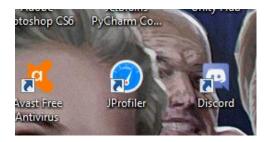
Sara Zavala 18893 Hoja de Trabajo 6 03 de Marzo de 2019

Profiler utilizado: JProfiler (IntelliJ)



HashMap: Mas lento

▼ m = 100.0% - 30,802 ms - 1 inv. Main.main

▶ @ 0.1% – 19,064 µs – 1 inv. MostrarElementosProfiler.mostrarElementos

▶ 0.0% - 12,002 µs - 1 inv. java.lang.ClassLoader.loadClass

TreeMap: Mas rápida

▼ m = 100.0% - 6,605 ms - 1 inv. Main.main

▶ 0.3% – 19,695 µs – 1 inv. MostrarElementosProfiler.mostrarElementos

▶ 0.2% - 15,220 µs - 1 inv. java.lang.ClassLoader.loadClass

LinkedMap: Intermedio

▼ m = 100.0% - 7,466 ms - 1 inv. Main.main

▶ 0.2% - 12,731 µs - 1 inv. java.lang.ClassLoader.loadClass

Calcule la complejidad de tiempo para la implementación HashMap, para mostrar todas las cartas. Indique como llegó a ese resultado.

Dos algoritmos pueden tener la misma complejidad, pero uno puede funcionar mejor que el otro. Recuerde que f(N) is O(N) significa que:

 $C1*N \le limit(f(N), N -> infinity) \le C2*N$

Donde C1 y C2 son constantes estrictamente positivas. La complejidad no dice nada sobre cuán pequeños o grandes son los valores de C Para dos algoritmos diferentes, las constantes probablemente serán diferentes.

Hashmap funciona según el principio del hash y utiliza internamente el código hash como base para almacenar el par clave-valor. Con la ayuda de hashcode, Hashmap distribuye los objetos a través de los cubos de tal manera que hashmap coloca los objetos y los recupera en tiempo constante O (1).

El mejor y promedio caso de Hashmap para búsqueda, inserción y eliminación es O (1) y el peor de los casos es O (n).