

1. Introducción al análisis de algoritmos

- 1. Introducción al concepto de algoritmia**
- 2. Eficiencia de un algoritmo**
- 3. Notación asintótica**
- 4. Reglas generales**

Bibliografía

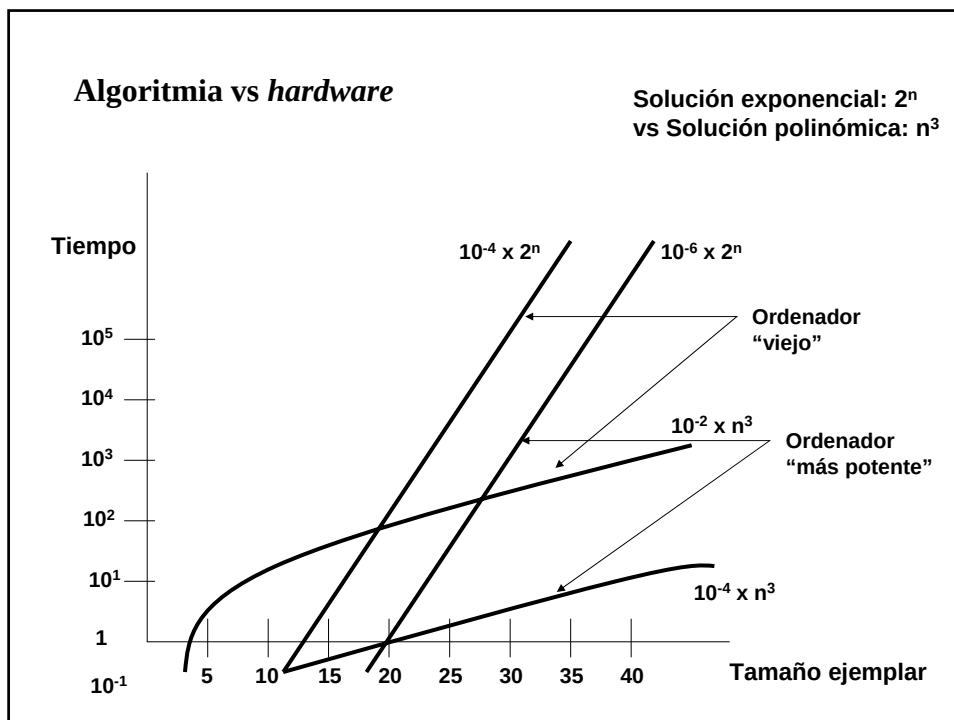
- Brassard y Bratley, 97; capítulos 2-4
- Weiss, 95; capítulo 2
- Knuth, 86 (Volumen 1); capítulo 1

1. 1. Introducción al concepto de algoritmia

- **Campo que estudia el análisis y diseño de algoritmos**
 - ¿Es realmente importante estudiar los algoritmos?
 - ¿Qué hemos estudiado hasta ahora?
 - Fundamentos de programación
 - Realizar programas correctos
 - ¿Es suficiente realizar “programas” correctos?
Ejemplos:
 - calcular el mínimo de tres números: número de comparaciones / asignaciones
 - ¿y si va dentro de un bucle que se repite N veces?
 - repetir operaciones de forma innecesaria

1. 1. Introducción al concepto de algoritmia

- ¿Qué vamos a estudiar?
 - Nuevas técnicas de diseño de algoritmos
 - Análisis de la eficiencia de las distintas soluciones
- ¿Cómo podemos saber si un algoritmo es eficiente?
 - Enfoque empírico
 - Enfoque teórico, con independencia de ordenadores concretos
- ¿Comparar dos versiones de un algoritmo?
 - Consumo de tiempo / memoria de procesador
 - El tamaño (de los ejemplares) sí importa



Evolución de las dos soluciones

n/seg.	1	10	20	30
N^3	1	1000	8000	27.000
2^n	2	1024	1.048.576	1.073.741.824

Hay 86400" en un día

8000" son 2,2 horas

1.048.576" aprox. 12,13 días

27.000" aprox. 7,5 horas

1.073.741.824" aprox 34 años

Sucesión de Fibonacci

n	10	20	30	50	100
Iterativa:	1/6 ms	1/3 ms	½ ms	3/4ms	11/2 ms
Rekursiva:	8 ms	1s	2'	21 días	10 ⁹ años

1. 2. Eficiencia de un algoritmo

- Cuándo es importante el
 - **Tiempo de ejecución**
 - Reservas on-line, disparo de alarmas,...
 - Mejor caso, caso promedio o peor caso
 - Alarmas (críticas) en centrales nucleares,
 - Análisis de la señal de un sismógrafo
 - **Consumo de memoria**
 - Manejo de bases de datos
 - Procesamiento de imágenes (p.e. de satélite)
 - Reconocimiento de patrones (cadenas de caracteres, voz)

1. 2. Eficiencia de un algoritmo

- ¿Sobre algún ordenador concreto/estándar? No
 - Usaremos un **Modelo**:
 - Ordenador secuencial
 - Repertorio estándar de operaciones (sencillas): asignación, comparación, operadores aritméticos,...
 - Todas las operaciones sencillas tienen el mismo coste
 - Capacidad de memoria infinita
 - Ejemplo 1:
 - distintos tamaños,
 - distintas instrucciones,
 - distintos tiempos de respuesta.

