. 89) [1]

A descrição da resolução de uma árvore de recursão pode ser encontrada no livro “Algoritmos: Teoria e Prática” [1] mas é uma descrição muito seca, com poucos detalhes para facilitar o entendimento de quem está começando a estudar o assunto. Meu objetivo, nas próximas seções, é explicar com mais detalhes um modo passo a passo para resolver uma árvore de recursão, e encontrar um palpite de solução para a recorrência que possa ser verificado pelo método da substituição.

2 Conhecimento preliminar

Antes de explicar como resolver uma árvore de recursão, você deve ser capaz de identificar várias informações em uma árvore de recursão como a ilustrada na Figura 1, por exemplo: a altura da árvore, a quantidade de nós em cada nível, o maior (último) nível da árvore, e o custo de cada nó. Esta seção explica como encontrar todas as informações necessárias para a resolução de uma árvore de recursão com a forma geral

$$T(n) = aT\left(\frac{n}{b}\right) + \Theta(f(n)) \tag{3}$$

onde:

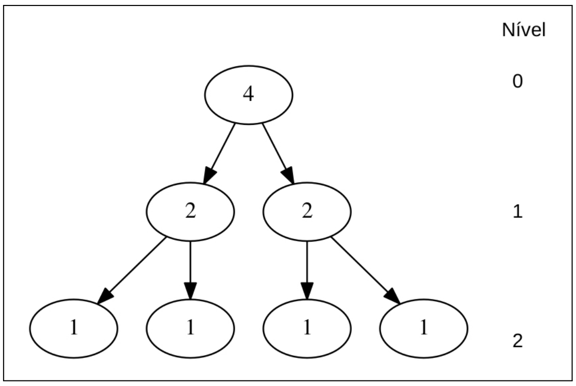
- a é o número de chamadas recursivas que são feitas, ou seja, é o número de subproblemas em que o problema original foi dividido;
- b é o fator que divide o tamanho do problema original, ou seja: cada subproblema terá $1/b$ do tamanho original; e
- $f(n)$ é uma função dada que representa o custo do algoritmo.

Para facilitar o entendimento e partir do abstrato para o concreto, utilizarei como exemplo a recursão $T(n) = 3T\left(\lfloor n/4 \rfloor\right) + \Theta(n^2)$ ilustrada na Figura 1.

2.1 Níveis e altura da árvore de recursão

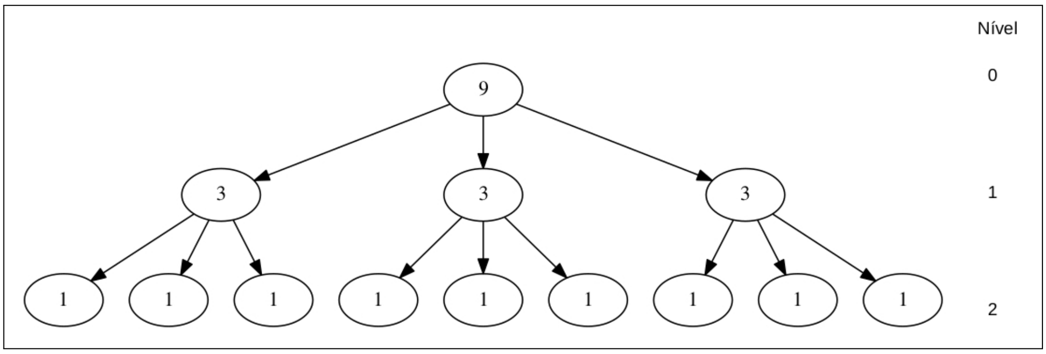
A primeira coisa que você precisa saber calcular é o número de níveis da árvore de recursão. Considere por exemplo a árvore na Figura 2, com $n = 4$, que faz duas chamadas recursivas ($a = 2$), cada uma divindo o problema em duas partes ($b = 2$). Cada nível é um "andar" da árvore, e o primeiro nível, a **raiz** da árvore é sempre o nível 0:

Figura 2: Árvore de recursão para $T(n) = 2T\left(n/2\right)$ e $n = 4$



Outro exemplo está na Figura 3, com $n = 9$, que faz três chamadas recursivas ($a = 3$), cada uma divindo o problema em três partes ($b = 3$). Note que nem sempre a será igual à b, depende da recursão realizada.

