Práctica 1: Introducción a máquinas de Turing

Compilado: 19 de marzo de 2025

Órdenes

1. Para los siguientes pares de funciones $f, g : \mathbb{N} \to \mathbb{R}$ decidir si $g = o(f)$, $f = o(g)$ o $f = \Theta(g)$.[1]

   (a) $f(n) = n^2$, $g(n) = 2n^2 + 100\sqrt{n}$.
   (b) $f(n) = n^{100}$, $g(n) = 2^{n/100}$.
   (c) $f(n) = n^{100}$, $g(n) = 2^{n^{1/100}}$.
   (d) $f(n) = \sqrt{n}$, $g(n) = 2^{\sqrt{\log n}}$.
   (e) $f(n) = n^{20}$, $g(n) = 2^{\log^2 n}$.
   (f) $f(n) = 50n$, $g(n) = n \log n$.

2. Indicar, para cada una de las siguientes funciones definidas recursivamente, su orden de crecimiento. Es decir, para cada función f encontrar una función g definida de forma cerrada (i.e. no recursivamente) tal que $f = \Theta(g)$. De ser posible, encontrar una expresión cerrada para f.[2]

   (a) $f(n) = f(n-1) + 10$.
   (b) $f(n) = f(n-1) + n$.
   (c) $f(n) = 2f(n-1)$.
   (d) $f(n) = 2f(n/2) + 10$.
   (e) $f(n) = 2f(n/2) + n$.
   (f) $f(n) = 3f(n/2) + n^3$.

   Para el resto de los ejercicios, sea $F = \{f : \mathbb{N} \to \mathbb{R} \mid f \text{ es no decreciente}\}$ el conjunto de funciones no decrecientes. Definimos la relación $\sim_\Theta$ sobre F como $f \sim_\Theta g \iff f = \Theta(g)$. Equivalentemente definimos $\sim_o$, $\sim_O$, $\sim_w$ y $\sim_\Omega$.

3. Probar que $\sim_\Theta$ una relación de equivalencia, y que tiene infinitas clases de equivalencia.[3]

4. Decidir y demostrar para qué casos $\sim_\star$ es reflexiva (con $\star \in \{o, O, w, \Omega\}$). Hacer lo mismo para simetría y transitividad.

5. Probar que existen funciones f y g tales que $f \neq O(g)$ y $g \neq O(f)$.

6. Probar que para todo $k \geq 0$ y $c > 1$ vale que $n^k = o(c^n)$. Concluir que todo polinomio crece más lento que cualquier función exponencial.

7. Sea $Poly_\leq \subseteq F$ el conjunto de funciones acotadas por arriba por un polinomio (i.e. $Poly_\leq = \{f \in F : \exists k \geq 0 \text{ tal que } f \in O(n^k)\}$, y $Exp_\geq$ el conjunto de funciones acotadas por debajo por alguna función exponencial (i.e. $Exp_\geq = \{f \in F : \exists c > 1 \text{ tal que } f \in \Omega(c^n)\}$). Probar que $F \neq Poly_\leq \cup Exp_\geq$. Es decir, probar que existen funciones que crecen más rápido que cualquier polinomio pero más lento que cualquier función exponencial. **Ayuda**: considerar $f(n) = n^{\log n}$.

[1] En general podría no pasar ninguna de las 3, ver Ejercicio 5.
[2] Para todos los casos, asumir que $f(i) = 1$ para todo $i \leq 10$.
[3] Opcionalmente, probar que la cantidad de clases de equivalencia es no numerable.

8. Decidir si son ciertas las siguientes afimaciones:

   (*a*) Si $f(n) = \Theta(g(n))$ entonces $2^{f(n)} = \Theta(2^{g(n)})$.

   (*b*) $f(n) = 2^{\Theta(\log n)}$ si y solamente si $f(n)$ es un polinomio.[4]

Módelos de cómputo

9. Argumentar que los siguientes modelos de cómputo son polinomialmente equivalentes a una máquina de Turing de una sola cinta de trabajo infinita en un solo sentido. Estimar la complejidad de la simulación.

   (*a*) Máquina de Turing con una sola cinta de trabajo infinita en ambos sentidos.

   (*b*) Máquina de Turing con k cintas de trabajo, todas infinitas en ambas direcciones, con $k \geq 1$.

   (*c*) Máquina de Turing con acceso RAM (i.e. una máquina de Turing con una sola cinta de trabajo infinita en un solo sentido, que tiene una cinta especial adicional sobre la cual puede escribir, y tiene una instrucción especial que mueve el puntero de la cinta de trabajo a la dirección que está escrita en la cinta especial)[5].

   (*d*) Máquina de Turing con memoria bidimensional infinita (i.e. reemplaza la cinta infinita unidireccional por una matriz M con casilleros en el rango $[0, \infty) \times [0, \infty)$).

   De ahora en más, a menos que se diga lo contrario, las máquina de Turing que consideramos tienen una sola cinta de trabajo, usan un alfabeto binario y la cinta de trabajo es infinita en un solo sentido.

10. Probar por cardinalidad que hay lenguajes que no son computables. Definir uno por diagonalización.

11. Argumentar que toda máquina de Turing que use tiempo $T(n)$ puede ser simulada por otra máquina que sea *oblivious* y use tiempo $T(n)^2$.

12. Argumentar que toda máquina de Turing que use espacio $T(n)$ puede ser simulada por otra máquina que sea *oblivious* y use espacio $T(n)$.

13. Argumentar que toda máquina de Turing con alfabeto Γ con $|\Gamma| \geq 2$ que use tiempo $T(n)$ puede ser simulada por una máquina de Turing que use tiempo $T(n) \log |\Gamma|$.

[4] El conjunto $2^{\Theta(g(n))}$ es el conjunto de funciones $h \in F$ tales que existen constantes c_0 y c_1 con $2^{c_0 g(n)} \leq h(n) \leq 2^{c_1 g(n)}$ para todo n a partir de un cierto n_0. Notar que podríamos extender esta idea y definir la clase de funciones $f(\Theta(g(n)))$ o equivalentemente para o, O, w o Ω.

[5] Por simplicidad, asumimos que a través del acceso RAM el puntero solo puede moverse a posiciones de memoria ya visitadas. Caso contrario, la máquina termina con un `Segmentation Fault`.