

Estructura de Datos

Pablo Argueta 13028

Héctor Möller 09002

HOJA 06

Call Tree - Method	Total Time [%]	Total Time	Total Time (CPU)
main		2,005 ms (100%)	273 ms
Maintst.main (String[])		2,005 ms (100%)	273 ms
javax.swing.JOptionPane.showOptionDialog (java.awt.Component, Object, String, int, int, javax.s		1,975 ms (98.5%)	242 ms
java.awt.Component.<clinit>		19.9 ms (1%)	19.9 ms
java.lang.ClassLoader.loadClass (String)		10.7 ms (0.5%)	10.7 ms
Self time		0.000 ms (0%)	0.000 ms

Fig. 1: profiler utilizando HashSet.

Call Tree - Method	Total Time [%]	Total Time	Total Time (CPU)
main		1,368 ms (100%)	272 ms
Maintst.main (String[])		1,368 ms (100%)	272 ms
javax.swing.JOptionPane.showOptionDialog (java.awt.Component, Object, String, int, int, javax.s		1,348 ms (98.5%)	252 ms
java.awt.Component.<clinit>		19.9 ms (1.5%)	19.9 ms
Self time		0.000 ms (0%)	0.000 ms

Fig. 2: profiler utilizando TreeSet.

Call Tree - Method	Total Time [%]	Total Time	Total Time (CPU)
main		1,423 ms (100%)	327 ms
Maintst.main (String[])		1,423 ms (100%)	327 ms
javax.swing.JOptionPane.showOptionDialog (java.awt.Component, Object, String, int, int, javax.s		1,393 ms (97.9%)	297 ms
java.awt.Container.<clinit>		10.7 ms (0.8%)	10.7 ms
java.awt.Component.<clinit>		9.95 ms (0.7%)	9.95 ms
java.lang.ClassLoader.loadClass (String)		9.65 ms (0.7%)	9.65 ms
Self time		0.000 ms (0%)	0.000 ms

Fig. 3: profiler utilizando LinkedHashSet

c. Observando el profiler, la más rápida es TreeSet.

d. En clase se dedujo que la complejidad del HashMap para el HashSet es n tendiendo a constante. Entonces se usa constante $O(1)$. Ahora bien, como se tiene que comparar n valores del conjunto desarrolladores Java con n valores del otro conjunto (desarrolladores Web), entonces la complejidad total en el peor de los casos es de $1 + (n * n) = n^2 + n$. Para simplificar las cosas, la complejidad es de $O(n^2)$.