

Projeto e Análise de Algoritmos

Teorema Mestre

Prof. Rodrigo de Barros Paes

rodrigo@ic.ufal.br

<https://sites.google.com/site/ldsicufal/disciplinas/projeto-e-analise-de-algoritmos>

O que é?

Fornece uma “receita” para resolver recorrências da forma:

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

onde:

$$a \geq 1$$

$$b > 1$$

$f(n)$ é assintoticamente > 0

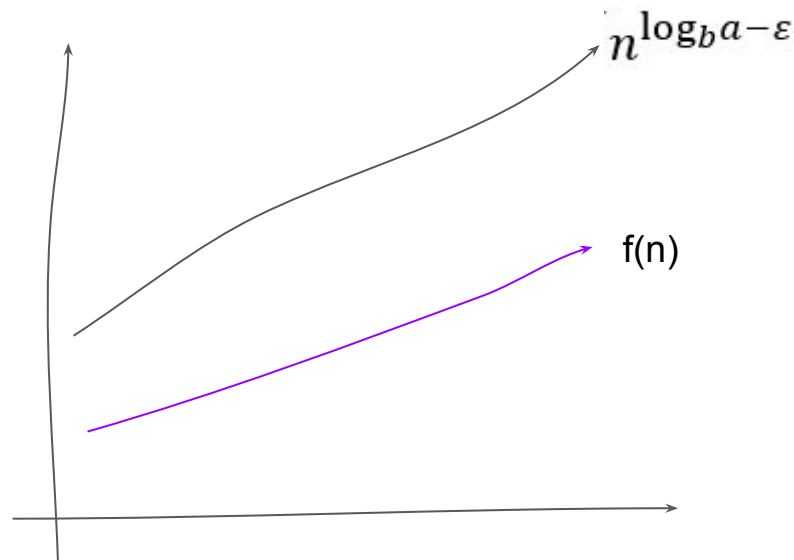
O método

Comparar $f(n)$ com $n^{\log_b a}$

CASO 1:

SE $f(n) = O(n^{\log_b a - \varepsilon})$, para $\varepsilon > 0$

ENTÃO $T(n) = \theta(n^{\log_b a})$



$f(n)$ não é apenas menor. Ela é assintoticamente menor por um fator n^ε . Ou seja, ela é polinomialmente menor.

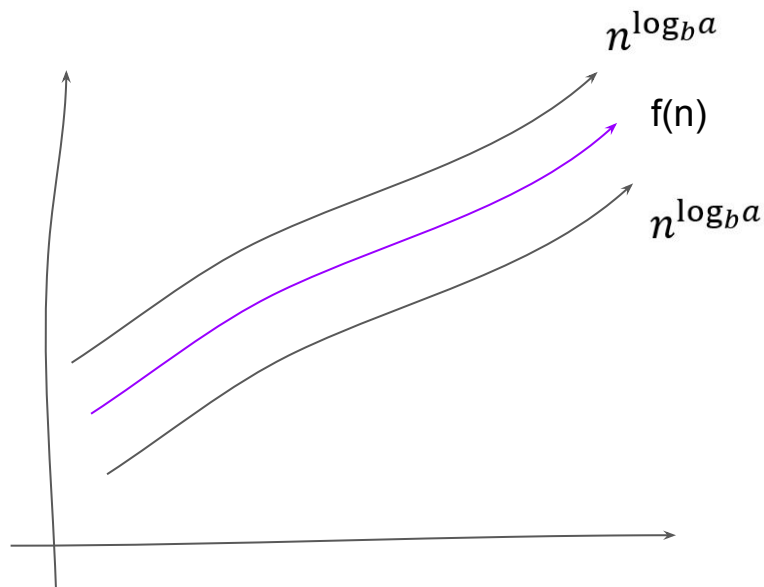
O método

Comparar $f(n)$ com $n^{\log_b a}$

CASO 2:

