Clase Práctica de Órdenes y Complejidad Algorítmica

Algoritmos y Estructuras de Datos II

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

16 de abril de 2018

Hoy en Algo 2

1. Notación de O, Θ y Ω
   - Repaso y ejercitación de O
   - Notación
   - Repaso y ejercitación de Ω
   - Repaso y ejercitación de Θ
   - Resumen
2. Propiedades
3. Álgebra de órdenes
   - Simplificando las cuentas
4. Ejercicios de órdenes
5. Complejidad
   - Funciones básicas
   - Un par de ejercicios
   - Funciones con parámetros formales
6. Ejercicios de Complejidad

Notación O

- $O(g) = \{f : \mathbb{N} \to \mathbb{N} \mid (\exists n_0, c)(\forall n > n_0)\ f(n) \leq cg(n)\}$
- $f \in O(g)$ 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 $f_1 + f_2 \in O(\max\{g_1, g_2\})$

Guarda

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:
  $f(n+1) = 2^{n+1} = 2.2^n \leq 2.c_n.1 = 2.c_n = c_{n+1} \in O(1)$

A la fresca, recórcholis, pamplinas, cielos, caramba. Hay algún error?, Cuál?

(Abusos de) 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 $(x_1, \ldots, x_k) > (y_1, \ldots, y_k)$ 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 Ω

- $\Omega(g) = \{f : \mathbb{N}^k \to \mathbb{N} \mid g \in O(f)\}$
- $f \in \Omega(g)$ si y sólo si $\exists \vec{n_0}, c$ tales que para todo $\vec{n} > \vec{n_0}$

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

Demostrar que

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

Notación Θ

- $\Theta(g) = \{f : \mathbb{N}^k \to \mathbb{N} \mid f \in O(g) \text{ y } g \in O(f)\}$
- $f \in \Theta(g)$ 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

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

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.

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

Propiedades

- Transitividad (de O, Ω y Θ)
- Reflexividad (de O, Ω y Θ)
- Simetría (de Θ)
- Simetría transpuesta (de O vs. Ω)
- (Antisimetría no vale)

Cuasi órdenes parciales

Para $\mathrm{f} \subseteqq \mathrm{g}$ si y sólo si $\mathrm{O}(\mathrm{f}) \subseteq \mathrm{O}(\mathrm{g})$

- Vale tricotomía? Es decir, este cuasi orden es total?
- Elemento máximo?
- Elemento mínimo?

Polinomios

Sea $P \in \mathbb{N}[n]$, $P = \sum_{i=0}^{d} a_i n^i$ con $a_d > 0$, y sea $k \in \mathbb{N}$.

- Si $k \geq d$, entonces $P \in O(n^k)$
- Si $k \leq d$, entonces $P \in \Omega(n^k)$
- Si $k = d$, entonces $P \in \Theta(n^k)$

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

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

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

Órdenes y expresiones

- Qué significa $f(n) + O(n)$?
- Por ejemplo, algún algoritmo de sort tiene complejidad $T(n) = 2T(n/2) + O(n)$?.
- 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 $c \in \mathbb{N}$.

Ejercicios para hacer en clase

Verdadero o Falso, justificando

- $2^n = O(n)$
- $7 = O(n)$
- $7 = O(1)$
- $n = O(n!)$
- Para todo $i, j \in \mathbb{N},\ in = O(jn)$
- $n = O(n-1)$
- $n + 999 = O(n)$
- $6n = O(n^2)$

Ejercicios para hacer en clase

Charlar

- Qué significa, intuitivamente, $\mathrm{O}(\mathrm{f}) \subseteq \mathrm{O}(\mathrm{g})$?
- $\mathrm{O}(\mathrm{n}^2) \cap \Omega(\mathrm{n}) = \Theta(\mathrm{n}^2)$?
- $\mathrm{O}(\mathrm{n}^2) \cap \Omega(\mathrm{n}) = \Theta(\mathrm{n})$?
- $\mathrm{O}(\mathrm{O}(\mathrm{f})) = \mathrm{O}(\mathrm{f})$? (ojo)

Ms ejercicios

- $O(\log n^2) = \ldots ?$
- $O(1 + \sin^2 n) = \ldots ?$
- $O(\log n) \subset O(\sqrt{n})$?
- $n + m = O(m)$?
- $n + m = O(m^2)$?
- $n + m = O(nm)$?
- $nm = O(n + m)$?
- $nm = O(n^2 + m^2)$?
- $n \log m + m \log n = O(n \log n + m \log m)$?
- $n \log n + m \log m = O(n \log m + m \log n)$?

