Análisis de Algoritmos
Primer Problemario

Prof. Miguel A. Pizaña

1 de Marzo de 2013

I Tareas

1. Dudar de todo, al menos una vez en la vida.
2. ¿Que dice la paradoja de Zenón de Elea? ¿Qué significa "paradoja"? ¿Cómo se resuelve la paradoja de Zenón?
3. Leer *Computer Machinery and Intelligence* de Alan Turing. Luego conteste: ¿Qué es la prueba de Turing (Turing's Test)? ¿Usted cree que alguna máquina podrá algún día aprobar la prueba de Turing? Suponiendo que sí, ¿Eso indicaría que la máquina tiene inteligencia? ¿Puede el cerebro humano hacer algo que las Máquinas de Turing no? Según Turing, ¿pueden las máquinas pensar? ¿Qué argumentos en favor y en contra da Turing? Según usted, ¿pueden las máquinas pensar? ¿por qué?
4. ¿Quién es Richard Stallman? ¿Qué es el Manifiesto GNU? ¿Cuáles son sus principales puntos?

II Modelos de la Computación.

1. Describa los modelos: RAM de costo uniforme, RAM de costo logarítmico, Máquina de Turing, Máquina de Turing multicintas. ¿Cuáles son las ventajas y desventajas relativas estos modelos? ¿Cuál de ellos se apega más a la verdad? ¿Qué es la verdad? ¿Cuáles son los supuesto falsos en los que se basan los modelos mencionados? ¿Qué sentido, propósito o justificación tiene el hacer modelos matemáticos de la computación basados en premisas falsas? ¿Cuánto le cuesta a la máquina A emular a la máquina B (con A y B en la lista de modelos de arriba)?

III Notación Asintótica.

1. Demostrar el criterio del límite:

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = c < \infty \text{ implica } f(n) = O(g(n))$$

2. Demuestre que

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = c \text{ con } 0 < c < \infty \text{ implica } f(n) = \Theta(g(n))$$

3. Demuestre que $f(n) = o(g(n))$ implica $f(n) = O(g(n))$.
4. Muestre que $f(n) = O(g(n))$ implica $|f(n)| + |g(n)| = \Theta(g(n))$.
5. Muestre que $f(n) = o(g(n))$ implica $g(n) \pm f(n) = \Theta(g(n))$.
6. Dar un ejemplo de funciones $T(n)$ y $A(n)$ tal que $T(n) = o(A(n))$, pero $A(n) = o(T(n+1))$. ¿Qué tiene que ver esto con Zenón, Aquiles y la Tortuga ?
7. Dar un ejemplo de funciones f y g tal que $f(n) \neq O(g(n))$ y $g(n) \neq O(f(n))$.
8. Dar un ejemplo tal que $f(n) = O(g(n))$, pero que f y g no cumplan con las hipótesis del criterio del límite.
9. Dar un ejemplo tal que $f(n) = O(g(n))$, $g(n) \neq O(f(n))$, pero $f(n) \neq o(g(n))$.
10. Sean $a, b, c, d \in \mathbb{R}$ con $0 < a < b$ y $1 < c < d$. Para este problema, escribiremos $f \prec g$ en lugar de $f(n) = o(g(n))$. Muestre que: $\log(n)^a \prec \log(n)^b \prec n^a \prec n^a \log(n) \prec n^b \prec n^{\log(n)} \prec c^n \prec d^n \prec \log(n)^n \prec n! \prec n^n$
11. Muestre que $\log(n) = \Theta(\log(n+1))$.
12. Muestre que
$$\sum_{k=1}^{n} \frac{1}{k} = \Theta(\log(n))$$
13. Muestre que
$$\sum_{k=1}^{n} \log(k) = \Theta(n \log(n))$$

IV Complejidad Asintótica

1. Suponga que tengo dos algoritmos A y B para resolver el mismo problema. El algoritmo A es de tiempo $O(n)$ y que el algoritmo B es de tiempo $O(n^2)$. ¿Esto significa que A es más rápido que B? Explique. ¿Es posible que B sea *siempre* más rápido que A? Explique.
2. Sean $a, b > 1$. Suponga que tengo dos algoritmos A y B para resolver el mismo problema, el algoritmo A es de tiempo $\Theta(n^a)$ y el algoritmo B es de tiempo $\Theta(b^n)$. Muestre que $T_A(n) = o(T_B(n))$. ¿Esto significa necesariamente que A es mejor que B en la práctica? Explique.
3. Calcule asintóticamente (usando "$= \Theta$") el mejor tiempo, el peor tiempo y el tiempo promedio de los algoritmos de ordenación: Burbuja, MergeSort, QuickSort, RadixSort.
4. Muestre que $\Omega(n \log(n))$ es una cota inferior asintótica absoluta para cualquier algoritmo que ordene n elementos diferentes usando exclusivamente decisiones binarias. ¿Cómo es entonces que RadixSort puede ordenar en tiempo lineal?
5. ¿Es posible ordenar n elementos en tiempo lineal? ¿Bajo que premisas? Explique.
6. Calcule una fórmula cerrada para los números de Fibonacci: $F(n) = F(n-1) + F(n-2)$ con $F(0) = 1$, $F(1) = 1$. Sugerencia: Suponga que existe una solución a la recurrencia de la forma $F(n) = q^n$, encuentre los dos valores q_1, q_2 que hacen que ese supuesto sea cierto, demuestre por inducción que cualquier combinación lineal de la forma $c_1 q_1^n + c_2 q_2^n$ es en efecto una solución a la recurrencia, use los valores iniciales ($F(0) = 1$, $F(1) = 1$) para determinar los valores exactos que c_1 y c_2 deben tener.
7. El tiempo que tarda, el algoritmo recursivo para calcular números de Fibonacci es $T_F(n) = T_F(n-1) + T_F(n-2) + 1$ con $T_F(0) = T_F(1) = 1$. Proporcione una fórmula cerrada exacta para el tiempo de ejecución de este algoritmo en términos de los números de Fibonacci $F(n)$. Sugerencia: primero itere la recurrencia, luego observe el patrón que emerge y finalmente pruebe su conjetura por inducción matemática.