

Fundamentos de análisis y diseño de algoritmos

Recurrencias

Método de iteración

Método maestro*

Método de sustitución

Análisis de algoritmos recursivos

Recurrencias

Método de iteración

Expandir la recurrencia y expresarla como una suma de términos que dependen de n y de las condiciones iniciales

Recurrencias

$$T(n) = n + 3T(\lfloor n/4 \rfloor), \quad T(1) = \Theta(1) \leftarrow \text{Constante}$$

Expandir la recurrencia 2 veces

$$T(n/4) = n/4 + 3T(n/16)$$

$$T(n) = n + \frac{3n}{4} + 3^2 T(n/4^2) \quad \text{Primera expansión}$$

$$T(n/4^2) = \frac{n}{4^2} + 3T(n/4^3)$$

$$T(n) = n + \frac{3n}{4} + \frac{3^2 n}{4^2} + 3^3 T(n/4^3) \quad \text{Segunda expansión}$$

Recurrencias

$$T(n) = n + 3T(\lfloor n/4 \rfloor)$$

$$n + 3 (\lfloor n/4 \rfloor + 3T(\lfloor n/16 \rfloor))$$

$$n + 3 (\lfloor n/4 \rfloor + 3(\lfloor n/16 \rfloor + 3T(\lfloor n/64 \rfloor)))$$

$$n + 3^* \lfloor n/4 \rfloor + 3^{2*} \lfloor n/16 \rfloor + 3^3 T(\lfloor n/64 \rfloor)$$

Recurrencias

$$T(n) = n + 3T(\lfloor n/4 \rfloor)$$

$$n + 3 (\lfloor n/4 \rfloor + 3T(\lfloor n/16 \rfloor))$$

$$n + 3 (\lfloor n/4 \rfloor + 3(\lfloor n/16 \rfloor + 3T(\lfloor n/64 \rfloor)))$$

$$n + 3^*\lfloor n/4 \rfloor + 3^2*\lfloor n/16 \rfloor + 3^3T(\lfloor n/64 \rfloor)$$

¿Cuándo se detienen las iteraciones?

Recurrencias

$$T(n) = n + 3T(\lfloor n/4 \rfloor)$$

$$n + 3(\lfloor n/4 \rfloor + 3T(\lfloor n/16 \rfloor))$$

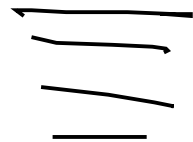
$$n + 3(\lfloor n/4 \rfloor + 3(\lfloor n/16 \rfloor + 3T(\lfloor n/64 \rfloor)))$$

$$n + 3\lfloor n/4 \rfloor + 3^2\lfloor n/16 \rfloor + \underline{3^3T(\lfloor n/64 \rfloor)}$$

$$3^3T(\lfloor n/4^3 \rfloor)$$

¿Cuándo se detienen las iteraciones?

Cuando se llega a $T(1)$



Condición inicial

Recurrencias

$$T(n) = n + 3T(\lfloor n/4 \rfloor)$$

$$n + 3(\lfloor n/4 \rfloor + 3T(\lfloor n/16 \rfloor))$$

$$n + 3(\lfloor n/4 \rfloor + 3(\lfloor n/16 \rfloor + 3T(\lfloor n/64 \rfloor)))$$

$$n + 3*\lfloor n/4 \rfloor + 3^2*\lfloor n/4^2 \rfloor + 3^3T(\lfloor n/4^3 \rfloor)$$

¿Cuándo se detienen las iteraciones?

Cuando se llega a $T(1)$, esto es, cuando $(n/4^i) = 1$

Expansión
i veces

$$\frac{n}{4^i} = 1 \quad n = 4^i \quad \log_4(n) = i \quad n/4^i = 1$$

Recurrencias

$$3^{\log_4 n} T(1)$$

$$T(n) = n + 3T(\lfloor n/4 \rfloor)$$

$$n + 3(\lfloor n/4 \rfloor + 3T(\lfloor n/16 \rfloor))$$

$$n + 3(\lfloor n/4 \rfloor + 3(\lfloor n/16 \rfloor + 3T(\lfloor n/64 \rfloor)))$$

$$n + 3\lfloor n/4 \rfloor + 3^2\lfloor n/4^2 \rfloor + 3^3\lfloor n/4^3 \rfloor + \dots + 3^{\log_4 n} T(1)$$

$$n + 3\lfloor n/4 \rfloor + 3^2\lfloor n/4^2 \rfloor + 3^3\lfloor n/4^3 \rfloor + 3^4\lfloor n/4^4 \rfloor + \dots + 3^i T(1)$$

¿Cuándo se detienen las iteraciones?

Cuando se llega a $T(1)$, esto es, cuando $(n/4^i)=1$

Recurrencias

$$T(n) = n + 3T(\lfloor n/4 \rfloor)$$

$$n + 3 (\lfloor n/4 \rfloor + 3T(\lfloor n/16 \rfloor))$$

$$n + 3 (\lfloor n/4 \rfloor + 3(\lfloor n/16 \rfloor + 3T(\lfloor n/64 \rfloor)))$$

$$n + 3^*\lfloor n/4 \rfloor + 3^{2*}\lfloor n/4^2 \rfloor + 3^3(\lfloor n/4^3 \rfloor) + \dots + 3^{\log_4 n} \Theta(1)$$

Después de iterar, se debe tratar de expresar como una sumatoria con forma cerrada conocida

Recurrencias

$$T(n) = n + 3T(\lfloor n/4 \rfloor)$$

$$n + 3 (\lfloor n/4 \rfloor + 3T(\lfloor n/16 \rfloor))$$

$$n + 3 (\lfloor n/4 \rfloor + 3(\lfloor n/16 \rfloor + 3T(\lfloor n/64 \rfloor)))$$

$$n + 3^*\lfloor n/4 \rfloor + 3^{2*}\lfloor n/4^2 \rfloor + 3^3(\lfloor n/4^3 \rfloor) + \dots + 3^{\log_4 n} \Theta(1)$$

$$\leq n + 3n/4 + 3^2n/4^2 + 3^3n/4^3 + \dots + 3^{\log_4 n} \Theta(1)$$

Recurrencias

$$T(n) = n + 3T(n/4)$$

$$n + 3 (n/4 + 3T(n/16))$$

$$n + 3 (n/4 + 3(n/16 + 3T(n/64)))$$

$$n + 3*n/4 + 3^2*n/4^2 + 3^3(n/4^3) + \dots + 3^{\log_4 n} \Theta(1)$$

$$\leq n + 3n/4 + 3^2n/4^2 + 3^3n/4^3 + \dots + 3^{\log_4 n} \Theta(1)$$

Recurrencias

$$T(n) = n + 3T(n/4)$$

$$n + 3 \left(n/4 + 3T(n/16) \right)$$

$$n + 3 \left(n/4 + 3(n/16 + 3T(n/64)) \right)$$

$$n + 3*n/4 + 3^2*n/4^2 + 3^3(n/4^3) + \dots + 3^{\log_4 n} \Theta(1)$$

$$\leq n + 3n/4 + 3^2n/4^2 + 3^3n/4^3 + \dots + 3^{\log_4 n} \Theta(1)$$

$$= n \sum_{i=0}^{\log_4 n} \left(\frac{3}{4} \right)^i + 3^{\log_4 n} \Theta(1)$$

$$= n \left(\frac{\left(\frac{3}{4} \right)^{(\log_4 n)+1} - 1}{\left(\frac{3}{4} \right) - 1} \right) + n^{\log_4 3} = n * 4 \left(1 - \left(\frac{3}{4} \right)^{(\log_4 n)+1} \right) + \Theta(n^{\log_4 3})$$

$$= O(n)$$

Recurrencias

Resuelva por el método de iteración

$$T(n) = 2T(n/2) + 1, T(1) = \Theta(1)$$

$\frac{1}{2} \times n$

expandir

$$T(n/2) = 2T(n/2^2) + 1$$

$$T(n) = 2(2T(n/2^2) + 1) + 1$$

$$1ra \quad T(n) = 2^2 T(n/2^2) + 2 + 1$$

$$T(n/2^2) = 2T(n/2^2) + 1$$

$$T(n) = 2^2 (2T(n/2^3) + 1) + 2 + 1$$

2 da $T(n) = 2^3 T(n/2^3) + 2^2 + 2 + 1$

$\left(\begin{array}{l} \text{1} = \frac{n}{2^i} \end{array} \right. \nearrow$
 $2^i = n \quad i = \log_2(n)$

$$T(n) = 2^i T(n/2^i) + 2^{i-1} + 2^{i-2} + \dots + 2^0$$

$$T(n) = 2^i T(n/2^i) + \sum_{p=0}^{i-1} 2^p$$

$$T(n) = 2^{\log_2 n} T(1) + \sum_{p=0}^{\log_2(n)-1} 2^p$$

$$T(n) = Cn + \frac{2^{\log_2(n)} - 1}{1}$$

$$\sum_{k=0}^n ar^k = \frac{ar^{n+1} - ar}{r-1}$$

$$T(n) = Cn + n - 1$$

$$T(n) = O(n)$$

Recurrencias

Resuelva por el método de iteración

$$T(n) = 2T(n/2) + 1, T(1) = \Theta(1) \quad \checkmark$$

$$T(n) = 2T(n/2) + n, T(1) = \Theta(1)$$

$$T(n) = 2T(n/2) + n, \quad T(1) = \Theta(1)$$

$$T(n/2) = 2T(n/4) + \frac{n}{2}$$

$$T(n) = 2^2 T(n/4) + \frac{2n}{2} + n$$

$$T(n/4) = 2T(n/8) + \frac{n}{2^2}$$

$$T(n) = 2^2 \times 2T(n/8) + \underbrace{\frac{2^2 n}{2^2} + \frac{2^1 n}{2^1} + n}_{i=1 \rightarrow 0}$$

