Projeto e analíse de algoritmos

Amanda Goulart

Dezembro 2021

1 Suponha que você tem 3 algoritmos que resolvem um certo problema:

1.1 Algoritmo I resolve o problema dividindo-o em 5 subproblemas de metade do tamanho cada, resolvendo-os recursivamente, e combinando suas soluções em tempo linear;

5 subproblemas = 5T
metade do tamanho = $\frac{n}{2}$
tempo linear = + n
Sendo assim com esses dados podemos estabelecer uma relação de recorrência, sendo ela:
$T(n) = 5T(\frac{n}{2}) + n$
Utilizando o teorema mestre:
a = 5 b = 2
$n^{\log_b a} = n^{\log_2 5}$
Sabendo que $\log_2 5$ é $\simeq 2,32$, temos o caso 1 do teorema mestre, resultando em:
T(n) = $O(n^{2,32})$

1.2 Algoritmo II resolve o problema de tamanho n resolvendo dois subproblemas de tamanho n-1 e combinando as soluções em tempo constante;

2 subproblemas = 2T
tamanho = n-1 tempo constante = 1
Sendo assim com esses dados podemos estabelecer uma relação de recorrência, sendo ela:
T(n) = 2T(n-1) + 1
Utilizando o método de substituição:
2(2T(n-1)+1 =
4T((n-2)+3) = 2((4T(n-2)+3)+1 =n 8T(n-3) + 7.
Como t(1) = 1 e supondo que n-1 = 1
$T(n) = 2^{n-1} + 2^{n-1} - 1$
logo,
$2^{n-1} + 2^{n-1} \leq cg(n)$
$1 * 2^n \leq cg(n)$, portanto c=1 e g(n) = 2^n, sendo assim:
$O(2^n)$

1.3 Algoritmo III resolve o problema de tamanho n dividindo-o em nove subproblemas de tamanho n/3 cada, resolvendo-os recursivamente, e combinando as soluções em tempo $\Theta(\mathbf{n^2})$

9 subproblemas = 9T

tamanho = $\mathrm{n}\frac{}{3}$
tempo = $\Theta(\mathbf{n^2})$
Sendo assim com esses dados podemos estabelecer uma relação de recorrência, sendo ela:
$T(n) = 9T(\frac{n}{3}) + \Theta(n^2)$
Utilizando o teorema mestre:
a = 9 b = 3
$n^{\log_b a} = n^{\log_3 9} = n^2$
Sabendo que f(n) = $\mathbf{n^2}$ e $n^{\log_3 9} = n^2$, k = 2 , logo k > 0. Caindo no caso 2, do teorema mestre, sendo assim $O(n^2 log(n))$

1.3.1 Determine os tempos de execução desses algoritmos utilizando a notação O. Qual algoritmo você escolheria?

Agora tendo conhecimento do tempo de execução de cada algoritmo, iremos analisar qual seria a melhor opção, olhando para
$O(n^{2,32}) < O(2^n)$ e $O(n^2 log(n) < O(2^n)$, portanto é possivel ver o algoritmo II, possui o maior tempo.
Dito isso, basta escolher anlisar as funções tendendo a ∞.
$\lim_{n\to\infty} = \frac{n^{2,32}}{n^2 \log(n)} = \lim_{n\to\propto} = \frac{n^{2+0,32}}{n^2 \log(n)}$
Sendo uma indeterminação, aplicarei L'Hospital:
$\lim_{n\to\propto} = \frac{n^{2,32'}}{n^2 \log(n)'} = \lim_{n\to\propto} = \frac{0,32*n*ln(10)}{n^0} * 68 = \lim_{n\to\propto} = 0.32 * ln(10) * x^{0.32}$
Temos que $n^{0,32} > log(n)$, portanto o algorimo escolhido pe o III, ou seja o $O(n^2 log(n)$

2 Seja o problema de se multiplicar dois números naturais com no máximo n dígitos cada. O algoritmo de multiplicação tradicional possui complexidade $\mathbf{O(n^2)}$. O algoritmo de Karatsuba resolve esse problema por meio de divisão e conquista com custo que pode ser descrito pela recorrência simplificada T(n) = 3T(n/2) + n.

2.1 Resolva a recorrência utilizando o método da árvore de recursão.

T(n) = 3T(n/2) + n
Desenvolvendo a árvore:

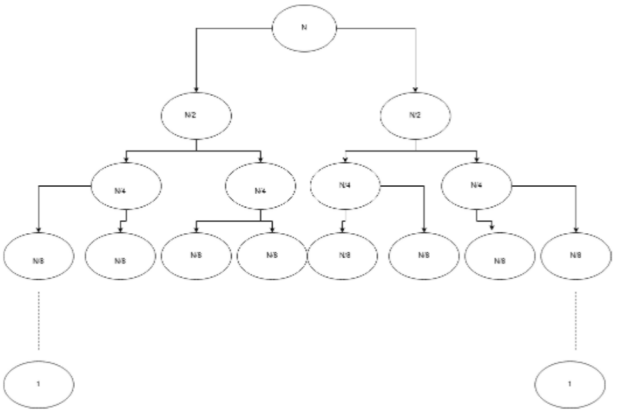

Figura 1: T(n) = 3T([n/2]) + n

Nível	Tamanho	nós	Tempo por nó
0	n	1	n
1	n/2	2	n/2
2	n/4	4	n/4
3	n/8	8	n/8
i	$\frac{n}{2^i}$	2^i	n/ 2^i
$\log_2 n$	1	n	1

