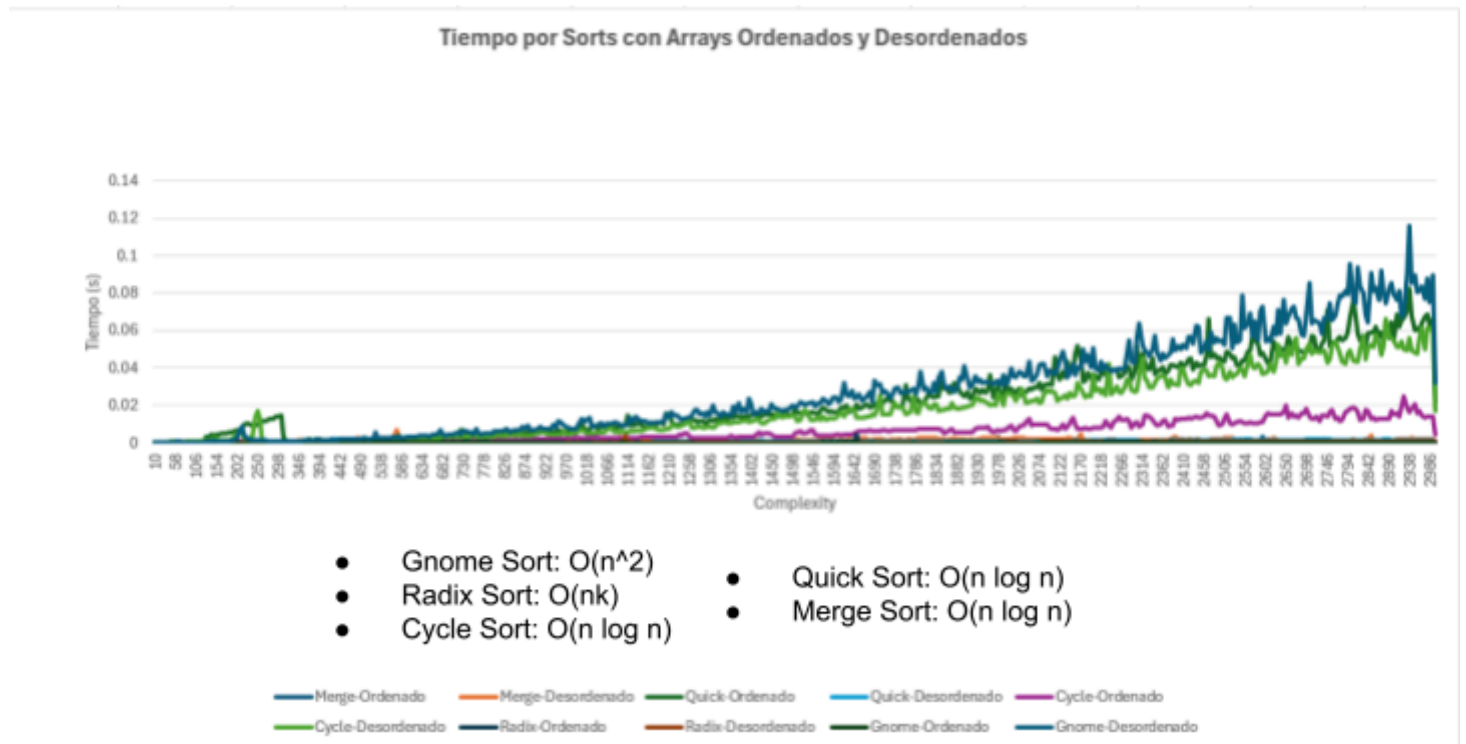


Hoja de Trabajo #3

a. Gráfico realizado en Excel:



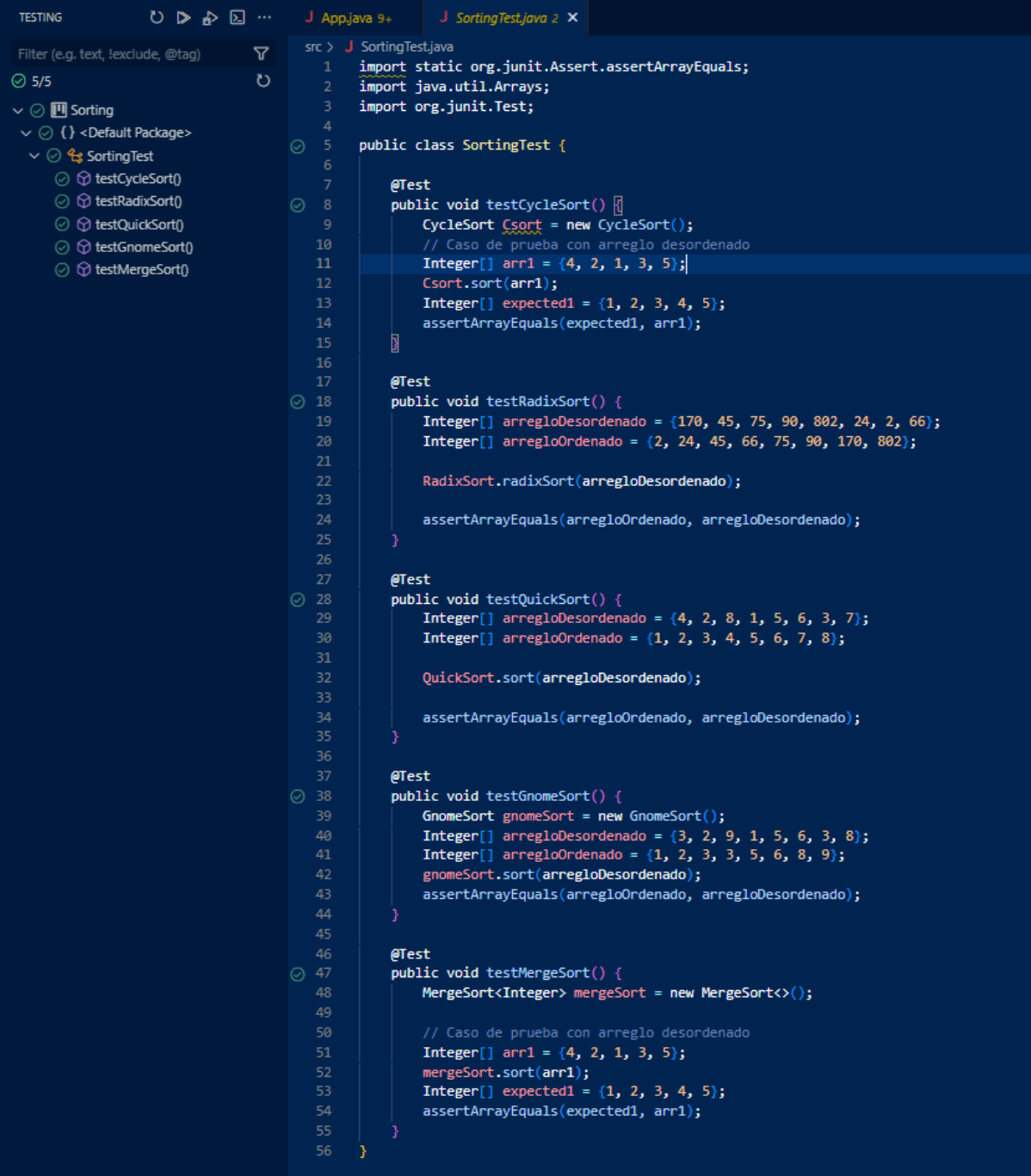
b. Link al repositorio (es público):

[Link a GitHub](#)

c. Profiler utilizado:

- i. Para esta hoja de trabajo se estuvo utilizando el profiler de JProfiler, este permite al usuario conocer cómo se maneja la memoria y los tiempos que le toma al procesador ejecutar diferentes instrucciones, entre otra vastedad de opciones. Este profiler tiene la ventaja que no depende de utilizar un IDE específico, sino que solamente se necesita tener un folder con los archivos .Class. Para poder utilizarlo se colocaba el working space como el folder con los archivos Class, y se indicaba el nombre del main. A continuación, se tenía que dirigir al apartado de CPU Memory, aquí se seleccionaba la opción de Complexity Analysis, se elegía el método que se deseaba estudiar, y posteriormente se exportaban los datos. Cada exportación fue colocada en una columna de un documento aparte, de esta manera se mantenía registro de cada uno de los tiempos que le tomó a los sorts finalizar su tarea, teniendo datos tanto ordenados como desordenados. Para la complejidad, se obtuvo que el quick, merge, y cycle sort tenían complejidad logarítmica, el gnome sort cuadrática, y el radix sort, lineal. El gráfico obtenido se colocó en el inciso "a".

d. Pruebas unitarias



The screenshot shows an IDE with a 'TESTING' tab active. On the left, a 'Filter (e.g. text, !exclude, @tag)' box shows '5/5' tests passed. Below it, a tree view shows the package structure: 'Sorting' (5/5) > '<Default Package>' > 'SortingTest' (5/5). The 'SortingTest' class contains five test methods: 'testCycleSort()', 'testRadixSort()', 'testQuickSort()', 'testGnomeSort()', and 'testMergeSort()'. The main editor displays the source code for 'SortingTest.java'. The code imports 'org.junit.Assert', 'java.util.Arrays', and 'org.junit.Test'. It defines a 'public class SortingTest' with five '@Test' methods. Each method tests a specific sorting algorithm by creating an instance, sorting a predefined array, and asserting it against an expected sorted array. The tests are: 'testCycleSort()' (uses CycleSort), 'testRadixSort()' (uses RadixSort), 'testQuickSort()' (uses QuickSort), 'testGnomeSort()' (uses GnomeSort), and 'testMergeSort()' (uses MergeSort). All tests are marked as passed with green checkmarks in the left margin.

```
src > J SortingTest.java
1  import static org.junit.Assert.assertEquals;
2  import java.util.Arrays;
3  import org.junit.Test;
4
5  public class SortingTest {
6
7      @Test
8      public void testCycleSort() {
9          CycleSort csort = new CycleSort();
10         // Caso de prueba con arreglo desordenado
11         Integer[] arr1 = {4, 2, 1, 3, 5};
12         csort.sort(arr1);
13         Integer[] expected1 = {1, 2, 3, 4, 5};
14         assertEquals(expected1, arr1);
15     }
16
17     @Test
18     public void testRadixSort() {
19         Integer[] arregloDesordenado = {170, 45, 75, 90, 802, 24, 2, 66};
20         Integer[] arregloOrdenado = {2, 24, 45, 66, 75, 90, 170, 802};
21
22         RadixSort.radixSort(arregloDesordenado);
23
24         assertEquals(arregloOrdenado, arregloDesordenado);
25     }
26
27     @Test
28     public void testQuickSort() {
29         Integer[] arregloDesordenado = {4, 2, 8, 1, 5, 6, 3, 7};
30         Integer[] arregloOrdenado = {1, 2, 3, 4, 5, 6, 7, 8};
31
32         QuickSort.sort(arregloDesordenado);
33
34         assertEquals(arregloOrdenado, arregloDesordenado);
35     }
36
37     @Test
38     public void testGnomeSort() {
39         GnomeSort gnomeSort = new GnomeSort();
40         Integer[] arregloDesordenado = {3, 2, 9, 1, 5, 6, 3, 8};
41         Integer[] arregloOrdenado = {1, 2, 3, 3, 5, 6, 8, 9};
42         gnomeSort.sort(arregloDesordenado);
43         assertEquals(arregloOrdenado, arregloDesordenado);
44     }
45
46     @Test
47     public void testMergeSort() {
48         MergeSort<Integer> mergeSort = new MergeSort<>();
49
50         // Caso de prueba con arreglo desordenado
51         Integer[] arr1 = {4, 2, 1, 3, 5};
52         mergeSort.sort(arr1);
53         Integer[] expected1 = {1, 2, 3, 4, 5};
54         assertEquals(expected1, arr1);
55     }
56 }
```