Figura 3: Árvore de recursão para $T(n) = 3T\left(n/3\right)$ e $n = 9$



O cálculo do **maior nível** de uma árvore de recursão no formato $T(n) = aT\left(\frac{n}{b}\right) + \Theta(f(n))$ é bem simples e dado por:

$$\text{Maior nível } = \log_b(n) \tag{4}$$

No exemplo da Figura 2, com $T(n) = 2T(n/2)$, o maior nível da árvore é dado por $\log_b(n) = \log_2(4) = 2$. De forma semelhante, o maior nível da árvore ilustrada na Figura 3, com $T(n) = 3T(n/3)$, é dado por $\log_b(n) = \log_3(9) = 2$.

Com a informação do maior nível podemos agora calcular a **altura** da árvore apenas somando-se 1 ao maior nível, pois o nível da raiz começa em 0:

$$\begin{aligned} \text{Altura } &= \text{maior nível } + 1 \\ &= \log_b(n) + 1 \end{aligned} \tag{5}$$

Qual seria o maior nível e a altura da recursão $T(n) = 3T\left(\lfloor n/4 \rfloor\right) + \Theta(n^2)$? O maior nível seria simplesmente $\log_4(n)$ e a altura seria $\log_4(n) + 1$.

2.2 Quantidade de nós em cada nível

Outra informação importante que você deve aprender a calcular é a quantidade de nós em cada nível i da árvore.

Considere, por exemplo, a árvore ilustrada na Figura 3: ela tem um nó no nivel 0, três nós no nível 1, e nove nós no nível 2.

O cálculo do **número de nós em cada nível** i de uma árvore de recursão na forma $T(n) = aT\left(\frac{n}{b}\right) + \Theta(f(n))$ é dado por:

$$\text{Nós no nível } i \ = a^i \tag{6}$$

Como um exemplo, considere novamente a recursão $T(n) = 3T(n/3)$: o número de nós em cada nível seria:

- Nós no nível 0: $a^i = 3^0 = 1$
- Nós no nível 1: $a^i = 3^1 = 3$
- Nós no nível 2: $a^i = 3^2 = 9$

Considere agora a recursão $T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2)$. Qual seria o número de nós em um nível i? Seria $a^i = 3^i$. Até aqui nenhum mistério. Agora imagine que você quer calcular o número de nós no último nível dessa recursão (o maior nível): como você faria isso considerando que não sabemos o valor exato de n?

Na Seção 2.1 você aprendeu que o maior nível é dado por $\log_b(n)$. Então o número de nós do maior (último) nível da árvore ocorre quando $i = \log_b(n)$, e é dado por:

$$\begin{aligned}\text{Nós no último nível } &= a^i \\ &= a^{\log_b(n)} \\ &= n^{\log_b(a)}\end{aligned} \tag{7}$$

Para a recursão $T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2)$, de acordo com a Equação 7, a quantidade de nós no último nível é dada por $n^{\log_b(a)} = n^{\log_4(3)}$.

2.3 Tamanho dos subproblemas em cada nível

A próxima informação que você precisa aprender a calcular em uma árvore de recursão é o tamanho de cada subproblema, em um determinado nível i da árvore.

Chamamos de n_i o tamanho do subproblema em um determinado nível i da árvore, e seu cálculo é dado por:

$$n_i = \frac{n}{b^i} \tag{8}$$

Por exemplo, o tamanho de cada subproblema no nível 1 da recursão $T(n) = 3T(n/3)$, com $n = 9$, ilustrada na Figura 3, é dado por $\frac{n}{b^i} = \frac{9}{3^1} = 3$.

