```
def PotenciaRec(b, n):
*1 OE — if n == 0:
  1 OE — return 1
*2 OE — if n % 2 == 0:
  2 OE — pot = PotenciaRec(b, n / 2)
  2 OE — return pot * pot
       else:
*3 OE — pot = PotenciaRec(b, (n - 1) / 2)
*3 OE — return pot * pot * b
```

$O(f(n)) = ?$

Assumint el pitjor cas, n és una (potència de 2) -1 (per a que **(n-1)/2** sempre sigui **imparell**). Així s'executa la part del codi amb més operacions elementals possibles.

Amb aquestes assumpcions, veiem que PotenciaRec() s'executa cada cop amb un valor inferior a la **meitat** per **n** cada cop. És a dir, que per a que n acabi siguent el valor mínim (0), el valor de **n** s'ha reduit a la meitat **log2(N)** vegades.

PotenciaRec(b, (n-1)/2)

Pròxima iter. $n_1 \simeq$ la meitat de $n_0$

Reducció a la meitat

↳ $\log_2 N$ crides a la funció per a $n = 1$ ($n = 0$ a la següent)

$\Rightarrow O(\log_2 N)$

↳ $\log_2 N \times OE$

$= \log_2 N \times 9$ operacions elementals