Complejidad de los Algoritmos

Que es un Algoritmo?

Webster: cualquier método especial para resolver cierta clase de problemas.

Horowitz: método preciso utilizable en una computadora para la solución de un problema.

CONJUNTO FINITO DE PASOS UTILIZADO POR UNA COMPUTADORA PARA RESOLVER UN PROBLEMA

Preguntas

- ¿Para todos los problemas, existe al menos un algoritmo?
- Si existen varios algoritmos para un problema, ¿Cómo hacer una selección en términos de eficiencia?

Características de los Algoritmos

•Cada paso requiere una o más operaciones

• Operaciones definida (No ambiguas):
> 5/0, 6 + 7 ó tal vez 8: No se permiten

• Cada paso puede realizarse en una cantidad finita de tiempo.

•Un algoritmo produce una o más salidas y requiere cero o más entradas (externas).

• Un algoritmo siempre termina en un número finito de pasos

Características

- Sin ambigüedad.
- Efectividad: tiempo finito de cómputo.
- Computabilidad. Alg que sea más listo que los humanos
- Complejidad.
- Claridad: programación estructurada.

Áreas de Estudio de los Algoritmos

- **¿Cómo construir Algoritmos?** -->Enfoques
  - Divide y vencerás, Programación dinámica, Exacta, estocástica, de vecindades,…
- **¿Cómo expresar algoritmos?** -->Enfoques
  - Programación estructurada, de objetos, de agentes, funcional, lógica,..
- **¿Cómo validar algoritmos?** -->Caminatas, verificación formal,…-->¿Cómo probar un programa?
- **¿Cómo analizar algoritmos?** -->Complejidad computacional, amigabilidad, robustez,..

Análisis de Algoritmos

- **Estudia la complejidad espacial y temporal de los algoritmos**
- **Y las otras propiedades relevantes!!!**

Tareas en el Análisis de Algoritmos

- Determinar qué operaciones se emplean y su costo relativo.
- Determinar conjuntos de datos para exhibir todos los patrones posibles de comportamiento.
- Análisis a priori: se determina una función (de ciertos parámetros) que acote el tiempo de cómputo del algoritmo.
- Análisis a posteriori: estadísticas reales sobre tiempo y memoria.

Desempeño de Algoritmos

Eficiencia --> Rapidez

Eficacia --> Precisión

Tipos de Algoritmos

- Algoritmos **determinísticos**: El resultado de cada operación está definido en forma única.
- Algoritmos **no-determinísticos**. El resultado de cada operación esta determinado por un conjunto de posibilidades.
- Algoritmo polinomial. Es un algoritmo de complejidad polinomial o inferior.
- Algoritmo No-Polinomial. Es un algoritmo de complejidad exponencial o mayor.

Complejidad Computacional

•Clasifica los algoritmos en buenos o malos.

•Clasifica los problemas de acuerdo a la dificultad inherente de resolverlos.

Complejidad:

Temporal: Tiempo requerido por un algoritmo para encontrar la solución.

Espacial: Almacenamiento requerido por un algoritmo para encontrar la solución.

Complejidad Temporal Asintótica (Espacial).

- comportamiento límite conforme el tamaño del problema se incrementa:
  - determina el tamaño del problema que puede ser resuelto por un algoritmo.

Complejidad Temporal

Parámetro de medición
Se toma el tamaño de la entrada n (descripción de la instancia) para medir los requerimientos de tiempo de un algoritmo.

El tiempo de ejecución se describe como una función de la entrada T(n)

Ejemplo:
$T(n)=n^2+2n$

Criterio de Análisis

Análisis del peor caso
Se toma el tiempo de ejecución del peor caso.

Ejemplo:

ALGORITMO	TIEMPO
suma=0;	1
for(i=1;i<=n;i++)	2n+1
suma+=i;	2n

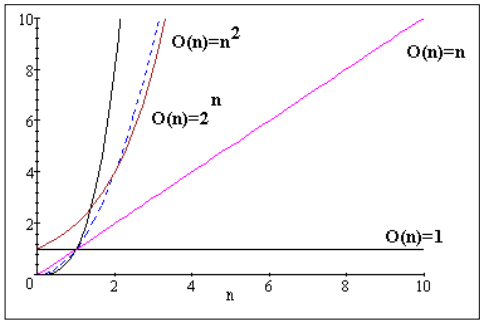
T(n) = 4n+2

Orden de un Algoritmo

El orden de un algoritmo indica el grado de complejidad de un algoritmo y que sirve de base para comparar su eficiencia.

