

Informe de Laboratorio 03

Tema: Insertion/Quicksort

Nota

Estudiante	Escuela	Asignatura
Mauricio Mendoza Choque Freddy Ancalla Marquez Jose Chino Nina mmendozac@ulasalle.edu.pe	Carrera Profesional de Ingeniería de Software	Lenguaje de Programación II Semestre: III Código: 20231001

Laboratorio	Tema	Duración
03	Insertion/Quicksort	06 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2025 - B	Del 15 setiembre 2025	Al 22 setiembre 2025

1. Trabajo

- En clase probamos un código de Algoritmo de ordenación por inserción, cual se no encargo agregar los commits a continuación:
- Agregar función para crear un arreglo random
- generar arreglos de 1 hasta N
- generar peores casos para arreglo
- evaluar tiempo de ejecución para cada arreglo
- graficar los tiempos de ejecución
- Enviar trabajo al profesor en un repositorio GitHub Privado, dándole permisos como colaborador.

2. Equipos, materiales y temas utilizados

- Sistema Operativo Windows 10/11 de 64 bits.
- Símbolo del sistema (CMD, Command Prompt).
- Windows PowerShell.
- OpenJDK 64-Bits 17.0.7.
- Cuenta en GitHub con el correo institucional.
- Algoritmo de ordenamiento por inserción

3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- <https://github.com/mmendozac-rgb/lp2.git>
- URL para el laboratorio 01 en el Repositorio GitHub.
- <https://github.com/mmendozac-rgb/lp2/tree/main/lab03>

4. Commits que se Realizaron

4.1. Realizando los commits

- Se realizaron los siguientes commits en la computadora:

Listing 1: Probando Algoritmo Insertion Sort GeeksForGeeks – Cormen

```
Se copi el cdigo de la pgina web GFG para probar algoritmo.
public class InsertionSort {
void sort(int arr[])
{
    int n = arr.length;
    for (int i = 1; i < n; ++i) {
        int key = arr[i];
        int j = i - 1;

        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
}
static void printArray(int arr[])
{
    int n = arr.length;
    for (int i = 0; i < n; ++i)
        System.out.print(arr[i] + " ");

    System.out.println();
}
public static void main(String args[])
{
    int arr[] = { 12, 11, 13, 5, 6 };

    InsertionSort ob = new InsertionSort();
    ob.sort(arr);

    printArray(arr);
}
}
```

Listing 2: Agregar función para crear arreglo random

```
Se creo una funcin que recibe N por parte del usuario y crea un arreglo de tamao N con
nmeros aleatorios desde el 1 hasta el 100.
import java.util.Scanner;
import java.util.Random;

public class InsertionSort {

    void sort(int arr[])
    {
        int n = arr.length;
        for (int i = 1; i < n; ++i) {
            int key = arr[i];
            int j = i - 1;

            while (j >= 0 && arr[j] > key) {
                arr[j + 1] = arr[j];
                j = j - 1;
            }
            arr[j + 1] = key;
        }
    }

    static void printArray(int arr[])
    {
        int n = arr.length;
        for (int i = 0; i < n; ++i)
            System.out.print(arr[i] + " ");
        System.out.println();
    }

    // Genera arreglo aleatorio de tamao N
    static int[] generarArreglo(int N) {
        int[] arr = new int[N];
        Random rand = new Random();

        for (int i = 0; i < N; i++) {
            arr[i] = rand.nextInt(100) + 1; // entre 1 y 100
        }

        return arr;
    }

    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        System.out.print("Ingrese el tamao del arreglo (N): ");
        int N = sc.nextInt();

        int arr[] = generarArreglo(N);

        System.out.println("Arreglo generado:");
        printArray(arr);

        InsertionSort ob = new InsertionSort();
        ob.sort(arr);

        System.out.println("Arreglo ordenado:");
```

```
        printArray(arr);  
  
        sc.close();  
    }  
}
```

Listing 3: Generar arreglos desde 1 hasta N

El programa genera arreglos desde tamaño 1 hasta tamaño N.

```
import java.util.Scanner;  
  
public class InsertionSort {  
    void sort(int arr[]) {} // innecesario, pero se mantiene  
    static void printArray(int n) {  
        System.out.print("(");  
        for (int i = 1; i <= n; i++) {  
            if (i > 1) System.out.print(",");  
            System.out.print(i);  
        }  
        System.out.println(")");  
    }  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        System.out.print("Ingrese el tamaño máximo del arreglo (N): ");  
        int N = sc.nextInt();  
        for (int tam = 1; tam <= N; tam++) {  
            System.out.print("Arreglo de tamaño " + tam + ": ");  
            printArray(tam);  
        }  
        sc.close();  
    }  
}
```

Listing 4: Generar peores casos para cada arreglo

Una función recibe el arreglo y lo inicializa con un peor caso de acuerdo a su tamaño.