$$T(n) = 2^i T(n/2^i) + \underbrace{n + n + \dots + n + n}_{i-1}$$

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

$$1 = \frac{n}{2^i} \rightarrow i = \log_2(n)$$

$$T(n) = 2^{\log_2(n)} T(1) + \sum_{p=0}^{\log_2(n)-1} n$$

$$T(n) = Cn + \log_2(n)n - n$$

$$\boxed{\Theta(n \log_2(n))}$$

$$\sum_{i=0}^n C = Cn$$

Recurrencias

Resuelva por el método de iteración

$$T(n) = 2T(n/2) + 1, T(1) = \Theta(1) \quad \checkmark \quad O(n)$$

$$T(n) = 2T(n/2) + n, T(1) = \Theta(1) \quad \checkmark \quad O(n \log_2(n))$$

$$T(n) = T(\lceil n/2 \rceil) + 1, T(1) = \Theta(1)$$

$$T(n) = T(n/2) + 1^0, \quad T(1) = \Theta(1)$$

$$T(n) = T(n/2^2) + 1^1 + 1^0$$

$$T(n) = T(n/2^3) + 1^2 + 1^1 + 1^0$$

$$T(n) = T(n/2^4) + 1^3 + 1^2 + 1^1 + 1^0$$

$$T(n) = T(n/2^i) + 1^{i-1} + 1^{i-2} + \dots + 1^0$$

$$T(n) = T(n/2^i) + \sum_{j=0}^{i-1} 1$$

$$\frac{n}{2^i} = 1 \quad i = \log_2(n)$$

$$T(n) = C + \sum_{p=0}^{\log_2(n)-1} 1$$

$$\log_a b = \frac{\log_x b}{\log_x a}$$

$$T(n) = C + \sum_{p=1}^{\log(n)-1} 1 + 1$$

$$\log_a b = K \log_x b$$

$$T(n) = C + \log_2(n) - 1 + 1$$

$$T(n) = C + \log_2(n)$$

$$T(n) = O(\log_2(n))$$


Recurrencias

Resuelva por el método de iteración

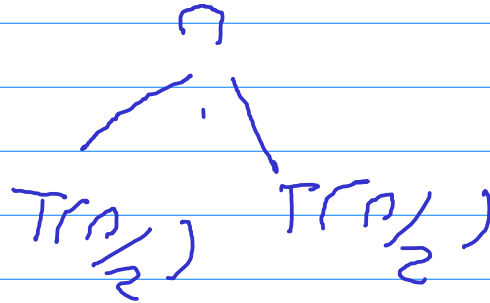
$$T(n) = 2T(n/2) + 1, T(1) = \Theta(1)$$

$$T(n) = 2T(n/2) + n, T(1) = \Theta(1) \quad \checkmark$$

$$T(n) = T(\lceil n/2 \rceil) + 1, T(1) = \Theta(1)$$

Demuestre que $T(n) = T(n/2) + n$, es $\Omega(n \log n)$ 

$$\overline{I}(n) = 2 \operatorname{Tr}(\rho_{1/2}) + n$$



$$i = 0$$

$$n \rightarrow \frac{2^0 n}{2^0}$$

$$T(1) = \Theta(1)$$

$$\log_2(n)$$

$$i = 1$$

$$\frac{n}{2}$$

$$\frac{n}{2}$$

$$\rightarrow \frac{2^1 n}{2^1}$$

$$+ 1$$

$$i = 2$$

$$T\left(\frac{n}{4}\right)$$

$$T\left(\frac{n}{4}\right)$$

$$T\left(\frac{n}{4}\right)$$

$$T\left(\frac{n}{4}\right) \leftarrow \frac{4n}{2^2}$$

⋮

$$i = \log_2(n) \quad T(1)$$

$$\frac{n}{2^i} = 1 \rightarrow i = \log_2(n)$$

$$T(n) = \sum_{i=0}^{\log_2(n)} \frac{2^i n}{2^i} = \sum_{i=0}^{\log_2(n)} n = \log_2(n)n + n$$

$$\Theta(n \log(n))$$

$$T(n) = T\left(\frac{n}{2}\right) + n \quad \Omega(n \log n)$$

$$\begin{array}{c} i=0 \quad n \\ | \\ i=1 \quad T\left(\frac{n}{2}\right) \end{array}$$

$$\sum_{k=0}^n ar^k = ar^{n+1} - a \quad \rightarrow \quad \begin{array}{c} i=0 \quad n \\ | \\ i=1 \quad \frac{n}{2} \\ | \\ i=2 \quad T\left(\frac{n}{4}\right) \end{array} \quad \begin{array}{c} i=0 \quad n \\ | \\ i=1 \quad \frac{n}{2^2} \\ | \\ i=2 \quad \frac{n}{4} = \frac{n}{2^2} \\ | \\ i=3 \quad T\left(\frac{n}{8}\right) \end{array}$$

$$\frac{n}{2^i} = 1 \quad i = \log_2(n)$$

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

$$T(n) = \left(\frac{\left(\frac{1}{2}\right)^{\log_2(n)} - 1}{-\frac{1}{2}} \right) n$$

