

1. Descarga la clase AnalisisAlgoritmos.java que se encuentra en Blackboard. Revisa su funcionamiento y discute en equipo su objetivo.

- ¿Para qué sirve la variable CANTIDAD\_ELEMENTOS?
- ¿Cuál es el objetivo del método nanoTime( ) de la clase System?
- ¿Qué imprime el programa?





2. Diseña un método que se encargue de instanciar y llenar un arreglo de enteros de tamaño size con valores enteros aleatorios. Incluye números negativos también. Prográmalo con la siguiente firma:

```
public static int[] refreshArray(int size)
```

Ejemplo: refreshArray(10) →

4	1	2	-3	1	8	5	3	-2	10
---	---	---	----	---	---	---	---	----	----

3. Selecciona un método de ordenamiento de cada columna, y agrega en la clase AnalisisAlgoritmos implementaciones de dichos algoritmos.

			
<ul style="list-style-type: none"> <li>• Bogo Sort (single swap)</li> <li>• Bogo Sort (100% random)</li> </ul>	<ul style="list-style-type: none"> <li>• Bubble Sort</li> <li>• Selection Sort</li> </ul>	<ul style="list-style-type: none"> <li>• Insertion Sort</li> <li>• Insertion Binary Sort</li> </ul>	<ul style="list-style-type: none"> <li>• Merge Sort</li> </ul>

4. Ejecuta y mide cada algoritmo de ordenamiento múltiples veces. Llena la tabla resultados.xlsx con los resultados obtenidos.

Algoritmo	# Elementos	Ejecución 1	Ejecución 2	Ejecución 3	Ejecución 4	Ejecución 5	Promedio
Bubble Sort	100						
Bubble Sort	1,000						
Bubble Sort	10,000						
Bubble Sort	100,000						
Bubble Sort	200,000						
Merge Sort	100						
Merge Sort	1,000						
Merge Sort	10,000						
Merge Sort	100,000						
Merge Sort	1,000,000						
Bogo Sort	3						
Bogo Sort	6						
Bogo Sort	9						
Bogo Sort	12						
Bogo Sort	15						

5. Grafica los resultados de la columna “Promedio” para cada uno de los algoritmos. Utiliza alguna herramienta de software como Excel, Wolfram Alpha, Matlab, etc.

6. Crea una presentación de PowerPoint con los resultados de tus mediciones. Incluye comparaciones entre los algoritmos, y responde las siguientes preguntas:

- ¿Cuál es el algoritmo de ordenamiento más rápido? Utiliza la notación Big-O para justificar tu respuesta.
- ¿Cuál es el algoritmo de ordenamiento más lento? Utiliza la notación Big-O para justificar tu respuesta.
- ¿Las gráficas se aproximan a la notación Big-O esperada?

- ¿Qué factores (de hardware, software, externos) existen que afecten los tiempos de ordenamiento entre distintas ejecuciones? Menciona por lo menos 5.

## Referencia:

### BogoSort (single swap)

```
public static void BogoSort(int[] array)
```

- i. Verifica si el arreglo está ordenado (de menor a mayor). De ser así, termine la ejecución del método.
- ii. Genera dos índices aleatorios e intercambia los elementos en dichas posiciones.
- iii. Verifique si arreglo está ordenado. De ser así, termine la ejecución.
- iv. Si el arreglo no está ordenado, continúa a partir del paso ii.

### BogoSort (100% random)

```
public static void BogoSort(int[] array)
```

- i. Verifica si el arreglo está ordenado (de menor a mayor). De ser así, termine la ejecución del método.
- ii. Intercambia todas las posiciones del arreglo de manera aleatoria.
- iii. Verifique si arreglo está ordenado. De ser así, termine la ejecución.
- iv. Si el arreglo no está ordenado, continúa a partir del paso ii.