Para a árvore de recursão $T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2)$, o tamanho do subproblema em cada nível i é dado por $\frac{n}{b^i} = \frac{n}{4^i}$.

Note também que a partir da Equação 8 podemos deduzir a Equação 4, para descobrir o maior nível da árvore: ele será simplesmente o nível i no qual $n_i = 1$, conforme a demonstração abaixo:

$$\begin{aligned} n_i &= \frac{n}{b^i} \\ 1 &= \frac{n}{b^i} \\ b^i &= n \\ \log_b(b^i) &= \log_b(n) \\ i \log_b(b) &= \log_b(n) \\ i &= \log_b(n) \end{aligned} \tag{9}$$

2.4 Custo de cada nó

Uma informação crucial para a resolução de uma árvore de recursão no formato $T(n) = aT\left(\frac{n}{b}\right) + \Theta(f(n))$ é o **custo de cada nó** na árvore.

Esse custo está relacionado com a função $\Theta(f(n))$ da recursão e deve ser calculado de duas maneiras diferentes:

- O custo para cada nó em todos os níveis, **exceto o último nível**, ou seja, o custo para os nós dos níveis $i = \{0, 1, 2, 3, \ldots, \log_b(n) - 1\}$; e
- O custo para cada nó do **último nível**, ou seja, o custo para os nós do nível $\log_b(n)$.

É necessário fazer essa divisão pois o custo dos nós do último nível, quando $T(n) = T(1)$ é, em tese, constante e sempre igual à 1. Isso não é verdade para os nós nos outros níveis, onde o tamanho de cada subproblema será diferente em cada nível.

De todas as informações que você precisa calcular, essa é uma das mais difíceis pois varia de acordo com a função $\Theta(f(n))$ dada.

É mais fácil explicar com um exemplo. Considere a recursão $T(n) = 3T\left(\lfloor n/4 \rfloor\right) + \Theta(n^2)$. Como a função de custo é n^2 e o tamanho de cada subproblema é dado por n_i, então o custo de cada nó, em cada nível exceto o último, é dado por $c(n_i)^2$:

$$c(n_i)^2 = c\left(\frac{n}{b^i}\right)^2 = c\left(\frac{n}{4^i}\right)^2 = c\frac{n^2}{16^i} \tag{10}$$

2.5 Séries geométricas

A última coisa que você precisa saber para poder resolver uma árvore de recursão são duas fórmulas para séries geométricas.

A primeira é a fórmula da **série geométrica exponencial** que nos diz que para $x \neq 1$:

$$\sum_{k=0}^{n} x^k = 1 + x + x^2 + \cdots + x^n = \frac{x^{n+1} - 1}{x - 1} \tag{11}$$

A segunda é a fórmula para **série geométrica decrescente infinita** que nos diz que para $|x| < 1$:

$$\sum_{k=0}^{\infty} x^k = 1 + x + x^2 + \cdots = \frac{1}{1 - x} \tag{12}$$

3 Resolução passo a passo

Com o conhecimento preliminar adquirido na Seção 2, basta seguir uma seqüência de passos para resolver a árvore de recursão no formato $T(n) = aT\left(\frac{n}{b}\right) + \Theta(f(n))$ e encontrar um bom palpite para a solução (que depois deverá ser verificada utilizando-se o método da substituição). Os passos para a solução da árvore de recursão são os seguintes:

1º: Determinar **quantos nós existem em cada nível**, para todos os níveis exceto o último (ou seja, níveis $i = \{0, 1, 2, \ldots, \log_b(n) - 1\}$), conforme a Equação 6 da Seção 2.2;

