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:

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

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

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

Soluciones

a) Nótese que para $n = 2^k$ resulta:

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

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

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

$$= n(a+b) - b \in \Theta(n)$$

- $\boxed{T(n) \in O(n)}$: 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 $n = 2, \ldots, \left\lfloor \frac{k+1}{2} \right\rfloor, \ldots, k$ 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 $n = 2^k$:

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

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

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

- $\boxed{T(n) \in O(n)}$: Probaremos por inducción que $0 \leq T(n) \leq cn \log_2(n)$:
  - Caso base $n = 2$: $T(n) = 2a + 2 = 2(a+1) \leq cn \log_2(n) = 2c \iff a + 1 \leq c$.
  - 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:

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

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

  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 $T(n) \in O\left[n \log_2(n)\right]$.
- $\boxed{T(n) \in \Omega(n)}$: COMPLETAR.

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

   Ayuda: Recuerde la siguiente propiedad: $a^{\log_b(n)} = n^{\log_b(a)}$.

   Solución

$$kn\begin{cases} k(n/2) & \begin{cases} k(n/4)\\ k(n/4)\\ k(n/4)\\ k(n/4)\end{cases} \quad \ldots T(1) \\ k(n/2) & \begin{cases} k(n/4)\\ k(n/4)\\ k(n/4)\\ k(n/4)\end{cases} \quad \ldots T(1) \\ k(n/2) & \begin{cases} k(n/4)\\ k(n/4)\\ k(n/4)\\ k(n/4)\end{cases} \quad \ldots T(1) \\ k(n/2) & \begin{cases} k(n/4)\\ k(n/4)\\ k(n/4)\\ k(n/4)\end{cases} \quad \ldots T(1) \end{cases} \qquad hojas = 4^{\log_2(n)} = n^{\log_2(4)} = n^2$$

$$altura = \log_2(n)$$

$$T(n) = kn + 4k(n/2) + 4^2k(n/4) + \ldots + n^2T(1) = \sum_{i=0}^{\log_2(n)-1} \frac{4^i kn}{2^i} + n^2T(1) =$$

$$= kn \sum_{i=0}^{\log_2(n)-1} 2^i + n^2T(1) = kn\frac{1-2^{\log_2(n)}}{1-2} + n^2T(1) = -kn(1-n) + n^2T(1) =$$

$$= kn^2 - kn + n^2T(1) \in \Theta\left(n^2\right)$$

- $\boxed{T(n) \in O(n)}$: Probaremos por inducción que $0 \leq T(n) \leq cn^2$:
  - Caso base $n = 1$: $T(n) = T(1) \leq cn^2 \iff T(1) \leq c$.
  - Caso inductivo: Supongamos que para $n = 1, \ldots, \left\lceil \frac{q}{2} \right\rceil, \ldots, q-1$ vale que $0 \leq T(n) \leq cn^2$. Luego:

    $$T(q) = 4T\left(\left\lceil \frac{q}{2} \right\rceil\right) + kq \leq 4c\left\lceil \frac{q}{2} \right\rceil^2 + kq \leq 4c\left(\frac{q+1}{2}\right)^2 + kq =$$

    $$= c(q+1)^2 + kq = cq^2 + 2cq + c + kq \leq cq^2 + 3cq + kq$$

    Finalmente $cq^2 + 3cq + kq \leq cq^2 \iff 3cq + kq \leq 0 \iff 3c + k \leq 0 \iff c \leq -\frac{k}{3}$.
- $\boxed{T(n) \in \Omega(n)}$: COMPLETAR.

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

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

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

Solución

$$k' + nk \to k' + (n-1)k \to k' + (n-2)k \to \ldots \to k'$$

$$T(n) = (n-a+1)k' + k\sum_{i=a+1}^{n} i = (n-a+1)k' + k\left[\frac{(n+a+1)(n-a)}{2}\right] =$$

$$= (n-a+1)k' + \frac{k}{2}\left[n^2 + na + n - na - a^2 - a\right] \in \Theta\left(n^2\right)$$

- $\boxed{T(n) \in O(n)}$: COMPLETAR.
- $\boxed{T(n) \in \Omega(n)}$: COMPLETAR.

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

   Soluciones

   a) El teorema maestro nos dice que:

   $$T(n) \in \begin{cases} \Theta\left[n^{\log_2(4)}\right] & \text{si } \exists\epsilon > 0/n \in O\left[n^{\log_2(4)-\epsilon}\right] \\ \vdots & \vdots \end{cases}$$

   y para $\epsilon = 1$ resulta $n \in O(n)$ luego $T(n) \in \Theta(n^2)$.

   b) El teorema maestro nos dice que:

   $$T(n) \in \begin{cases} \vdots & \vdots \\ \Theta\left[n^{\log_2(4)}\log_2(n)\right] & \text{si } n^2 \in \Theta\left[n^{\log_2(4)}\right] \\ \vdots & \vdots \end{cases}$$

   y como $n^2 \in \Theta(n^2)$ entonces $T(n) \in \Theta\left[n^2\log_2(n)\right]$.

