3) Complejidad algoritmos:

$t(n) = C_0 T(n/2) + C_1 \qquad t(1) = C_1$

- Primero vamos a reemplazar valores de n para ver si hay algun patron

$T(1) = t_1$

$t(2) = C_0 t(1) + C_1 = C_0 t_1 + C_1$

$T(4) = C_0 t(2) + C_1 = C_0 (C_0 t_1 + C_1) + C_1$

$= C_0^2 t_1 + C_0 C_1 + C_1$

$t(8) = C_0 t(4) + C_1 = C_0 (C_0^2 t_1 + C_0 C_1 + C_1) + C_1$

$= C_0^3 t_1 + C_0^2 C_1 + C_0 C_1 + C_1$

- Dandonos cuenta que $2^0 = 1$; $2^1 = 2$; $2^2 = 4$; $2^3 = 8$

Deducimos que:

$t(2^n) = C_0^{n-1} t_1 + C_0^{n-1} + C^{n-3} C_1 + \dots + C_1 \qquad (*)$

- Si $C_0 = 1$ $(*)$ nos queda.

$t(2^n) = t_1 + n C_1$

- Haciendo el cambio de variable:

$2^n = N \Rightarrow n = \log_2 N$

-tenemos que:

$$t(2^{\log_2 N}) = t_1 + C_1 \log_2 N$$

$$t(N) = t_1 + C_1 \log_2 N$$

$\Rightarrow$ Con el cambio $N \to n$ la complejidad de $t(n)$ es de orden $O(\log_2 n)$

- Para $C_0 = 2$ (*) nos queda:

$$t(2^n) = 2^{n-1} t_1 + 2^{n-1} C_1 + 2^{n-2} C_1 + \dots + 2^0 C_1$$

$$t(2^n) = 2^{n-1} t_1 + \sum_{k=0}^{n-2} C_1 2^k$$

- Recordado que la suma geometrica viene dada por:

$$\sum_{k=0}^{n} a r^k = a \left(\frac{1 - r^{n+1}}{1 - r} \right)$$

- tenemos que

$$t(2^n) = 2^{n-1} t_1 + C_1 \left(\frac{1 - 2^{n-1}}{1-2} \right)$$

$$t(2^n) = 2^{n-1} t_1 + C_1 2^{n-1} - C_1$$

$$t(2^n) = 2^{n-1}(t_1 + C_1) - C_1$$

- Con el cambio $2^n = N \Rightarrow n = \log N \wedge \log_2 2 = 1$

$$t(N) = 2^{\log_2 \frac{N}{2}} (t_1 + C_1) - C_1 = \frac{N}{2}(t_1 + C_1) - C_1$$

$\Rightarrow$ Con el cambio $N \to n$ la complejidad de $t(n)$ es de orden $O(n)$