COMPLEJIDAD DE UN ALGORITMO

Camilo José Villalba
Facultad de Ingenieria, Estructuras de Datos
Universidad Nacional de Colombia, Bogotá

1. INTRODUCCIÓN

Todos los algoritmos consumen recursos en tiempo (cuanto tiempo demora el algoritmo en ejecutarse) y en espacio (cuanta memoria ocupa el algoritmo a la hora de ejecutarse) esto determina la eficiencia del algoritmo, es decir un algoritmo que consume pocos recursos será mas eficiente que uno que consuma artos.

2. COMPLEJIDAD EN TIEMPO

El tiempo que demora un algoritmo en ejecutarse depende de la cantidad de procesos que debe realizar el computador durante la ejecución de este, como la velocidad del computador depende del procesador de este y de otros factores ajenos al algoritmo, el tiempo del algoritmo no puede expresarse en una unidad de tiempo como los segundos, ya que en un computador rápido el algoritmo tardara menos tiempo en ejecutarse que en uno que sea lento, por esto vamos a denotar como la función T(n) el tiempo que tarda en ejecutarse un algoritmo, donde "n" es la entrada del algoritmo. Como la entrada del algoritmo nos determina el tiempo de ejecución entonces podemos establecer tres casos de acuerdo a la entrada, el mejor caso (mínimo numero de pasos con una entrada de tamaño "n"), el caso promedio y el pero caso.

Calcular el tiempo de ejecución de un algoritmo puede ser realizado de diferentes maneras dependiendo del algoritmo. Para el caso en que el algoritmo no sea recursivo ni llame ninguna función, se puede realizar la suma de los valores de tiempo a cada uno de los pasos del algoritmo de acuerdo a el tipo de operaciones que realice, estos valores los podemos ver en la tabla 1. Donde le vamos a asignar un tiempo de 1 a cada una de las siguientes operaciones simples.

Tipo de operación	Ejemplo	Tiempo
(+, -, *,/)	4+4	1
Asignación	x=1;	1
Comparación	4>6	1
Operadores lógicos	C1 && C2	1
Acceso a memoria	Solicitar A[j]	1
Condición	If (c1)	1 si c1 = falso, 2 si c1 = verdadero
Ciclos	for()	La complejidad en tiempo para evaluar la condición mas la complejidad para evaluar el cuerpo del ciclo por el numero de veces que se repite el ciclo.

Tabla 1.

En caso de que el algoritmo llame alguna función la complejidad en tiempo seria la del algoritmo sin el llamado de las funciones mas la complejidad del llamado de la función y del cuerpo de esta.

La complejidad en tiempo del algoritmo también esta dado por la tasa de crecimiento del algoritmo, esta tasa de crecimiento puede ser de tipo logarítmico, polinomial o exponencial. Esto nos puede facilitar escoger entre varios algoritmos que realicen la misma tarea el mas eficiente ya que seria teóricamente el que tenga una menor tasa de crecimiento, además con esta tasa podemos hallar instancias mas grandes del algoritmo.

3. COMPLEJIDAD EN ESPACIO

La complejidad en espacio es cuanta memoria ocupa el algoritmo al ejecutarse, el factor determinante esta dado por los datos que entran al programa los cuales se pueden almacenar en espacios dinámicos o estáticos.

Un espacio estático es aquel que se define en el algoritmo, es decir en la ejecución del programa ya se sabe cuanto espacio va a ocupar pera entender esto de puede dar el ejemplo de un arreglo, al crear el arreglo se le da un tamaño inicial, esto hace que al ejecutar el programa se reserve un espacio de memoria del tamaño del arreglo así este esté vacío. El costo en espacio lo vamos a denotar como Cm(T) donde "T" es un tipo de datos definido. Para calcular el costo en espacio debemos tener en cuenta lo siguiente, al instanciar los tipos de datos primitivos como int, double, boolean o char tienen un costo de Cm(T)=1, los registro y los arreglos ocupan un espacio de la suma del espacio de cada uno de sus componentes mas el espacio como tal del registro o el arreglo, los apuntadores tienen un costo de Cm(T)=1.

Los espacios dinámicos son espacios en los cuales el tamaño varia durante la ejecución del algoritmo, el costo de estos se puede calcular como la máxima cantidad de objetos en un momento dado durante la ejecución del programa, como ejemplo podemos poner un objeto de tipo lista, al cual podemos añadir objetos indefinidos sin embargo el costo de este será que tantos objetos contenga la lista en el momento.

Bibliografía

Skiena, S. S. *The Algorithm Design Manual* (2 ed.). New York: Springer.
Scalise , E., & Carmona, R. *Análisis de Algoritmos y Complejidad* . Caracas, Venezuela.
TÉCNICAS DE DISEÑO DE ALGORITMOS. (s.f.). Recuperado el 25 de 09 de 2013, de http://www.lcc.uma.es/~av/Libro/CAP1.pdf