2º: Determinar o **custo de cada nó** para todos os níveis exceto o último (ou seja, o custo dos nós nos níveis $i = \{0, 1, 2, \ldots, \log_b(n) - 1\}$), conforme explicado na Seção 2.4 (você terá que levar em conta o tamanho dos subproblemas em cada nível, conforme a Equação 8 da Seção 2.3, e a função de custo $\Theta(f(n))$);

3º: Determinar o **custo total de cada nível** para todos os níveis exceto o último (ou seja, níveis $i = \{0, 1, 2, \ldots, \log_b(n) - 1\}$), multiplicando a quantidade de nós em cada nível (1º passo) pelo custo de cada nó (2º passo);

4º: Determinar a **soma de todos os custos de todos os níveis**, exceto o último (ou seja, a soma dos custos de todos os níveis $i = \{0, 1, 2, \ldots, \log_b(n) - 1\}$);

5º: Determinar a **soma dos custos do último nível** através da quantidade de nós existente no último nível, conforme a Equação 7 da Seção 2.2 (note que como supomos que o custo $T(n)$ de cada nó no último nível é $T(1) = 1$, ou seja, constante, a quantidade de nós nesse último nível já corresponderá ao custo total do último nível);

6º: Determinar a **soma dos custos de todos os níveis**, do primeiro ao último, somando os custos encontrados no 4º e 5º passos;

7º: Utilizar uma das **séries geométricas** (Equação 11 ou Equação 12, conforme a situação) para simplificar (limitar) as somas e encontrar um palpite para $T(n)$ — a rigor, aqui termina a resolução da árvore de recursão; e

8º: Utilizar o **método da substituição** para verificar se o palpite encontrado no 7º passo está correto.

3.1 Exemplo de resolução: 1

Os passos listados anteriormente ficarão claros com um exemplo concreto. Vamos utilizar o método da árvore de recursão para encontrar um bom palpite para a recorrência $T(n) = 3T\left(\lfloor n/4 \rfloor\right) + \Theta(n^2)$ ilustrada na Figura 1. Seguindo o passo a passo:

1º: Determinar **quantos nós existem em cada nível**, para todos os níveis exceto o último:

$$a^i = 3^i$$

2º: Determinar o **custo de cada nó** para todos os níveis exceto o último: como a função de custo é $\Theta(f(n^2))$, consideramos que o custo é então cn^2, e como o tamanho n de cada subproblema em cada nível i é $\frac{n}{b^i}$, o custo de cada nó é:

$$c\left(\frac{n}{b^i}\right)^2 = c\left(\frac{n}{4^i}\right)^2 = c\frac{n^2}{16^i}$$

3º: Determinar o **custo total de cada nível** para todos os níveis exceto o último:

$$3^i \times c\frac{n^2}{16^i} = \frac{3^i}{16^i}cn^2 = \left(\frac{3}{16}\right)^i cn^2$$

4º: Determinar a **soma de todos os custos de todos os níveis**, exceto o último: como o último nível é dado por $\log_b(n) = \log_4(n)$, o penúltimo nível é $\log_4(n) - 1$, e a soma será:

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

5º: Determinar a **soma dos custos do último nível** através da quantidade de nós existente no último nível:

$$n^{\log_b(a)} = n^{\log_4(3)} = \Theta(n^{\log_4(3)})$$

6º: Determinar a **soma dos custos de todos os níveis**, do primeiro ao último:

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

7º: Utilizar uma das **séries geométricas** para simplificar as somas e encontrar um palpite: a série adequada aqui é uma série geométrica descrescente infinita, assim:

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

Note que como $n^{\log_4(3)} \approx n^{0.79}$, o termo n^2 domina assintoticamente o tempo de execução, por isso nosso palpite é o de que $T(n) = O(n^2)$. A verificação pelo método da substituição não será feita aqui.

