Introducción al análisis de algoritmos

Pedro O. Pérez M., MTI

Análisis y diseño de algoritmos
Tecnológico de Monterrey

pperezm@tec.mx

01-2019

Contenido

Un poco de historia

- En 1834 Charle Babbage ya había concebido, pero de forma mecánica, el concepto de un algoritmos mediante su "Maquina Analítica".
- El concepto matemático formal de algoritmo fue formulado en 1930 (antes de la aparición de computadoras).
- Alan Turing tuvo gran peso durante el proceso de formalización del algoritmo.
- John Von Neumann también hizo importante contribuciones.

¿Qué es un algoritmo?

Algoritmo: (Quizá del lat. tardío *algobarismus, y éste del árabe clásico *hisabu lgubar, cálculo mediante cifras arábigas).

- Conjunto ordenado y finito de operaciones que permite hallar la solución de un problema.
- Método y notación en las distintas formas del cálculo.

Real Diccionario de la Lengua Española

Un algoritmo es una secuencia de instrucciones inequívocas para la solución de un problema , es decir , para la obtención de una salida requerida para cualquier entrada legítima en una cantidad finita de tiempo.

Introduction to the Design and Analisys of Algorithms, Anany Levitin

Un algoritmo es un procedimiento computacional bien definido que toma algún valor, o conjunto de valores, como entrada y produce un cierto valor, o conjunto de valores, como salida. Un algoritmo es por lo tanto una secuencia de pasos computacionales que transforman una dada entrada en una salida.

Introduction to Algorithms, Thomas H. Cormen

En matemáticas, ciencias de la computación, y disciplinas relacionadas, un algoritmo (del latín, dixit algorithmus y éste a su vez del matemático persa al-Jwarizmi) es una lista bien definida, ordenada y finita de operaciones que permite hallar la solución a un problema. Dado un estado inicial y una entrada, a través de pasos sucesivos y bien definidos se llega a un estado final, obteniendo una solución. Los algoritmos son objeto de estudio de la algoritmia, y su definición queda formalizada por el modelo computacional de la Máquina de Turing.

Wikipedia

¿Cuáles son sus características?

- Específico o bien definido.
- Conjunto de datos de entrada.
- Conjunto de datos de salida.
- Finito.
- Preciso.
- Eficaz.
- Correcto.

¿Porqué es importante estudiar los algoritmos?

- Internet: Búsqueda web, ruteo de paquetes, archivos compartidos.
- Biología: Proyecto del genoma humano, análisis de proteínas.
- Computadoras: Circuitos eléctricos, sistemas de archivos, compiladores.
- Gráficas computacionales: Películas, vídeo juegos, realidad virtual.
- Seguridad: Teléfonos celulares, comercio electrónico.
- Transporte: Calendarización de vuelos, selección de rutas.
- Física: Simulación de colisiones de partículas.

- Viejos algoritmos, nuevas oportunidades.
- La posibilidad de resolver problemas de formas no exploradas.
- Es un reto intelectual.
- Gráficas computacionales: Películas, vídeo juegos, realidad virtual.
- Poder desbloquear los secretos del universo.
- Por diversión y dinero.

¿Cómo analizamos los algoritmos?

- Cuando tenemos varios algoritmos para resolver un mismo problema, necesitamos una forma de determinar la mejor opción.
- La respuesta es el análisis asintótico de complejidad.
- Pero, ¿qué es la complejidad de un algoritmo?
  - Es la medida de los recursos que necesita un algoritmo para su ejecución.
  - Complejidad temporal: El tiempo que necesita un algoritmo para terminar su ejecución.
  - Complejidad espacial: La cantidad de memoria que requiere un algoritmo durante su ejecución.

- El tiempo de ejecución de un algoritmo depende de:
  - Factores externos: La computadora donde se va a realizar la ejecución, el compilador (o interprete) usado, la experiencia del programador, los datos de entrada.
  - Factores internos: El número de instrucciones asociadas al algoritmo.
- Entonces, ¿cómo podemos estudiar el tiempo de ejecución del algoritmo?

- Análisis empírico (a posteriori):
  - Generando ejecuciones del algoritmo para distintos valores de entrada y cronometrando el tiempo de ejecución.
  - Factores internos: Los resultados dependen de factores externos e internos.
- Análisis analítico (a priori):
  - Obtener una función que represente el tiempo de ejecución del algoritmo para cualquier valor de entrada.
  - Depende solo de los factores internos.

¿Big Ω, Big Θ?, Big O?

