3. El siguiente algoritmo calcula el mínimo elemento de un arreglo a : **array**$[1..n]$ **of nat** mediante la técnica de programación *divide y vencerás*. Analizá la eficiencia de $minimo(1, n)$.

```
fun minimo(a : array[1..n] of nat, i, k : nat)  ret m : nat
    if i = k then m := a[i]
    else
        j := (i + k) div 2
        m := min(minimo(a, i, j), minimo(a,j+1,k))
    fi
end fun
```

La función ***minimo*** toma como valor de entrada un segmento de un arreglo no vacío, donde la operación relevante de la misma parece ser las asignaciones a la variable *m* (es la que más se repite).
Entonces defino una función recursiva **f(t)** que calculará la cantidad de asignaciones a *m* que realiza ***minimo*** según el tamaño del segmento del arreglo, cuya longitud va desde *i* a *k*.

$$\mathrm{f(t)} = \begin{cases} 1 & \text{si } n = 1 \\ 1 + \mathrm{f(t\ div\ 2)} + \mathrm{f(t\ div\ 2)} & \text{si } n > 1 \end{cases}$$

$$\mathrm{f(t)} = \begin{cases} 1 & \text{si } n = 1 \\ 2\ \mathrm{f(t\ div\ 2)} + 1 & \text{si } n > 1 \end{cases}$$

donde los componentes *a*, *b*, *g(n)* y *k* son:

a = 2	b = 2	g(n) = 1	k = 0

$\therefore$ como $a > b^k$, luego f(t) es del orden $t^{log_2(2)} = t^1 = t$ (complejidad lineal)