$$\left(\left(\frac{1}{2}\right)^{\log_2(n)} - 1 \right) n + 2n$$

$$T(n) = -n \left(\frac{1}{2}\right)^{\log_2(n)} + 2n$$

$$\log_2(2^3)$$

$$T(n) = -n \left(2^{-1}\right)^{\log_2(n)} + 2n$$

$$3 \log_2(2)$$

$$T(n) = -n 2^{\log_2\left(\frac{1}{n}\right)} + 2n$$

$$T(n) = -n \frac{1}{n} + 2n = 2n - 1$$

$$c \times n \log(n) \leq 2n - 1$$


La afirmación que $T(n)$ es $\mathcal{O}(n \log(n))$ NO ES CIERTA

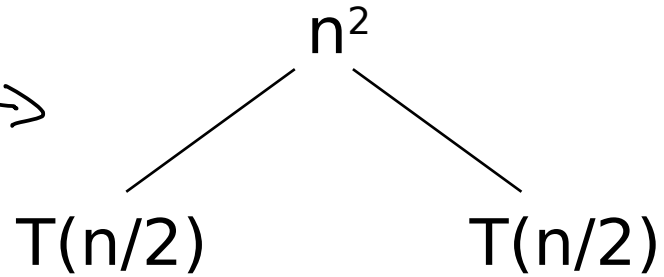
Recurrencias

Iteración con árboles de recursión

$$T(n) = 2T(n/2) + n^2$$

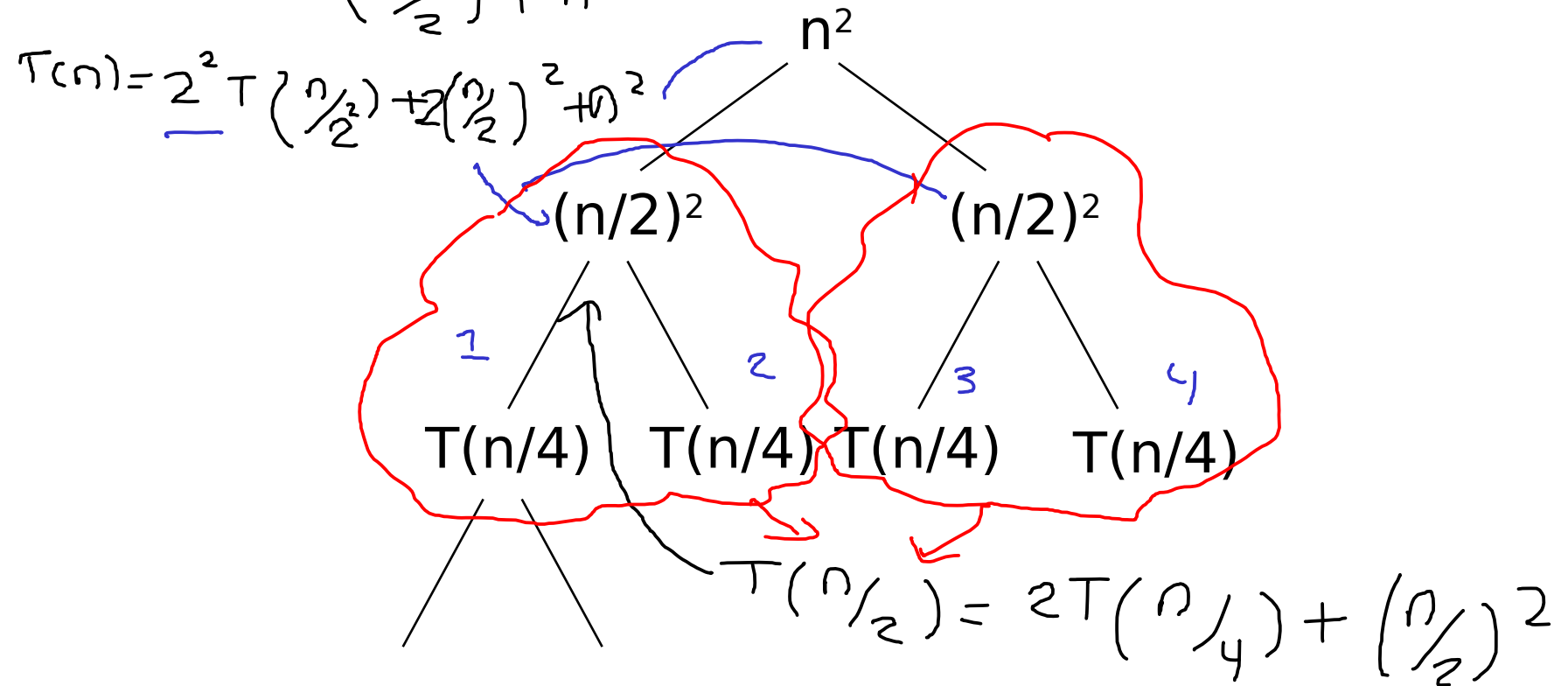
Recurrencias

$$2T(n/2) + n^2$$




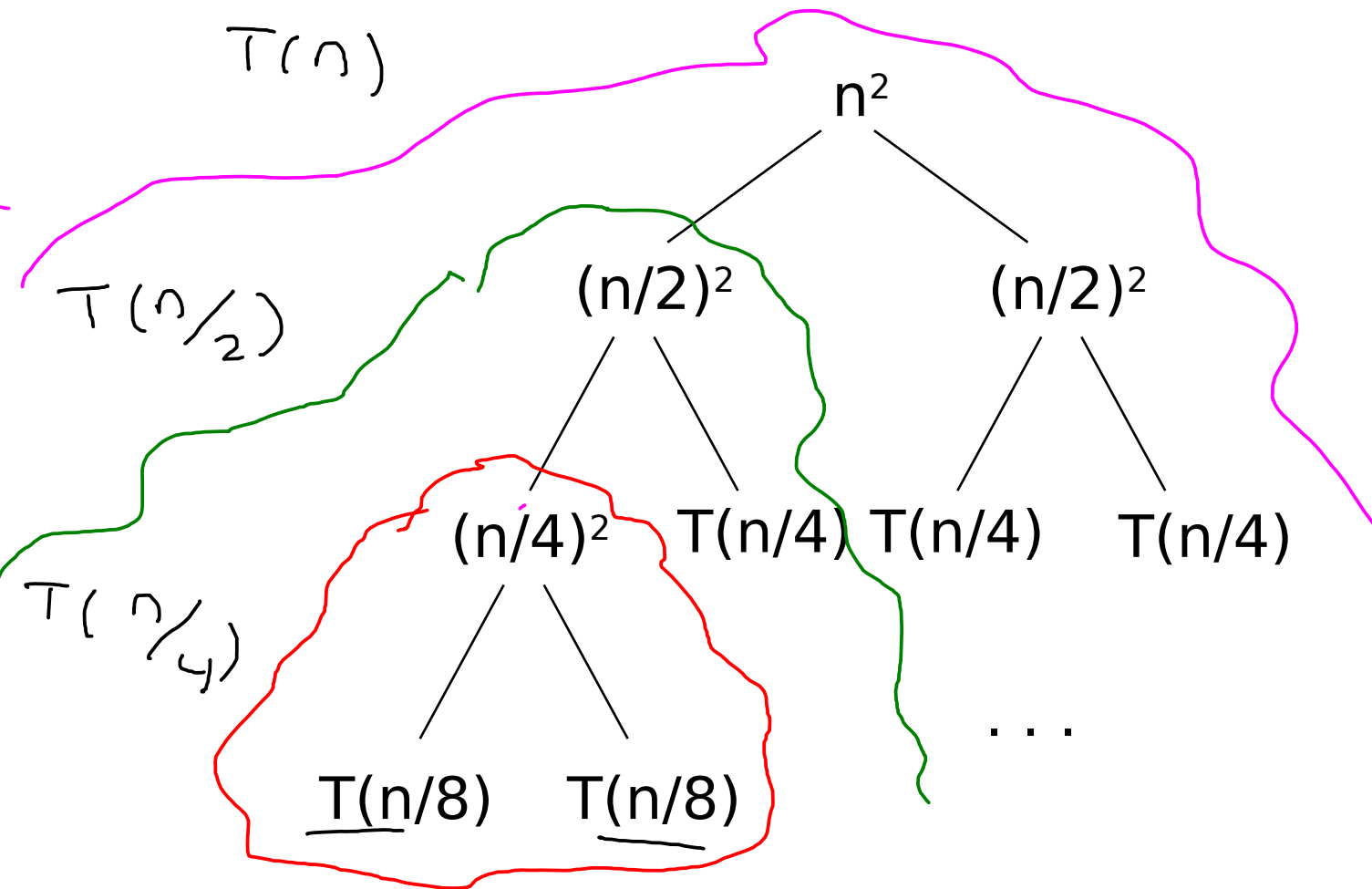
Recurrencias

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

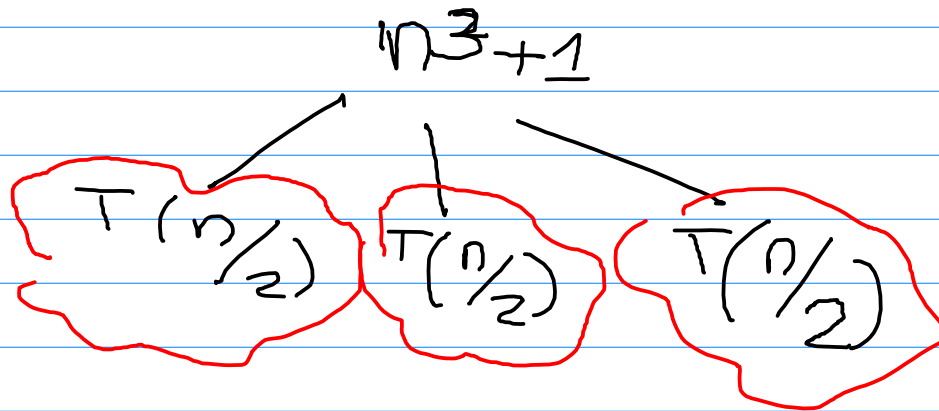


$$T\left(\frac{n}{4}\right) = 2T\left(\frac{n}{8}\right) + \left(\frac{n}{4}\right)^2$$

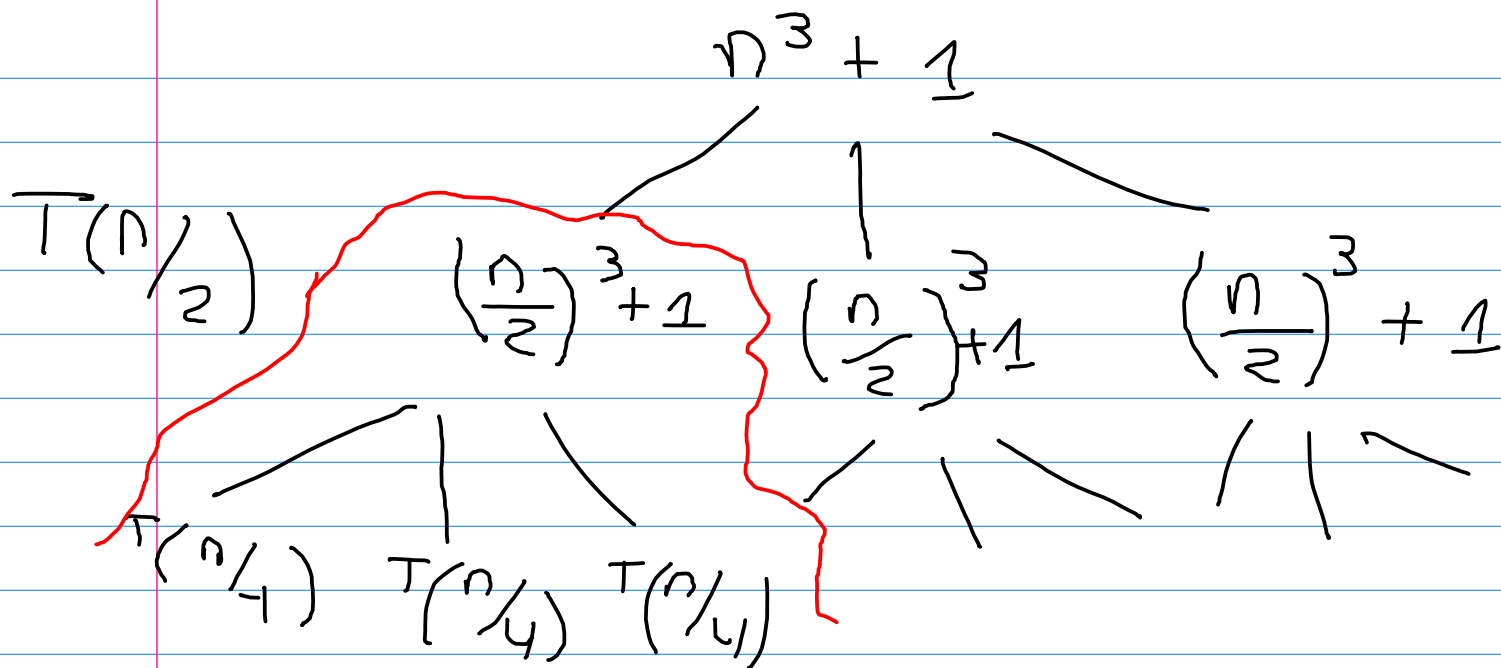
Recurrencias



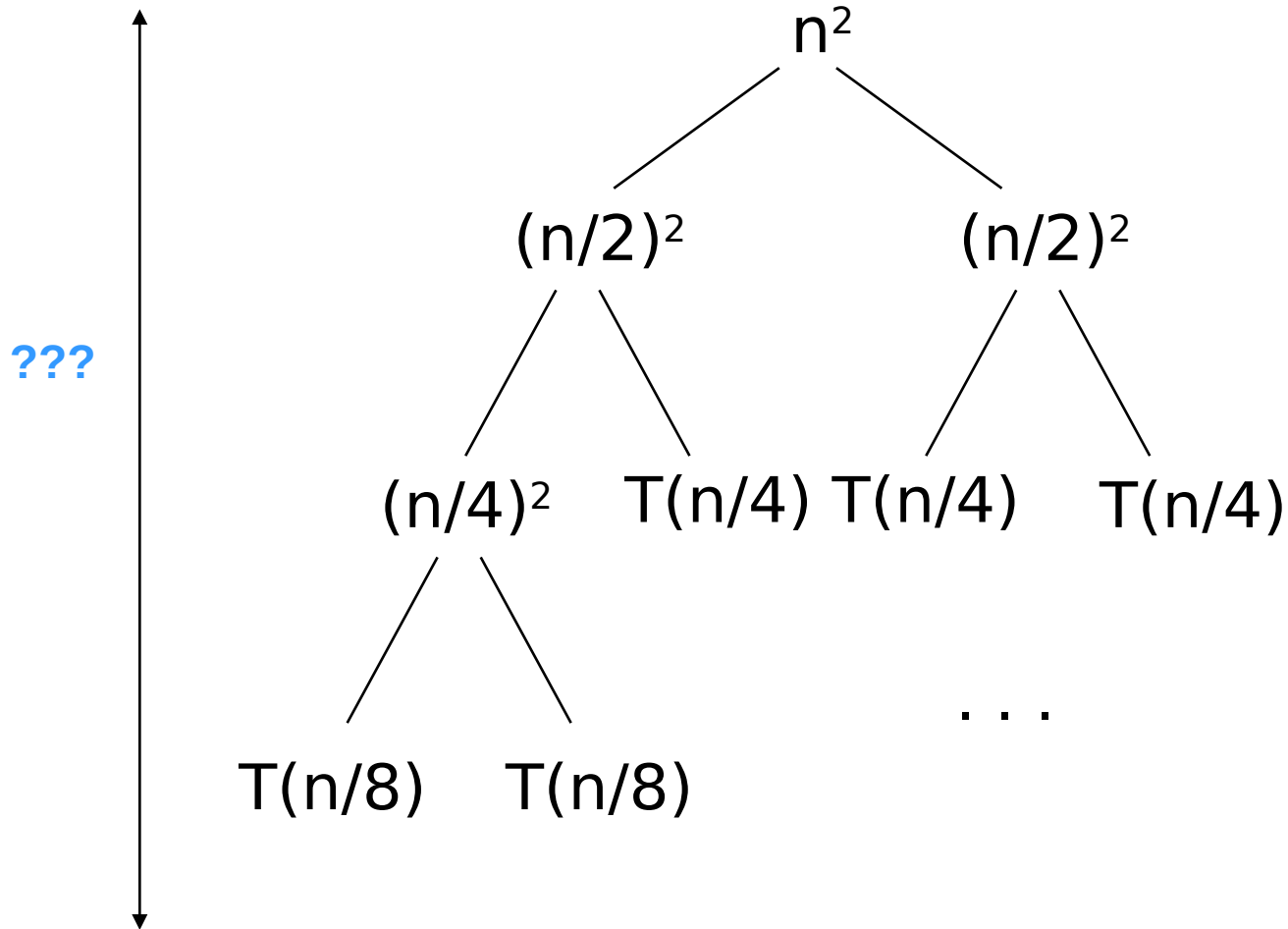
$$T(n) = 3T\left(\frac{n}{2}\right) + n^3 + 1$$