Tempo:
$\sum_{i=0}^{\log_n 1}(\frac{3}{2}) * n + n^{\log_2 3}$
Note que é uma série geometrica, logo podemo utilizar a formula, sabendo que :
a = n
r = $\frac{3}{2}$,
n = $\log_2(n+1)$
$T(n) = \frac{n((\frac{3}{2}^{\log_2(n-1)}-1))}{\frac{3}{2}-1}$
$T(n) = \frac{n((3^{\log_2(n-1)}-2^{\log_2(n-1)}))}{3(2^{\log_2(n-1)})-2^{\log_2(n-1))}}$
$T(n) = \frac{n((3^{\log_2(n)}-2^{\log_2(n)}))}{3(2^{\log_2(n)})-2^{\log_2(n))}}$
$T(n) = \frac{n((3^{\log_2 3}-2n))}{3n-2n}$
$T(n) = 3n((3^{\log_2 3} - 2n))$
Portanto,
$\Theta(n^{\log_2 3})$

2.2 Mostre pelo método da substituição que a complexidade encontrada em a) vale como um limite superior.

Chute : T(n) é $\Theta(n^{\log_2 3})$
Vamos provar por indução que T(N) $\leq$c* $n^{\log_2 3}$
Base : n= 2 , Como $T(2) = 3T(\frac{2}{2}) + 2 = 6ec * 1 * 2,32 = 2,32c$, vale se
$6 \leq 2,32c = c \geq 2,32$.

Queremos provar que T(n) $\leq$c* $n^{\log_2 3}$, se n > 2,32

HIPOTESE: $T(n) \leq ck \log k$
$\forall 2, 32 \leq k \leq n$
Como $T(n) = 3T(\frac{n}{2}) + ne\frac{n}{2} < n$, por hípotese vale $T(\frac{n}{2} \leq c(\frac{n}{2} \log \frac{n}{2})$
Então
$T(n) = 3T(\frac{n}{2}) + n$ e $\leq 2[c\frac{n}{2}(\log n - 1)] + n =$
$=$ c n $\log n - cn + m \leq cn \log n$
E veja que $-cn + n \leq 0$ se $n \leq cn, oic \geq 1$.
Sabemos assim que a complexidade deste problema não deve superar $\Theta(n^{\log_2 3})$, uma vez que existe um algoritmo desta complexidade que o resolve.

3 Seja um conjunto A de n elementos e uma dada função f que mapeia elementos de A para A, considere o problema de encontrar um subconjunto S $\subseteq$ A de maior tamanho possível tal que a função f mapeia todo elemento de S a outro elemento de S e que nenhum de seus elementos é mapeado por mais do que um elemento, ou seja, f deve fazer mapeamento um-para-um em S.

3.1 Faça o projeto do algoritmo que resolva o problema acima por indução.

3.2 Escreva o pseudo-código do algoritmo.

3.3 Determine a complexidade do algoritmo em termos de n.

4 Gostaríamos de descobrir a resistência a quedas de um novo modelo de celular super resistente e suponhamos que essa resistência é a mesma para todas as unidades e que se o celular se quebra ao cair de uma altura x, então também irá se quebrar ao cair de qualquer altura $>$ x. O teste de resistência será feito em um prédio de 100 andares e o objetivo é descobrir a partir de qual andar o celular será quebrado ao ser jogado desse andar até o térreo. Um celular usado no teste que não se quebrou pode ser usado novamente. Assuma que, se o celular não se quebrar em uma queda, sua condição permanece inalterada.

4.1 Qual a quantidade ótima de tentativas do pior caso com uma quantidade infinita de celulares disponíveis?

Para solucionar este problema de forma ótima, podemos utilizar uma árvore binária de busca, onde iniciamos no 50, e dependendo da resposta, iriamos para números maiores ou menores, e assim sucessivamente. Com base nisso sabemos que a busca em uma ABB de tamanh N, é dada por $\log_2 N$

Suponhamos que temos uma árvore de tamanho N:
Para a interação 2, temos que o tamanho se torna $\frac{N}{2}$.
Para a interação 3, temos que o tamnho é $\frac{\frac{n}{2}}{2} = \frac{n}{2^2}$.
Logo para a interação K:

$\frac{n}{2^k}$
Até que o tamanho se torne 1
$\frac{n}{2^k} = 1$
$n = 2^k$
$\log_2 n = log2(2^k$
$\log_2 n = klog2(2)$
Obs:
$\log_a(a) = 1$
Portanto k = $\log_2(n)$
Como temos 100 andares, temos que $\log_2 100$ = 6,6438 tentativas.

4.2 Qual a quantidade ótima de tentativas do pior caso para 2 celulares disponíveis para o teste?

Suponha que soltaremos o celular no andar N, caso ele não quebre, será necessário ir ao próximo andar n + (n-1), e ao próximo n + (n-1) + (n-2), e assim sucessivamente:
n + (n-1) + (n-2) + (n-3) + (n-4) + ... + 1> = 100
Percebesse que temos um somátorio:
$\sum_{k=1}^{n} k = 1 + 2 + 3 + ... + (n-2) + (n-1) + n$, este somatório é conhecido pelo número triângular, e sia fórmula é dada por
$Tn = \frac{n(n-1)}{2} = 100$
Temos que N, ou seja a quantidade de tentativas com apenas dois celulares disponíveis é 13,651.