Estructura de datos y algoritmos

Rodrigo Alvarez
rodrigo.alvarez2@mail.udp.cl

Problema computacional

Un problema computacional se puede describir como un enunciado que describe la relación entre un conjunto de datos de entrada y otro de salida.

Algoritmo

Un algoritmo es un conjunto de instrucciones finitas y bien definidas que transforman cada instancia de entrada de un problema en un valor de salida especifico.

Análisis de algoritmos

- Funcionamiento correcto
- Eficiencia en el tiempo de ejecución
- Eficiencia en el espacio

Métricas de comparación de algoritmos

- Tiempo de ejecución
- Espacio en memoria
- Uso de procesadores
- Uso de dispositivos de entrada/salida

Es importante tener en cuenta que la **elección de un algoritmo** o una implementación **depende** en gran medida del **contexto** y el **hardware** a utilizar.

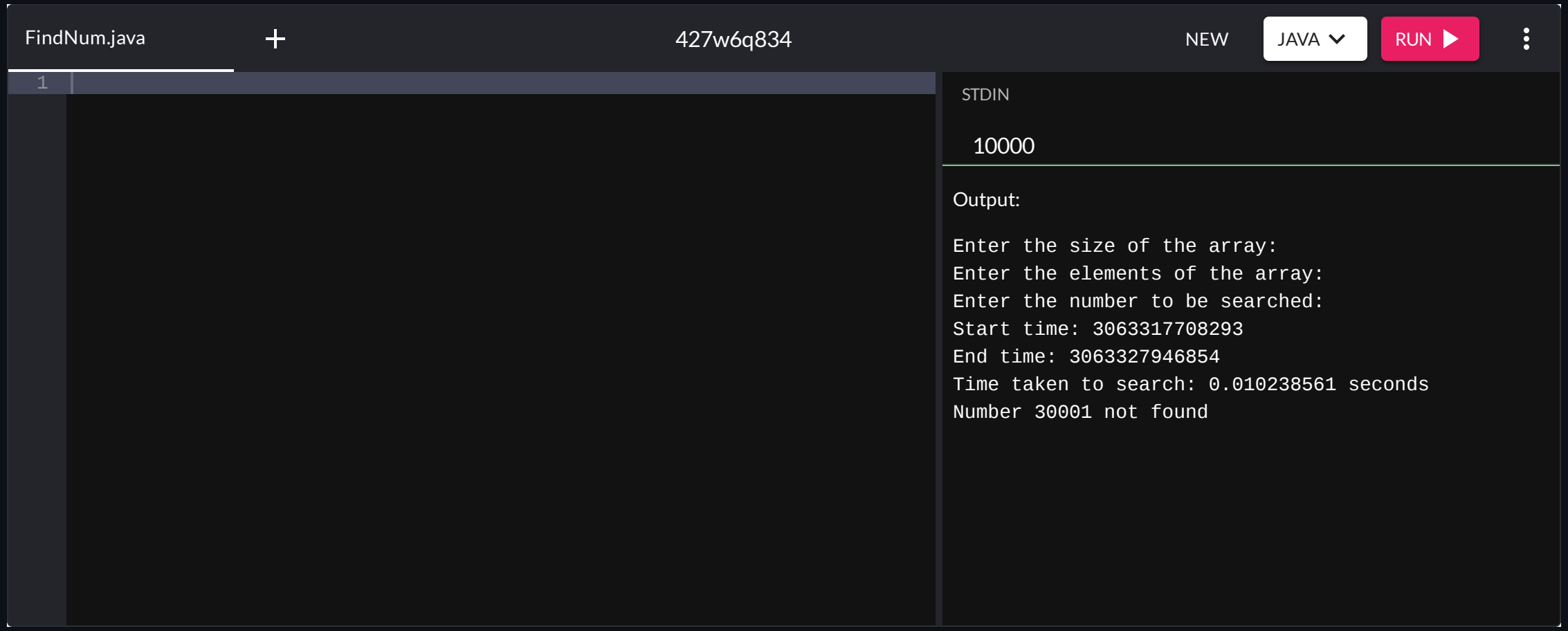
Análisis de complejidad temporal

El análisis de complejidad temporal es una técnica que permite determinar cuánto tiempo tardará un algoritmo en ejecutarse en función del tamaño de los datos de entrada.

