

Clase Práctica de Órdenes y Complejidad Algorítmica

Alexis Tcach
Algoritmos y Estructuras de Datos II

Departamento de Computación,
Facultad de Ciencias Exactas y Naturales,
Universidad de Buenos Aires

23 de Septiembre de 2015

1 Notación de O , Θ y Ω

- Repaso y ejercitación de O
- Repaso y ejercitación de Θ
- Repaso y ejercitación de Ω

2 Álgebra de órdenes

- Simplificando las cuentas

3 Complejidad

- Funciones básicas
- Funciones con parámetros formales

Notación O

- $O(g) = \{f : \mathbb{N} \rightarrow \mathbb{N} \mid (\exists n_0, c)(\forall n > n_0) f(n) \leq cg(n)\}$
- $f \in O(g)$ si y sólo si $\exists n_0, c$ tales que para todo $n > n_0$
$$f(n) \leq cg(n)$$

Demostrar que si

- $f(n) = 4n^2 + 5n + 2$ entonces $f \in O(g)$ donde $g(n) = n^2$
- $f_1 \in O(g_1)$ y $f_2 \in O(g_2)$ entonces $f_1 + f_2 \in O(\max\{g_1, g_2\})$

Peligro!

Sea $f(n) = 2^n$. Veamos que para todo n dado, $f(n) \in O(1)$.

- Caso base: $f(1) = 2^1 = 2 = c_1 \in O(1)$
- Paso inductivo:
$$f(n+1) = 2^{n+1} = 2 \cdot 2^n \leq 2 \cdot c_n \cdot 1 = 2 \cdot c_n = c_{n+1} \in O(1)$$

¿Hay algún error?, ¿Cuál?

Notación

- $f \in O(g)$ si y sólo si $f(n) \leq cg(n)$ para todo $n > n_0$
- Notamos:
 - $f(n) = O(g(n))$, $f(n) = O(g)$, y $f \in O(g(n))$ si y sólo si $f \in O(g)$
 - $f(n) \neq O(g(n))$, $f(n) \neq O(g)$, y $f \notin O(g(n))$ si y sólo si $f \notin O(g)$
- Recordar:
 - $O(g)$ u $O(g(n))$ representa un **conjunto de funciones**.
- Ejemplos:
 - $n \log n = O(n^2)$
 - $n^n \neq O(n!)$

Cuidado: $f(n) = O(g(n)) \not\Rightarrow g(n) = O(f(n))$.

Múltiples parámetros

- $f : \mathbb{N}^k \rightarrow \mathbb{N}$, $f \in O(g)$ si y sólo si $f(\vec{n}) \leq cg(\vec{n})$ para todo $\vec{n} > \vec{n}_0$.
donde $(x_1, \dots, x_k) > (y_1, \dots, y_k)$ sii $x_i > y_i$ para todo $1 \leq i \leq k$.
- Otra vez, notamos:
 - $f(\vec{n}) = O(g(\vec{n}))$, $f(\vec{n}) = O(g)$, y $f \in O(g(\vec{n}))$ si y sólo si $f \in O(g)$
- Ejemplos:
 - $m \log n = O(mn)$

Parámetros vs. constantes.

- ¿Qué significa $f \in O(n^k)$?
- Hay que aclarar cuáles son las constantes.
- Lo que no es constante, es parámetro de f
- Ejemplos:
 - Para todo $k \in \mathbb{N}$, si $f \in O(n^k)$ entonces $f \in O(n^{k+1})$.
 - $n^k \in O(2^n)$ para todo $k \in \mathbb{N}$.
 - Si f es un polinomio, entonces $f \in O(n^k)$ para algún $k = O(1)$.

Notación Θ

- $\Theta(g) = \{f : \mathbb{N}^k \rightarrow \mathbb{N} \mid f \in O(g) \text{ y } g \in O(f)\}$
- $f \in \Theta(g)$ si y sólo si $\exists \vec{n}_0, c_0, c_1$ tales que para todo $\vec{n} > \vec{n}_0$

$$c_0 g(\vec{n}) \leq f(\vec{n}) \leq c_1 g(\vec{n})$$

Demostrar que si

- $4n^2 + 5n + 2 = \Theta(n^2)$

Notación Ω

- $\Omega(g) = \{f : \mathbb{N}^k \rightarrow \mathbb{N} \mid g \in O(f)\}$
- $f \in \Omega(g)$ si y sólo si $\exists \vec{n}_0, c$ tales que para todo $\vec{n} > \vec{n}_0$

$$cg(\vec{n}) \leq f(\vec{n})$$

Los ejercicios son similares.

Resumiendo...

