

Introducción al curso

ANÁLISIS Y DISEÑO DE ALGORITMOS

MTI. PEDRO O. PÉREZ MURUETA

TECNOLÓGICO DE MONTERREY, CAMPUS QUERÉTARO

Contenido

- Un poco de historia.
- ¿Qué es un algoritmo?
- ¿Cuáles son sus características?
- ¿Porqué es importante estudiar los algoritmos?
- ¿Cómo analizamos los algoritmos?
- ¿Big O, Big Ω , Big Θ ?
- Jerarquía de los algoritmos.
- Complejidad vs. Tiempo.
- Clasificación de los problemas.

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 este abrev. del ár. clás. ḥisābu lġubār, cálculo mediante cifras arábigas).
 - m. Conjunto ordenado y finito de operaciones que permite hallar la solución de un problema.
 - m. Método y notación en las distintas formas del cálculo.

¿Qué es un algoritmo?

- 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.

¿Qué es un algoritmo?

- 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.

¿Qué es un algoritmo?

- 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.

¿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 el estudio de los algoritmos?

- Por que su impacto es muy amplio.
 - 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.

¿Porqué es importante el estudio de los algoritmos?

- Viejos algoritmos, nuevas oportunidades.
- La posibilidad de resolver problemas de formas no exploradas.
- Es un reto intelectual.
- Poder desbloquear los secretos del universo.
- Por diversión y dinero.

¿Cómo analizamos un algoritmo?

- 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.

¿Cómo analizamos un algoritmo?

- 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?

¿Cómo analizamos un algoritmo?

- Análisis empírico (a posteriori):
 - Generando ejecuciones del algoritmo para distintos valores de entrada y cronometrando el tiempo de ejecución.
 - 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.

¿Cómo analizamos un algoritmo?

- 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

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
$O(m^n) \ m \geq 2$	Exponencial
$O(n!)$	Factorial

Complejidad vs Tiempo

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

Clasificación de los problemas

- Problema P (Polinomial):
 - Informalmente, es la clase de problemas de decisión resolubles por algún algoritmo dentro de un número de pasos delimitados 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.

Clasificación de los problemas

- Problema NP (No 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 "sí"-instances pueden aceptarse en tiempo polinómico por una máquina de Turing no determinista.
 - Ejemplo: Encontrar los factores primos de un número entero muy largo, problema del agente viajero, el problema de satisfactibilidad de ecuaciones booleanas.

Clasificación de los problemas

- 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.

Clasificación de los problemas

