Universidad del Valle de Guatemala

Algoritmos y Estructuras de Datos – Sección 31

Guatemala, Agosto 1 de 2016

Análisis de Complejidad *Sorts* – Hoja de Trabajo No. 3

Profiler utilizado

El *profiler* que se empleó fue VisualVM, que hace un enlace directo de las clases en Java con la máquina virtual para llevar un control de los métodos basados la cantidad de veces que fueron llamados y la cantidad de operaciones que realizan.

Captura de pantalla del *profiling* de los algoritmos de ordenamiento, con n = 3000

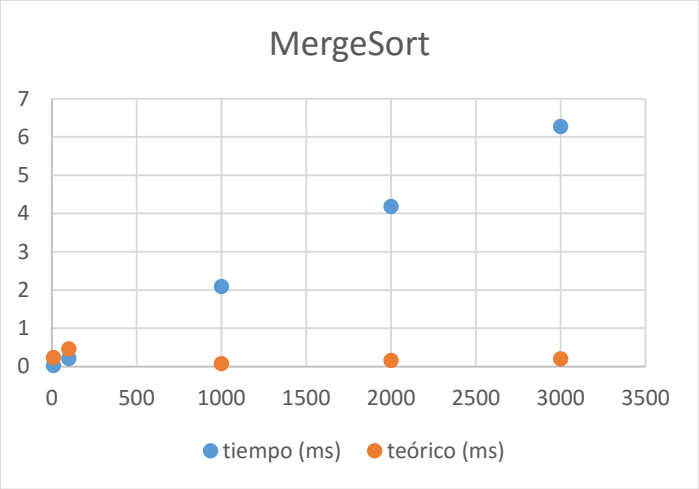
Call Tree - Method	Total Time [%]	Total Time	Invocations
main		23.0 ms (100%)	1
Sorts.GnomeSort (Estructura)		14.8 ms (64.3%)	1
Sorts.MergeSort (int[])		6.33 ms (27.4%)	1
Sorts.MergeSort (int[])		6.27 ms (27.2%)	2
Self time		0.039 ms (0.2%)	1
Sorts.Merge (int[], int[], int[])		0.012 ms (0.1%)	1
Sorts.RadixSort (Estructura)		1.74 ms (7.6%)	1
Sorts.QuickSort (int[], int, int)		0.160 ms (0.7%)	1
Sorts.QuickSort (int[], int, int)		0.084 ms (0.4%)	2
Sorts.QuickSort (int[], int, int)		0.080 ms (0.3%)	4
Sorts.Particion (int[], int, int)		0.002 ms (0%)	2
Self time		0.002 ms (0%)	2
Self time		0.066 ms (0.3%)	1
Sorts.Particion (int[], int, int)		0.008 ms (0%)	1
Thread-0		0.281 ms (100%)	1
DestroyJavaVM		0.000 ms (0%)	1

Complejidad de los algoritmos empleados

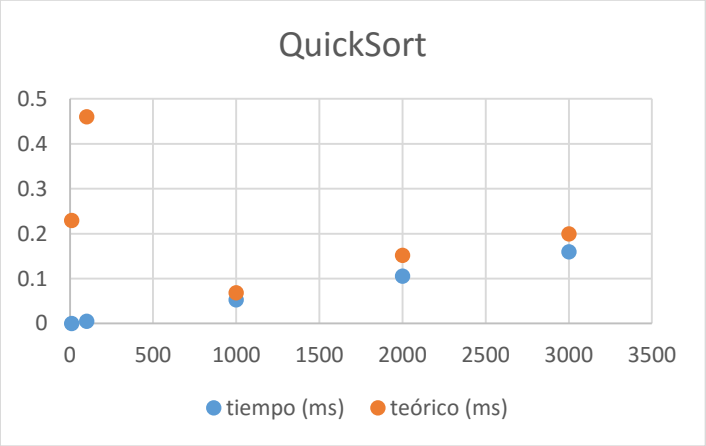
Algoritmo	Mejor	Peor	Promedio
QuickSort	O(nlogn)	$O(n^2)$	O(nlogn)
MergeSort	O(nlogn)	O(nlogn)	O(nlogn)
RadixSort	O(nk)	O(nk)	O(nk)
GnomeSort	O(n)	$O(n^2)$	$O(n^2)$

Comparación de los tiempos de ejecución de los algoritmos, tiempo teórico vs tiempo dado por el *profiler*

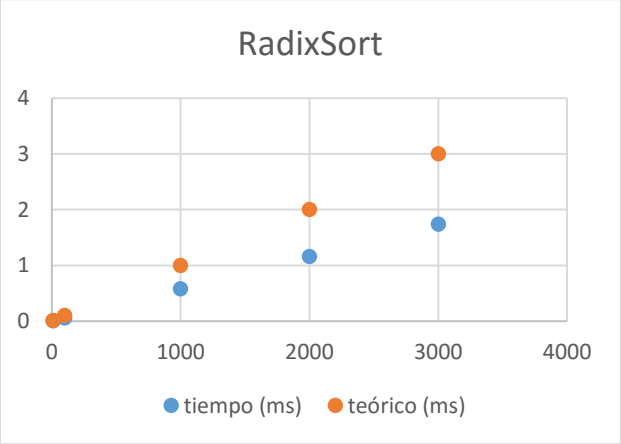
MergeSort



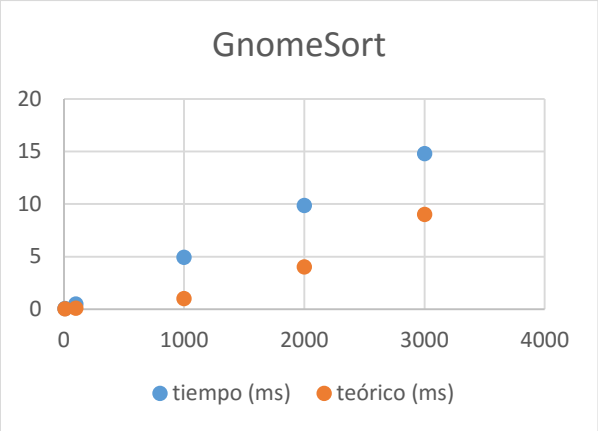
QuickSort



RadixSort



GnomeSort


GnomeSort
20
15
10
5
0
0
1000
2000
3000
4000
tiempo (ms)
teórico (ms)