Sep 17

Complejidad

Esteban

Búsqueda de

- Eficiencia
  - Tiempo de ejecución ← + importante para la materia
  - Espacio (memoria) ← el que le sigue.
  - # de CPUs
  - % usado de red de comunicación

- Espacio: Se puede reutilizar
- Tiempo: "De un solo uso!"

Ansiedad.

Ejemplo

Búsqueda en Array

Búsqueda Sequencial

Algoritmo:

- I←1
- encontré ←falso
- Mientras ~encontré
  - Si A[I] = x entonces encontré ← verdadero
  - I←I+1
- print(I-1)

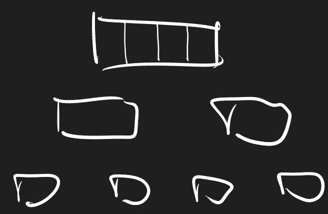
Cuánto tarda?

↳ Buscar (5, [2, 6, 3, 5, 8])

Cuánta memoria necesito?

Otro ejemplo

"Búsqueda Binaria"



Análisis Teórico

- Basado en un "modelo de máquina" de cómputo" consensuado
- En función del tamaño del input
- Para distintos tipos de input
- Análisis asintótico

o "modelo

1. Modelo de cómputo

Máquina Teórica ideal

de pasos/tareas/operaciones

Operaciones elementales

$t(I)$: # de operaciones elementales requeridas para la instancia I

Consideraremos OE las operaciones aritméticas básicas, comparaciones lógicas, transferencias de control, asignaciones a variables de tipos básicos, etc.

En función del tamaño de entrada

Análisis del caso:

↳ Peor $T_{peor}(n) = \max_{\text{instancias } I / |I| = n} \{ t(I) \}$

↳ Medio ($T_{peor}(n)$: tiempo de la peor instancia posible)

↳ Mejor $T_{mejor}(n) = \min_{\text{instancias } I / |I| = n} \{ t(I) \}$

Medio:

$$T_{prom}(n) = \mathbb{E}[X]$$

con X v.a. definida por todas las posibles ejecuciones del algoritmo para un tamaño de entrada dado.

$P(I)$: proba. de que el input sea la instancia I

$$T_{prom}(n) = \sum_{\text{instancias } I / |I| = n} \{ P(I) \cdot t(I) \}$$

Principio de Invarianza

- Dado un algoritmo y dos máquinas (o dos implementaciones) M_1 y M_2, que tardan $T_1(n)$ y $T_2(n)$ respectivamente sobre inputs de tamaño n, existe una cte. real $c > 0$ y un $n_0 \in N$ tales que $\forall\, n \geq n_0$ se verifica que:

$$T_1(n) \leq c T_2(n)$$

No necesitamos unidades de tiempo.

Orden de magnitud.

Análisis Asintótico

$$\begin{cases} OE \times N_2^2 = 1 \text{ hora} \\ \frac{OE}{100} \times N_x^2 = 1 \text{ hora} \end{cases} \Rightarrow \frac{OE}{100} \cdot N_x^2 = OE \times N_2^2$$

← 100 veces más rápidas las OE

$$N_x^2 = \frac{\cancel{OE} \times N_2^2}{\frac{\cancel{OE}}{100}}$$

N_2 : maximo tamaño de un problema resoluble en una hora para una compu. actual (complejidad n^2)

$$N_x = 10\, N_2^2$$

otro)

2^n:

$$\begin{cases} OE \times 2^{N_s} = 1 \text{ hora} \\ \frac{OE}{100} \times 2^{N_x} = 1 \text{ hora} \end{cases} \Rightarrow 2^{N_x} = 100 \cdot 2^{N_s}$$

$$\log_2(2^{N_x}) = \log_2(100 \cdot 2^{N_s})$$

$$N_x = \log_2(100) + N_s$$

$$\approx 6{,}644 + N_s$$

Comportamiento Asintótico

Medidas

- O ("big O") — O grande — cota superior
- Ω (omega) cota inferior
- Θ (theta) orden exacto