SE $f(n) = \theta(n^{\log_b a})$

ENTÃO $T(n) = \theta(n^{\log_b a} \lg n)$



O método

Comparar $f(n)$ com $n^{\log_b a}$

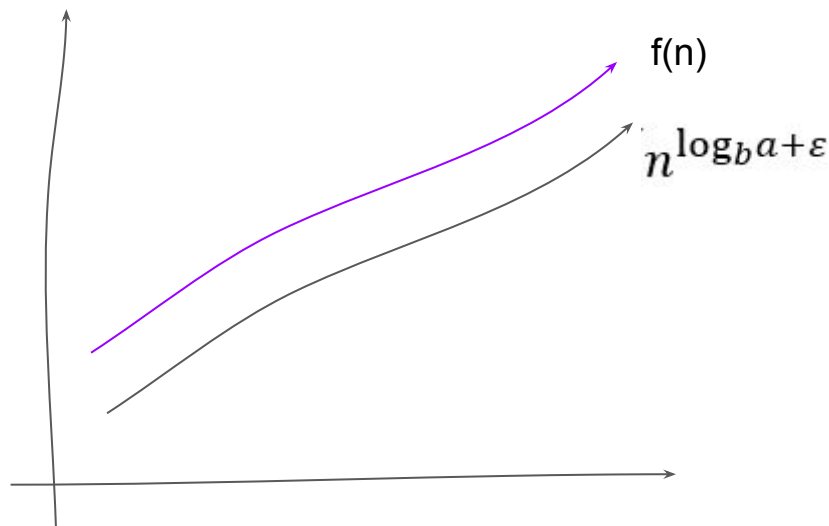
CASO 3:

SE ($f(n) = \Omega(n^{\log_b a + \varepsilon})$, para $\varepsilon > 0$)

E

($af(n/b) \leq cf(n)$, para $c < 1$)

ENTÃO $T(n) = \theta(f(n))$



Exemplo de uso

$$T(n) = 9T\left(\frac{n}{3}\right) + n$$

$$a = 9$$

$$b = 3$$

$$f(n) = n$$

Comparar $f(n)$ com $n^{\log_b a}$:

$$n^{\log_3 9}$$

$$= n^2$$

logo, $f(n)$ é assintoticamente menor que n^2 . \Rightarrow CASO 1 $\Rightarrow T(n) = \theta(n^2)$

“cola” do método

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

$$n^{\log_b a}$$

Exemplo 2

$$T(n) = T\left(\frac{2n}{3}\right) + 1$$

$$T(n) = T\left(\frac{n}{\frac{3}{2}}\right) + 1$$

$$a = 1$$

$$b = 3/2$$

$$f(n) = 1$$

Comparar $f(n)$ com $n^{\log_b a}$:

$$= n^{\log_{\frac{3}{2}} 1} = n^0 = 1$$

“cola” do método

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

$$n^{\log_b a}$$

CASO 2

$$T(n) = \theta(n^{\log_b a} \lg n)$$

$$T(n) = \theta(1 \times \lg n)$$

$$T(n) = \theta(\lg n)$$

Exemplo 3

$$T(n) = 3T\left(\frac{n}{4}\right) + n \lg n$$

$$a = 3$$

$$b = 4$$

$$f(n) = n \lg n$$

Comparar $f(n)$ com $n^{\log_b a}$:

$$n^{\log_4 3}$$

$$n^{0.793}$$

Será que ...

$$n \lg n = \Omega(n^{0.793+\epsilon})?$$

“cola” do método

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

$$n^{\log_b a}$$

Por exemplo, se $\epsilon \approx 0.2$, teríamos n^1 , ou seja, n é polinomialmente menor que $n \lg n$.

