Edgar Alcantara Ginera

DEFINICIÓN DE UN ALGORITMO DE ORDEN CONSTANTE

Esta abreviatura es mejor conocida como **orden de complejidad**, y de esta manera logramos agrupar todas las complejidades que crecen de igual forma, es decir, que pertenecen al mismo orden. En conclusión, la complejidad algorítmica o ritmo de crecimiento es una **métrica que te permite como programador evaluar la factibilidad de las diferentes soluciones de un problema**, y poder decidir con un argumento matemático cuál es mejor mediante comparaciones.

Este tipo de herramienta teórica tiene aparentemente poca utilidad en el día a día de un programador habitual, pues **requiere de algoritmos mucho más complejos y pilas de datos muy grandes**, y a menos que trabajes con Big Data, Machine Learning o hardware con recursos muy limitados como Arduino donde la optimización es crucial, la complejidad algorítmica tiene poca importancia a nivel práctico. Sin embargo, es un concepto bastante básico e importante de conocer, pues te permite dimensionar y pensar acerca del comportamiento de tus soluciones y la manera en la que los solucionas, y no solo codear un algoritmo que termine costándote más en un futuro.

Edgar Alcantara Ginera

DEFINICIÓN Y EJEMPLO DE UN ALGORITMO DE ORDEN LOGARITMICO

En computación y matemáticas un algoritmo de ordenamiento es un algoritmo que pone elementos de una lista o un vector en una secuencia dada por una relación de orden, es decir, el resultado de salida ha de ser una permutación —o reordenamiento— de la entrada que satisfaga la relación de orden dada.

Los algoritmos de ordenamiento estable mantienen un relativo preorden total. Esto significa que un algoritmo es *estable* solo cuando hay dos registros *R* y *S* con la misma clave y con *R* apareciendo antes que *S* en la lista original.

Cuando elementos iguales (indistinguibles entre sí), como números enteros, o más generalmente, cualquier tipo de dato en donde el elemento entero es la clave, la estabilidad no es un problema. De todas formas, se asume que los siguientes pares de números están por ser ordenados por su primer componente:

```
(4, 1)  (3, 7)  (3, 1)  (5, 6)
```

En este caso, dos resultados diferentes son posibles, uno de los cuales mantiene un orden relativo de registros con claves iguales, y una en la que no:

```
(3, 7)  (3, 1)  (4, 1)  (5, 6)   (Orden mantenido)
(3, 1)  (3, 7)  (4, 1)  (5, 6)   (Orden cambiado)
```

Pueden cambiar el orden relativo de registros con claves iguales, pero los algoritmos estables nunca lo hacen. Los algoritmos inestables pueden ser implementados especialmente para ser estables. Una forma de hacerlo es extender artificialmente el cotejamiento de claves, para que las comparaciones entre dos objetos con claves iguales sean decididas usando el orden de las entradas original. *Recordar* este orden entre dos objetos con claves iguales es una solución poco práctica, ya que generalmente acarrea tener almacenamiento adicional.

Ordenar según una clave primaria, secundaria, terciara, etc., puede ser realizado utilizando cualquier método de ordenamiento, tomando todas las claves en consideración (en otras palabras, usando una sola clave compuesta). Si un método de ordenamiento es estable, es posible ordenar múltiples ítems, cada vez con una clave distinta. En este caso, las claves necesitan estar aplicadas en orden de aumentar la prioridad.

Ejemplo: ordenar pares de números, usando ambos valores

```
(4, 1)  (3, 7)  (3, 1)  (4, 6) (Original)
(4, 1)  (3, 1)  (4, 6)  (3, 7) (Después de ser ordenado por el segundo valor)
(3, 1)  (3, 7)  (4, 1)  (4, 6) (Después de ser ordenado por el primer valor)
```

Por otro lado:

```
(3, 7)  (3, 1)  (4, 1)  (4, 6) (Después de ser ordenado por el primer valor)
(3, 1)  (4, 1)  (4, 6)  (3, 7) (Después de ser ordenando por el segundo valor,
                                El orden por el primer valor es perturbado)
```

DEFINICIÓN Y EJEMPLO DE UN ALGORITMO DE ORDEN LINEAL

Por ordenar se entiende el proceso de reorganizar un conjunto de objetos en una cierta secuencia de acuerdo a un criterio especificado. En general, el objetivo de este proceso es facilitar la posterior búsqueda de elementos en el conjunto ordenado.

Ejemplo:

Si el arreglo a ordenar es a = ['a','s','o','r','t','i','n','g','e','x','a','m','p','l','e'],

el algoritmo va a recorrer el arreglo de izquierda a derecha. Primero toma el segundo dato 's' y lo asigna a *v* y *i* toma el valor de la posición actual de *v*.

Luego compara esta 's' con lo que hay en la posición j-1, es decir, con 'a'. Debido a que 's' no es menor que 'a' no sucede nada y avanza i.

Ahora v toma el valor 'o' y lo compara con 's', como es menor recorre a la 's' a la posición de la 'o'; decrementa j, la cual ahora tiene la posición en dónde estaba la 's'; compara a 'o' con a[j-1] , es decir, con 'a'. Como no es menor que la 'a' sale del for y pone la 'o' en la posición a[j]. El resultado hasta este punto es el arreglo siguiente: a = ['a','o','s','r',....]

Así se continúa y el resultado final es el arreglo ordenado :

a = ['a','a','e','e','g','i','l','m','n','o','p','r','s','t','x']

Edgar Alcantara Ginera

DEFINICIÓN Y EJEMPLO DE UN ALGORITMO DE ORDEN NLONG

`L` es una lista ordenada que contiene `n` enteros con signo (siendo `n` suficientemente grande), por ejemplo `[-5, -2, -1, 0, 1, 2, 4]` (aquí, `n` tiene un valor de 7). Si se sabe que `L` contiene el entero 0, ¿cómo puedes encontrar el índice de 0?

```
a = 0
b = n-1
while True:
  h = (a+b)//2 ## // is the integer division, so h is an integer
  if L[h] == 0:
    return h
  elif L[h] > 0:
    b = h
  elif L[h] < 0:
    a = h
```

`a` y `b` son los índices entre los que 0 es que se encuentran. Cada vez que entramos en el bucle, utilizamos un índice entre `a` y `b` y lo utilizan para reducir el área de búsqueda.

En el peor de los casos, tenemos que esperar hasta que `a` y `b` sean iguales. ¿Pero cuántas operaciones lleva eso? No n, porque cada vez que entramos en el bucle, dividimos la distancia entre `a` y `b` en alrededor de dos. Más bien, la complejidad es O (log n).

DEFINICIÓN Y EJEMPLO DE UN ALGORITMO DE ORDEN CUADRATICO

Todos son una parábola y tendrán un comportamiento similar con respecto al aumento de la cantidad de entradas. Estas presentan un comportamiento muy diferente al conjunto anterior. Este grupo de O grandes las llamaremos de **Órden Cuadrática**. También las podemos expresar como:

$$O(100n^2 + 101n) \in O(n^2)$$

DEFINICIÓN Y EJEMPLO DE UN ALGORITMO DE ORDEN EXPONENCIAL

Es uno de las peores complejidades algorítmicas. Sube demasiado a medida que crecen los datos de entrada. Un ejemplo de este código es la solución de Fibonacci, el cual genera bucles 2 recursividades en cada ejecución. Ejemplo:

```
function exponencial($n){
  if (n==1 || n==2) return 1;
  return exponencial($n-1)+exponencial($n-2);
}
def exponencial(n):
  if n==1 or n==2:
    return 1
  return exponencial(n-1)+exponencial(n-2)
```