



**INSTITUTO POLITÉCNICO NACIONAL**

**ESCUELA SUPERIOR DE CÓMPUTO**

**SISTEMAS DISTRIBUIDOS**

**PRÁCTICA 1: PROCESOS E HILOS**

**ALUMNO: GONZÁLEZ CASTILLO ALEXIS GABRIEL**

**GRUPO: 7CM1**

**PROFESOR: CHADWICK CARRETO ARELLANO**

**SEMESTRE: 25-26/2**

**FECHA DE ENTREGA: 20 DE FEBRERO DE 2025.**

## 1. Tabla de contenido.

1. Tabla de contenido. ....	2
2. Antecedente .....	3
3. Planteamiento del problema.....	3
4. Propuesta de solución.....	3
5. Materiales y Métodos.....	4
6. Desarrollo de la solución.....	4
7. Resultados .....	7
8. Conclusiones .....	8
9. Link a Github.....	8

## **2. Antecedente**

Hoy en día, procesar grandes volúmenes de datos es una necesidad común en muchas aplicaciones de software. La búsqueda de elementos en arreglos es una operación clave en sistemas como bases de datos, motores de búsqueda y aplicaciones científicas. Sin embargo, cuando los arreglos son demasiado grandes, la búsqueda secuencial puede volverse lenta y poco eficiente.

En el contexto de los sistemas distribuidos, el procesamiento concurrente es una estrategia fundamental para mejorar el rendimiento y la escalabilidad de las aplicaciones. A través del uso de múltiples hilos de ejecución, es posible dividir tareas y ejecutarlas en paralelo, reduciendo los tiempos de respuesta y optimizando el uso de los recursos disponibles.

Esta práctica explora cómo el uso de hilos puede mejorar la eficiencia en la búsqueda de un número dentro de arreglos ordenados y desordenados. Se analizan distintas estrategias de concurrencia para dividir la tarea y evaluar su impacto en el desempeño del sistema.

## **3. Planteamiento del problema**

Dado un arreglo de 4 millones de elementos, se requiere buscar un número específico proporcionado por el usuario. El arreglo puede estar ordenado o desordenado, y la búsqueda debe realizarse de manera eficiente en términos de tiempo de ejecución.

El problema principal es ¿cómo mejorar el tiempo de búsqueda en arreglos grandes utilizando técnicas de concurrencia? Por lo tanto, es necesario implementar un proceso en Java donde se hará una búsqueda lineal y una búsqueda binaria, ambos casos buscarán un número ingresado por el usuario.

## **4. Propuesta de solución**

La solución propuesta es la siguiente.

- Dividir el arreglo en partes iguales: El arreglo se dividirá en subarreglos más pequeños, donde cada subarreglo es procesado por un hilo independiente.
- En el arreglo ordenado, se utilizará una búsqueda binaria, aprovechando sus propiedades.
- En el arreglo desordenado, se utilizará una búsqueda lineal.
- Se crean múltiples hilos para procesar los subarreglos de manera concurrente, aprovechando los recursos de hardware disponibles.

## 5. Materiales y Métodos

- **Lenguaje de programación:** Java (versión 17 o superior).
- **Entorno de desarrollo:** IntelliJ IDEA o cualquier IDE compatible con Java.
- **Archivo de datos:** Un archivo de texto (4millones.txt) que contiene 4 millones de números, uno por línea y el mismo archivo de datos pero ordenados.
- **Hardware:** Computadora con procesador multinúcleo para aprovechar la concurrencia.

## 6. Desarrollo de la solución

Clase main:

Solicito al usuario un número a buscar y se procedo a ejecutar los procesos de las dos búsquedas en hilos separados. Se ejecuta ambos hilos en paralelo y se espera a que terminen las búsquedas antes de continuar. Al finalizar se imprime un mensaje si se logró encontrar el número ingresado.

```
public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Ingrese un número para buscar: ");
        int numero = scanner.nextInt();

        Thread procesoOrdenado = new Thread(new BusquedaOrdenada(numero));
        procesoOrdenado.start();

        Thread procesoDesordenado = new Thread(new BusquedaDesordenada(numero));
        procesoDesordenado.start();

        try {
            procesoOrdenado.join();
            procesoDesordenado.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        System.out.println("Búsqueda completada.");
    }
}
```

## Clase BúsquedaOrdenada

En esta clase, se lee el archivo y almacena los datos en un arreglo, además de ordenar el arreglo.