Empírica o experimental

Consiste en medir el tiempo de ejecución en una computadora específica para una entrada dada.

Ejemplo de análisis empírico



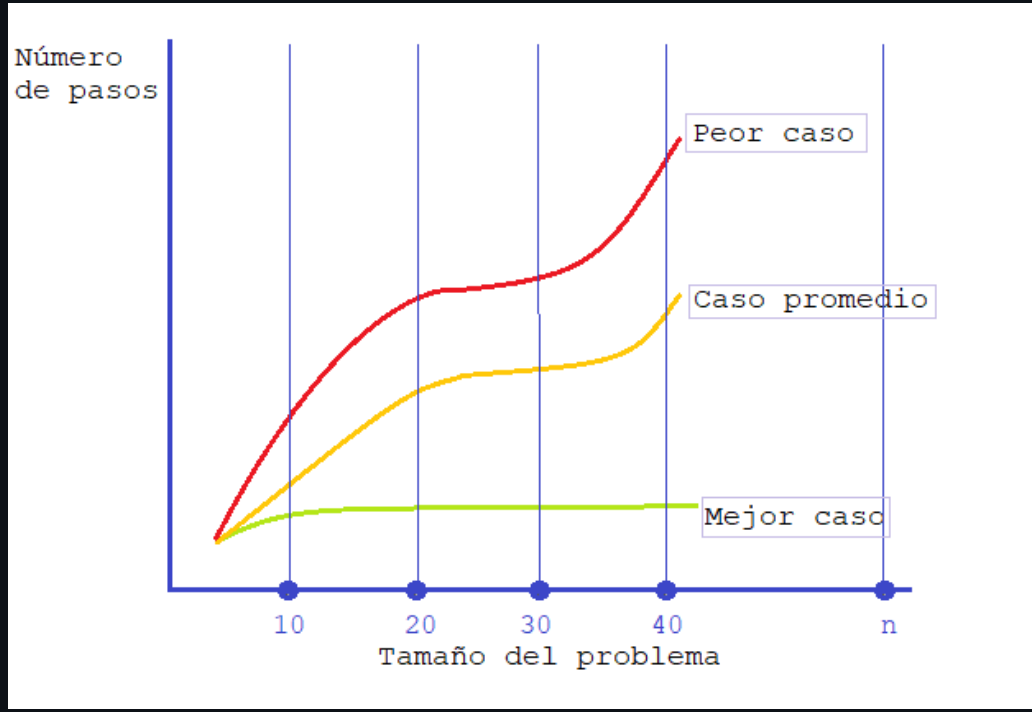
Análisis teórico

El análisis de complejidad es una herramienta fundamental para el diseño y optimización de algoritmos. Este análisis es basado en un "modelo de máquina" o "modelo de cómputo consensuado" se presenta como una medida "universal" válida para distintas implementaciones del algoritmo.

- La **medida del tiempo** se refiere al número de instrucciones u operaciones elementales (OE) que se ejecutan en la máquina "ideal" para determinado input en función del tamaño del mismo.
- La **medida espacial** se refiere a la cantidad de memoria necesaria para resolver un problema en la máquina ideal.

Análisis de los distintos casos

Diferentes entradas de la misma longitud pueden causar que el algoritmo se comporte distinto, por lo que se podría analizar el algoritmo desde tres perspectivas: **el mejor caso**, **el caso promedio** y **el peor caso**.



