###### Licenciatura en Ingeniería de Software

###### Taller de Programación Visual III

|  |  |
| --- | --- |
| UNIDAD I, Introducción a la Programación visual | Tarea # 2 |
| Nombre: Jaime Alejandro Sanchez Laguna | Fecha:16 de Marzo del 2019 |

NOTATION BIG O

Este se trata de que función a tiempo cuanto tu algoritmo puede tardar en ejecutarse y se refleja en cuestión al peso del archivo, a medida que crece los datos de entrada más tiempo tarda en ejecutar el programa esto nos dice que algoritmo es ineficiente por la cantidad de operaciones que utiliza para completar su función tardaría mucho tiempo.

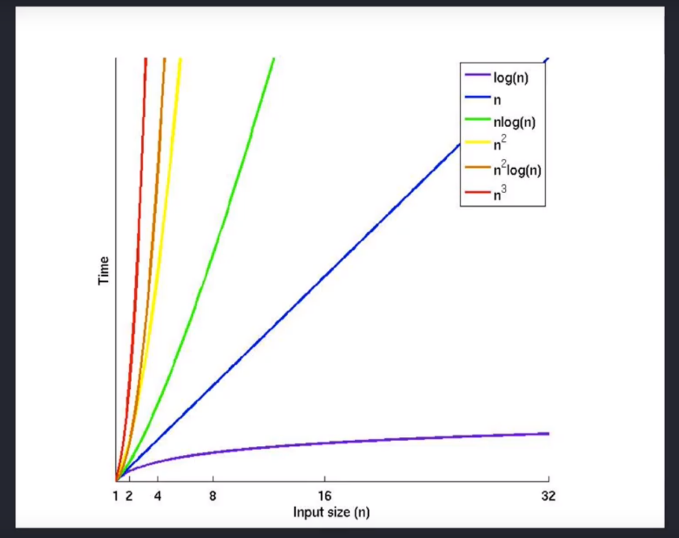
Un ejemplo del Big o es donde se tiene un archivo con un peso de 1 TB y se quiere compartir, se tiene dos opciones poderlo enviar vía internet este método dura 2 horas o pasarlo en algún formato físico, el formato físico te dura 7 horas en completarse el viaje hasta llegar.   
Si este archivo incrementa su tamaño a 100TB ahora por vía internet se tardaría 20 horas y por físico duraría lo mismo, entonces en teoría es más factible por vía física.

Este consiste en algunas reglas o normas

Se refiere a las constantes

A esto se refiere evitar el uso de constantes ya que puede haber dos procesos que hagan exactamente o mismo pero eso no importa ya que la entrada será la misma.

Termino no dominantes

Para calcular la complejidad o el tiempo debemos ver cómo funcionan los términos estos son algunos de los términos.

O(logN)

O(N log N)

O(N)

(N)

(N\*N)

O(2N)

Algoritmo multi parte

Se tiene que tomar en cuenta que no es lo mismo tener dos arreglos con el mismo nombre a tener dos arreglos de diferentes nombres ya que esto aumenta la complejidad algo tan simple puede alterar el tiempo de ejecución, al igual que estar buscando en los arreglos uno adentro de otro.

Log N Búsqueda binaria

Una forma de verlo es si quieres buscar el numero 9 en un arreglo de N posiciones puedes situar un punto en medio de ese arreglo y ver si este numero 9 es menor o mayor a la mitad de arreglo este se parte si solo se deja la parte que contiene el numero a buscar, se repite y esto nos ayuda a no estar buscar buscando en un montón y esto nos ayuda a reducir el tiempo de ejecución ya que no busca entre todos si no cada vez en menos datos.

Recursividad

Se puede pensar que este método es sencillo si lo vemos en código pero a cuanto eficiencia esto no nos ayuda mucho ya que para sacar o encontrar el resultado se tiene que crear más pasos aunque estos no se vean.

Conclusión:

La notation Big O nos ayuda a ver como estructurar nuestro código para mayor eficiencia y así al momento de cargar datos el proceso para encontrar la solución será menor o no estará tan cargado de cosas innecesarias

Y esto nos ayuda a que no se consuma memoria innecesariamente.