Se crean dos hilos y a cada hilo se le pasa una parte del arreglo en donde buscará el número utilizando la búsqueda binaria. Como en el caso de la clase main, se espera a que terminen los hilos.

```
public class BusquedaOrdenada implements Runnable {
    private final int numero;

    public BusquedaOrdenada(int numero) {
        this.numero = numero;
    }

    @Override
    public void run() {
        try {
            File archivo = new File("src/main/resources/4millones.txt");
            Scanner scanner = new Scanner(archivo);

            int[] arreglo = new int[4000005];

            int index = 0;
            while (scanner.hasNextLine() && index < 4000000) {
                arreglo[index] = Integer.parseInt(scanner.nextLine().trim());
                index++;
            }

            Arrays.sort(arreglo, 0, index);

            int numHilos = 2;
            int tamCacho = arreglo.length / numHilos;

            Thread[] hilos = new Thread[numHilos];
            for (int i = 0; i < numHilos; i++) {
                int inicio = i * tamCacho;
                int fin = (i == numHilos - 1) ? arreglo.length : inicio + tamCacho;
                hilos[i] = new Thread(() -> buscarEnSubarreglo(arreglo, inicio, fin,
numero));
                hilos[i].start();
            }

            for (Thread hilo : hilos) {
                hilo.join();
            }

        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private void buscarEnSubarreglo(int[] arreglo, int inicio, int fin, int numero) {
        int izquierda = inicio;
        int derecha = fin - 1;

        while (izquierda <= derecha) {
            int medio = izquierda + (derecha - izquierda) / 2;
```

```

        if (arreglo[medio] == numero) {
            System.out.println("Número " + numero + " encontrado en posición " +
medio + " (Ordenado)");
            return;
        }

        if (arreglo[medio] < numero) {
            izquierda = medio + 1;
        } else {
            derecha = medio - 1;
        }
    }

    System.out.println("Número " + numero + " no encontrado en el subarreglo [" +
inicio + ", " + (fin - 1) + "] (Ordenado)");
}
}

```

## Clase BúsquedaDesordenada

En esta clase, se lee el archivo y almacena los datos en un arreglo.

Se crean dos hilos y a cada hilo se le pasa una parte del arreglo en donde buscará el número utilizando la búsqueda lineal. Como en el caso de la clase main, se espera a que terminen los hilos.

```

public class BusquedaDesordenada implements Runnable {
    private final int numero;

    public BusquedaDesordenada(int numero) {
        this.numero = numero;
    }

    @Override
    public void run() {
        try {
            File archivo = new File("src/main/resources/4millones.txt");
            Scanner scanner = new Scanner(archivo);

            int[] arreglo = new int[4000005];

            int index = 0;
            while (scanner.hasNextLine() && index < 4000000) {
                arreglo[index] = Integer.parseInt(scanner.nextLine().trim());
                index++;
            }

            int numHilos = 2;
            int tamCacho = arreglo.length / numHilos;

            Thread[] hilos = new Thread[numHilos];
            for (int i = 0; i < numHilos; i++) {
                int inicio = i * tamCacho;

