

Ayudantía 1

ESTRUCTURA DE DATOS Y ALGORITMOS

PROFESOR: YERKO ORTIZ

AYUDANTE: VICENTE DIAZ

SECCIÓN 1



Contacto

Discord: trapitokid

Wsp: +569 5749 6014

LoL: Saki TvT#woof



Java

- Lenguaje de programación orientada a objetos.
- Una clase posee atributos y métodos que definen el comportamiento de un objeto.
- Existen expresiones las cuales se ocupan al programar.
- Se ocupan los tipos de datos primitivos (int, float, String, boolean, etc.).

J Main.java X

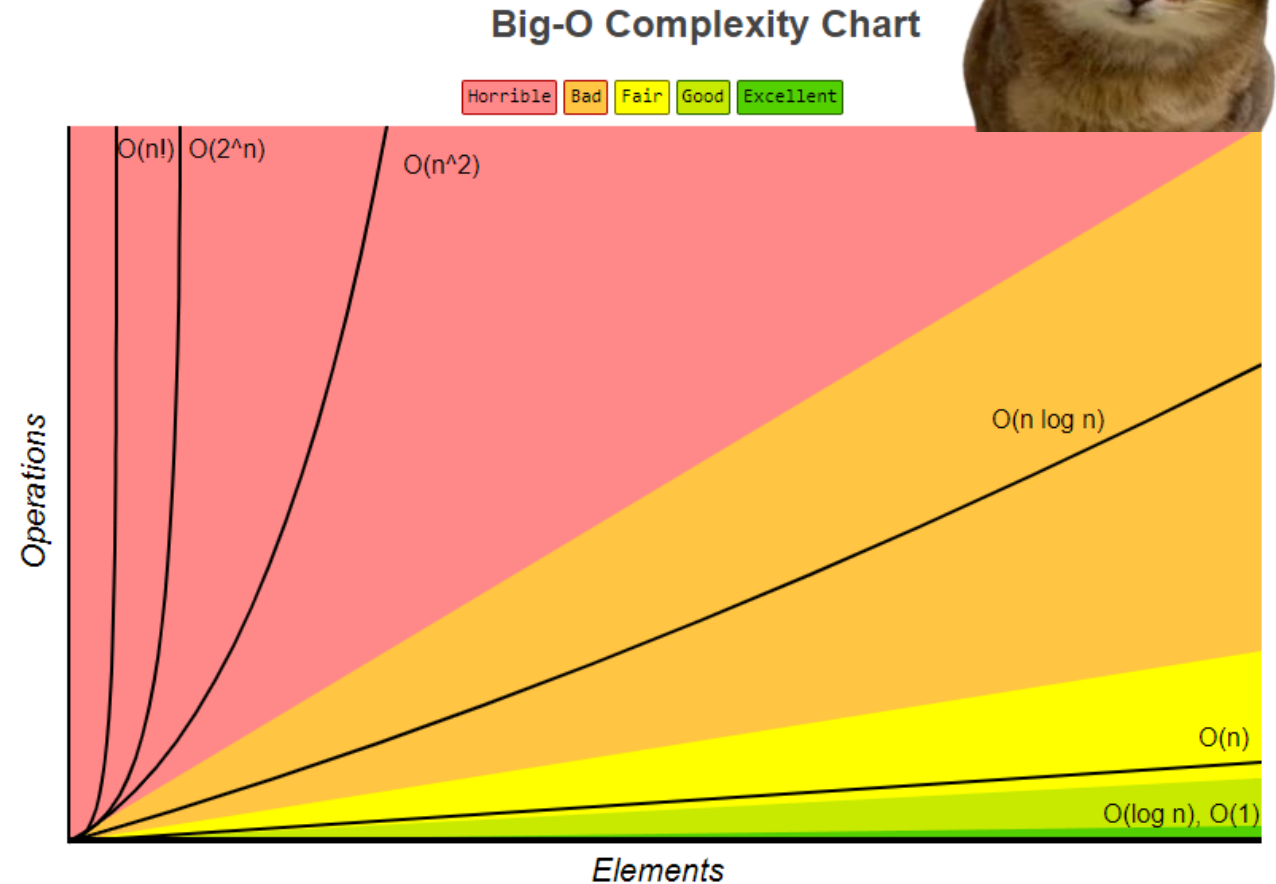
lab1 > Correctoquizas > J Main.java > ...

```
10 public class Main {
11
12     private static SongController songController = new SongController();
13     private static HistoryController historyController = new HistoryCont
14     private static PlayerController playerController = new PlayerControl
15     private static PlayerController playerController2 = new PlayerContro
16
17     /* Esta función sirve para leer información desde un archivo CSV. */
18     public static void ReadCSV(SongController songController, int MaxSon
19         /* Array de songs */
20
21         long start = System.nanoTime();
22
23         String csvFile = "data.csv";
24         String line = "";
25         String cvsSplitBy = ",(?![^\\[\\]]*\\])";
26         int yearIndex = 1;
27         int artistsIndex = 3;
28         int idIndex = 8;
29         int nameIndex = 14;
30
31         int count = 0;
32
33         try (BufferedReader br = new BufferedReader(new FileReader(csvFi
34             br.readLine());
35             while ((line = br.readLine()) != null) {
```



