Universidad del Valle de Guatemala
Algoritmos y Estructura de Datos
Sección 20

04/03/2022

Hoja de Trabajo 3

Javier Ramírez 21700

Ángel Castellanos 21600

Mario Cristales 21631

Repositorio de Github:

https://github.com/J4v1er-502/Hoja-de-Trabajo-3

Resultados con Visual VM Profiler

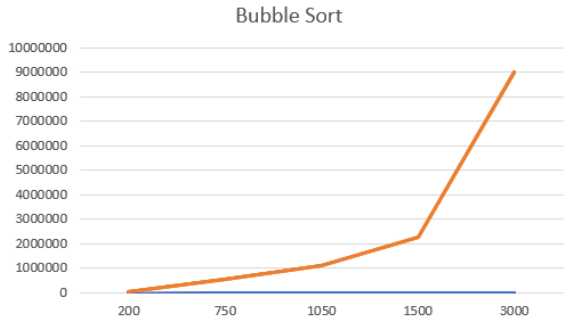
No. Datos	Bubble	Gnome	Radix	Quick	Merge
200	217	6.66	17.1	0.163	4.3
750	336	2.42	12.3	0.19	1.5
1050	294	6.14	6.6	0.209	3.44
1500	523	1.93	16.7	0.239	6.36
3000	732	0.027	15.1	0.247	5.21

Gráfica 1: Tiempo de corrida en ms sin ser previamente ordenado.

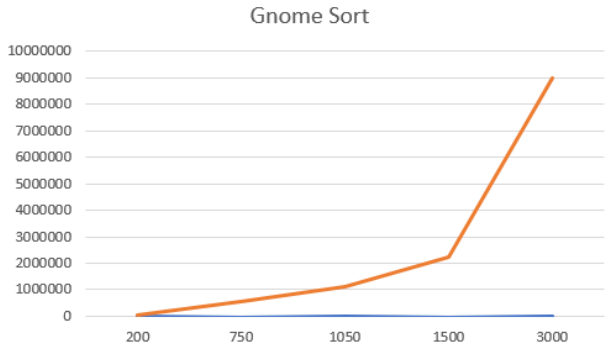
No. Datos	Bubble 2	Gnome 2	Radix 2	Quick 2	Merge 2
200	376	6.12	6.9	0.121	4.56
750	42.1	2.35	1.78	0.179	2.35
1050	179	164	4.34	0.141	4.68
1500	124	0.035	5.21	0.082	4.39
3000	593	8.34	7.81	0.117	4.95

Gráfica 2: Tiempo de corrida en ms previamente ordenado.

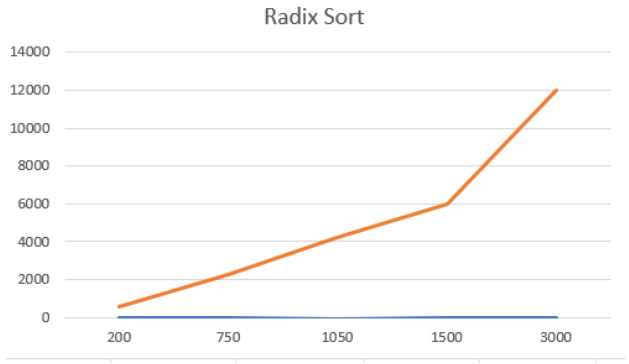
Gráficas:



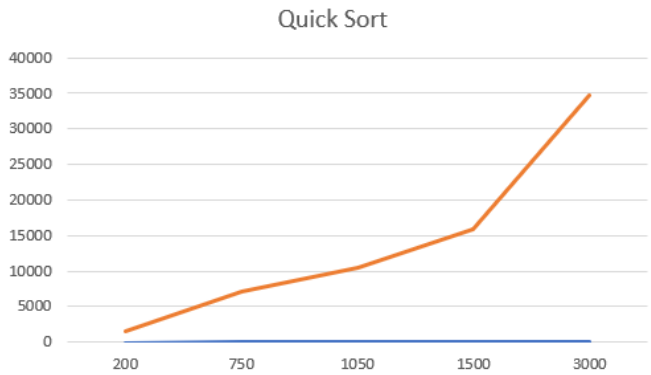
Gráfica 1: Bubble sort rendimiento práctico comparado con el rendimiento teórico.



