Informe Final: Análisis Práctico de un Algoritmo

Daniel Steven Cañizares Ortega - 1152203
Andrés Alfonso Parra Garzón - 1152185
Mauricio Di Donato Sánchez - 1152186
Mónica Gil González - 1152183
Yohan Camilo Botello Maldonado - 1152137

Universidad Francisco de Paula Santander
Ingeniería de Sistemas
Cúcuta, Colombia
2023

Datos tomados

https://docs.google.com/spreadsheets/d/1feJjEKz0QT8ALsR48QlZxZKZhV_tm3PCAqo5IGTbDaE/edit?usp=sharing

Script desarrollado

https://github.com/byandrev/execution-timer-algorithm

El link de la línea anterior, redirecciona a un Script desarrollado en Python que genera las matrices establecidas por el profesor (2x2, 5x5, 10x10, 50x50, 100x100), genera los tiempos de cada test y los guarda en un documento output.

El script `generate_tests.py` genera los inputs para cada test en un archivo txt dentro de la carpeta inputs, donde primero viene el tamaño de la matriz (cuadrada), seguido por un guión bajo " _" y el número del test. Por ejemplo: `2_1.txt`.

El script script.py genera los tiempos de cada test. Esto lo hace con ayuda de las librerías subprocess y datetime. Subprocess tiene un método llamado "check_output" el cual nos permite ejecutar comandos de terminar en python. Medimos el tiempo actual antes de ejecutar el comando y los restamos después de finalizar el comando, con esto se obtiene el tiempo de ejecución de cada test en segundos**.**

Complejidad del Algoritmo (Java)

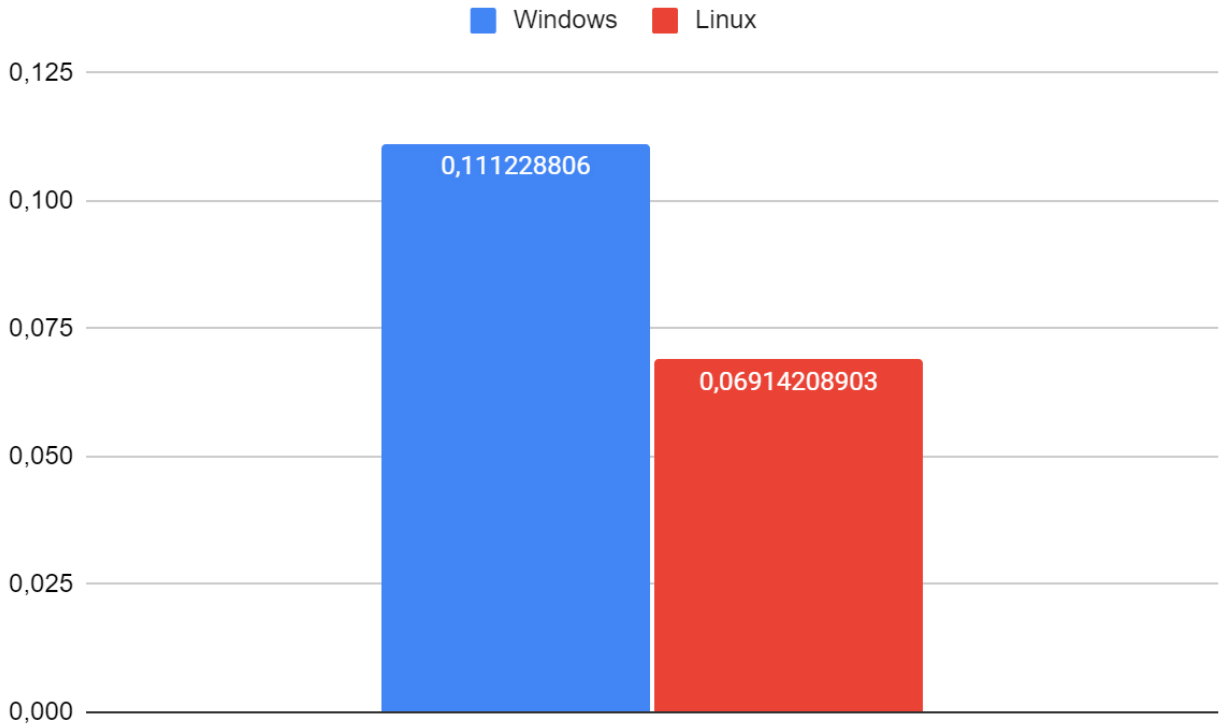
Ecuación -> 23AB + 9A + 9
Ecuación sólo algoritmo -> 17AB + 3A + 2
Complejidad -> $O(n^2)$

```java
import java.util.Scanner;
public class Main
{
    public static void main(String []args){//1+1+1+1+1+(3AB + 3A + 1)+(17AB + 3A + 2)+(3AB + 3A + 2) -> 23AB + 9A + 9
        Scanner sc = new Scanner(System.in);//1

        byte A = sc.nextByte();//1
        byte B = sc.nextByte();//1
        byte matriz[][] = new byte[A][B];//1
        //1
        for(byte i=0; i<A; i++){//2+(3B+1) -> (3B + 3) * A -> 3AB + 3A + 1
            //1
            for(byte j=0; j<B; j++){//2+1 -> 3*B -> 3B + 1
                matriz[i][j] = sc.nextByte();//1
            }
        }
        procesarMatriz(matriz);//1+(17AB + 3A + 1) --> 17AB + 3A + 2
        imp(matriz);// 1+(3AB + 3A + 1) --> 3AB + 3A + 2
    }

    public static void procesarMatriz(byte matriz [][]){// 17AB + 3A + 1
        //1
        for(byte i=0; i<matriz.length; i++){//2+(17B+1) -> (17B + 3) * A -> 17AB + 3A + 1
            //1
            for(byte j=0; j<matriz[i].length; j++){//2+(2+2+2+2+5+2) -> 17*B -> 17B + 1
                if(i%2==0){//2
                    matriz[i][j] += 1;//2
                }
                if(j%2==0){//2
                    matriz[i][j] += 2;//2
                }
                if(i%2!=0 && j%2!=0){//5
                    matriz[i][j] -= 3;//2
                }
            }
        }
    }

    public static void imp(byte matriz [][]){// 3SAB + 3A + 1
        //1
        for(byte i=0; i<matriz.length; i++){//2+(3B + 1) -> (3B + 3) * A -> 3AB + 3A + 1
            //1
            for(byte j=0; j<matriz[i].length; j++){//2+1 -> 3*B -> 3B + 1
                System.out.print(""+matriz[i][j]+(j<matriz[i].length-1?" ":"\n"));//1
            }
        }
    }
}
```