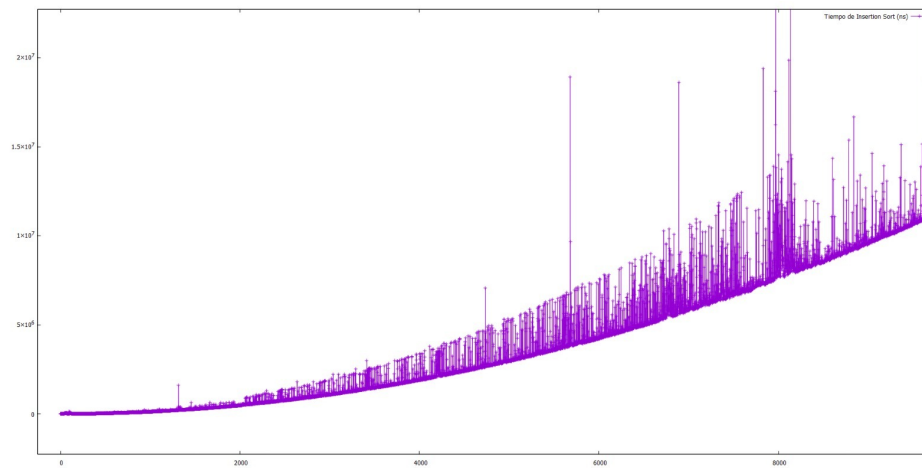
```
import java.util.Scanner;  
  
public class InsertionSort {  
    void sort(int arr[]) {} // innecesario, pero se mantiene  
  
    static void printArray(int n) {  
        System.out.print("(");  
        for (int i = n; i >= 1; i--) {  
            System.out.print(i);  
            if (i > 1) System.out.print(",");  
        }  
        System.out.println(")");  
    }  
  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        System.out.print("Ingrese el tamaño máximo del arreglo (N): ");  
        int N = sc.nextInt();
```

```
for (int tam = 1; tam <= N; tam++) {  
    System.out.print("Arreglo de tamaño " + tam + ": ");  
    printArray(tam);  
}  
  
sc.close();  
}  
}
```

][H] En un archivo de texto plano en cada línea debe estar el tiempo de ejecución de cada arreglo.

Listing 5: Graficar los tiempos de ejecución

Usando el programa GNUPplot "tiempos.dat" with linespoints title "Tiempo de Insertion Sort" se pudo sacar la grafica correspondiente:



Listing 6: tiempo en nanosegundos

```
Esto es una representacion de como se ejecuta en nanosegundos Code:
import java.util.Scanner;
import java.io.FileWriter;
import java.io.IOException;

public class InsertionSort {
    void sort(int arr[]) {}

    static void printArray(int n) {
        System.out.print("(");
        for (int i = n; i >= 1; i--) {
            System.out.print(i);
            if (i > 1) System.out.print(",");
        }
        System.out.println(")");
    }

    static void insertionSort(int[] arr) {
        int n = arr.length;
        for (int i = 1; i < n; ++i) {
            int key = arr[i];
            int j = i - 1;
            while (j >= 0 && arr[j] > key) {
                arr[j + 1] = arr[j];
                j = j - 1;
            }
            arr[j + 1] = key;
        }
    }

    static int[] generateArray(int n) {
        int[] arr = new int[n];
        for (int i = 0; i < n; i++) arr[i] = n - i;
        return arr;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Ingrese el tamaño máximo del arreglo (N): ");
        int N = sc.nextInt();

        try (FileWriter fw = new FileWriter("tiempos.dat")) {
            for (int tam = 1; tam <= N; tam++) {
                System.out.print("Arreglo de tamaño " + tam + ": ");
                printArray(tam);

                int[] arr = generateArray(tam);

                long inicio = System.nanoTime();
                insertionSort(arr);
                long fin = System.nanoTime();

                long tiempo = fin - inicio;

                fw.write(tam + " " + tiempo + "\n");
            }
        }
    }
}
```

```
    }  
  } catch (IOException e) {  
    System.out.println("Error escribiendo el archivo: " + e.getMessage());  
  }  
  
  sc.close();  
  
  System.out.println("\nArchivo 'tiempos.dat' generado con tiempos en nanosegundos.");  
}  
}
```

```
1 800  
2 700  
3 600  
4 600  
5 700  
6 800  
7 1200  
8 1200  
9 1300  
10 1600  
11 1800  
12 2000  
13 2300  
14 2700  
15 2800  
16 3100  
17 3400  
18 4700  
19 4000  
20 4500  
21 5000  
22 5600  
23 6200  
24 6700  
25 6100  
26 7700  
27 7900  
28 7900  
29 9400  
30 9100  
31 11300  
32 11400  
33 11200  
34 11900  
35 13100  
36 13400  
37 12900  
38 14500  
39 15000  
40 15500  
41 17400  
42 15900
```

5. Calificación

Tabla 1: Rúbrica para contenido del Informe y evidencias

Contenido y demostración		Puntos	Checklist	Estudiante	Profesor
1. GitHub	Repositorio se pudo clonar y se evidencia la estructura adecuada para revisar los entregables. (Se descontará puntos por error o observación)	4	X	4	
2. Commits	Hay porciones de código fuente asociado a los commits planificados con explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	X	4	
3. Ejecución	Se incluyen comandos para ejecuciones y pruebas del código fuente explicadas gradualmente que permitirían replicar el proyecto. (Se descontará puntos por cada omisión)	4	X	4	
4. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	X	2	
7. Ortografía	El documento no muestra errores ortográficos. (Se descontará puntos por error encontrado)	2	X	2	
8. Madurez	El Informe muestra de manera general una evolución de la madurez del código fuente con explicaciones puntuales pero precisas, agregando diagramas generados a partir del código fuente y refleja un acabado impecable. (El profesor puede preguntar para refrendar calificación).	4			
Total		20		16	

6. Referencias

- <https://www.geeksforgeeks.org/dsa/quick-sort-algorithm/>
- <https://www.geeksforgeeks.org/insertion-sort/>

Listing 7: Foto leyendo el libro

Esto es la foto leyendo el libro de corman