- $f \in O(g)$ cuando f está acotada **superiormente** por g
- $f \in \Omega(g)$ cuando f está acotada **inferiormente** por g
- $f \in \Theta(g)$ cuando $f = O(g)$ y $f = \Omega(g)$.
- Hay que tener cuidado con las constantes.

$$O(1) \subset O(\log n) \subset O(n) \subset O(n^k) \subset O(k^n) \subset O(n!) \subset O(n^n)$$

Álgebra de órdenes

- ¿Qué significan las siguientes expresiones?
 - $O(f) + O(g) = O(h)$
 - $O(f) \cdot O(g) = O(h)$
 - $\sum_{i=1}^n O(n) = O(n^2).$

Álgebra de órdenes

- Definiciones.

- $O(f) + O(g) = O(f + g) = O(\max\{f, g\})$
- $O(f) \cdot O(g) = O(fg)$
- En general, $O(f) \bullet O(g) = O(f \bullet g)$.
- $\sum_{i=1}^k O(f) = O\left(\sum_{i=1}^k f\right) = O(kf)$.
- En particular, si k es constante, entonces $\sum_{i=1}^k O(f) = O(f)$
- $\prod_{i=1}^k O(f) = O\left(\prod_{i=1}^k f\right) = O(f^k)$.

Nota: $O(f) + O(g) \neq O(f) \cup O(g)$.

Álgebra de órdenes (ejercicios)

- Evaluar la validez de las siguientes ecuaciones (justificar):

- $\sum_{i=1}^n O(1) = O(n).$

- Si $k \in \mathbb{N}$, entonces $\sum_{i=1}^{2^k} O(n) = O(2^k n) = O(n).$

- Si $k \in \mathbb{N}$, entonces $\prod_{i=1}^n O(k) = O(1).$

Órdenes y expresiones

- ¿Qué significa $f(n) + O(n)$?
- Por ejemplo, merge sort toma tiempo $T(n) = 2T(n/2) + O(n)$?
- En general, si $f(n) = g(n) \bullet O(h(n))$ significa que
 - $f(n) = g(n) \bullet h'(n)$ para algún $h' \in O(h)$, i.e.
 - existen $c, n_0 \in \mathbb{N}$ tales que $f(n) \leq g(n) \bullet (c \cdot h(n)) \forall n \geq n_0$.
- Ejemplos:
 - $f(n) = 3n + O(\log n)$ significa $f(n) \leq 3n + c \log n$ para $c \in \mathbb{N}$.
 - $T(n) = 2T(n/2) + O(n)$ significa $T(n) \leq 2T(n/2) + cn$ para $c \in \mathbb{N}$.
 - $f(n) = n^{O(1)}$ significa $f(n) = n^c$ para $c \in \mathbb{N}$.

¿Qué es la complejidad de un algoritmo?

- **Función** para medir los **recursos** que consume un algoritmo
 - Tiempo
 - Memoria
 - Ancho de banda
 - Escrituras a disco, etc
 - **Operaciones elementales**
 - **Operaciones en general**
- Peor caso, mejor caso, caso promedio.

Sumatoria

SUM(**in** A : arreglo(nat)) $\rightarrow res: nat$

1. $res \leftarrow 0$
2. **for** $i \leftarrow 1$ **to** $tam(A)$:
3. $res \leftarrow res + A[i]$

Sumatoria exponencial

SUMEXP(**in** A : arreglo(nat)) $\rightarrow res: nat$

1. **var** i : nat
2. $res \leftarrow 0$
3. $i \leftarrow 1$
4. **while** $i \leq tam(A)$:
5. $res \leftarrow res + A[i]$
6. $i \leftarrow i * 2$

Búsqueda secuencial

BÚSQUEDASECUENCIAL(**in** A : arreglo(nat), **in** e : nat) $\rightarrow res:bool$

1. **var** i : nat
2. $i \leftarrow 1$
3. **while** $i \leq tam(A)$ **and** $A[i] \neq e$:
4. $i \leftarrow i + 1$
5. $res \leftarrow i < tam(A)$

Parámetros formales y subrutinas

- ¿Qué pasa si tenemos llamadas a subrutinas?
- Veámoslo en el pizarrón
- ¿Qué pasa si tenemos parámetros formales?
- Veámoslo en los ejemplos

Búsqueda secuencial

Parámetros formales

generos α

operaciones

$\bullet =_{\alpha} \bullet : \alpha \times \alpha \rightarrow bool$

BÚSQUEDASECUENCIAL(**in** A : arreglo(α), **in** e : α) $\rightarrow res:bool$

1. **var** i : nat
2. $i \leftarrow 1$
3. **while** $i \leq tam(A)$ **and** $A[i] \neq_{\alpha} e$:
4. $i \leftarrow i + 1$
5. $res \leftarrow i < tam(A)$

Resumiendo...

- Complejidad (modo AED 2): Contar *operaciones elementales* de los algoritmos.
- Peor caso vs. Mejor caso; no confundir con O vs. Ω .
- Las OE son las operaciones que provee una máquina RAM.
- **Suponemos** que las OE toman tiempo constante $O(1)$.
 - ¡En Algo 3 se cuestiona si esto es cierto!