Windows vs Linux

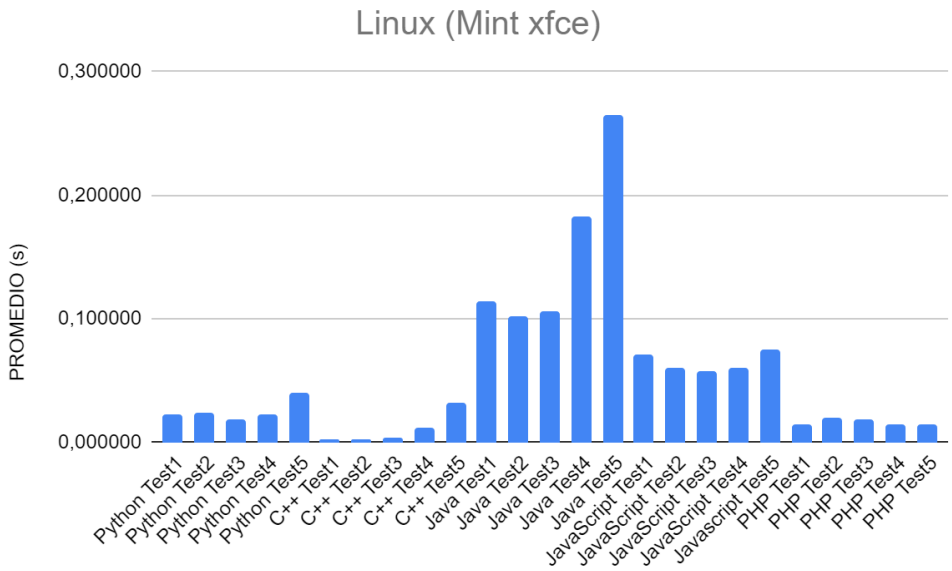
Comparación de tiempo promedio de ejecución en los 5 lenguajes.

