

Teorema Mestre

Considere a recorrência $T(n) \leq aT(n/b) + c \cdot n^d$

O Teorema Mestre diz que

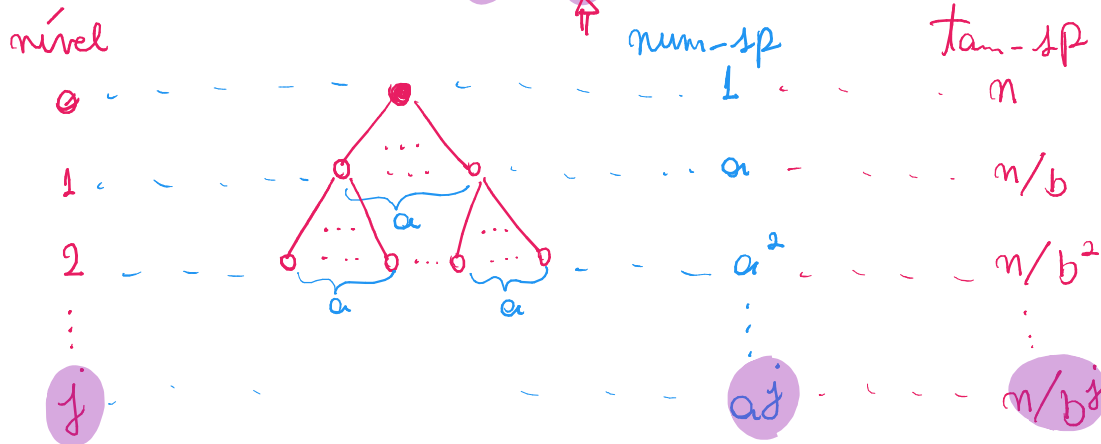
$$1 - a = b^d \Rightarrow T(n) = O(n^d \log n)$$

$$2 - a < b^d \Rightarrow T(n) = O(n^d)$$

$$3 - a > b^d \Rightarrow T(n) = O(n^{\log_b a})$$

Vamos entender o porquê construindo uma árvore de recursão

- para a recorrência $T(n) \leq aT(n/b) + c \cdot n^d$



Recorrência $T(n) \leq a \cdot T(n/b) + c \cdot n^d$

Qual o número de subproblemas no nível j da árvore? $\text{num-sp}(j) = a^j$

Qual o tamanho de um subproblema do nível j ? $\text{tam-sp}(j) = n/b^j$

Qual o trabalho local (não recursivo) em um problema no nível j ?

R.: $\text{trab-sp}(j) \leq c \cdot \text{tam-sp}(j)^d = c \cdot (n/b^j)^d$

Qual o trabalho realizado no nível j ?

$$\begin{aligned} \text{trab}(j) &= \text{num-sp}(j) \cdot \text{trab-sp}(j) \\ &= a^j \cdot c \cdot (n/b^j)^d = c \cdot \frac{a^j \cdot n^d}{b^{dj}} = c \cdot n^d \cdot \left(\frac{a}{b^d}\right)^j \end{aligned}$$

Note que, só o termo a/b^d é influenciado pelo nível j . Por isso,

- interpretamos a como a taxa com que o trabalho aumenta por nível e
- b^d como a taxa com que o trabalho diminui por nível da árvore.

Quantos níveis tem a árvore de recursão?

- Sabemos que no último nível h o tamanho do subproblema é 1,
 - ou algum número pequeno que corresponda ao caso base.
- Assim,

$$T_{\text{sub}}(h) = \frac{n}{b^h} = 1 \Rightarrow b^h = n \Rightarrow h = \log_b n$$

Por fim, qual o trabalho total realizado?

- Vamos somar ao longo de todos os níveis da árvore

$$T(n) = \sum_{j=0}^h T_{\text{sub}}(j) = \sum_{j=0}^h c \cdot n^d \left(\frac{a}{b^d}\right)^j = c \cdot n^d \sum_{j=0}^{\log_b n} \left(\frac{a}{b^d}\right)^j$$

Chegamos à soma dos termos de uma PG com razão

- e sabemos que o resultado de tal soma
 - depende particularmente da razão ser maior ou menor que 1.
- É daí que surgem os casos do Teorema Mestre.