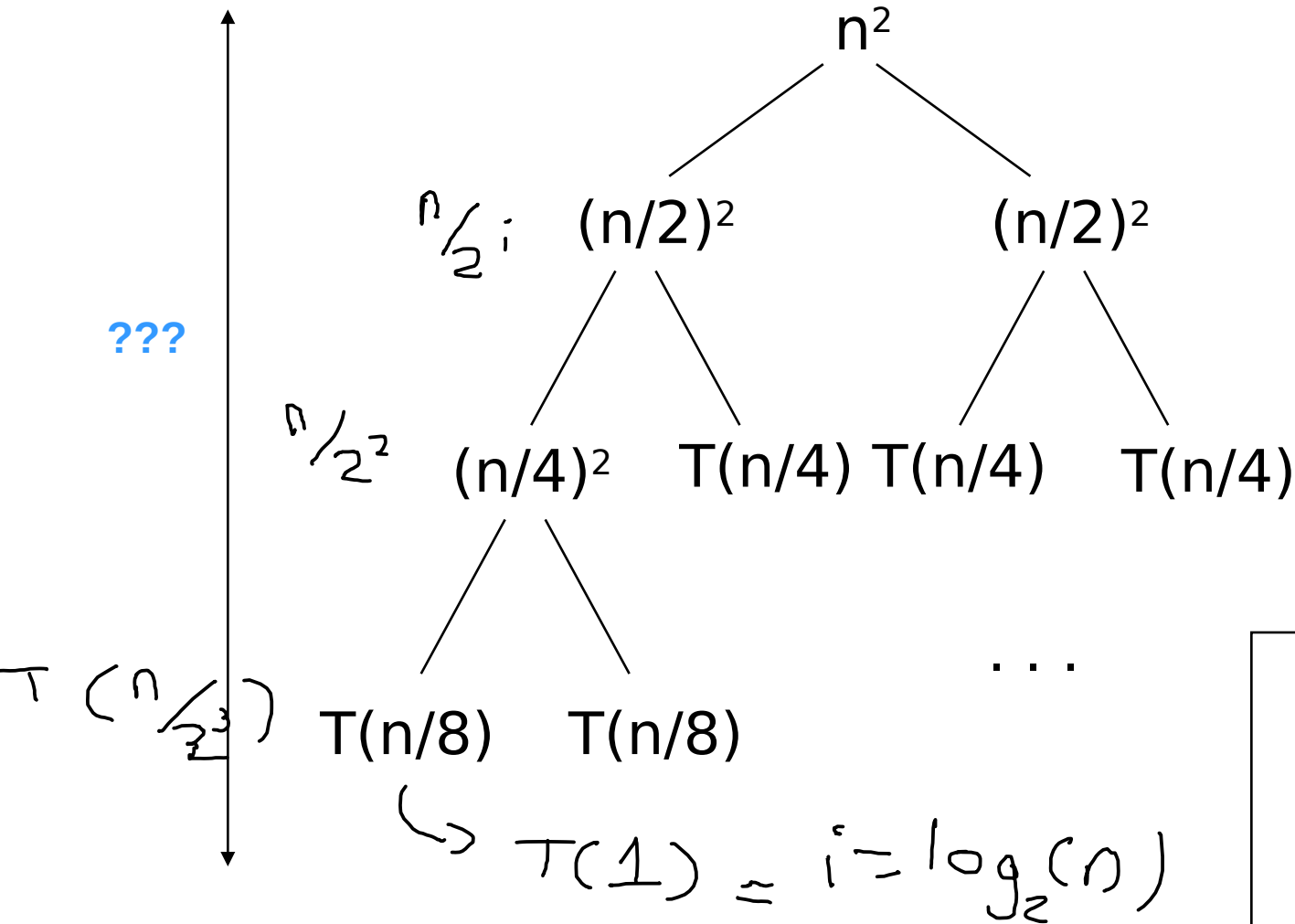
$$T\left(\frac{n}{2}\right) = 3T\left(\frac{n}{4}\right) + \left(\frac{n}{2}\right)^3 + 1$$