Análisis asintótico

- Algoritmo: secuencia de instrucciones claras y precisas.
- Eficiencia: ciclos de CPU (tiempo de ejecución) o memoria.
- El análisis asintótico es una relación entre tiempo y tamaño de la entrada, uso de memoria y tamaño de la entrada.
- Para describir estas complejidades se ocupan 3 notaciones:
 - Big-O: peor de los casos $\{O[g(n)]\}$
 - Omega: mejor de los casos $\{\Omega[g(n)]\}$
 - Theta: tiempo promedio $\{\theta[g(n)]\}$



Casos Big-O

$O(1)$ = cualquier línea de código que no sea un ciclo ni recursión.

$O(\log[n])$ = la variable del ciclo no aumenta de manera constante (*, /).

$O(n)$ = la variable del ciclo va aumentando o decreciendo constantemente.

$O(n \log[n])$ = combinación de otros casos.

$O(n^x)$ = x depende de cuantos ciclos anidados hay e iteran conforme a la entrada.

```
void exampleAlgorithm(int[] arr) {  
    for (int i = 0; i < arr.length; i++) {  
        System.out.println(arr[i]);  
    }  
}
```

```
void anotherAlgorithm(int[] arr) {  
    for (int i = 0; i < arr.length; i++) {  
        for (int j = 0; j < arr.length; j++) {  
            System.out.println(arr[i] + arr[j]);  
        }  
    }  
}
```

```
void binarySearch(int[] arr, int target) {  
    int left = 0;  
    int right = arr.length - 1;  
    while (left <= right) {  
        int mid = left + (right - left) / 2;  
        if (arr[mid] == target) {  
            System.out.println("Encontrado");  
            return;  
        }  
    }  
}
```

```
int count = 0;  
for (int i = n; i > 0; i /= 2) {  
    for (int j = 0; j < n; j++) {  
        count++;  
    }  
}
```



Ejercicio 1

Se te otorga un array de n enteros. Quieres modificar el array para que sea creciente, es decir, cada elemento sea al menos tan grande como el elemento anterior. ¿Cuál es el número mínimo de movimientos requeridos?

Input:

La primera línea de entrada contiene un entero n (tamaño del array).

Luego, la segunda línea contiene n enteros (contenido del array).

Output:

Imprime el número mínimo de movimientos.



Link Github

Repositorio:

<https://github.com/DiegoBan/EDDA2024-1-S4>

Github personal:

<https://github.com/VicenteDiazH>



Ejercicio 2

Dada una matriz $M \times N$, imprime todos los elementos de la matriz en orden espiral.

Example 1:

1	→	2	→	3
4	→	5		↓
↑				6
7	←	8	←	9
				↓

Input: matrix = [[1,2,3],[4,5,6],[7,8,9]]

Output: [1,2,3,6,9,8,7,4,5]



Example 2:

1	→	2	→	3	→	4
5	→	6	→	7		↓
↑						8
9	←	10	←	11	←	12
						↓

Input: matrix = [[1,2,3,4],[5,6,7,8],[9,10,11,12]]

Output: [1,2,3,4,8,12,11,10,9,5,6,7]