Tarea 2

Parte II - Práctica (60p)

Ejercicio II.1 (20 puntos)

Genere vectores de tamaño N de número enteros aleatorios, donde N comienza en 50.000 y crece hasta 1.000.000, con incrementos de 50.000. El dominio de los valores de cada elemento va de 10.000 a 1.000.000. Luego realice **N** búsquedas aleatorias utilizando la estrategia de **búsqueda binaria** (esto es, busque un elemento aleatorio en el propio arreglo) y **N** búsquedas aleatorias usando una **búsqueda lineal**.

Cada función de búsqueda debe hacerse en un método que debe ser definido como genérico.

Genere la tabla parecida a la que se muestra abajo:

N	Búsqueda binaria				Búsqueda Lineal			
	T(n)en ms	T/N	$T/(N \log_2 N)$	T/N^2	T(n) en ms	T/N	$T/(N \log_2 N)$	T/N^2
50000								
100000								
150000								
..								

Observaciones:

- Para aplicar la búsqueda binaria el vector debe estar ordenado, para ello implemente el algoritmo de *quicksort* utilizando la mediana de tres para la elección del pivot (ver método en libro de WEISS). Para ello crea una clase separada denominada *Util*, donde deberá estar el código del algoritmo de ordenación. *El cálculo de tiempo no debe incluir la ordenación*. El método debe ser genérico para los tipos adecuados de ordenación.

- Su programa debe generar la tabla mencionada, es decir no debe solo transcribir los resultados en un documento. Utilice un formato legible para los decimales, utilice *printf()* para ello con las cadenas de formato adecuadas.

- Aquí se presenta un breve código de como calcular el tiempo de ejecución (en ms) en Java:

```
public  class MedicionTiempo {
    public static void main ( String [] args) {
        final int MAX_ITER = 100000000;
        long t1, t2;
        t1 = System.currentTimeMillis();
        /* Medimos cuanto tiempo lleva una iteracion de 10000000 */
        for (int k=0; k < max_iter; k++);
        t2 = System.currentTimeMillis();
        System.out.println("Tiempo : " + (t2 - t1) + "ms");
    }
}
```

Al final de su implementación escriba de forma clara y precisa, en la opción de comentarios de la actividad o enviando un archivo adicional, los resultados que obtuvo en sus experimentos. Explique los valores obtenidos y las tendencias de cada columna (que corresponden a la Búsqueda Binaria y la Búsqueda Lineal). Aquí debe colocar los resultados que obtuvo en su entorno de pruebas además indique qué plataforma de pruebas utilizó (CPU, RAM, Sistema Operativo, versión de Java, etc.). (2p)

Rúbrica	**Puntaje**
Implementación correcta de funciones de búsqueda definidos como generic	4
Implementación del Algoritmo de ordenación en clase separada denominada Util	2

Implementación correcta del experimento con generar la generación de la tabla de acuerdo al modelo mostrado	10
Comentarios relevantes sobre los hallazgos en el experimento. Se debe de incluir todo lo solicitado.	4

Ejercicio II.2 (20 puntos)

Dado un array de *n* elementos, se sabe que algunos elementos están duplicados; esto es, aparecen más de una vez en el array.

Implemente una clase Java genérica capaz de eliminar elementos duplicados en el array en tiempo $O(n \log(n))$.

Fundamentar el tiempo requerido en la opción de agregar comentario en la actividad o enviando un archivo adicional, o puede comentarlo en el mismo código.

Rúbrica	**Puntaje**
Implementación correcta, sin errores de compilación y de acuerdo a restricciones	10
Uso de generics	2
Provisión de datos de prueba	5
Fundamentación del tiempo requerido	3

Ejercicio II.3 (20 puntos)

Implemente los dos algoritmos `fib1` y `fib2` de abajo que calculan el n-ésimo número de Fibonacci.

```
function fib1(n)
if n = 0:   return 0
if n = 1:   return 1
return fib1(n − 1) + fib1(n − 2)
```

```
function fib2(n)
if n = 0 return 0
create an array f[0...n]
f[0] = 0, f[1] = 1
for i = 2...n:
   f[i] = f[i − 1] + f[i − 2]
return f[n]
```

Mida el tiempo que le toma a cada subrutina retornar un valor para y escriba un programa que genere la tabla de abajo.

N	**Tiempo de `fib1` en microsegundos**	**Tiempo de `fib2` en microsegundos**
1		
2		
3		
…		
50		

Observaciones:

- Su programa debe generar la tabla mencionada, es decir no debe solo transcribir los resultados en un documento. Utilice un formato legible para los decimales, utilice *printf()* para ello con las cadenas de formato adecuadas.

Al final de su implementación escriba de forma clara y precisa, en la opción de comentarios de la actividad o enviando un archivo adicional, los resultados que obtuvo en sus experimentos. Explique los valores obtenidos y las tendencias de cada columna. También se debe colocar los resultados que obtuvo en su entorno de pruebas. Escriba qué plataforma de pruebas utilizó (CPU, RAM, Sistema Operativo, versión de Java, etc.).

Estime analíticamente el tiempo probable de ejecución para N = 1000 (Fibo(1000)). Para ello debe determinar el rendimiento de cada algoritmo.

Rúbrica	**Puntaje**
Implementación correcta de `fib1`.	3
Implementación correcta de `fib2`.	3
Implementación correcta del experimento con la generación de la tabla de acuerdo al modelo mostrado	4
Comentarios relevantes sobre los hallazgos en el experimento. Se debe de incluir todo lo solicitado.	5
Estimación para un número N = 1000.	5