```

```

        int fin = (i == numHilos - 1) ? arreglo.length : inicio + tamCacho;
        hilos[i] = new Thread(() -> buscarEnSubarreglo(arreglo, inicio, fin,
numero));

        hilos[i].start();
    }

    for (Thread hilo : hilos) {
        hilo.join();
    }

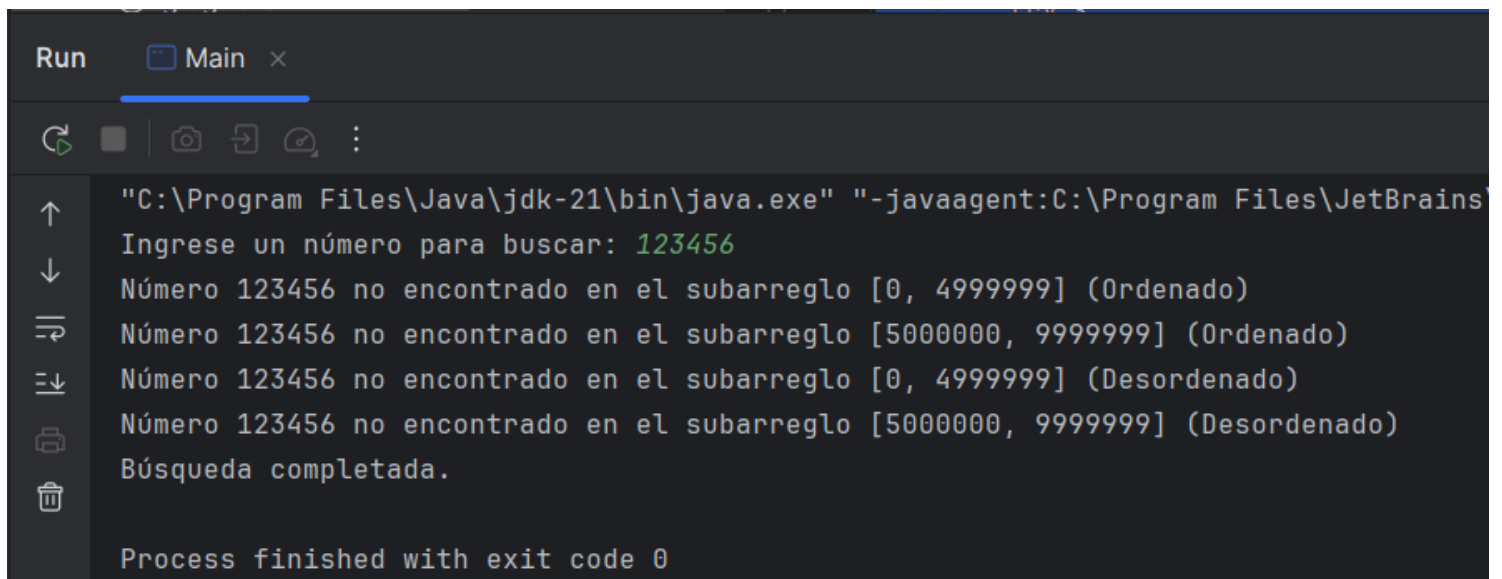
    } catch (Exception e) {
        e.printStackTrace();
    }
}

private void buscarEnSubarreglo(int[] arreglo, int inicio, int fin, int numero) {
    for (int i = inicio; i < fin; i++) {
        if (arreglo[i] == numero) {
            System.out.println("Número " + numero + " encontrado en posición " + i +
" (Desordenado)");
            return;
        }
    }
    System.out.println("Número " + numero + " no encontrado en el subarreglo [" +
inicio + ", " + (fin - 1) + "] (Desordenado)");
}
}

