Ejercicio 9

rec 2

```
public int rec2(int n){
    if(n<=1)
        return 1
    else
        return (2 * rec2(n-1));
}
```

$$T(n)\begin{cases} c_1 & , n \leq 1 \\ c_2 + T(n-1) & , n > 1 \end{cases}$$

c_2 → costo de multiplicar por 2 (operación constante)

- paso 1: $T(n) = c_2 + T(n-1)$ si $n > 1$
- paso 2: $T(n) = c_2 + (c_2 + T(n-1)-1)$ ($T(n)$)
  $= 2c_2 + T(n-2)$
- paso 3: $T(n) = 2c_2 + (c_2 + T(n-1) - 2)$
  $= 3c_2 + T(n-3)$
- paso i: $T(n) = i \cdot c_2 + T(n-i)$

encontré la fórmula general, ahora busco hasta el valor donde termina la recursión (1) → $T(n) = 1$ cuando $n - i = 1$

$\therefore (n-1) = i$

$$T(n) = (n-1) \cdot cte_2 + T(n-(n-1))$$
$$T(n) = (n-1) \cdot cte_2 + T(n-n+1)$$
$$T(n) = (n-1) \cdot cte_2 + T(1)$$
$$T(n) = (n-1) \cdot cte_2 + cte_1 \quad \therefore O(n)$$

∴ por lo que el orden del algoritmo rec 2 es de $O(n)$

rec 1

```
static public int rec1(int n){
    if (n ≤ 1)
        return 1;
    else
        return (rec1(n-1) + rec1(n-1));
}
```

$$T(n) \begin{cases} c_1 & , n \leq 1 \\ c_2 + 2T(n-1) & , n > 1 \end{cases}$$

c_2 (la operación constante) + las dos llamadas recursivas

- paso 1: $T(n) = c_2 + 2T(n-1)$ si $n \geq 1$
- paso 2: $T(n) = c_2 + 2(c_2 + 2T(n-1)-1)$
  $= 2 \cdot c_2 + 2^2 T(n-2)$
- paso 3: $T(n) = 2c_2 + 2^2(c_2 + 2T(n-1)-2)$
  $= 3c_2 + 2^3 T(n-3)$
- paso i: $T(n) = i c_2 + 2^i T(n-i)$

encontré la fórmula general ahora busco para el valor donde termina la recursión, caso 1

$T(n) = 1 \rightarrow$ cuando $n - i = 1 \rightarrow i = n-1$

reemplazo i por n-1

$T(n) = (n-1)c_2 + 2^{n-1} \cdot T(1)$

$T(n) = (n-1)c_2 + 2^{n-1} c_1$

* por lo que el orden del algoritmo es de $O(2^n)$

rec 3

```
static public int rec3(int n){
    if (n == 0)
        return 0;
    else{
        if (n == 1)
            return 1;
        else
            return (rec3(n-2) * rec3(n-2));
    }
}
```

$T(n) \begin{cases} c_0 & \text{si } n \leq 1 \\ c_2 + 2T(n-2) & \text{si } n > 1 \end{cases}$

$\Updownarrow$

$T(n) \begin{cases} c_0 & \text{si } n = 0 \\ c_1 & \text{si } n = 1 \\ c_2 + 2T(n-2) & \text{si } n > 1 \end{cases}$

multiplicación (hay factor constante)

- paso 1: $T(n) = c_2 + 2T(n-2)$ si $n > 1$
- paso 2: $T(n) = c_2 + 2(c_2 + 2T(n-2)-2)$

  $= 2c_2 + 2^2 T(n-4) \iff 2c_2 + 2^2 T(n-2\,2)$
- paso 3: $T(n) = 2c_2 + 2^2(c_2 + 2T(n-2)-4)$

  $= 3c_2 + 2^3 T(n-6) \iff 3c_2 + 2^3 T(n-2\,3)$
- paso i: $T(n) = i\,c_2 + 2^i T(n-2\,i)$
- encontre la formula general ahora busco, para el valor donde termina la recursión $\Omega(1 \leq)$

$T(n) = 1 \rightarrow$ cuando $n - 2\,i = 1$

$2i = n-1 \iff i = \frac{n-1}{2}$

$T(n) = i\,c_2 + 2^i T(n-2\,i) \rightarrow i = \frac{n-1}{2}$

$T(n) = \frac{n-1}{2} c_2 + 2^{\frac{n-1}{2}} T(1)$, etc

$T(n) = \frac{n-1}{2} c_2 + 2^{\frac{n-1}{2}} c_1 \quad \therefore O(2^{\frac{n}{2}})$

* por lo que el orden del Algoritmo es de $2^{\frac{n}{2}}$

potencia_iter

```
static public int potencia_iter (int x , int n){
    int potencia;
    if (n==0)
        potencia = 1;
    else
        if (n==1)
            potencia =x;
        else{
            potencia=x
            for(int i = 2 ; i<=n; i++){
                potencia *= x
            }
        }
    return potencia
}
```

$n \leq 1$ → cte 1

$n > 1$ → cte 2

$$T(n) = c_1 + c_2 + \sum_{i=2}^{n} c_3$$

$$T(n) = c_1 + c_2 + (n-1)c_3$$

$\therefore O(n)$

Hace lo que el Algoritmo es de Orden $O(n)$

potencia_rec

$$T(n) \begin{cases} c_0 & , si\ n = 0 \\ c_1 & , si\ n = 1 \\ T(n/2) + c_2 & , si\ n > 1 \end{cases} \iff T(n) \begin{cases} c_1 & , si\ n \leq 1 \\ T\left(\frac{n}{2}\right) + c_2 & , si\ n > 1 \end{cases}$$

- paso 1: $T(n)_1 = T(n/2) + c_2$ si $n > 1$
- paso 2: $T(n) = \left(\frac{T(n/2)+c_2}{2}\right) + c_2$

  $= T\left(\frac{n}{4}\right) + 2 \cdot c_2$
- paso 3: $T(n) = T\left(\frac{n}{8}\right) + 3c_2$
- reemplazo i:

$$T(n) = T\left(\frac{n}{2^{\log_2(n)}}\right) + \log_2(n) \cdot c_2$$

$$T(n) = T\left(\frac{n}{n}\right) + \log_2(n) c_2 = T(1) + \log_2(n) c_2$$

$\therefore$ Hace lo que el Algoritmo es de $O(\log(n))$

- paso i = $T(n) = T\left(\frac{n}{2^i}\right) + i c_2$
- encontre la formula general ahora busco, hasta el valor donde termina la recursion osea (1) → $T(n) = 1$

  cuando $\frac{n}{2^i} = 1$

  $n = 2^i$ → $\log_b(b^x) = x$

  $\log_2(n) = i$

prop log

$\log_b a = b$