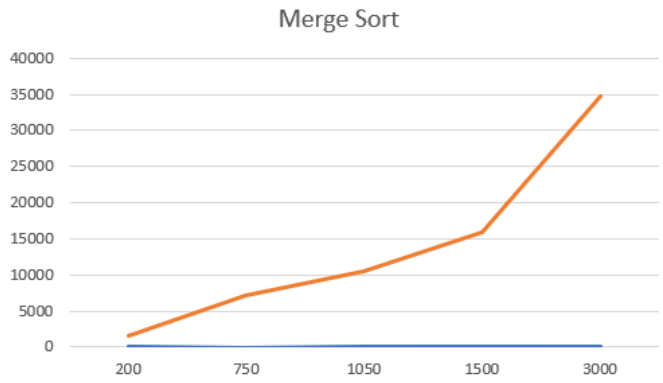
Gráfica 2: Gnome sort rendimiento práctico comparado con el rendimiento teórico.



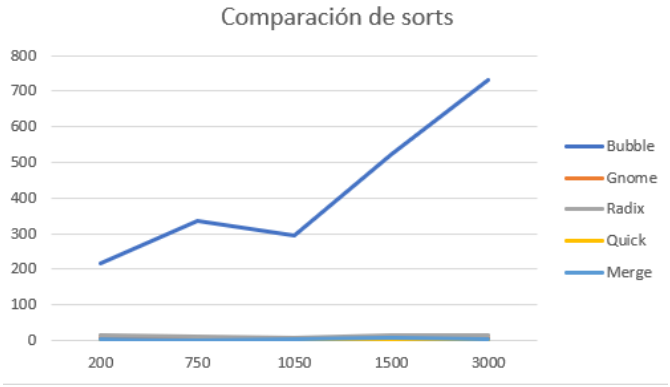
Gráfica 3: Radix sort rendimiento práctico comparado con el rendimiento teórico.



Gráfica 4: Quick sort rendimiento práctico comparado con el rendimiento teórico.



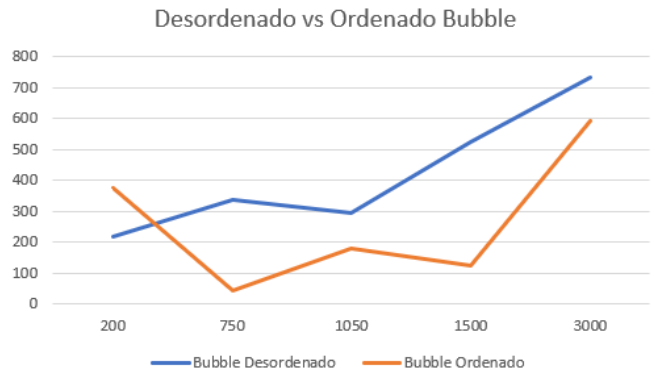
Gráfica 5: Merge sort rendimiento práctico comparado con el rendimiento teórico.



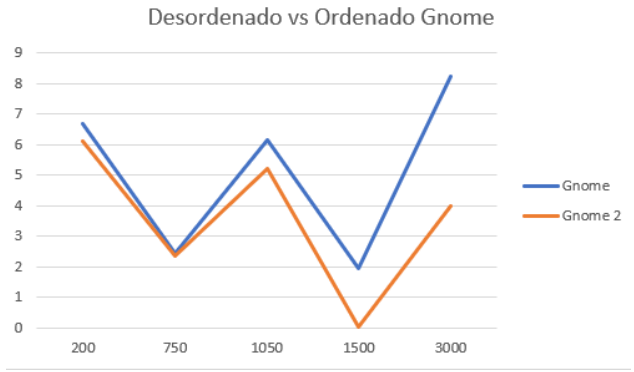
Gráfica 6: Comparación de rendimiento entre todos los sorts.