Resta saber se a segunda condição do caso 3 se aplica.

$$af(n/b) \leq cf(n)$$

Exemplo 3 (cont.)

$$af(n/b) \leq cf(n)$$

$$3f\left(\frac{n}{4}\right) \leq cf(n)$$

$$\therefore 3\left(\frac{n}{4} \lg \frac{n}{4}\right) \leq cn \lg n$$

$$\therefore \frac{3}{4}n \lg \frac{n}{4} \leq cn \lg n$$

Lembrando que $C < 1$

$$\therefore \frac{3}{4}n \lg \frac{n}{4} \leq \frac{3}{4}n \lg n$$

$$\therefore \lg \frac{n}{4} \leq \lg n$$

Verdade para n grande

Exemplo 4

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

$$a = 2$$

$$b = 2$$

$$f(n) = n \lg n$$

Comparar $f(n)$ com $n^{\log_b a}$:

$$n^{\log_2 2}$$

$$\therefore n^1$$

$$\therefore n$$

Logo $f(n)$ é assintoticamente maior que n .

Mas precisamos que seja polinomialmente maior

“cola” do método

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

$$n^{\log_b a}$$

$$n \lg n = \Omega(n^{1+\epsilon})$$

Nesse caso, qualquer que seja o ϵ ($\epsilon > 0$), teríamos a $n^{1+\epsilon}$ assintoticamente maior que $n \lg n$. Ou seja, é maior, mas não é polinomialmente maior.

Exemplo: $\epsilon = 0.01$

https://github.com/r0drigopaes/paa/blob/master/polinomialmente_maior.ods

Polinomialmente maior

Basta dividir uma função por outra e verificar se o resultado é assintoticamente do que a variável elevada a alguma constante positiva.

Exemplo:

$$f(x) = x^3$$

$$g(x) = x^2$$

$f(x)$ é polinomialmente maior que $g(x)$?

$$x^5/x^2 = x^3$$

x^3 é maior que x^ϵ ? sim, para qualquer $\epsilon < 3$

$$x^3/x^2 = x^1$$

x^1 é maior que x^ϵ ? sim, para qualquer $0 < \epsilon < 1$

$$x^3/x^3 = 1$$

1 é maior que x^ϵ ? Não. Não existe esse $\epsilon > 0$.

$$\lg n / n = \lg n$$

$\lg n$ é maior que n^ϵ ? Não, é assintoticamente menor para qualquer $\epsilon > 0$

Exemplo 5

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

Alunos, vou trocar isso aqui por algo mais didático em breve.

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

characterizes the running times of the divide-and-conquer algorithm for both the maximum-subarray problem and merge sort. (As is our practice, we omit stating the base case in the recurrence.) Here, we have $a = 2$, $b = 2$, $f(n) = \Theta(n)$, and thus we have that $n^{\log_b a} = n^{\log_2 2} = n$. Case 2 applies, since $f(n) = \Theta(n)$, and so we have the solution $T(n) = \Theta(n \lg n)$.

Exemplo 6

$$T(n) = 8T(n/2) + \Theta(n^2)$$

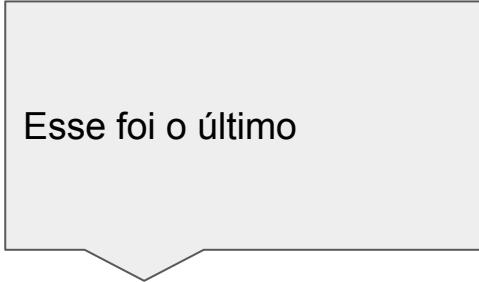


Esse aqui também

describes the running time of the first divide-and-conquer algorithm that we saw for matrix multiplication. Now we have $a = 8$, $b = 2$, and $f(n) = \Theta(n^2)$, and so $n^{\log_b a} = n^{\log_2 8} = n^3$. Since n^3 is polynomially larger than $f(n)$ (that is, $f(n) = O(n^{3-\epsilon})$ for $\epsilon = 1$), case 1 applies, and $T(n) = \Theta(n^3)$.