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.

Cuidado: no confundir peor, mejor, promedio, con O, Ω, Θ

Sumatoria

```
Sum(in A: arreglo(nat)) → res:nat
    res ← 0
    for i ← 1 to tam(A):
        res ← res + A[i]
```

$$T(n) = c_1 + \sum_{i=1}^{tam(A)} c_2 = \Theta(n), \text{ donde } n = tam(A)$$

Sumatoria exponencial

```
SumExp(in A: arreglo(nat)) → res:nat
        var i: nat
        res ← 0
        i ← 1
        while i ≤ tam(A):
                res ← res + A[i]
                i ← i * 2
```

$$T(n) = c_1 + \sum_{i=1, i=2^j}^{\mathrm{tam}(A)} c_2 = \Theta(\log n), \text{ donde } n = \mathrm{tam}(A)$$

Búsqueda secuencial

```
BúsquedaSecuencial(in A: arreglo(nat), in e: nat) → res:bool
    var i: nat
    i ← 1
    while i ≤ tam(A) and A[i] ≠ e:
        i ← i + 1
    res ← i < tam(A)
```

"Peor caso, mejor caso, caso promedio."?

$$T_{peor}(n) = c_1 + \sum_{i=1, A[i] \neq e}^{n} c_2 =^{e \notin A} \Theta(n), \text{ donde } n = tam(A) + 1$$

$$T_{mejor}(n) = c_1 + \sum_{i=1, A[i] \neq e}^{n} c_2 =^{e = A[1]} \Theta(1), \text{ donde } n = tam(A) + 1$$

Un par de ejercicios con fors

```
ForFor1(in A: arreglo(nat))
1.    for i ← 1 to tam(A):
2.        for j ← 1 to 10:
3.            A[i] ← A[i] + A[i]
```

```
ForFor2(in A: arreglo(nat))
1.    for i ← 1 to tam(A):
2.        for j ← 1 to 1000000000000000:
3.            A[i] ← A[i] + A[i]
```

Otro par de ejercicios con fors

```
ForFor3(in A: arreglo(nat))
1.    for i ← 1 to tam(A):
2.        for j ← 1 to tam(A):
3.            A[i] ← A[i] + A[j]
```

```
ForFor4(in A: arreglo(nat))
1.    for i ← 1 to tam(A):
2.        for j ← i + 1 to tam(A):
3.            A[i] ← A[i] + A[j]
```

Parámetros formales y subrutinas

- Qué pasa si tenemos llamadas a subrutinas?
- Qué pasa si tenemos parámetros formales?
- Veámoslo en los ejemplos siguientes

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 ← 1
3. **while** i ≤ tam(A) **and** A[i] $\neq_\alpha$ e:
4.   i ← i + 1
5. res ← i < tam(A)

$$T_{peor}(n) = c_1 + \sum_{i=1,A[i]\neq e}^{n} c_2 + cmp_\alpha(A[i]) =^{e\notin A} \Theta(\sum_{i=1,A[i]\neq e}^{n} cmp_\alpha(A[i])) = O(n\max_{i\in[1..n]}(cmp_\alpha(A[i])))$$

$$T_{mejor}(n) = c_1 + \sum_{i=1,A[i]\neq e}^{n} c_2 + cmp_\alpha(A[I]) =^{e=A[1]} \Theta(cmp_\alpha(A[1])),$$

$$\text{donde } n = \text{tam}(A) + 1$$

Agregar atrás

Vector_nat se representa con *vector* donde
vector = *tupla*⟨ a: *arreglo(nat)*, t: *nat*⟩

```
AgregarAtrás(inout V: vector, in n: nat)
    if V.t > tam(V.a) then:
        nuevo : arreglo(nat)
        nuevo ← crear(2 ∗ tam(V.a))
        for i ← 1 to tam(V.a)
            nuevo[i] ← V.a[i]
        V.a ← nuevo
    V.a[V.t] ← n
    V.t ← V.t + 1
```

Calcule la complejidad de AgregarAtrás

Agregar muchos

Cuánto cuesta el algoritmo?

```
AgregarMuchos(inout V: vector, in n: nat)
1.      for i ← 1 to n
2.          AgregarAtrás(V, i)
```

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 $\mathrm{O}(1)$.
  - En Algo 3 se cuestiona si esto es cierto!