Notas de Puntos Relevantes de Videos

José Luis Sandoval Pérez
Teoría de la Complejidad Computacional
Profesor: Miguel Ángel Meza De Luna

Video 1

Dentro del mundo de los desarrolladores el concepto de eficiencia es de suma importancia, ya sea para algoritmos, escritura de código, etc. Todo esto lo demanda el usuario que busca la velocidad.

Debido a esto se han desarrollado para analizar algoritmos bajo un mismo campo de análisis. Pero este análisis no es igual para el todo de los casos, ya que depende del hardware.

Es por eso por lo que la notación asintótica nos permite determinar el comportamiento del algoritmo al igual que su eficiencia, comparándolo con diferentes opciones. Este análisis se hace en base al coste del algoritmo en función de varios parámetros por ejemplo el tamaño de su entrada.

así pues, las funciones que definen los costes del algoritmo cuentan con la siguiente clasificación:

1. O Grande (Cota superior de ajuste): funciones que cuando tienden a infinito son inferiores o iguales a alguna función lineal. Constante que multiplicada por la función cumpla la condición anterior. Crecimiento mayor o igual a la complejidad en el tiempo
2. Omega Grande (Cota inferior de ajuste): funciones que cuando tienden a infinito son superiores o iguales a alguna función lineal. Constante que multiplicada por la función cumpla la condición anterior. Crecimiento menor o igual a la complejidad
3. Theta Grande (Cota ajustada): Este es un concepto clave en la notación asintótica que se refiere a funciones que, cuando tienden a infinito, son equivalentes a alguna función lineal. Esto significa que la tasa de crecimiento de la función es proporcional a una constante multiplicada por la función. En otras palabras, si tenemos una función f(n), podemos decir que es Theta Grande si existe otra función g(n) tal que para algún valor constante C y para todos los valores de n lo suficientemente grandes, C * g(n) es igual a f(n). Crecimiento igual a la complejidad
4. O pequeña (Cota inferior ajustada): Este concepto se refiere a funciones que, cuando tienden a infinito, son equivalentes a alguna función multiplicada por una constante y un término

más pequeño. En otras palabras, si tenemos una función f(n), podemos decir que es O pequeña si existe otra función g(n) tal que para algún valor constante C y para todos los valores de n lo suficientemente grandes, C * g(n) + o(g(n)) es igual a f(n). Crecimiento mayor a la complejidad en el tiempo.

5. Omega pequeña (Cota superior ajustada): Este concepto se refiere a funciones que, cuando tienden a infinito, son inferiores o equivalentes a alguna función multiplicada por una constante y un término más grande. En otras palabras, si tenemos una función f(n), podemos decir que es Omega pequeña si existe otra función g(n) tal que para algún valor constante C y para todos los valores de n lo suficientemente grandes, C * g(n) + ω(g(n)) es igual a f(n). Crecimiento menor a la complejidad.

Con estas clasificaciones es más fácil agrupar los costes algoritmos en base a la familia que pertenecen.

Video 2

Términos de mayor Crecimiento

+ Mayor /Peor 2^n

n^

n

log (n)

\- Menor/Mejor 1

Para que se cumpla la notación debemos de tener un punto a partir de donde se cumple la condición. Valor mínimo.

Video 3

Antes de mencionar los métodos para el cálculo de la notación asintótica debemos de destacar que la notación estándar de la industria es O-Grande, ya que nos señala el peor de los casos al ejecutar un algoritmo.

Al aumentar la entrada aumenta el tiempo.

Enfoque en grado mayor de crecimiento.

Existen varios métodos

1. Método de revisión de código: método basado en asignar un valor a cada estructura de código. (Suposición)
2. Método midiendo el tiempo, por grafica (intuitivo): No es una estrategia práctica, pero es más comprensible.