Casos do Teorema Mestre

Lembrando que trabalho total é dado por

- nossos casos dependem do valor da razão (a / b^d) .

$$T(n) \leq c \cdot n^d \cdot \sum_{j=0}^{\log_b n} \left(\frac{a}{b^d} \right)^j$$

$$\log_b^n = \frac{\log_{10}^n}{\log_{10} b}$$

Caso 1: $a = b^d \Rightarrow \frac{a}{b^d} = 1 \Rightarrow$ trab. uniforme por nível

$$T(n) = c \cdot n^d \cdot \sum_{j=0}^{\log_b n} \underbrace{\left(\frac{a}{b^d} \right)}_1^j = c \cdot n^d (\log_b n + 1) = O(n^d \log n)$$

- $T(n)$ é # de níveis vezes trabalho por nível

Caso 2: $a < b^d \Rightarrow \frac{a}{b^d} < 1 \Rightarrow$ trab. diminui por nível

$$T(n) = c \cdot n^d \cdot \sum_{j=0}^{\log_b n} \left(\frac{a}{b^d} \right)^j \leq c \cdot n^d \frac{1}{1 - \frac{a}{b^d}} = \underbrace{\frac{c}{1 - \frac{a}{b^d}}}_{\text{constante}} \cdot n^d = O(n^d)$$

- $T(n)$ é dominado pelo trabalho no primeiro nível da árvore

Caso 3: $a > b^d \Rightarrow \frac{a}{b^d} > 1 \Rightarrow$ Matr. aumenta por nível

$$T(n) \leq a T(n/b) + c \cdot n^d$$

$$\begin{aligned} T(n) &\leq c \cdot n^d \cdot \sum_{j=0}^{\log_b n} \left(\frac{a}{b^d}\right)^j \leq c \cdot n^d \cdot \frac{\left(\frac{a}{b^d}\right)^{\log_b n} - 1}{\left(\frac{a}{b^d}\right) - 1} \\ &\leq \frac{c \cdot n^d \cdot \left(\frac{a}{b^d}\right) \cdot a^{\log_b n}}{(b^d)^{\log_b n} (a/b^d - 1)} = \frac{c \cdot n^d \cdot (a/b^d) \cdot a^{\log_b n}}{n^d (a/b^d - 1)} \\ &= \frac{c(a/b^d)}{(a/b^d - 1)} \cdot n^{\log_b a} = O(n^{\log_b a}) \end{aligned}$$

Se $q > 1$

Temos

$$\sum_{j=0}^k q^j = \frac{q^{k+1} - 1}{q - 1}$$

- $T(n) = O(a^{\log_b n})$ é o número de folhas no último nível da árvore.
 - Assim, $T(n)$ é dominado pelo trabalho no último nível da árvore.
- Para verificar que $a^{\log_b n} = n^{\log_b a}$
 - aplique a operação $\log_b$ aos dois lados da equação.

$$\begin{aligned} \overline{a^{\log_b n}} &= \overline{n^{\log_b a}} \\ \log_b a^{\log_b n} &= \log_b n^{\log_b a} \end{aligned}$$

$$\log_b n \cdot \log_b a = \log_b a \cdot \log_b n$$

Curiosidade: existe uma versão mais poderosa (e complicada) do Teorema Mestre,

- para lidar com recorrências em que o trabalho local não é um polinômio.
 - Desafio: troque $c n^d$ por $f(n)$ na recorrência e refaça árvore e análise.

Aplicando o Teorema Mestre

Considerando a recorrência $T(n) \leq a \cdot T(n/b) + c \cdot n^d$, o Teorema Mestre diz que

1. $a = b^d \rightarrow T(n) = O(n^d \log n)$
2. $a < b^d \rightarrow T(n) = O(n^d)$
3. $a > b^d \rightarrow T(n) = O(n^{\log_b a})$

Algoritmo de Multiplicação de Inteiros de Karatsuba

- $T(n) = 3 T(n/2) + c n$

Handwritten notes for Karatsuba:
// a // b // d
 $a=3$
 $b=2$
 $d=1$
 $a=3 > 2^1 = b^d \Rightarrow T(n) = O(n^{\log_2 3}) \approx O(n^{1.585})$
Case 3
melhor que n^2

Algoritmo de Ordenação mergeSort e Algoritmo para Encontrar o Par Mais Próximo

- $T(n) = 2 T(n/2) + c n$

Handwritten notes for mergeSort:
// a // b // d
 $a=2$
 $b=2$
 $d=1$
 $a=2 = 2^1 = b^d \Rightarrow T(n) = O(n \log n)$
Case 1

Algoritmo de Multiplicação de Matrizes de Strassen

- $T(n) = 7 T(n/2) + c n^2$

Handwritten notes for Strassen:
// a // b // d
 $a=7$
 $b=2$
 $c=2$
 $a=7 > 2^2 = b^d \Rightarrow T(n) = O(n^{\log_2 7}) \approx O(n^{2.81})$
Case 3
melhor que n^3