ANÁLISIS DE ALGORITMOS

ν El análisis de algoritmos estima el consumo de recursos de un algoritmo.

ν Esto nos permite comparar los costos relativos de dos o más algoritmos para resolver el mismo problema.

ν El análisis de algoritmos también les da una herramienta a los diseñadores de algoritmos para estimar si una solución propuesta es probable que satisfaga las restricciones de recursos de un problema.

ν El concepto de razón de crecimiento, es la razón a la cual el costo de un algoritmo crece conforme el tamaño de la entrada crece.

Complejidad en tiempo

TIEMPO DE EJECUCIÓN

El tiempo de Ejecución de un programa se mide en función de N, lo que designaremos como T(N).

Esta función se puede calcular físicamente ejecutando el programa acompañados de un reloj, o calcularse directamente sobre el código, contando las instrucciones a ser ejecutadas y multiplicando por el tiempo requerido por cada instrucción. Así, un trozo sencillo de código como:

```
S1;
for(x = 0; x < N; x++)
S2;
```

Demanda: $T(N) = t1 + t2 * N$

Donde t1 es el tiempo que lleva ejecutar la serie S1 de sentencias, y t2 es el que lleva la serie S2.

Habitualmente todos los algoritmos contienen alguna sentencia condicional o selectiva, haciendo que las sentencias ejecutadas dependan de la condición lógica, esto hace que aparezca más de un valor para T(N), es por ello que debemos hablar de un rango de valores:

$Tmin(N) \leq T(N) \leq Tmax(N)$

Estos extremos son llamados "el peor caso" y "el mejor caso" y entre ambos se puede hallar "el caso promedio" o el más frecuente, siendo este el más difícil de estudiar; nos centraremos en el "el peor caso" por ser de fácil cálculo y se acerca a "el caso promedio", brindándonos una medida pesimista pero fiable.

Toda función T(N) encierra referencias al parámetro N, y a una serie de constantes Ti dependientes de factores externos al algoritmo. Se tratará de analizar los algoritmos dándoles autonomía frente a estos factores externos, buscando estimaciones generales ampliamente válidas, a pesar de ser demostraciones teóricas.

COMPLEJIDAD EN ESPACIO.

Es la memoria que utiliza un programa para su ejecución; es decir el espacio de memoria que ocupan todas las variables propias del algoritmo.

Esta se divide en Memoria Estática y Memoria Dinámica.

v Memoria estática. Para calcularla se suma de memoria que ocupan las variables declaradas en el algoritmo.

v Memoria dinámica. Su cálculo no es tan simple ya que depende de cada ejecución del algoritmo.

Ejemplo: algoritmo de búsqueda en arboles.

Función búsqueda _arboles.

Devuelve una solución o fallo.

Inicializa un árbol de búsqueda con estado inicial.

Bucle hacer

Si no hay candidatos para expandir.

Entonces devolver fallo.

En otro caso escoger nodo para expandir.

Si el nodo es el objetivo.

Entonces devolver solución.

En otro caso expandir nodo.

M = profundidad máxima del árbol (puede ser infinita)

D = profundidad de la mejor solución (menor costo)

B = factor de ramificación (número máximo de sucesiones) = 10

Depth	Nodes	Time	Memory
0	1	1 milisecond	100 bytes
2	111	.1 second	11 Kb
4	11111	11 second	1 Mb
6	10 ^ 6	18 minutos	111 Mb

EFICIENCIA DE LOS ALGORITMOS.

La Eficiencia de un algoritmo es comprobar que dicho análisis debe ser empírico y teórico, en cualquiera de los diferentes métodos de los algoritmos de ordenamiento, los cuales son:

Ø Bubble sort

Ø Selection sort

Ø Insertion sort

Ø Shell sort

Ø Ha sort

Ø Merge sort

Ø Quick sort

¿Por qué estudiar la eficiencia de los algoritmos?

Porque nos sirve para establecer la frontera entre lo factible y lo imposible. Ejemplo: Algoritmos de ordenación Observación El tiempo de ejecución depende del tamaño del conjunto de datos.

Objetivo Parametrizar el tiempo de ejecución en función del tamaño del conjunto de datos, intentando buscar una cota superior que nos sirva de garantía.

Tipos de Análisis

• Peor de los Casos: Se corresponde con el peor tiempo. T(n) es el tiempo máximo sobre las entradas.

• Mejor Caso: Límite inferior en el tiempo. T(n) es el menor tiempo de todas las posibles entradas

• Caso Promedio: Es el tiempo medio esperado sobre todas las posibles entradas de tamaño n. Se considera una distribución de probabilidad sobre las entradas.

• Análisis Probabilístico: Es el tiempo de ejecución esperado para una entrada aleatoria. Se expresa tanto el tiempo de ejecución y la probabilidad de obtenerlo

. • Análisis Amortizado: El tiempo que se obtiene para un conjunto de ejecuciones, dividido por el número de ejecuciones.

Notación Big O:

La notación es la representación relativa de la complejidad algorítmica. Trata de simplificar la comparación entre algoritmos a una sola variable.

O(N) ^ n ; if n=3; entonces es de grado 3.

Fuente:

Clase de Estructura de Datos con el profesor Ray Parra (1-2pm)

http://rhomarycristobal.blogspot.com/2011/12/septima-unidad-analisis-de-algoritmos.html