Recurrencias



Recurrencias



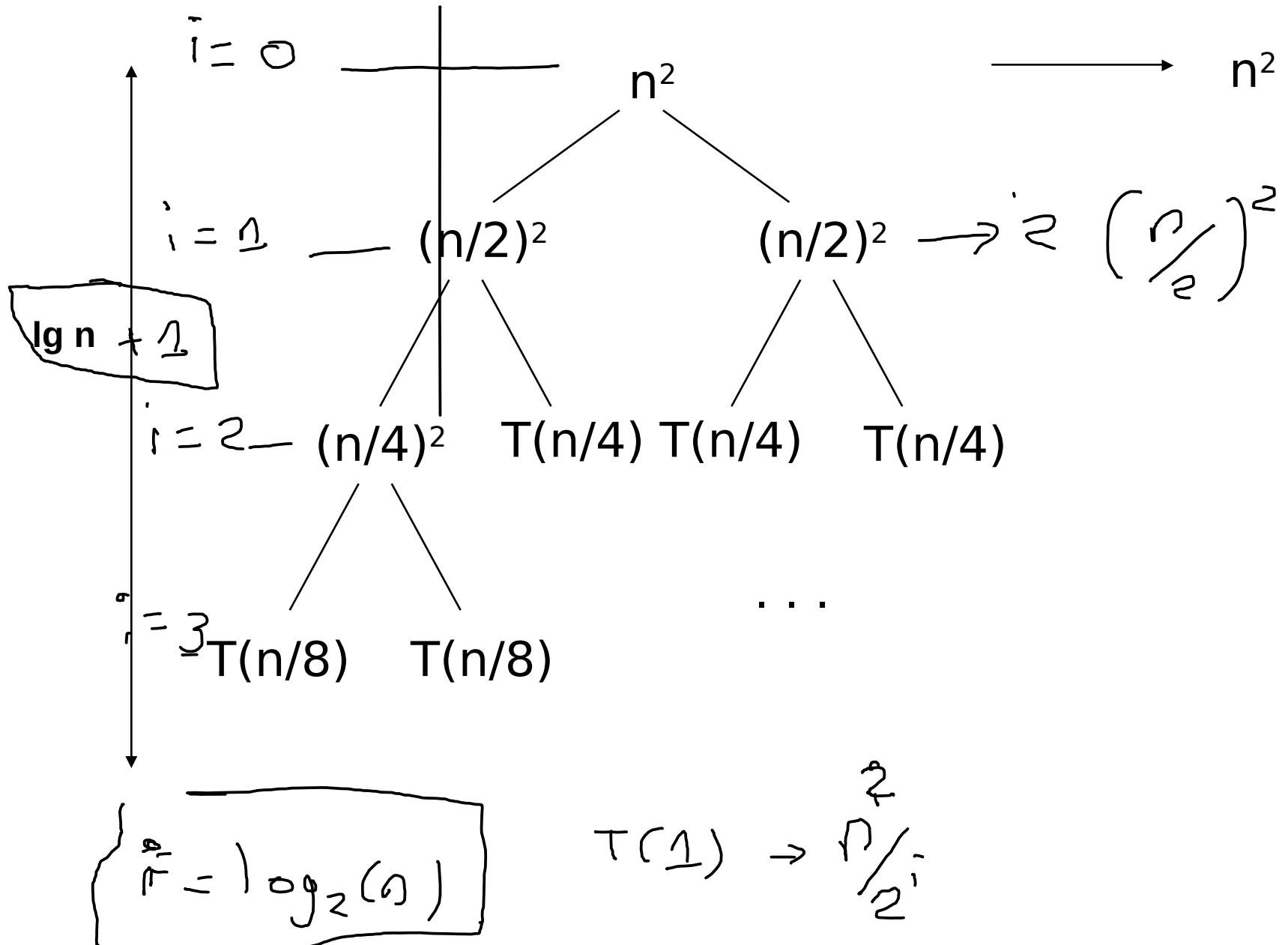
$$(n/2^i)^2 = 1$$

$$(n/2^i) = 1$$

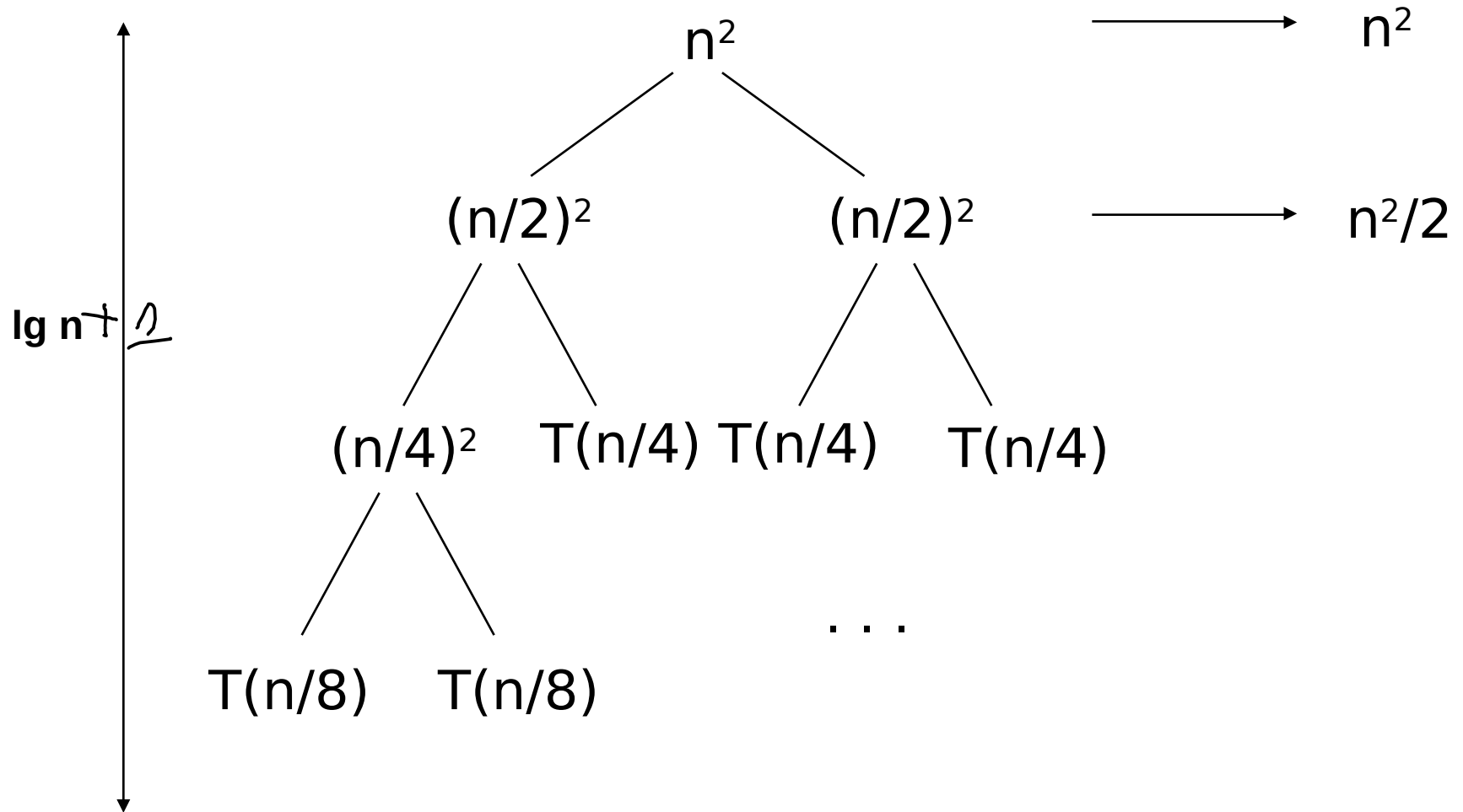
$$n = 2^i$$

$$\log n = i$$

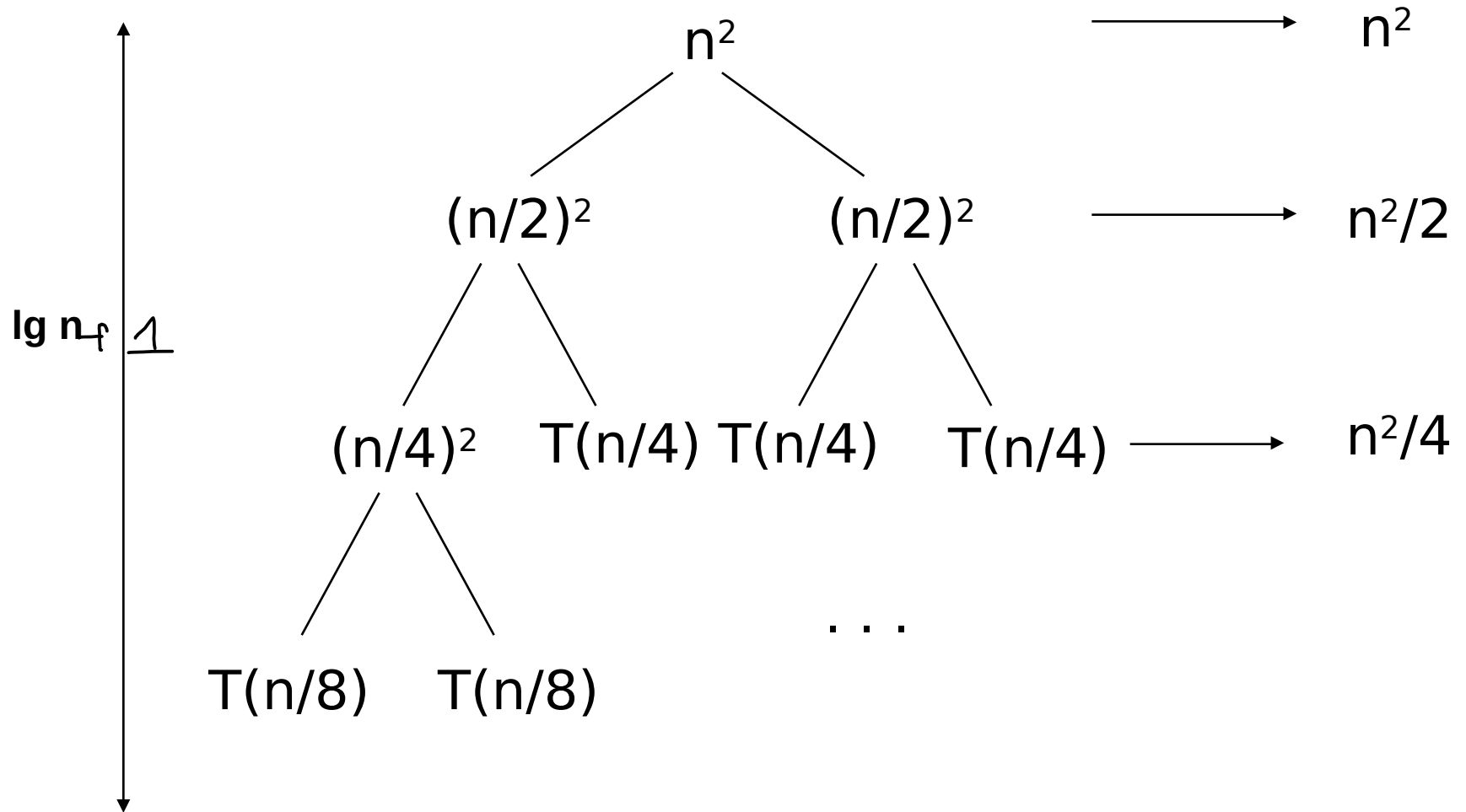
Recurrencias



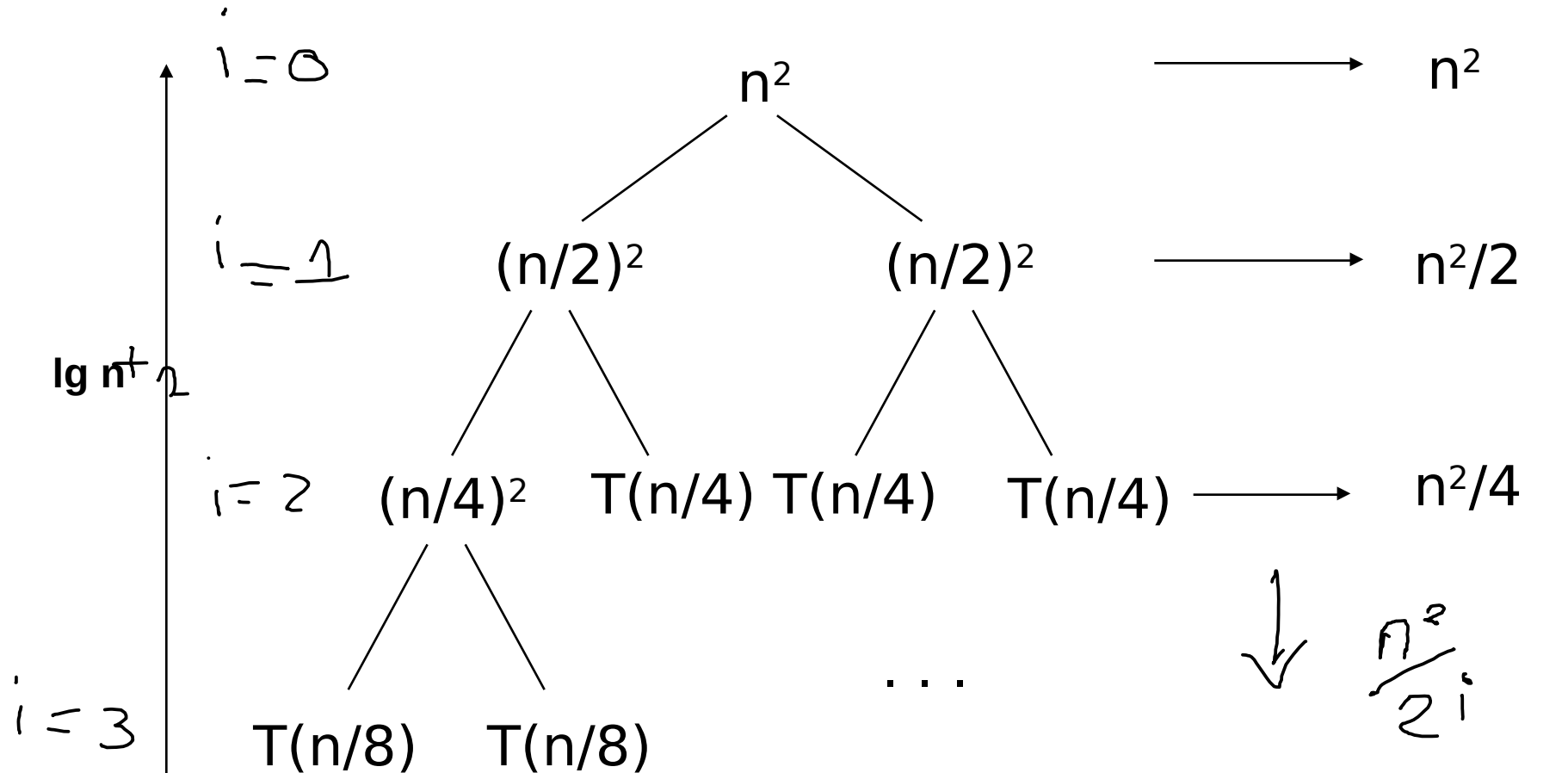
Recurrencias



Recurrencias



Recurrencias



$$\text{Total} = \sum_{i=0}^{\lg n} \frac{n^2}{2^i} = n^2 * \sum_{i=0}^{\lg n} \left(\frac{1}{2}\right)^i$$

Recurrencias

$$Total = \sum_{i=0}^{\lg n} \frac{n^2}{2^i} = n^2 * \sum_{i=0}^{\lg n} \left(\frac{1}{2}\right)^i$$

$$Total = n^2 * \frac{(1/2)^{(\lg n + 1)} - 1}{1/2 - 1} = \Theta(n^2)$$

Recurrencias

Resuelva construyendo el árbol

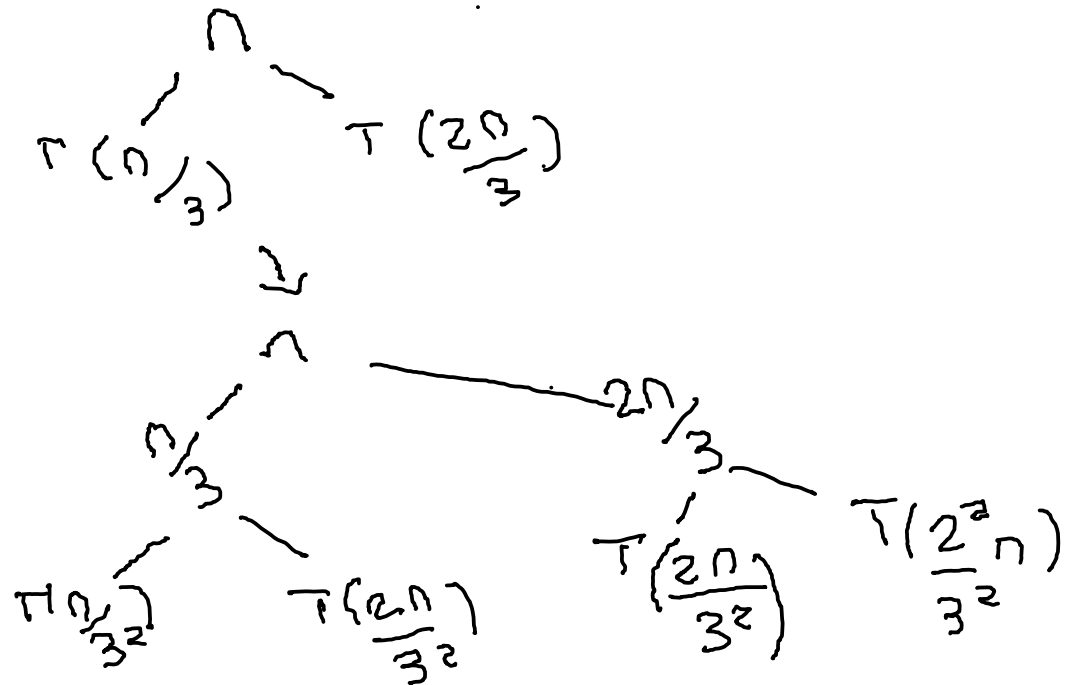
$$T(n) = 2T(n/2) + 1, T(1) = \Theta(1)$$