   c) El teorema maestro nos dice que:

   $$T(n) \in \begin{cases} \vdots & \vdots \\ \Theta(n^3) & \text{si } \exists\epsilon > 0/n^3 \in \Omega\left[n^{\log_2(4)+\epsilon}\right] \\ & \text{y } \exists c < 1, N \in \mathbb{N}/\forall n > N \Rightarrow 4(n/2)^3 \leq cn^3 \end{cases}$$

   Sean $\epsilon = 1$, $c = 1/2$ y $N = 0$ luego $n^3 \in \Omega\left[n^{\log_2(4+\epsilon)}\right]$ y ademas $4(n/2)^3 = \frac{1}{2}n^3 \leq cn^3$ para cualquier $n > N$; por lo tanto $T(n) \in \Theta(n^3)$.

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

   a) $\ln(n)$.
   b) n^2.
   c) n^n.

Soluciones

a) Sabemos que $\ln(n)$ es estrictamente creciente y ademas para $b = e \geq 2$ resulta $\ln(bn) = 1 + \ln(n) \in O[\ln(n)]$ por lo tanto ln es suave.

b) Sabemos que para $n \geq 0$, n^2 es estrictamente creciente y ademas para $b = 2$ resulta $(2n)^2 \in O(n^2)$ por lo tanto n^2 es suave.

c) COMPLETAR.

7. Encontrar cotas asintóticas 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 $T(2^k) = T\left(\lfloor 2^{k/2} \rfloor\right) + 1$.

Soluciones

a) Como $1 \in \Theta\left[n^{\log_2(1)}\right]$, por el teorema maestro $T(n) \in \Theta\left[n^{\log_2(1)} \log_2(n)\right]$.

b) COMPLETAR.

c) COMPLETAR.

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))
```

Solución

- exp_1
  - $W_{exp1}(0) = 1$
  - $W_{exp1}(n) = 1 + W_{exp1}(n-1)$
  - $W_{exp1} \in \Theta(n)$
- exp_2
  - $W_{exp2}(0) = 1$
  - $W_{exp2}(n) = 1 + \begin{cases} 1 + W_{exp2}(n/2) & \text{n es par} \\ 1 + W_{exp2}(n-1) & \text{n es impar} \end{cases}$
  - Para potencias de 2, por el teorema maestro resulta $W_{exp2} \in \Theta[\log_2(n)]$ y como $\log_2$ es suave, podemos concluir que $W_{exp2} \in \Theta[\log_2(n)]$

El algoritmo mas eficiente es exp_2.

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.

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

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

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

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

Solución

- $reverse_1$
  - $W_{r1}(0) = 1$
  - $W_{r1}(n) = W_{r1}(n-1) + W_{++}(n-1) + 1 =$
    $= W_{r1}(n-2) + W_{++}(n-2) + 1 + W_{++}(n-1) + 1 =$
    $= \ldots = n + \sum_{i=0}^{n-1} \underbrace{W_{++}(i)}_{\in \Theta(n)}$
- $W_{r1} \in \Theta(n^2)$
- $reverse_2$
  - $W_{rs}(0) = 1$
  - $W_{rs}(n) = W_{rs}(n-1) + 1 \in \Theta(n)$
  - $W_{r2}(n) = W_{rs}(n) \in \Theta(n)$

La función mas eficiente es $reverse_2$.

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

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

split :: [a] -> ([a], [a])
split []     = ([], [])
split [x]    = ([x], [])
split (x:y: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.

Soluciones

a)

- $W_{tt}(0) = 1$.
- $W_{tt}(1) = 1$.
- $W_{tt}(n) = 2W_{tt}(n/2) + W_s(n)$.
- $S_{tt}(0) = 1$.
- $S_{tt}(1) = 1$.
- $S_{tt}(n) = S_{tt}(n/2) + S_s(n)$.

b) El teorema maestro nos dice que:

$$W_{tt}(n) \in \begin{cases} \vdots & \vdots \\ \Theta\left[n^{\log_2(2)} \log_2(n)\right] & \text{si } n \in \Theta\left[n^{\log_2(2)}\right] \\ \vdots & \vdots \end{cases}$$

y como $n \in \Theta(n)$ entonces $W_{tt}(n) \in \Theta[n \log_2(n)]$.
Ademas:

$$S_{tt}(n) \in \begin{cases} \vdots & \vdots \\ \Theta(n) & \text{si } \exists \epsilon > 0 / n \in \Omega\left[n^{\log_2(1+\epsilon)}\right] \\ & \text{y } \exists c < 1, N \in \mathbb{N} / \forall n > N \Rightarrow n/2 \leq cn \end{cases}$$

Sean $\epsilon = 1$, $c = 1/2$ y $N = 0$ luego $n \in \Omega\left[n^{\log_2(1+\epsilon)}\right]$ y ademas $n/2 =\leq cn = n/2$ para cualquier $n > N$; por lo tanto $S_{tt}(n) \in \Theta(n)$.