Operaciones elementales

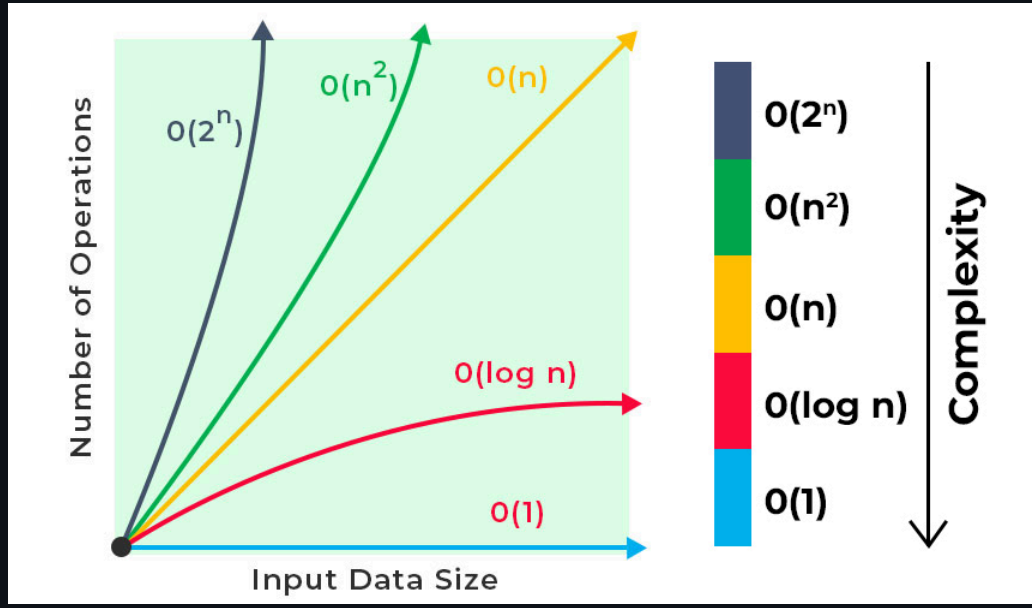
Las operaciones elementales son aquellas operaciones básicas que el procesador puede realizar en un tiempo acotado por una constante, es decir, que no dependen del tamaño de la entrada. Estas operaciones incluyen las operaciones aritméticas básicas como la suma, resta, multiplicación y división, las comparaciones lógicas como mayor que, menor que, igual a, y las asignaciones a variables de tipos básicos como enteros, flotantes o caracteres.

Es importante tener en cuenta que el conjunto de operaciones elementales puede variar según el modelo de cómputo que se utilice.

Análisis asintótico

El análisis asintótico es una técnica de análisis que permite describir la complejidad espacial o temporal de un algoritmo mediante el orden de crecimiento.

El orden de crecimiento se caracteriza mediante una función matemática, esta es la que describe la relación entre tiempo y tamaño de la entrada, o uso de memoria y tamaño de la entrada.



- ¿Por qué quiero algoritmos que sean eficientes asintóticamente?

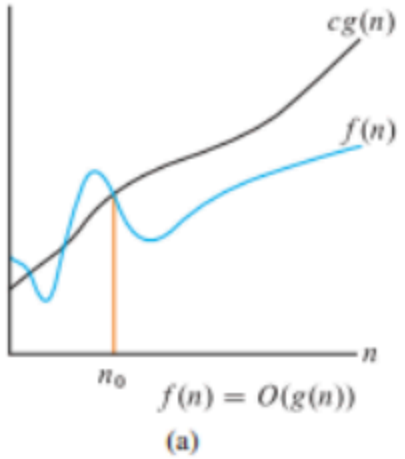
	Tamaño n					
Complejidad Temporal	10	20	30	40	50	60
n	.00001 segundos	.00002 segundos	.00003 segundos	.00004 segundos	.00005 segundos	.00006 segundos
n^2	.0001 segundos	.0004 segundos	.0009 segundos	.0016 segundos	.0025 segundos	.0036 segundos
n^3	.001 segundos	.008 segundos	.027 segundos	.064 segundos	.125 segundos	.216 segundos
n^5	.1 segundos	3.2 segundos	24.3 segundos	1.7 minutos	5.2 minutos	13.0 minutos
2^n	.001 segundos	1.0 segundos	17.9 minutos	12.7 días	35.7 años	366 siglos
3^n	.059 segundos	58 minutos	6.5 años	3855 siglos	$2x10^8$ siglos	$1.3x10^{13}$ siglos

