1. Probar utilizando el método de sustitución que $T(n) \in O(\log_2(n))$.

$$T(n) = \begin{cases} 1 & n = 1 \\ T(\lfloor n/2 \rfloor) + 1 & n > 1 \end{cases}$$

Solución

- Caso base $n = 2$: $0 \leq 2 = T(n) \leq 2\log_2(n) = 2$.
- Caso inductivo: Supongamos que para $n = 1, \ldots, \left\lfloor \frac{k+1}{2} \right\rfloor, \ldots, k$ vale que $0 \leq T(n) \leq 2\log_2(n)$. Luego:

$$T(k+1) = T\left(\left\lfloor \frac{k+1}{2} \right\rfloor\right) + 1 \leq 2\log_2\left(\left\lfloor \frac{k+1}{2} \right\rfloor\right) + 1 \leq 2\log_2\left(\frac{k+1}{2}\right) + 1 =$$

$$= 2\left[\log_2(k+1) - \log_2(2)\right] + 1 = 2\log_2(k+1)$$

2. Sean $a, b \in \mathbb{R}^+$, utilizar el método de sustitución para encontrar funciones $f(n)$ tales que $T(n) \in \Theta[f(n)]$ para las siguientes recurrencias:

a) $T(n) = \begin{cases} a & n = 1 \\ 2T(\lfloor n/2 \rfloor) + b & n > 1 \end{cases}$.

b) $T(n) = \begin{cases} a & n = 1 \\ 2T(\lfloor n/2 \rfloor) + n & n > 1 \end{cases}$.

Ayuda: Recuerde las propiedades:

- $x \neq 1 \Rightarrow \sum_{j=0}^{n} ax^j = \frac{a - ax^{n+1}}{1-x}$.
- COMPLETAR.
- COMPLETAR.

Soluciones

a) Notese que para $n = 2^k$ resulta:

$$T(n) = 2T\left(2^{k-1}\right) + b = 2\left[2T\left(2^{k-2}\right) + b\right] + b = 2^2T\left(2^{k-2}\right) + 3b =$$

$$= 2^3T\left(2^{k-1}\right) + 2^2b + 2b + b = \ldots = 2^ka + \sum_{i=0}^{k-1} 2^ib = 2^ka + b\sum_{i=0}^{k-1} 2^i =$$

$$= 2^ka + b\frac{1-2^k}{1-2} = 2^ka - b\left(1-2^k\right) = 2^ka - b + b2^k = 2^k(a+b) - b =$$

$$= n(a+b) + b \in O(n)$$

- $\boxed{T(n) \in O(n)}$: Probaremos por inducción que $0 \leq T(n) \leq (a+b)n - b$:
  - Caso base $n = 1$: $T(n) = a \leq a + b - b \leq n(a+b-b) = (a+b)n - nb \leq (a+b)n - b$.
  - Caso inductivo: Supongamos que para $n = 1, \ldots, \left\lfloor \frac{k+1}{2} \right\rfloor, \ldots, k$ vale que $0 \leq T(n) \leq (a+b)n - b$. Luego:

  $$T(k+1) = 2T\left(\left\lfloor \frac{k+1}{2} \right\rfloor\right) + b \leq 2(a+b)\left\lfloor \frac{k+1}{2} \right\rfloor - 2b + b =$$

  $$= 2(a+b)\left\lfloor \frac{k+1}{2} \right\rfloor - b \leq (a+b)(k+1) - b$$

  y como $b \geq 0$ entonces $0 \leq T(n) \leq (a+b)n - b \leq (a+b)n$ por lo que $T(n) \in O(n)$.
- $\boxed{T(n) \in \Omega(n)}$: COMPLETAR.

b) Observemos que pasa para $n = 2^k$:

$$T(n) = 2T\left(2^{k-1}\right) + 2^k = 2\left[2T\left(2^{k-2}\right) + 2^{k-1}\right] + 2^k = 2^2T\left(2^{k-2}\right) + 2 \cdot 2^{k-1} + 2^k =$$

$$= 2^2\left[2T\left(2^{k-3}\right) + 2^{k-2}\right] + 2 \cdot 2^{k-1} + 2^k = 2^3T\left(2^{k-3}\right) + 2^2 \cdot 2^{k-2} + 2 \cdot 2^{k-1} + 2^k =$$

$$= 2^3T\left(2^{k-3}\right) + 2^k + 2^k + 2^k = \ldots = 2^ka + k2^k = na + \log_2(n)\,n \in O\left[n\log_2(n)\right]$$

