El Método Maestro (Master Method)

El método maestro es una manera para estimar funciones a partir de desigualdades recursivas. Es muy útil para estimar el tiempo de ejecución de algoritmos "divide y conquista" como Strassen y Karatsuba.

Teorema [Master method]

Suponga que $T(n)$ satisface

$$T(n) \leq a\, T\left(\frac{n}{b}\right) + O(n^d)$$

para algunos números $a \geq 1$, $b > 1$, $d \geq 0$. Entonces

$$T(n) \sim \begin{cases} O(n^d \log(n)), & \text{si } a = b^d \\ O(n^d), & \text{si } a < b^d \\ O(n^{\log_b(a)}), & \text{si } a > b^d \end{cases}$$

Parámetros:

a = # llamadas recursivas

b = factor por el que reducimos el input

d = orden del trabajo hecho afuera de llamadas recursivas.

n longitud del input $= n$

Ejemplos:

$$\underbrace{\overset{n}{\widetilde{ab}}}_{\text{dígitos}} \times \overset{n}{\widetilde{cd}} = (10^{\frac{n}{2}} a + b)(10^{\frac{n}{2}} c + d)$$

$$= 10^n a\cdot c + 10^{\frac{n}{2}}(a\cdot d + b\cdot c) + b\cdot d$$

Multiplicación Recursiva:

$$T(n) \leq \underset{a}{4}\, T\left(\frac{n}{\underset{b}{2}}\right) + O(n^1) \qquad d = 1$$

$$\Rightarrow T(n) = O(n^{\log_2(4)}) = O(n^2)$$

a vs b^d $\Leftrightarrow$ 4 vs 2^1 $\Rightarrow$ $4 > 2^1$

Karatsuba:

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

mucho mjor!

$$a \text{ vs } b^d \iff 3 \overset{>}{\text{vs}} 2^1 \Rightarrow T(n) = O\left(n^{\log_2(3)}\right) = O\left(n^{1.59}\right)$$

Ficticio:

$$T(n) \leq 2T\left(\frac{n}{2}\right) + O(n^2)$$

$$a \text{ vs } b^d \iff 2 \underset{<}{\text{vs}} 2^2 \Rightarrow T(n) = O(n^2)$$

Multiplicación de matrices: (de matrices $n \times n$)

Recursiva:

$$T(n) \leq \underset{\substack{\shortparallel \\ a}}{8}\, T\left(\frac{n}{\underset{\substack{\shortparallel \\ b}}{2}}\right) + O(n^{2}) \quad d$$

$$a \text{ vs } b^d \iff 8 \text{ vs } 2^2 = 8 \underset{>}{\text{vs}} 4$$

$$T(n) = O\left(n^{\log_2(8)}\right) = O(n^3)$$

igual que la no recursiva!

Strassen:

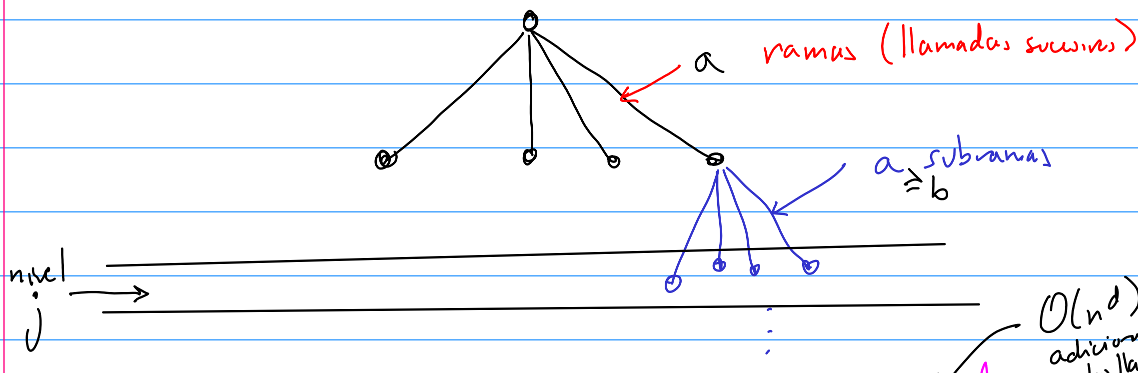
$$T(n) \leq 7T\left(\frac{n}{2}\right) + O(n^2)$$

$$7 \text{ vs } 2^2 = 7 > 4$$

$$T(n) = O\left(n^{\log_2(7)}\right) = O(n^{2.8})$$

exponent of Matrix multiplication. Estimar este número es un problema abierto de mucho interés...

Dem: En una recursión podemos usar un árbol para imaginar y contar el trabajo realizado



$O(n^d)$ adicional a las llamadas recursivas

Trabajo en nivel j (en el merge) $= \underbrace{a^j}_{\text{número de problemas}} \cdot c\Big(\underbrace{\frac{n}{b^j}}_{\text{tamaño del input}}\Big)^d$

$$= cn^d \left(\frac{a}{b^d}\right)^j$$

luego total satisface

$$T(n) \le \sum_{j=0}^{L} cn^d \left(\frac{a}{b^d}\right)^j$$

¿Qué es L? $L = \log_b(n)$

$b^L = n$

si partes cada paso en b partes.

Concluimos

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

Hay 3 posibilidades:

(i) Trabajo igual en cada nivel $\frac{a}{b^d} = 1$

$T(n) \le c\, n^d \log_b(n)$ ✓

(ii) Tenemos que estudiar la serie geométrica

$$1 + r + r^2 + \dots + r^{k-1} = S$$

$$rS - S = r^k - 1$$

$$S = \frac{r^k - 1}{r-1} \le \frac{r^k}{r-1} = r^{k-1}\left(\frac{r}{r-1}\right)$$

Si $r > 1$ entonces $S \le r^{k-1}$ (último término)

Si $r < 1$ entonces $S \le \frac{1}{1-r}$ (acotada por una constante)

$$T(n) \le C n^d \overbrace{\sum_{k=0}^{\log_b(n)} \left(\frac{a}{b^d}\right)^i}^{S}$$

Si $\frac{a}{b^d} > 1 \Rightarrow T(n) \le C n^d \left(\frac{a}{b^d}\right)^{\log_b(n)}$

$$= O\left(a^{\log_b(n)}\right).$$

Si $\frac{a}{b^d} < 1 \Rightarrow T(n) \le C n^d \cdot K = O(n^d)$

∎