$$T(n) = 2T(n/2) + n, T(1) = \Theta(1)$$

Recurrencias

Resuelva la recurrencia $T(n) = T(n/3) + T(2n/3) + n$

Indique una cota superior y una inferior



$$\frac{2}{3}n = \frac{2n}{3} + \frac{n}{3}$$

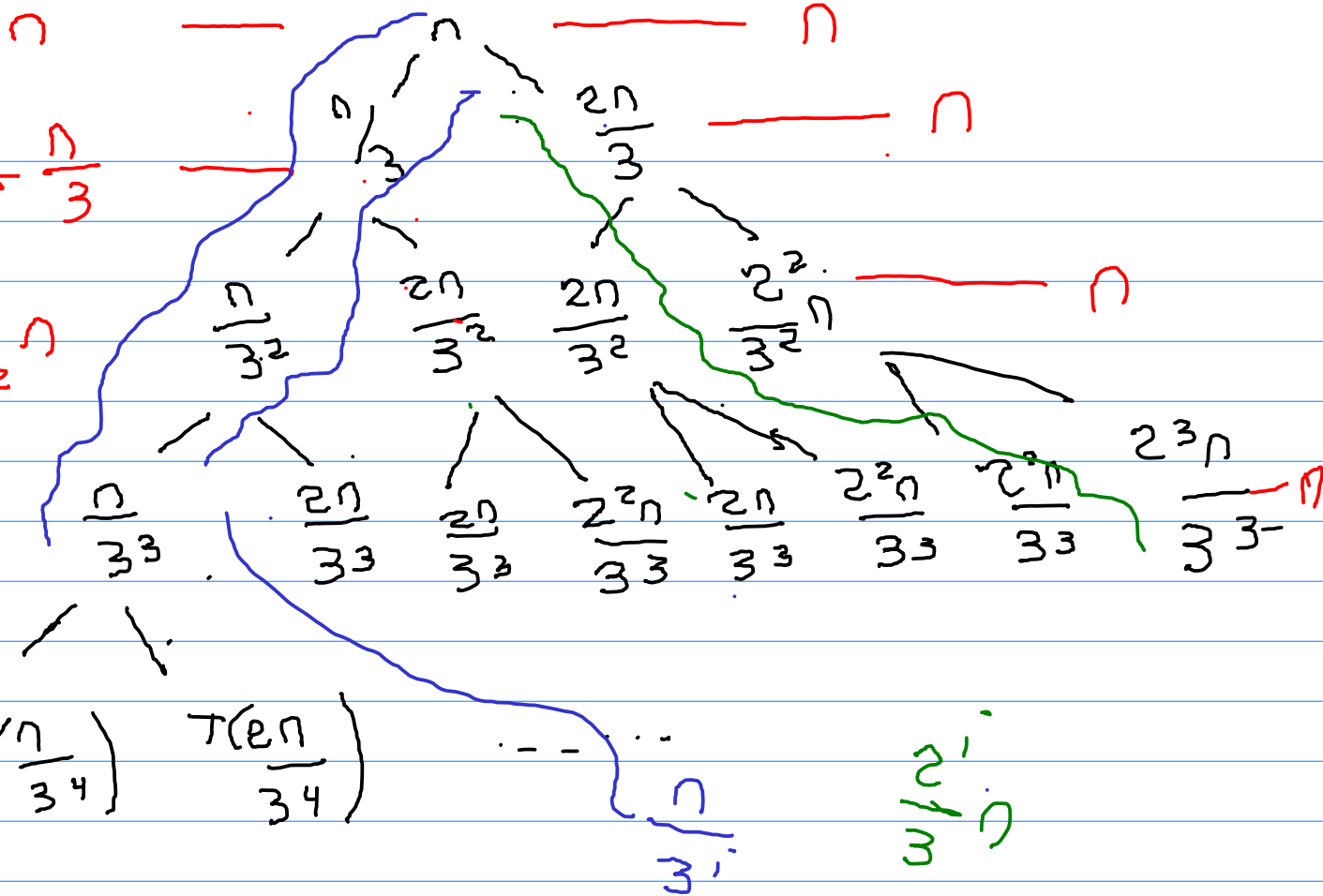
$$\frac{n}{3^2}$$

$$\frac{27n}{3^3}$$

$$T\left(\frac{n}{3^4}\right) \quad T\left(\frac{2n}{3^4}\right)$$

$$\sum_{i=0}^x n$$

$$T(1) ??$$



$$\frac{n}{3^i} = 1$$

$$\frac{2^k}{3^k} = 1$$

$$n = \frac{3^k}{2^k}$$

$$k = \log_{\frac{3}{2}}(n)$$

$$i = \log_3(n)$$

major profund.

$$\log_3(n) = 4$$

major profund.

$$n = 27$$

$$1 = 3$$

$$\log_{\frac{3}{2}} 27 = \frac{\log_{10}(27)}{\log_{10}(\frac{3}{2})} = 8$$

$$\sum_{i=0}^{\log_3(n)}$$

$$n \rightarrow \log_3(n) \times n + n \rightarrow \Omega(n \log_3(n))$$

$$\sum_{i=0}^{\log_{\frac{3}{2}}(n)}$$

$$n \rightarrow \log_{\frac{3}{2}}(n) \times n + n \rightarrow \Theta(n \log_{\frac{3}{2}}(n))$$

Recurrencias

Método maestro

Permite resolver recurrencias de la forma:

$$T(n) = aT(n/b) + f(n), \text{ donde } a \geq 1, b > 1$$

Recurrencias

Dado $T(n) = aT(n/b) + f(n)$, donde $a \geq 1$, $b > 1$, se puede acotar asintóticamente como sigue:

1. $T(n) = \Theta(n^{\log_b a})$

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

2. $T(n) = \Theta(n^{\log_b a} \lg n)$

Si $f(n) = \Theta(n^{\log_b a})$ para algún $\varepsilon > 0$

3. $T(n) = \Theta(f(n))$

Si $f(n) = \Omega(n^{\log_b a + \varepsilon})$ para algún $\varepsilon > 0$ y si $af(n/b) \leq cf(n)$

para algún $c < 1$

Recurrencias

Dado $T(n) = 9T(n/3) + n$

$$n^{\log_3 9} = n^2$$

vs

$$f(n) = n$$

$$a = 9$$

$$b = 3$$

$$F(n) = n$$

$$\text{Es } f(n) = O(n^{\log_b a - \epsilon}) \quad ?$$

$$\text{Es } n = O(n^{2 - \epsilon}) \quad ?$$

$$c n^{\log_3 9} > F(n), \quad n > k$$

$$c n^{\log_3 9 - \epsilon} > n$$

$$c n^{2 - \epsilon} > n$$

\rightarrow

$$c n > n$$



$$\text{Si } c = 2$$

Recurrencias

Dado $T(n) = 9T(n/3) + n$

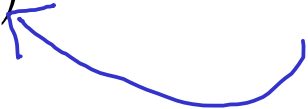
$$n^{\log_3 9} = n^2 \quad \textbf{vs} \quad f(n) = n$$

Es $f(n) = O(n^{\log_b a - \varepsilon})$?

Es $n = O(n^{2-\varepsilon})$?

Si $\varepsilon = 1$ se cumple que $n = O(n)$, por lo tanto, se cumple que:

$$T(n) = \Theta(n^2)$$

$$T(n) = \Theta(n^{\log_3 9})$$


Recurrencias

$$T(n) = T(2n/3) + 1$$

$$\phi T(n/b) + f(n)$$

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

vs

$$f(n) = 1$$

$$\text{Es } f(n) = O(n^{\log_b a - \epsilon}) \quad ?$$

$$\phi = 1$$

$$\text{Es } 1 = O(n^{0 - \epsilon}) \quad ? \quad \text{No}$$

$$b = 3/2$$

No existe $\epsilon > 0$

$$1 = O\left(\frac{1}{2}\right) \rightarrow$$

$$\frac{1}{2} \geq n, c > 0$$

El primer caso no se cumple

Recurrencias

$$T(n) = T(2n/3) + 1$$

$$n^{\log_{3/2} 1} = n^0 = 1 \quad \text{vs} \quad \boxed{f(n) = 1}$$

$$\text{Es } f(n) = \Theta(n^{\log_b a}) \quad ?$$

$$\Theta(1) = 1 \quad \checkmark$$

$$\text{Es } 1 = \Theta(1) \quad ?$$

Si, por lo tanto, se cumple que:

$$T(n) = \Theta(1 * \lg n) = \boxed{\Theta(\lg n)}$$

Recurrencias

$$T(n) = 3 T(n/4) + n \lg n$$

$$n^{\log_4 3} = n^{0.793} \quad \text{vs} \quad f(n) = n \lg n$$

$$\text{Es } f(n) = O(n^{\log_b a - \epsilon}) \quad ? \quad O(n^{0.793 - 1}) \quad \times$$

$$\text{Es } f(n) = \Theta(n^{\log_b a}) \quad ? \quad \Theta(n^{0.793}) \quad \times$$

$$\text{Es } f(n) = \Omega(n^{\log_b a + \epsilon}) \quad ? \quad \Omega(n^{1.793}) \quad \checkmark$$

Si, y además, $af(n/b) \leq cf(n)$

$$3(n/4) \lg(n/4) \leq cn \lg n \quad c < 4$$

$$3(n/4) \lg n - 3(n/4) \cdot 2 \leq cn \lg n$$

$$(3/4)n \lg n \leq cn \lg n \rightarrow \boxed{c=3/4} \text{ y se concluye que } T(n) = \Theta(n \lg n)$$

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

Recurrencias

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

Muestre que no se puede resolver por el método maestro

$$\underline{\underline{\underline{+ a r p g}}}$$

$$a=2 \quad b=2 \quad f(n) = n \lg(n)$$

$$n \lg_b a = n \lg_2 2 = \textcircled{n}$$

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

$$qT(n/b) + F(n)$$

$$q=2 \quad b=2 \quad F(n) = \log(n)$$

$$\log_b q = \log_2 2 = 1$$

$$1) F(n) \text{ es } O(n^{\log_b q - \epsilon}) \quad \epsilon > 0, \log(n) \text{ es } O(n^{1-\epsilon})$$

$$\epsilon = 1, \log(n) \text{ es } O(1) \quad \times$$

$$2) F(n) \text{ es } \Theta(n^1) \quad \times \rightarrow \log(n) \text{ es } O(n) \text{ y } \Omega(n) \quad \times$$

$$3) F(n) \text{ es } \Omega(n^{1+\epsilon}) \rightarrow \log(n) \text{ es } \Omega(n^2) \quad \times$$