3.2 Exemplo de resolução: 2

Agora vamos utilizar o método da árvore de recursão para encontrar um bom palpite para a recorrência $T(n) = 3T(\lfloor n/2 \rfloor) + \Theta(n)$. Seguindo o passo a passo:

1º: Determinar **quantos nós existem em cada nível**, para todos os níveis exceto o último:

$$a^i = 3^i$$

2º: Determinar o **custo de cada nó** para todos os níveis exceto o último: como a função de custo é $\Theta(f(n))$, consideramos que o custo é então cn, e como o tamanho n de cada subproblema em cada nível i é $\frac{n}{b^i}$, o custo de cada nó é:

$$c\left(\frac{n}{b^i}\right) = c\left(\frac{n}{2^i}\right) = c\frac{n}{2^i}$$

3º: Determinar o **custo total de cada nível** para todos os níveis exceto o último:

$$3^i \times c\frac{n}{2^i} = \frac{3^i}{2^i}cn = \left(\frac{3}{2}\right)^i cn$$

4º: Determinar a **soma de todos os custos de todos os níveis**, exceto o último: como o último nível é dado por $\log_b(n) = \log_2(n)$, o penúltimo nível é $\log_2(n) - 1$, e a soma será:

$$\sum_{i=0}^{\log_2(n)-1} \left(\frac{3}{2}\right)^i cn$$

5º: Determinar a **soma dos custos do último nível** através da quantidade de nós existente no último nível:

$$n^{\log_b(a)} = n^{\log_2(3)} = \Theta(n^{\log_2(3)})$$

6º: Determinar a **soma dos custos de todos os níveis**, do primeiro ao último:

$$T(n) = \sum_{i=0}^{\log_2(n)-1} \left(\frac{3}{2}\right)^i cn + \Theta(n^{\log_2(3)})$$

7º: Utilizar uma das **séries geométricas** para simplificar as somas e encontrar um palpite: a série adequada aqui é uma série geométrica exponencial, assim:

$$\begin{aligned}
T(n) &= \sum_{i=0}^{\log_2(n)-1} \left(\frac{3}{2}\right)^i cn + \Theta(n^{\log_2(3)}) \\
&= \frac{(3/2)^{(\log_2(n)-1)+1} - 1}{(3/2) - 1} cn + \Theta(n^{\log_2(3)}) \\
&= \frac{(3/2)^{\log_2(n)}}{(3/2) - 1} cn + \Theta(n^{\log_2(3)}) \\
&= \frac{(n)^{\log_2(3/2)}}{(1/2)} cn + \Theta(n^{\log_2(3)}) \\
&= 2\left[n^{\log_2(3/2)}\right] cn + \Theta(n^{\log_2(3)}) \\
&= 2\left[n^{\log_2(3)-\log_2(2)}\right] cn + \Theta(n^{\log_2(3)}) \\
&= 2\left[n^{\log_2(3)-1}\right] cn + \Theta(n^{\log_2(3)}) \\
&= O(n^{\log_2(3)})
\end{aligned}$$

Note que como $n^{\log_2(3)}$ domina assintoticamente o termo $n^{\log_2(3)-1}$, nosso palpite é o de que $T(n) = O(n^{\log_2(3)})$. A verificação pelo método da substituição não será feita aqui.

3.3 Cuidados

O passo a passo descrito aqui funciona muito bem para as recorrências do tipo genérico $T(n) = aT\left(\frac{n}{b}\right) + \Theta(f(n))$. Se, por outro lado, você tem recorrência mais complicadas, como por exemplo

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

você ainda poderá utilizar o método da árvore de recursão, mas teríamos que considerar, por exemplo, o caminho mais longo da raiz até o último nível. Consulte o livro do Cormen [1] para maiores detalhes.

Outro cuidado importante é com a simplificação das séries geométricas: dependendo da recursão considerada a álgebra pode se tornar bem complicada e é fácil se perder nas contas e acabar cometendo um erro na simplificação.

Referências

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, Cambridge, 3 edition, 2009. ISBN 9780262033848. URL https://www.amzn.com/0262033844.