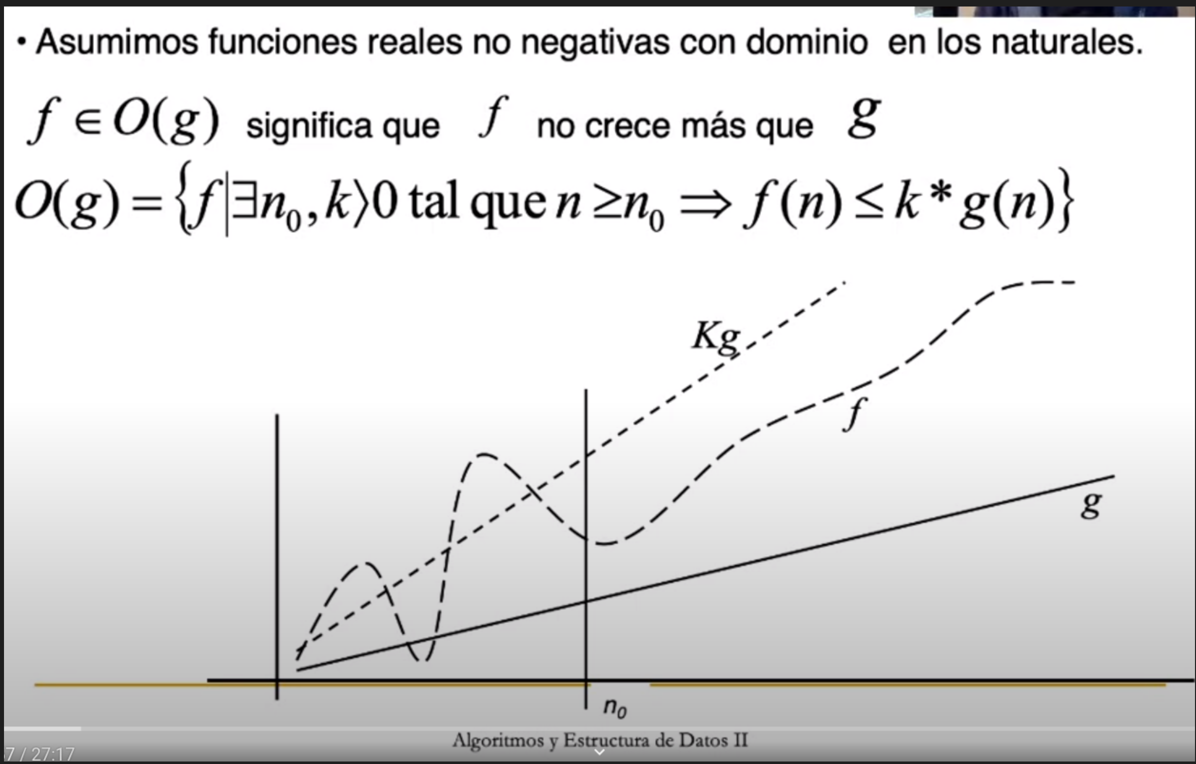
Cota Superior - Notación O

$f \in O(g)$:

"f no crece más rápido que alguna función proporcional a g"

$T_{peor} \in O(g)$:

En Todos los casos el tiempo será a lo sumo proporcional a la cota



ej:

$$100 \cdot n^2 + 300 \cdot n \in O(n^2)$$

Puedo decir que:

$$\exists n_0, K > 0 \ / \ n \geq n_0 \Rightarrow 100n^2 + 300n \leq K \cdot n^2$$

Veamos lo

$$100n^2 + 300n \leq K \cdot n^2$$

$$\cdot \frac{1}{n^2}$$

$$100 + \frac{300}{n} \leq K$$

Para cualquier no que elija, puedo encontrar el k

$n_0 = 1$:

$$K = 400$$

$n_0 = 2$:

$$K = 250$$

Propiedades de O

1. Para cualquier función f se tiene que $f \in O(f)$.
2. $f \in O(g) \Rightarrow O(f) \subset O(g)$.
3. $O(f) = O(g) \Leftrightarrow f \in O(g)$ y $g \in O(f)$.
4. Si $f \in O(g)$ y $g \in O(h) \Rightarrow f \in O(h)$.
5. Si $f \in O(g)$ y $f \in O(h) \Rightarrow f \in O(\min(g,h))$.
6. Regla de la suma:
   Si $f1 \in O(g)$ y $f2 \in O(h) \Rightarrow f1 + f2 \in O(\max(g,h))$.
7. Regla del producto: Si $f1 \in O(g)$ y $f2 \in O(h) \Rightarrow f1 * f2 \in O(g*h)$.
8. Si existe $\lim_{n\to\infty} \frac{f(n)}{g(n)} = k$, según los valores que tome k:

a) Si $k \neq 0$ y $k < \infty$ entonces $O(f) = O(g)$.
b) Si $k = 0$ entonces $f \in O(g)$, es decir, $O(f) \subset O(g)$, pero sin embargo se verifica que $g \notin O(f)$.

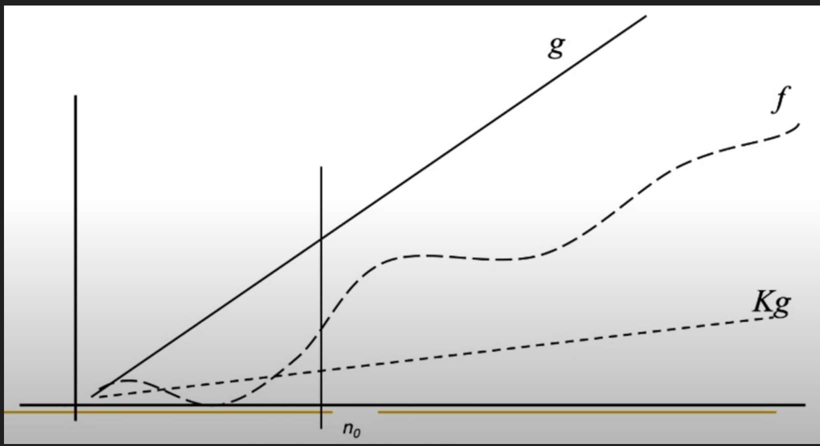
Notacion Ω ← omega

$f \in \Omega(g)$

A g se le llama Cota inferior de f

"f crece al menos como g"

$$\Omega(g) = \left\{ f \mid \exists\, n_0, k > 0 \;/\; n \geq n_0 \rightarrow f(n) \geq k \cdot g(n) \right\}$$