Recurrencias

Resuelva

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

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

$$T(n) = 4T(n/2) + n^3$$

$$T(n) = 4T(n/2) + n \quad a=4 \quad b=2 \quad F(n)=n$$

$$n^{\log_b a} = n^{\log_2 4} = n^2$$

$$\underline{1)} \quad F(n) \text{ es } O(n^{2-\epsilon}) \rightarrow n \text{ es } O(n^{2-\epsilon}) \quad \epsilon=1$$

$$n \text{ es } O(n) \rightarrow cn \geq n, \quad n \geq k$$

$$c=2 \quad k=1$$

$$T(n) = \Theta(n^{\log_b a}) = \Theta(n^2)$$

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

$$a=4 \quad b=2 \quad f(n)=n^2$$

$$1) \quad f(n) \text{ is } O(n^{2-\epsilon}) \quad n^2 \text{ is } O(n) \quad \times$$

$$2) \quad f(n) \text{ is } \Theta(n^2) \quad n^2 \text{ is } \Theta(n^2)$$

$$\begin{array}{cc} \swarrow & \searrow \\ n^2 \text{ is } \Theta(n^2) & n^2 \text{ is } \Omega(n^2) \end{array}$$

$$cn^2 \geq n^2, n \geq k, \quad cn^2 \leq n^2, n \geq k$$

$$T(n) = \Theta(n^{\log_b a} \log(n))$$

$$T(n) = \Theta(n^2 \log(n))$$

$$T(n) = 4T(n/2) + n^3 \quad a=4 \quad b=2 \quad f(n)=n^3$$

$$1) \quad n^3 \text{ es } O(n^{2-\epsilon}) \xrightarrow{\epsilon > 0} n^3 \text{ es } O(n) \quad \times$$

$$2) \quad n^3 \text{ es } \Theta(n^2) \quad \times \Rightarrow n^3 \text{ es } O(n^2) \quad \times$$

$$n^3 \text{ es } \Omega(n^2) \quad \checkmark$$

$$3) \quad n^3 \text{ es } \Omega(n^{2+\epsilon}) \Rightarrow n^3 \text{ es } \Omega(n^3) \quad \checkmark$$

$$a f(n/b) < c f(n) \quad c < \frac{1}{4} \quad \leftarrow c = \frac{3}{4}$$

$$4 \left(\frac{n}{2} \right)^3 < c n^3 \Rightarrow \frac{1}{2} n^3 < c n^3 \quad c > \frac{1}{2}$$

$$T(n) = \Theta(f(n)) = \Theta(n^3)$$

Recurrencias

Método de sustitución

Suponer la forma de la solución y probar por inducción matemática

Recurrencias

$$T(n) = 2T(\underline{n/2}) + n, \quad T(1) = 1$$

$$\begin{array}{c} n \\ \nearrow \\ n+1 \\ \hline \end{array}$$

Suponer que la solución es de la forma $T(n) = O(n \lg n)$

Probar que $T(n) \leq cn \lg n$.

Se supone que se cumple para $n/2$ y se prueba para n

Hipotesis inductiva: $T(n/2) \leq cn/2 \lg(n/2)$

$$\underline{cn \lg(n/2)}$$

$$f(n) \text{ es } O(g(n)) \rightarrow \exists g(n) \geq f(n) \quad \forall n \geq k$$

Recurrencias

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

Probar que $T(n) \leq cn \lg n$.

Hipótesis inductiva: $T(n/2) \leq cn/2 \lg(n/2)$

$$c \underbrace{\lg(2)} = c$$

Paso inductivo:

$$T(n) \leq 2(cn/2 \lg(n/2)) + n$$

$$\leq cn \lg(n/2) + n \rightarrow cn(\lg(n) - \lg(2)) + n$$

$$= \underbrace{cn \lg(n)} - \underbrace{cn} + n, \text{ para } c \geq 1, \text{ haga } \underbrace{c=1}$$

$$\underbrace{T(n)} \leq cn \lg n$$

Recurrencias

$$T(n) = 2T(n/2) + n, T(1) = 1$$

Probar que $T(n) \leq cn \lg n$.

Paso base: si $c=1$, probar que $T(1)=1$ se cumple

$$T(n) \leq n \lg(n)$$

$$T(1) \leq 1 * 1 \lg 1 ?$$

$$1 \leq 0 ?$$

No, se debe escoger otro valor para c

Recurrencias

$$T(n) = 2T(n/2) + n, T(1) = 1$$

Probar que $T(n) \leq cn \lg n$.

Paso base: si $c=2$, probar que $T(1)=1$ se cumple

$$T(1) \leq 2 * 1 \lg 1 ?$$

$$1 \leq 0 ?$$

No, se puede variar k.

Para esto, se calcula $T(2)$ y se toma como valor inicial

$$c = 3$$

$$c = 4$$

$$n > 1$$

Recurrencias



Probar que $T(n) \leq cn \lg n$.

$$T(2) = 2T(0) + 2 = 4$$

Paso base: si $c=1$, probar que $T(2)=4$ se cumple

$$T(2) \leq 1 * 2 \lg 2 ?$$

$$4 \leq 2 ?$$

No, se puede variar c .



Recurrencias

Probar que $T(n) \leq cn \lg n$.

$$T(2) = 2T(1) + 2 = 4$$

Paso base: si $c=3$, probar que $T(2)=4$ se cumple

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

$$T(2) \leq 3 \cdot 2 \lg 2 ?$$

$$T(4) = 2T(2) + 4 = 8$$

$$4 \leq 6 ?$$

Si, se termina la demostración

$$8 \leq 3 \cdot 8 \cdot \lg(4)$$

$$T(n) = 2T(n/2) + n \text{ es } O(n \lg n)$$

$$\boxed{8 \leq 48}$$

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

$$\Theta(n^2 \log(n)) \quad T(1) = 1$$

Para base

$$O(n^2 \log(n)) \quad \Omega(n^2 \log(n))$$

Se cumple para $n/2$

entonces demostramos para n

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

$$\Theta(n^2 \log(n))$$

$$O(n^2 \log(n))$$



Suponemos $T(n/2)$ es correcto

$$T(n/2) \leq c \times \frac{n^2}{2^2} \log\left(\frac{n}{2}\right) \Rightarrow T(n) = O(n^2 \log(n))$$

$$T(n) \leq 4c \frac{n^2}{2^2} \log\left(\frac{n}{2}\right) + n^2$$

$$T(n) \leq cn^2 \log(n) + \underline{(1 - c \log(2)) n^2}$$

$$T(n) \leq cn^2 \log(n)$$

$$c=2$$

$$T(1)=1$$

$$1 \leq 0^x$$

$$T(2) = 4T(1) + 4 = 8$$

$$8 \leq 2 \times 4 \times 1 \rightarrow 8 \leq 8 \checkmark$$

Demostrado

$$T(n) = 4T(n/2) + n^2 \quad \text{e.s. } \Omega(n^2 \log(n)) \quad \checkmark$$

↓

$$T(n/2) \geq c \times \left(\frac{n}{2}\right)^2 \log\left(\frac{n}{2}\right)$$

$$T(n) \geq 4c \times \left(\frac{n}{2}\right)^2 \log\left(\frac{n}{2}\right) + n^2$$

$$T(n) \geq cn^2 \log(n) + (1 - \overbrace{\log(2)}^4 c) n^2$$

$$T(n) \geq cn^2 \log(n) + (1 - c) n^2$$

$$c = 1$$

$$T(n) \geq n^2 \log(n)$$

$$T(1) \geq 1$$

$$I \geq 0 \quad \checkmark$$

$$\left\{ \begin{array}{ll} T(2) = 8 & T(4) = 48 \end{array} \right.$$

$$8 \geq 8 \quad \checkmark$$

$$48 \geq 32 \quad \checkmark \checkmark$$

Análisis de algoritmos recursivos

Un algoritmo recursivo tiene las siguientes partes:

- 1) Una condición de parada
- 2) Un llamado recursivo

El análisis de estos algoritmos lo realizaremos analizando

- 1) Su complejidad en un llamado
- 2) Cómo es el llamado recursivo y cómo cambia la entrada a medida que se realizan los llamados
- 3) Cómo es la forma de la entrada para llegar a la condición de parada

Análisis de algoritmos recursivos

Ejemplo, pensemos en este algoritmo para calcular la serie de Fibunnaci para un número (n) dado

Recuerda:

$$f(n) = f(n-1) + f(n-2), f(0) = 1, f(1) = 1$$

fibunnaci(n)

Si $n = 0$ retorne 1

Sino si $n = 1$ retorne ~~1~~

Sino fibunnaci(n - 1) + fibunnaci(n - 2)

Análisis de algoritmos recursivos

Si

Recuerda:

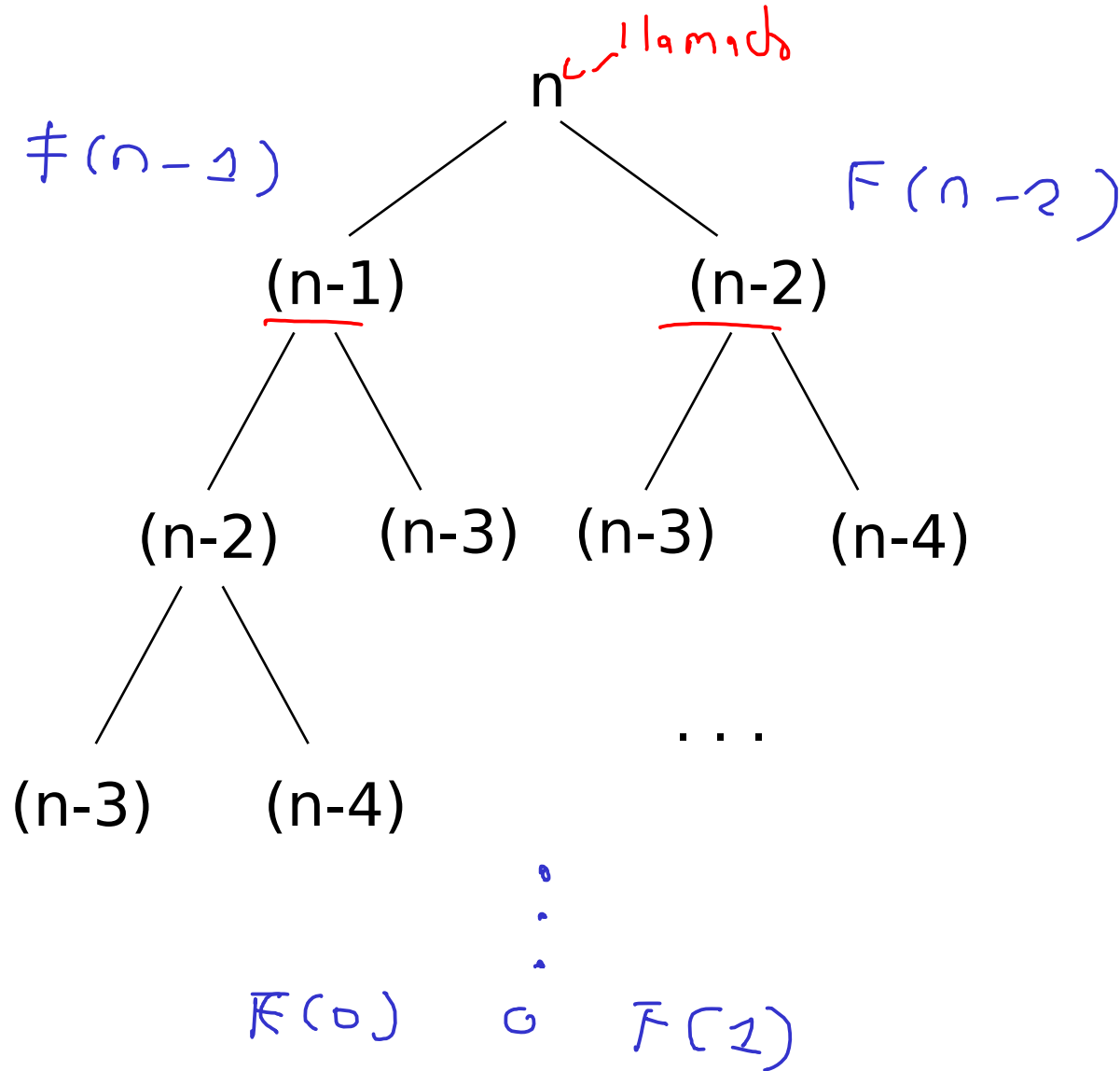
$$f(n) = \underline{f(n-1)} + \underline{f(n-2)}, f(0) = 1, f(1) = 1$$

fibunnaci(n)