Ejemplo:
Un algoritmo cuya función de tiempo es :
$T(n)=(n^2-n)/2$
Es de orden $O(n^2)$

Comparación de Orden



NOTACION ASINTOTICA O(n)

- $O(n^2)$: si un algoritmo de complejidad temporal asintótica para un problema de tamaño n procesa entradas en c n^2, su complejidad es $O(n^2)$.
- Una función f(n) es O(g(n)) si

  $| f(n) | <= c | g(n) |$

  C = constante

Notación O(n)

- Conforme es mayor la velocidad de las computadoras:
  - Los problemas a resolver son mayores.
  - El análisis de la complejidad es más importante.

Notación con O(n)

O(1)	CONSTANTE
O(log log(n))	Log log(n)
O(log(n))	Logarítmica
O(n)	Lineal
O(nlog(n))	nlog (n)
$O(n^2)$	Cuadrática
$O(n^3)$	Cúbica
$O(n^m)$	Polinomial
$O(m^n)$	exponencial
O(n!)	factorial

Buenos y Malos Algoritmos

- ***Convención 1:*** Todos los algoritmos, desde constantes hasta polinomiales, son polinomiales.
- ***Convención 2 :*** Todos los algoritmos exponenciales y factoriales, son exponenciales.
- ***Convención 3 :*** Los algoritmos polinomiales son "buenos" algoritmos.
- ***Convención 4:*** Los algoritmos exponenciales son "malos" algoritmos.

Crecimiento de Funciones Polinomiales y Exponenciales

Tiempo de evaluación de la función en unidades de tiempo, en función del tamaño de la entrada n

Función	***Valores Aproximados***		
n	10	100	1000
$n \log n$	33	664	9966
n^3	1,000	1,000,000	10^9
$10^6\, n^8$	1,014	10^{22}	10^{30}
2^n	1,024	1.27×10^{30}	1.05×10^{301}
$n^{\log n}$	2,099	1.93×10^{13}	7.89×10^{29}
$n!$	3,628,800	10^{158}	$4X10^{2567}$

Teorema:

- SI $A(n) = a_m n^m + \ldots\ldots + a_1 n + a_0$,
- entonces $A(n) = O(n^m)$

Tipos de Algoritmos

Algoritmos de tiempo polinomial:
Cuya función de tiempo tiene un orden $O(n^x)$.

Algoritmos de tiempo exponencial:
Cuya función de tiempo tiene un orden $O(x^n)$

Un problema está bien resuelto hasta que exista un algoritmo de tiempo polinomial para él.

Ejemplo: $t(n) = 60n^2 + 5n + 1$. Peor Caso

Tiempo de evaluación de t(n) en unidades de tiempo en función del tamaño de la entrada n

n	**$t(n) = 60n^2+5n+1$**	**$60n^2$**
10	6,051	6,000
100	600,501	600,000
1,000	60,005,001	60,000,000
10,000	6,000,050,001	6,000,000,000

Efecto de la Tecnología

Función	*Tamaño de la Instancia Solucionada en 1 Día*	*Tamaño de la Instancia Solucionada en un Día en una Computadora 10 Veces Más Rápida*
n	10^{12}	10^{13}
$n \log n$	0.948×10^{11}	0.87×10^{12}
n^2	10^6	3.16×10^6
n^3	10^4	2.15×10^4
$10^8 n^4$	10	18
2^n	40	43
10^n	12	13
$N^{\log n}$	79	95
$n!$	14	15

Modelos de Cómputo

Máquina Determinista:

Es aquella en que para una misma entrada, siempre se obtiene el mismo resultado.

Máquina no-determinista:

Es aquella que para una misma entrada, se pueden obtener resultados diferentes.

Fase de adivinación+Fase de verificación