Rendimiento esperado y obtenido:

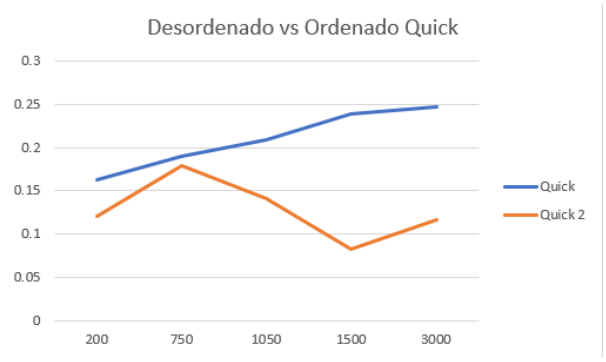
1) BubbleSort: El rendimiento esperado de BubbleSort es de $O(n^2)$ operaciones realizadas, con el rendimiento más alto obtenido de 732 ms.
2) GnomeSort: De igual manera el rendimiento esperado para GnomeSort es de $O(n^2)$ operaciones realizadas, en el rendimiento obtenido se obtuvo un rendimiento más alto de 8.24 ms.
3) RadixSort: Opera con tiempo O(nw), donde n es el número de datos y w la longitud de los datos. Se obtuvo el rendimiento más alto de 16.7 ms.
4) QuickSort: El rendimiento esperado de QuickSort es de O(n log(n)), mientras se obtuvo el rendimiento más alto de 0.247 ms.
5) MergeSort: El rendimiento de MergeSort es de O(n log(n)), y se obtuvo el rendimiento más alto de 6.36 ms.



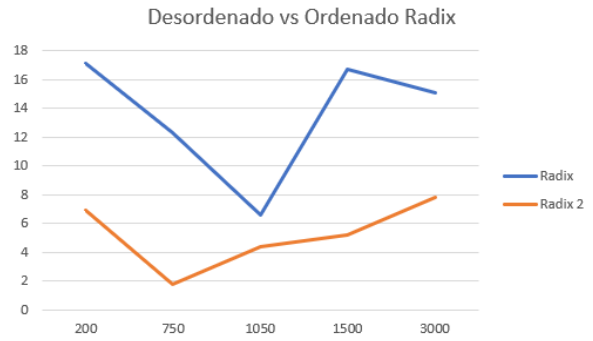
Gráfica 7: Ordenamiento BubbleSort



Gráfica 8: Ordenamiento GnomeSort



Gráfica 9: Ordenamiento QuickSort



Gráfica 10: Ordenamiento RadixSort.

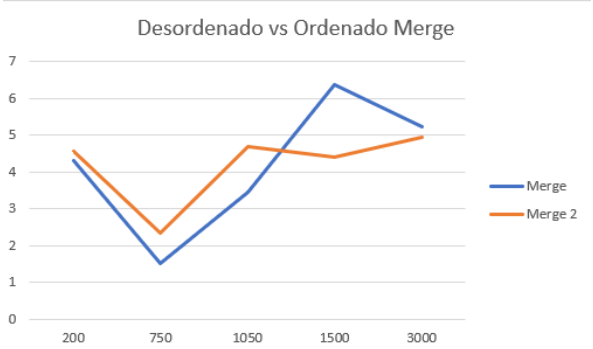


Gráfico 11: Ordenamiento MergeSort

Para realizar estos estudios se utilizó el Profiler de Visual VM, pues es un profiler bastante accesible y sencillo de utilizar. La instalación no requiere tantos pasos, y al momento de analizar la recolección de datos se pueden identificar estos de manera fácil. Además, Visual VM cuenta con demasiada información en internet y video tutoriales, por lo que cualquier problema o duda que ocurriera, se resolvía rápidamente. Se utilizó en Eclipse, pues instalarlo y hacer uso de Visual VM en Eclipse IDE fue explicado en clase y es sencillo de realizar. Solo se utiliza el .exe al momento de correr el programa, y Visual VM se encarga del resto.