- $\boxed{T(n) \in O(n)}$: Probaremos por inducción que $0 \leq T(n) \leq cn\log_2(n)$:
  - Caso base $n = 2$: $T(n) = a \leq cn\log_2(n) = 2 \iff a/2 \leq c$.
  - Caso inductivo: Supongamos que para $n = 1, \ldots, \left\lfloor \frac{k+1}{2} \right\rfloor, \ldots, k$ vale que $0 \leq T(n) \leq cn\log_2(n)$. Luego:

    $$T(k+1) = 2T\left(\left\lfloor \frac{k+1}{2} \right\rfloor\right) + k + 1 \leq c(k+1)\log_2\left(\left\lfloor \frac{k+1}{2} \right\rfloor\right) + k + 1 =$$

    $$\leq c(k+1)\left[\log_2(k+1) - 1\right] + k + 1 = c(k+1)\log_2(k+1) - (c-1)(k+1)$$

    Finalmente, tomando $c = \max\{a/2, 1\}$ vale:

    $$c(k+1)\log_2(k+1) - (1-c)(k+1) < c(k+1)\log_2(k+1)$$

    por lo que $T(n) \in O[n\log_2(n)]$.
- $\boxed{T(n) \in \Omega(n)}$: COMPLETAR.

3. Utilice un árbol de recurrencia para encontrar una cota asintotica Θ para la recurrencia $T(n) = 4T(\lceil n/2 \rceil) + cn$ donde c es una constante. Verifique que la cota encontrada es correcta.

4. Utilizar un arbol de recurrencia para obtener una cota asintotica para

$$T(n) = \begin{cases} c' & n \leq a \\ T(n-1) + T(a) + cn & n > a \end{cases}$$

donde $a \geq 1$, $c > 0$ son constantes.

5. Utilizar el teorema maestro para encontrar cotas asintoticas Θ para las siguientes recurrencias (asumir que $T(1) > 0$):

   a) $T(n) = 4T(n/2) + n$.
   b) $T(n) = 4T(n/2) + n^2$.
   c) $T(n) = 4T(n/2) + n^3$.

6. Para cada una de las siguientes funciones, determinar si son suaves o no. Demostrar.

   a) $\ln(n)$.

b) n^2.

c) n^n.

7. Encontrar cotas asintoticas Θ para cada una de las siguientes recurrencias y demostrarlas. Asumir que $T(1) > 0$.

   a) $T(n) = T(n/2) + 1$.

   b) $T(n) = T(n-1) + n$.

   c) $T(n) = T(\lfloor\sqrt{n}\rfloor) + 1$. Ayuda: use «renombre de variable» con $n = 2^k$. En otras palabras, calcule primero una cota Θ para $T \circ 2^k$, usando $T\left(2^k\right) = T\left(\left\lfloor 2^{k/2} \right\rfloor\right) + 1$.

8. Dadas las siguientes definiciones en pseudocodigo de `exp1` y `exp2`, calcular el trabajo de cada una de ellas y determinar que función es mas eficiente.

```
exp1 0 = 1
exp1 n = 2 * exp1 (n-1)

exp2 0 = 1
exp2 n = if   even n
         then square (exp2 (div n 2))
         else 2 * (exp2 (n-1))
```

9. Dados los siguientes pseudocodigos que implementan distintos algoritmos para invertir los elementos de una lista, calcular el trabajo de `reverse1` y `reverse2` y determinar que función es mas eficiente.

```
reverse1 :: [a] -> [a]
reverse1 []     = []
reverse1 (x:xs) = (reverse1 xs) ++ [x]

(++) :: [a] -> [a]
(++) [] ys     = ys
(++) (x:xs) ys = x : (xs ++ ys)

revStack :: [a] -> [a]
revStack [] ys     = ys
revStack (x:xs) ys = revStack xs (x:ys)
```

```haskell
reverse2 :: [a] -> [a]
reverse2 xs = revStack xs []
```

10. Dado el siguiente pseudocodigo de un algoritmo que construye un árbol binario a partir de una lista:

```haskell
data Tree a = Empty | Leaf a | Node (Tree a) (Tree a)

split :: [a] -> ([a], [a])
split []     = ([], [])
split [x]    = ([x], [])
split (x:xs) = let (ys, zs) = split xs
               in (x:ys, y:zs)

toTree :: [a] -> Tree a
toTree []       = Empty
toTree [x]      = Leaf x
toTree (x:y:xs) = let (ys, zs) = split (x:y:xs)
                      (t1, t2) = toTree ys ||| toTree zs
                  in Node t1 t2
```

   a) Expresar las recurrencias correspondientes al trabajo y a la profundidad de la función `toTree`, asumiendo que $W_{split}(n) = S_{split}(n) = O(n)$, siendo n la longitud de la lista que recibe.

   b) Resolver la recurrencia encontrada en el apartado anterior utilizando el teorema maestro. Expresar la solución utilizando la notación Θ.