Práctica 1

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

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

2. Utilizar un árbol de recurrencia para obtener una cota asintótica para

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

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

3. 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.

1.

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

2.

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

Ayuda: Recuerde las propiedades:

- $\forall n, a, b \in \mathbb{Z}, a \neq 0 \wedge b \neq 0 \Rightarrow \lfloor \lfloor n/a \rfloor / b \rfloor = \lfloor n/(ab) \rfloor$
- $\forall n, a, b \in \mathbb{R}^+, a^{\log_b(n)} = n^{\log_b(a)}$

4. Utilizar el teorema Maestro para encontrar cotas asintóticas Θ para las siguientes recurrencias (asumir que $T(1) > 0$):

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

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

1. $ln(n)$
2. n^2
3. n^n

6. Utilice un árbol de recurrencia para encontrar una cota asintótica Θ para la recurrencia

$$T(n) = \begin{cases} a & n = 1 \\ 4T(\lceil n/2 \rceil) + cn & n > 1 \end{cases}$$

donde a y c son constantes positivas. Verifique que la cota encontrada es correcta. Ayuda: demostrar por suavidad.

7. Encontrar cotas asintóticas Θ para cada una de las siguientes recurrencias (asumir que $T(1) = a > 0$). Demostrar.

- $T(n) = T(n/2) + 1$
- $T(n) = T(n-1) + n$
- $T(n) = T(\lfloor \sqrt{n} \rfloor) + 1$

8. Dadas las siguientes definiciones en pseudocódigo de exp_1 y exp_2, calcular el trabajo de cada una de ellas y determinar qué función es más eficiente.

$$
\begin{array}{lcl}
exp_1\ 0 & = & 1 \\
exp_1\ (n+1) & = & 2 \times exp\ n
\end{array}
$$

$$
\begin{array}{lcl}
exp_2\ 0 & = & 1 \\
exp_2\ n & = & case\ par\ n\ of \\
 & & \quad true \to square\ (exp_2\ (div\ n\ 2)) \\
 & & \quad false \to 2 \times (exp_2\ (n-1))
\end{array}
$$

9. Dados los siguientes pseudocódigos que implementan distintos algoritmos para invertir los elementos de una lista, calcular el trabajo de $reverse_1$ y $reverse_2$ y determinar qué función es más eficiente.

$$
\begin{array}{lcl}
reverse_1 & : & [a] \to [a] \\
reverse_1\ [] & = & [] \\
reverse_1\ (x \lhd xs) & = & (reverse_1\ xs)\ @\ [x]
\end{array}
$$

$$
\begin{array}{lcl}
[]\ @\ ys & = & ys \\
(x \lhd xs)\ @\ ys & = & x \lhd (xs\ @\ ys)
\end{array}
$$

$$
\begin{array}{lcl}
revStack & : & [a] \to [a] \\
revStack\ []\ ys & = & ys \\
revStack\ (x \lhd xs)\ ys & = & revStack\ xs\ (x \lhd ys)
\end{array}
$$

$$
\begin{array}{lcl}
reverse_2 & : & [a] \to [a] \\
reverse_2\ xs & = & revStack\ xs\ []
\end{array}
$$

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

$$data\ Tree\ a\ =\ Empty\ \mid\ Leaf\ a\ \mid\ Node\ (Tree\ a)\ (Tree\ a)$$

$$
\begin{array}{lcl}
split & : & [a] \to [a] \times [a] \\
split\ [] & = & ([], []) \\
split\ [x] & = & ([x], []) \\
split\ (x \lhd y \lhd xs) & = & let\ \ (ys, zs)\ =\ split\ xs \\
 & & in\ \ (x \lhd ys, y \lhd zs)
\end{array}
$$

$$
\begin{array}{lcl}
toTree & : & [a] \to Tree\ a \\
toTree\ [] & = & Empty \\
toTree\ [x] & = & Leaf\ x \\
toTree\ (x \lhd y \lhd xs) & = & let\ \ (ys, zs)\ = split\ (x \lhd y \lhd xs) \\
 & & \qquad (t1, t2)\ = toTree\ ys\ ||\ toTree\ zs \\
 & & in\ \ Node\ t1\ t2
\end{array}
$$

1. 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.
2. Resolver la recurrencia encontrada en el apartado anterior utilizando el teorema Maestro. Expresar la solución utilizando la notación Θ.