Propiedades

1. Para cualquier función f se tiene que $f \in \Omega(f)$.
2. $f \in \Omega(g) \Rightarrow \Omega(f) \subset \Omega(g)$.
3. $\Omega(f) = \Omega(g) \Leftrightarrow f \in \Omega(g)$ y $g \in \Omega(f)$.
4. Si $f \in \Omega(g)$ y $g \in \Omega(h) \Rightarrow f \in \Omega(h)$.
5. Si $f \in \Omega(g)$ y $f \in \Omega(h) \Rightarrow f \in \Omega(\max(g,h))$.
6. Regla de la suma: Si $f1 \in \Omega(g)$ y $f2 \in \Omega(h) \Rightarrow f1 + f2 \in \Omega(g+h)$.
7. Regla del producto: Si $f1 \in \Omega(g)$ y $f2 \in \Omega(h) \Rightarrow f1 * f2 \in \Omega(g*h)$.
8. Si existe $\displaystyle Lim_{n\to\infty} \frac{f(n)}{g(n)} = k$, según los valores que tome k:

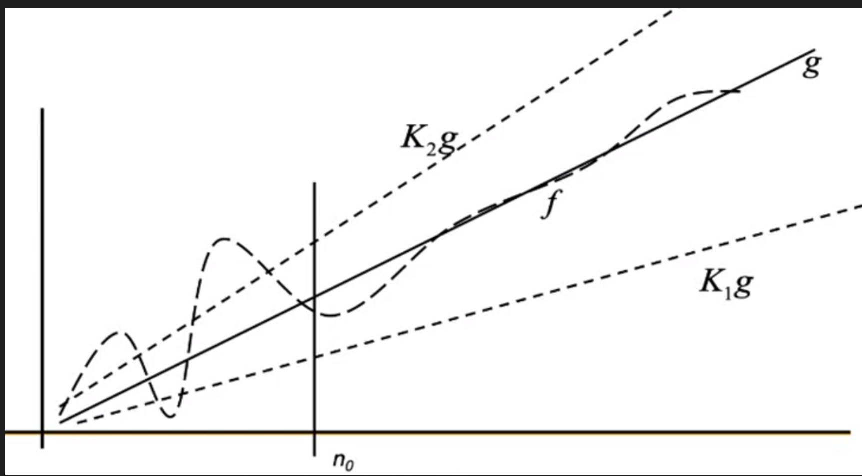
a) Si $k \neq 0$ y $k < \infty$. entonces $\Omega(f) = \Omega(g)$.
b) Si $k = 0$ entonces $g \in \Omega(f)$, es decir, $\Omega(g) \subset \Omega(f)$, pero sin embargo se verifica que $g \notin O(f)$.

Orden Exacto - Notación Θ

$$\Theta(f) = O(f) \cap \Omega(f)$$

$f \in \Theta(g)$ significa que f crece (a partir de cierto punto) igual que g

$$\Theta(g) = \{ f \mid \exists\, n_0, k_1, k_2 > 0 \;/\; n \geq n_0 \Rightarrow k_1 \cdot g(n) \leq f(n) \leq k_2 \cdot g(n) \}$$



Propiedades

1. Para cualquier función f se tiene que $f \in \Theta(f)$.
2. $f \in \Theta(g) \Rightarrow \Theta(f) = \Theta(g)$.
3. $\Theta(f) = \Theta(g) \Leftrightarrow f \in \Theta(g)$ y $g \in \Theta(f)$.
4. Si $f \in \Theta(g)$ y $g \in \Theta(h) \Rightarrow f \in \Theta(h)$.
5. Si $f \in \Theta(g)$ y $f \in \Theta(h) \Rightarrow f \in \Theta(\max(g,h))$.
6. Regla de la suma:
   Si $f1 \in \Theta(g)$ y $f2 \in \Theta(h) \Rightarrow f1 + f2 \in \Theta(\max(g+h))$.
7. Regla del producto: Si $f1 \in \Theta(g)$ y $f2 \in \Theta(h) \Rightarrow f1 * f2 \in \Theta(g*h)$.
8. Si existe $\lim\limits_{n \to \infty} \frac{f(n)}{g(n)} = k$,según los valores que tome k:

a) Si $k \neq 0$ y $k < \infty$. entonces $\Theta(f) = \Theta(g)$.
b) Si $k = 0$ entonces $\Theta(g) \neq \Theta(f)$.

Preguntar
Para qué usamos

Complejidad de Algoritmos Recursivos

$$T(n) = n + T(n-1)$$