- Cuando analizamos un algoritmos debemos tener en cuenta tres situaciones:
  - El mejor de los casos (Cota inferior - $\Omega(n)$)
  - El caso promedio (Cota promedio - $\Theta(n)$)
  - El peor de los casos (Cota superior - $O(n)$)

Jerarquía de los algoritmos

Notación O	**Nombre**
$O(1)$	Constante
$O(\log \log(n))$	log log
$O(\log(n))$	Logarítmica
$O(n)$	Lineal
$O(n \log n)$	n log n
$O(n^2)$	Cuadrática
$O(n^3)$	Cúbica
$O(n^m)$	Polinomial
$O(m^n)$ m >= 2	Exponencial
$O(n!)$	Factorial

Complejidad vs. tiempo

N	**10**	**100**	**1,000**	**10,000**	**100,000**
$O(1)$	1 μs	1 μs	1 μs	1 μs	1 μs
$O(\log n)$	3 μs	7 μs	10 μs	13 μs	17 μs
$\sqrt{n}$	3 μs	10 μs	31 μs	100 μs	316 μs
n	10 μs	100 μs	1,000 μs	10,000 μs	100,000 μs
$n \log n$	33 μs	664 μs	10,000 μs	133,000 μs	1.6 *seg*
n^2	100 μs	10,000 μs	1 *seg*	1.7 min	16.7 min
n^3	1 *ms*	1 *seg*	16.7 min	11.6 *dia*	31.7 *año*
2^n	1.024 *ms*	$4*10^{16}$ *año*	$3.39*10^{287}$ *año*	…	…
$n2^n$	10.24 *ms*	$4*10^{18}$ *año*	…	…	…
$n!$	4 *seg*	$2.95*10^{144}$ *año*	…	…	…

Problema P (Polinomial)

- Informalmente, es la clase de problemas de decisión resolubles por algún algoritmo dentro de un número de pasos delimitadas por algunos polinomio fijo en la longitud de la entrada.
- Formalmente se define como los problemas de decisión que pueden ser resueltos por una máquina de Turing determinista utilizando una cantidad de tiempo polinomial, o tiempo polinomial.
- Ejemplos: Máximo común divisor, determinar si un número es primo, el camino más corto de A a B en un grafo, el ciclo Euleriano en un grafo.

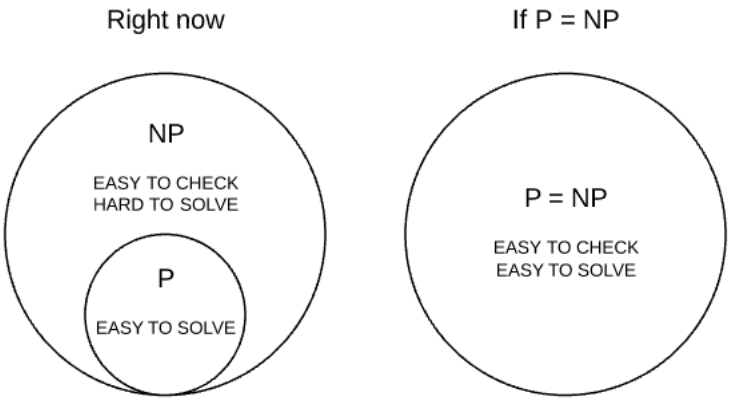
Problema NP (Non-Polinomial)

- Informalmente, es el conjunto de problemas cuyas soluciones se puede verificar en tiempo polinomial. Pero por lo que cualquiera puede ver, muchos de esos problemas llevan tiempo exponencial de resolver.
- Formalmente, es el conjunto de los problemas de decisión en el que "síinstances pueden aceptarse en tiempo polinómico por una máquina de Turing no determinista.
- Ejemplos: Encontrar los factores primos de un número entero muy largo, problema del agente viajero, el problema de satisfactibilidad de ecuaciones booleanas.

Familia de problemas NP

- NP-Hard: Clase de problemas que son al menos tan duro como los problemas más difíciles en NP . Los problemas en NP- duro no tienen que ser elementos de NP, de hecho, que ni siquiera pueden ser problemas decidibles.
- NP-Complete: Clase de problemas que contiene los problemas más difíciles en N . Cada elemento de NP-completo tiene que ser un elemento de NP.
- NP-Easy: A lo sumo tan duro como NP, pero no necesariamente en NP, ya que puede que no sean los problemas de decisión.
- NP-Equivalent: Exactamente tan difícil como los problemas más difíciles en NP, pero no necesariamente en NP.

P vs. NP



Referencia: `https://goo.gl/JnAHbP`