- Si $n = 0$ retorne 1
- Sino si $n = 1$ retorne 2

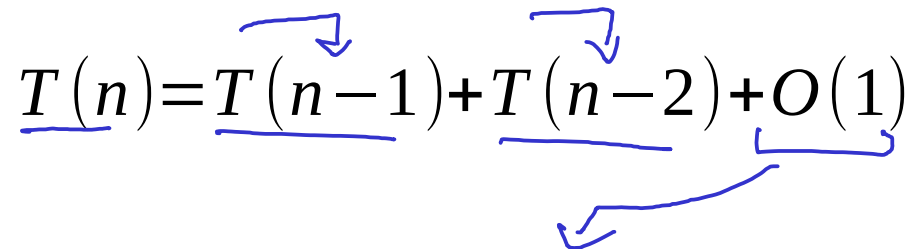
Sino fibunnaci(n - 1) + fibunnaci(n - 2)

Análisis de algoritmos recursivos



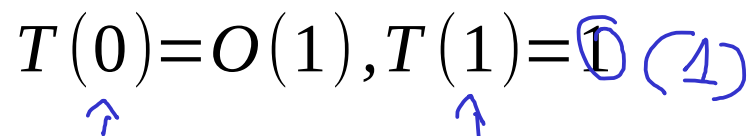
Análisis de algoritmos recursivos

Si observamos para cada llamado de n se realiza el llamado para $n-1$ y $n-2$, la parada se encuentra cuando $n = 0$ y $n = 1$ entonces, la complejidad de este algoritmo está dada por la relación

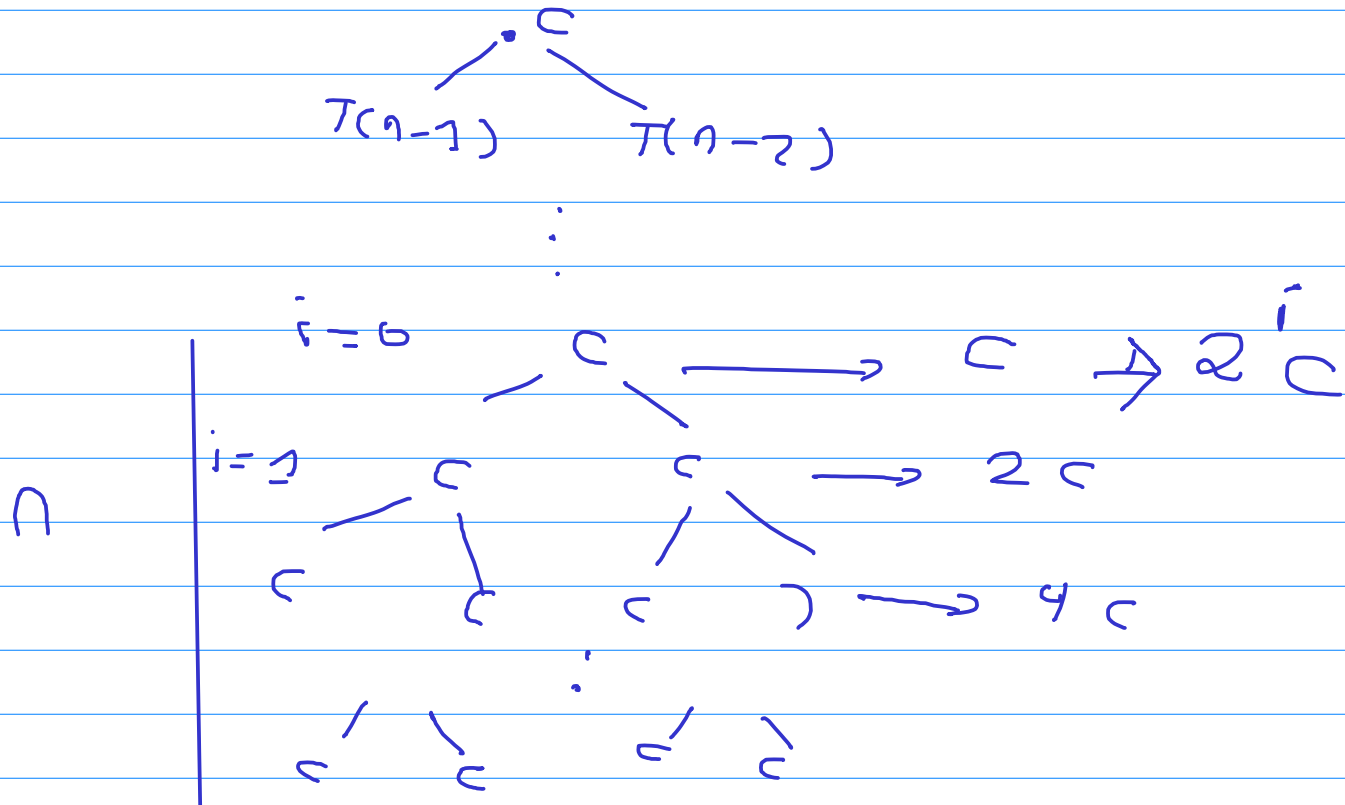
$$T(n) = T(n-1) + T(n-2) + O(1)$$
The equation $T(n) = T(n-1) + T(n-2) + O(1)$ is shown with blue annotations. Blue arrows point from $T(n)$ to $T(n-1)$ and $T(n-2)$. The terms $T(n-1)$, $T(n-2)$, and $O(1)$ are underlined. A blue arrow points from the $O(1)$ term down towards the text below.

Se cuenta $O(1)$ en cada llamado debido a la verificaciones que debe realizar, las cuales son ejecutadas en **tiempo constante**

Si observa en las condiciones de parada el tiempo también es constante, entonces:

$$T(0) = O(1), T(1) = 1$$
The base cases $T(0) = O(1)$ and $T(1) = 1$ are shown. Blue arrows point up to 0 and 1 . The 1 in $T(1)$ is circled in blue, and the $O(1)$ in the first equation is also circled in blue.

$$T(n) = T(n-1) + T(n-2) + c$$



$$T(n) = \sum_{i=0}^n 2^i c \rightarrow c \left(\frac{2^{n+1} - 1}{2 - 1} \right) \rightarrow O(2^n)$$

Análisis de algoritmos recursivos

Siempre que se trabaje con algoritmos recursivos el cálculo de la complejidad va tener la forma

$$T(n) = T(x_1) + T(x_2) + \dots + T(x_n) + f(n)$$

$$x_1 = \Delta n$$

Donde $T(x_i)$ indica cómo cambia la entrada en cada llamado recursivo y $f(n)$ representa la complejidad de los procesos realizados en un **sólo** paso.

Iterativo

$$F_0 = 1$$

$$F_1 = 1$$

$$\text{for}(i=2; i < n; i++) \{ \quad n-1-2+1+1 = (n-1)$$

$$F_x = F_{x-1} \quad (n-2)$$

$$F_1 = F_1 + F_0 \quad (n-2)$$

$$F_0 = F_{x-1} \quad (n-2)$$

}

$$O(n) \text{ vs } O(2^n)$$

Iterativo

Recursivo

Análisis de algoritmos recursivos

Tarea

Analizar el algoritmo Merge Sort para estas condiciones

Mejor caso

Particiones: $n/2$ y $n/2$ complejidad ordenar $O(1)$. $\leftarrow O(n)$

Caso promedio

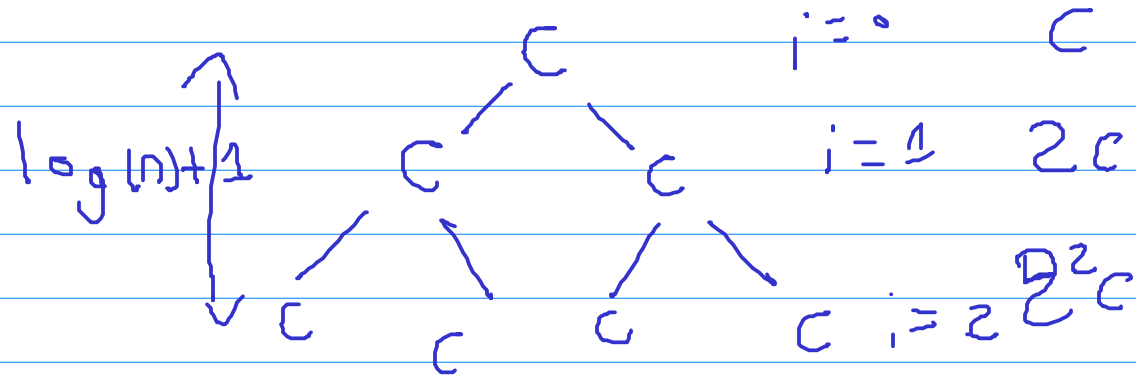
Particiones: $n/8$, $7n/8$ complejidad ordenar $O(n)$

Peor caso

Particiones $n/2$ y $n/2$ complejidad ordenar $O(n)$

Merge Sort

$$T(n) = \underbrace{2T\left(\frac{n}{2}\right)}_{\text{partitioning}} + C \quad \text{order } n \log n$$



$$C(2n-1) \leftarrow T(n) = \sum_{j=0}^i 2^j C = C \sum_{j=0}^{\log(n)} 2^j = C \left(\frac{2^{\log(n)+1} - 2}{2-1} \right)$$

$O(n)$ Major case

$$T(n) = T\left(\frac{n}{8}\right) + T\left(\frac{7n}{8}\right) + n \quad \text{Easo promedio}$$

↓

$$O(n^2)$$

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

per caso

↓

$$O(n^2)$$

Análisis de algoritmos recursivos

Tarea

Análizar algoritmos que tienen el siguiente comportamiento.

Algoritmo 1

Particiones: $n - 2$ y $n - 4$, complejidad cada paso $O(n)$

Algoritmo 2

Particiones: $n/2$ y $n/3$, complejidad cada paso $O(\log(n))$

Análisis de algoritmos recursivos

Tarea

Analizar el siguiente algoritmo:

Algoritmo(n)

Si $n = 0$ retorne 1

Si $n = 1$ retorne 2

Sino Si n es par retorne $n + f(\lfloor n/2 \rfloor)$