Gráfica tiempo promedio total de ejecución en Windows vs Linux

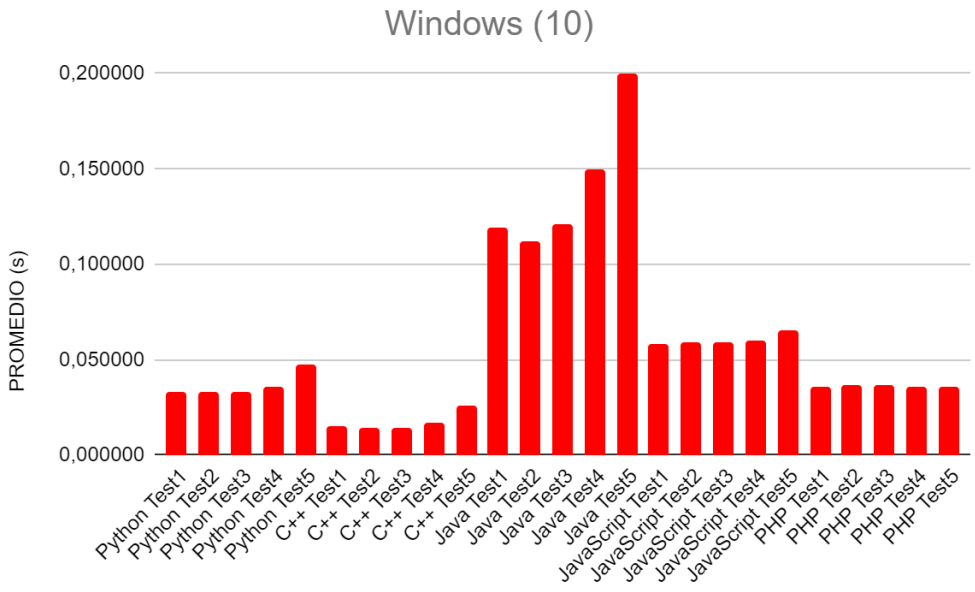


Algunas gráficas

Gráfica tiempo promedio de ejecución en Linux (Core i5-6300U 2.40Ghz)



Gráfica tiempo promedio de ejecución en Windows (Ryzen 7 5700g 3.80Ghz)



Conclusiones

- De los 5 lenguajes implementados en este análisis, el que destacó por rapidez en todos los tests expuestos fue C++ y el más lento Java. Esto se puede deber a que Java se ejecuta en una máquina virtual (JVM).

- A medida que la matriz de entrada para la prueba aumentaba de tamaño para el test, así mismo tomaba más tiempo para la ejecución del mismo, a excepción de los tests en el lenguaje php, el cual es siempre estable.

- La estabilidad de las pruebas de matrices en el lenguaje de programación PHP se puede deber a la forma en la que el lenguaje maneja la memoria y el acceso a los elementos en la matriz, ya que en PHP las matrices son arrays asociativos, almacenando las matrices como una tabla hash. PHP utiliza la función de hash para buscar la ubicación de un elemento en la matriz, independientemente del tamaño de la matriz. Por lo tanto una vez que la matriz se ha creado, el tiempo de acceso a los elementos será constante y no surgirá el tamaño de la matriz.

- Las pruebas en Linux resultaron ser considerablemente más rápidas que en Windows, esto puede ser debido a que Linux cuenta con una arquitectura más eficiente lo que permite un mejor uso de los recursos del sistema, Windows realiza muchas tareas en segundo plano (actualizaciones, análisis de virus, entre otras tareas de mantenimiento) que pueden influir en el desarrollo de los test y a demás se podría contar que la Linux permite una mayor personalización del sistema operativo permitiendo así eliminar software innecesario y optimizar la configuración del sistema.

- El tiempo de ejecución de los tests puede llegar a estar limitado por factores como la capacidad de procesamiento de la CPU, la cantidad de memoria RAM disponible en el sistema, la cantidad de procesos en segundo plano y la eficiencia y/o manejo de los datos de entrada en cada lenguaje en específico (por lo general el tiempo de ejecución de un algoritmo es proporcional al tamaño de la entrada, en este caso de la matriz).