1. 2. Eficiencia de un algoritmo

- Concepto de **eficiencia** será función de
 - **Tiempo de ejecución**, que es función de las operaciones elementales
Ejemplo 2: búsqueda secuencial vs dicotómica
 - **Tamaño de los ejemplares** (memoria/representación)
Ejemplo 3: términos de la serie de Fibonacci
- Ambos factores están relacionados
 - Un ejemplar de tamaño o valor grande puede hacer que una operación no sea elemental, que no pueda representarse,...
 - Una solución recursiva o ineficiente puede consumir muchos recursos

1. 3. Notación asintótica

- **$t(n)$:** $\mathbb{N} \rightarrow \mathbb{R}^+$
recursos (tiempo/espacio) que consume un algoritmo para solucionar un ejemplar de tamaño **n**
- **Principio de Invarianza**
 - Dos implementaciones distintas de un mismo algoritmo no diferirán más que en alguna constante multiplicativa
- **Notación asintótica:**
concerniente al funcionamiento (tiempo de ejecución/número de instrucciones ejecutadas) de los algoritmos en casos límite (para ejemplares suficientemente grandes)

1. 3. 1. Notación $O()$: en el / del orden de

- $f(n)$ y $g(n)$ dos funciones monótonas crecientes
- $n, n_0 \in \mathbb{Z}^+$
- $c \in \mathbb{R}^+$

Conjunto de cotas superiores: $O(\cdot)$

$f(n) \in O(g(n)) \Leftrightarrow \exists c > 0, n_0 > 0: \forall n > n_0: f(n) \leq c \cdot g(n)$
(¡¡cuidado con las constantes multiplicativas!!)

Notaciones más usadas

- $\log n$, *logarítmica*
- n , *lineal*
- n^2 , *cuadrática* n^3 , *cúbica* n^k , *polinómica*
- 2^n , *exponencial*
- $n!$, *factorial*

Expresiones útiles:

- $f(n) \in O(f(n))$
- $O(f(n)) + O(g(n)) \equiv O(f(n) + g(n)) \equiv O(\max(f(n), g(n)))$
- $k \cdot O(f(n)) \equiv O(k \cdot f(n)) \equiv O(f(n))$
- $O(f(n)) \cdot O(g(n)) \equiv O(f(n) \cdot g(n))$
- $O(a_0 + a_1 \cdot n + \dots + a_d \cdot n^d) \in O(n^d)$

1. 3. 2. Notaciones $\Omega()$ y $\Theta()$

Conjunto de cotas inferiores: $\Omega(\cdot)$

$$f(n) \in \Omega(g(n)) \Leftrightarrow \exists c > 0, n_0 > 0: \forall n > n_0: f(n) \geq c \cdot g(n)$$

Conjunto de cotas estrictas: $\Theta(\cdot)$

$$f(n) \in \Theta(g(n)) \Leftrightarrow f(n) \in O(g(n)) \text{ y } f(n) \in \Omega(g(n))$$

Jerarquías

- $O(1) \subset O(\lg n) \subset O(n^{1/2}) \subset O(n) \subset O(n \cdot \lg n) \subset O(n^2) \subset O(2^n) \subset O(n!)$
- $\Omega(1) \supset \Omega(\lg n) \supset \Omega(n^{1/2}) \supset \Omega(n) \supset \Omega(n \cdot \lg n) \supset \Omega(n^2) \supset \Omega(2^n) \supset \Omega(n!)$
- $\Theta(1) \neq \Theta(\lg n) \neq \Theta(n^{1/2}) \neq \Theta(n) \neq \Theta(n \cdot \lg n) \neq \Theta(n^2) \neq \Theta(2^n) \neq \Theta(n!)$

1. 4. Reglas generales

- Secuencias
 $O(f(n)) + O(g(n)) \equiv O(f(n) + g(n)) \equiv O(\max(f(n), g(n)))$
- Anidamiento
 $O(f(n)) \cdot O(g(n)) \equiv O(f(n) \cdot g(n))$
- Bucles para
 $O(c \cdot n)$

Ej1: para i desde 1 hasta m hacer

 p(i)

fin para

Ej2: para i desde 1 hasta m hacer

 para j desde i+1 hasta m hacer

 intercambia(A(i,j), A(j,i));

 finpara

finpara

1. 4. Reglas generales (cont.)

- Bucles mientras/repetir
inf := 1; sup := m;
mientras inf < sup hacer
 med := (inf + sup) div 2;
 si v(med) = x entonces
 inf := med;
 sup := med;
 sino
 si v(med) > x entonces
 sup := med - 1;
 sino
 inf := med + 1;
 finsi
fin mientras
- Esquemas recursivos

Trabajo

- Introducción al concepto de Algoritmia
- ¿Por qué es importante?
- Medidas de la eficiencia (teórica) de un algoritmo:
 $O(\cdot)$, $\Omega(\cdot)$, $\Theta(\cdot)$
- Aplicarla a dos esquemas de búsqueda: secuencia y binaria
- Probarlo en C:
 - Generar un vector ordenado de 1000 elementos
 - Generar aleatoriamente 100 números
 - Buscarlos y calcular el número medio de iteraciones