```

## 7. Resultados

Número a buscar: 123456



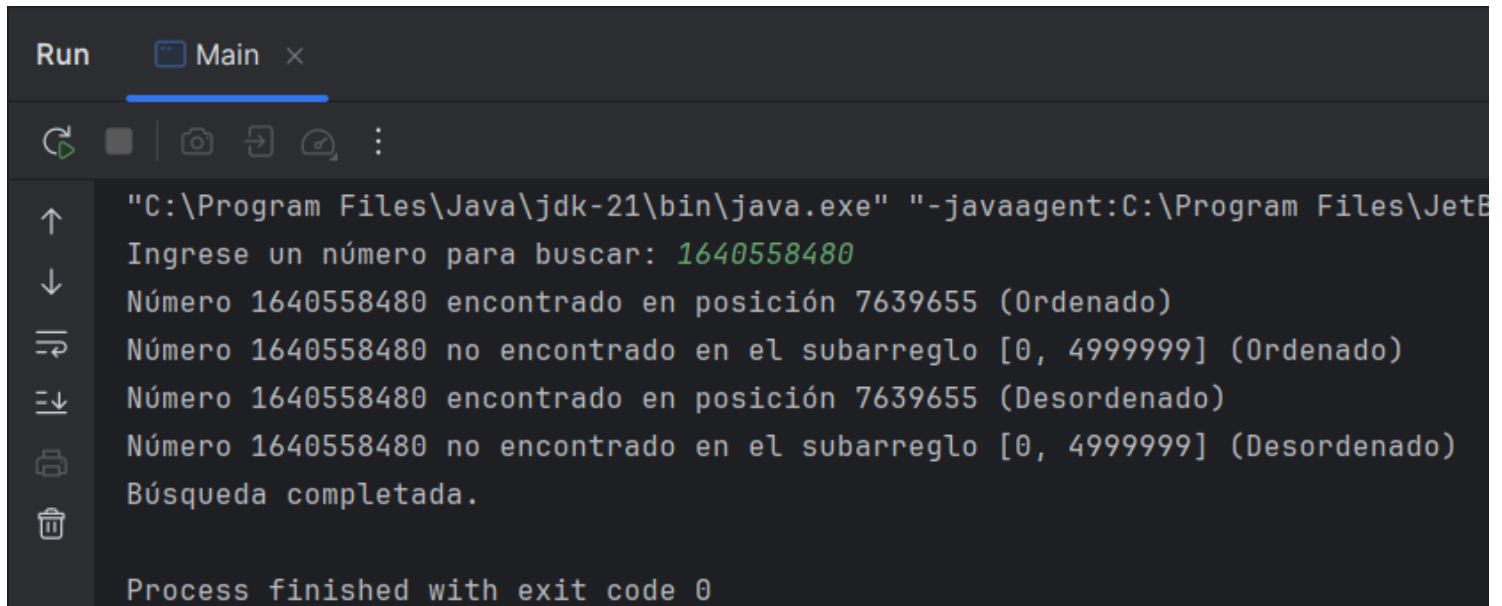
```

Run Main x
C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program Files\JetBrains
Ingrese un número para buscar: 123456
Número 123456 no encontrado en el subarreglo [0, 4999999] (Ordenado)
Número 123456 no encontrado en el subarreglo [5000000, 9999999] (Ordenado)
Número 123456 no encontrado en el subarreglo [0, 4999999] (Desordenado)
Número 123456 no encontrado en el subarreglo [5000000, 9999999] (Desordenado)
Búsqueda completada.
Process finished with exit code 0

```

Resultado: No se encontró el número, se puede observar cómo cada hilo devolvió que no encontró el número en el subarreglo que le correspondía.

Número a buscar: 1640558480



```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program Files\JetB
Ingrese un número para buscar: 1640558480
Número 1640558480 encontrado en posición 7639655 (Ordenado)
Número 1640558480 no encontrado en el subarreglo [0, 4999999] (Ordenado)
Número 1640558480 encontrado en posición 7639655 (Desordenado)
Número 1640558480 no encontrado en el subarreglo [0, 4999999] (Desordenado)
Búsqueda completada.

Process finished with exit code 0
```

Resultado: El número si se encontraba en el arreglo, cada hilo devolvió si lo encontró en su subarreglo, se observa como solo un subarreglo en cada búsqueda devolvió true.

## 8. Conclusiones

En esta práctica, se logró implementar una estrategia de procesamiento concurrente utilizando hilos en Java para la búsqueda de un número en arreglos ordenados y desordenados. A través de este enfoque, se pudo observar cómo los hilos permiten ejecutar tareas en paralelo, optimizando el tiempo de búsqueda en comparación con una ejecución secuencial.

Además, esta implementación sirvió como una introducción al concepto de sistemas distribuidos, ya que el uso de múltiples hilos es una técnica fundamental en la programación concurrente y distribuida. La práctica permitió comprender cómo dividir tareas y coordinarlas de manera eficiente, lo cual es clave en sistemas que requieren procesamiento paralelo y distribución de carga.

En conclusión, se cumplieron los objetivos planteados, demostrando el funcionamiento de los hilos y su importancia dentro del desarrollo de sistemas de alto rendimiento y distribuidos.

## 9. Link a Github

<https://github.com/alexis7gabriel/P1-Distribuidos>