Exemplo 7

$$T(n) = 7T(n/2) + \Theta(n^2)$$



Esse foi o último

which describes the running time of Strassen's algorithm. Here, we have $a = 7$, $b = 2$, $f(n) = \Theta(n^2)$, and thus $n^{\log_b a} = n^{\log_2 7}$. Rewriting $\log_2 7$ as $\lg 7$ and recalling that $2.80 < \lg 7 < 2.81$, we see that $f(n) = O(n^{\lg 7 - \epsilon})$ for $\epsilon = 0.8$. Again, case 1 applies, and we have the solution $T(n) = \Theta(n^{\lg 7})$.

Exercícios

a. $T(n) = 2T(n/4) + 1.$

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

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

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

Por que o teorema mestre funciona?

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

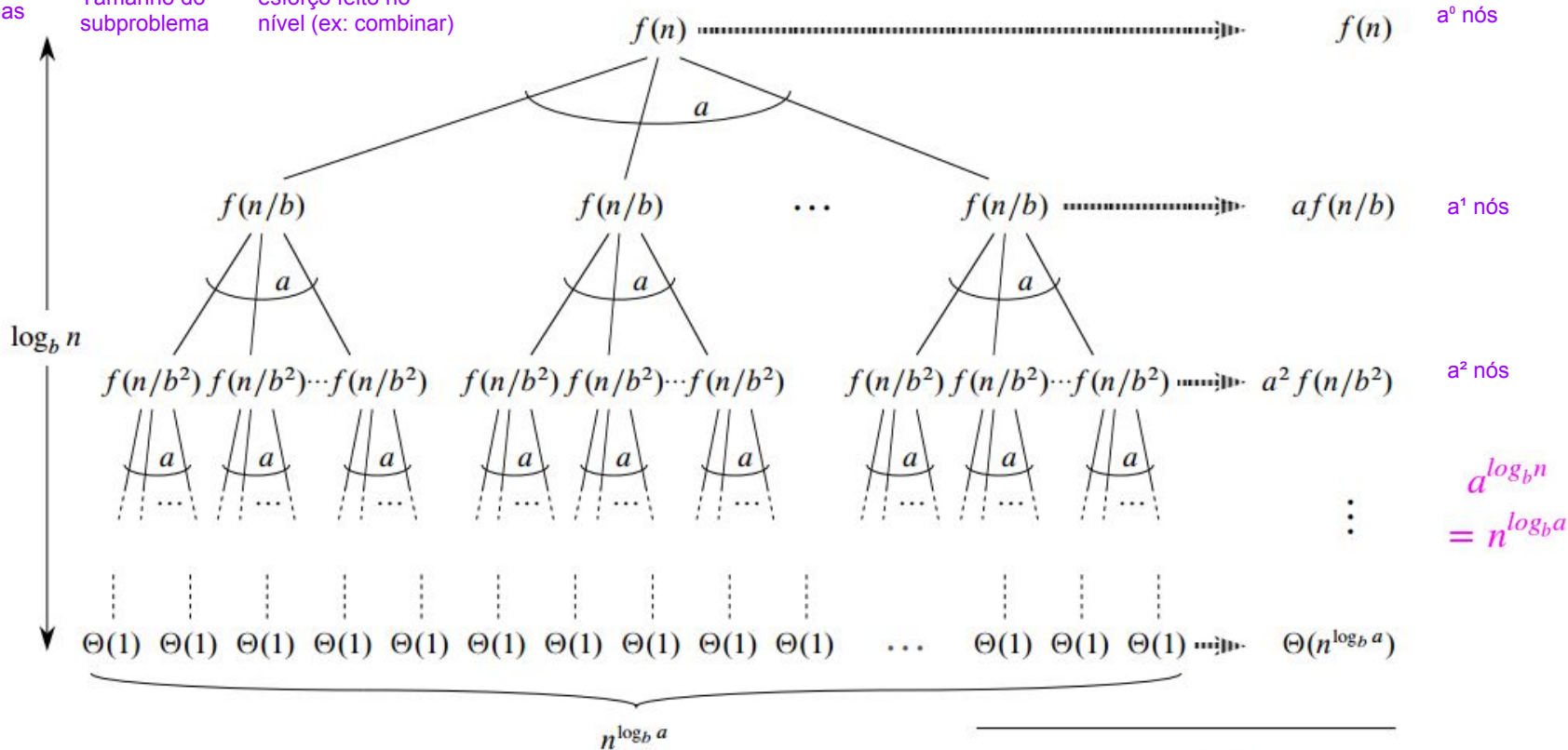
assuma que n é uma potência exata de $b > 1$

