**Informe**

**Tiempos de ejecución de los algoritmos RadixSort, MergeSort y BubbleSort**

El siguiente informe tiene como fin exponer las diferencias en tiempo de ejecución de los algoritmos de ordenamiento RadixSort, MergeSort y BubbleSort.

Las siguientes pruebas se hicieron sobre arreglos de tamaño: 100 (cien), 1.000 (mil), 10.0000 (diez mil), 100.000 (cien mil), 1.000.000 (un millón) y 10.000.000 (diez millones) sobre un rango de valores numéricos que abarca desde el 0 (cero) hasta el 10.000 (diez mil), con la propiedad de ser generados aleatoriamente mediante el uso de los métodos provistos en la clase ArrayGenerator.

En la siguiente tabla se exponen los resultados de tiempo de ejecución de dichos algoritmos. Algunas pruebas de tiempo fueron realizadas sobre la unidad de medida de nano segundos, y las restantes en segundos.

|  |  |  |  |
| --- | --- | --- | --- |
| **Cantidad de elementos** | **Tiempos de**  **algoritmo RadixSort** | **Tiempos de**  **algoritmo MergeSort** | **Tiempos de**  **algoritmo BubbleSort** |
| **100** | 1.720.400 ns | 88.000 ns | 389.900 ns |
| **1.000** | 1.649.100 ns | 701.700 ns | 9.532.100 ns |
| **10.000** | 8.403.500 ns | 2.889.300 ns | 434.879.600 ns |
| **100.000** | 41.921.900 ns | 43.117.800 ns | 53 segundos |
| **1.000.000** | 490.209.800 ns | 374.109.800 ns | - |
| **10.000.000** | 5 segundos | 4 segundos | - |

Como conclusión vemos que BubbleSort solo se desempeña correctamente para arreglos de tamaños muy pequeños, mientras que en el resto de casos su tiempo lo hace inutilizable.

En los casos de RadixSort y MergeSort, ambos se desempeñan correctamente con grandes cantidades de elementos, pero teniendo un costo adicional en uso de memoria.