Comparación de algoritmos con distintas complejidades, polinomiales y exponenciales

Fuente: Aho, Hopcroft, Ullman

Big O

$\mathcal{O}(g(N)) = \{f(N)$: existen constantes positivas c y N_0 de forma que $0 \leq f(N) \leq cg(N)$ para todo $N \geq N_0$ $\}$

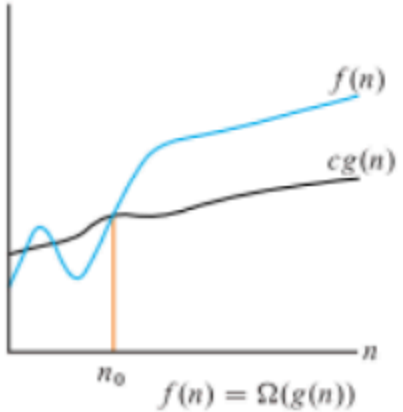


Big O

- Caracteriza la cota superior de una función, es decir que la función en cuestión no crece más rápido que un cierto ratio.
- Esta notación se usa para caracterizar el peor caso de tiempo de ejecución o memoria para un algoritmo.
- Digamos que un determinado algoritmo tiene tiempo de ejecución en función de $T(N) = 7N^3 + 5N^2 - 20N + 7$. Esto en términos de Big O podría traducirse como $T(N) = O(N^3)$.
- La razón es que $7N^3 + 5N^2 - 20N + 7 \leq cN^3$ con c como una constante positiva.

Big Omega

$\Omega(g(N)) = \{f(N)$: existen constantes positivas c y N_0 de forma que $0 \leq cg(N) \leq f(N)$ para todo $N \geq N_0\}$

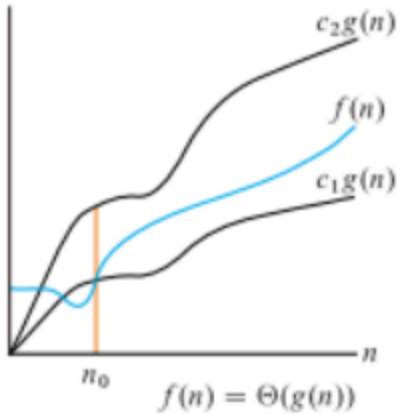


Big Omega (Ω)

- La notación Notación Ω, conocida como Omega caracteriza la cota inferior de una función.
- Para el caso del ejemplo $T(N) = 7N^3 + 5N^2 - 20N + 7$, podemos decir que en términos de notación Omega el tiempo de ejecución es en magnitud de $\Omega(N^3)$, $\Omega(N^2)$ o incluso $\Omega(1)$, siempre y cuando exista un N para ese caso.

Big Theta

$\Theta(g(N)) = \{f(N)$: existen constantes positivas c_1, c_2 y N_0 de forma que $0 \leq c_1 g(N) \leq f(N) \leq c_2 g(N)$ para todo $N \geq N_0$ $\}$



(c)

Big Theta (Θ)

- La notación Notación Θ, conocida como Theta caracteriza la cota ajustada de una función.
- Para el caso del ejemplo $T(N) = 7N^3 + 5N^2 - 20N + 7$, si $T(N) = \Omega(N^3)$ y $T(N) = O(N^3)$, entonces se puede decir $T(N) = \Theta(N^3)$.

Como analizar algoritmos

Guidelines for Asymptotic Analysis

Máquina de turing