qtd de subproblemas

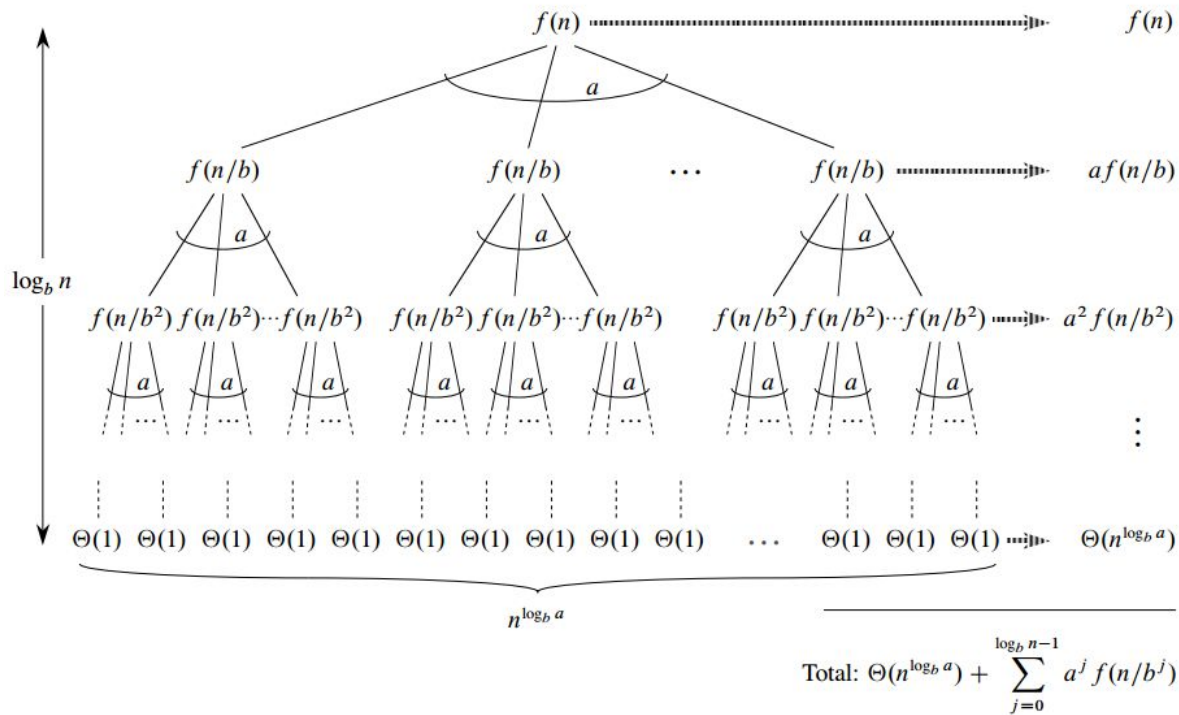
Tamanho do subproblema

esforço feito no nível (ex: combinar)

número de "quebras" do problema original



custo de todas as folhas + custo de todos os níveis intermediários --> Total: $\Theta(n^{\log_b a}) + \sum_{j=0}^{\log_b n - 1} a^j f(n/b^j)$



Caso 1: o custo é dominado pelos custos das folhas ($f(n)$ é assintoticamente menor)

Caso 2: o custo está igualmente distribuído entre os níveis da árvore

Caso 3: os custos são dominados pelo custo da raiz