José Antonio Mérida Castejón - 201105

Adrián López - 21357

11-03-2024

Algoritmos y Estructuras de Datos

Hoja de Trabajo #6

Link al Repositorio en Github: https://github.com/TonitoMC/HDT6_AED

Análisis de Tiempo de Corrida Total:

El tiempo tomado para ejecutar la totalidad del método showAvailable() para cada implementación de la interfaz Map utilizada dentro del programa:

HashMap

```
Nombre: Vendread RevenantsTipo: Monstruo
Nombre: Ballista of Rampart SmashingTipo: Hechizo
Tiempo Tomado: 21002701
```

TreeMap

```
Nombre: Zure, Knight of Dark WorldTipo: Monstruo
Nombre: Zushin the Sleeping GiantTipo: Monstruo
Tiempo Tomado: 20071000
```

LinkedHashMap

```
Nombre: ZW - Ultimate ShieldTipo: Monstruo
Nombre: ZW - Unicorn SpearTipo: Monstruo
Tiempo Tomado: 18307800
```

La implementación más rápida fue LinkedHashMap, aunque no fue por mucho. Las tres implementaciones tuvieron un tiempo de corrida similar, únicamente difiriendo por unos pocos milisegundos. También vale la pena recalcar que hay situaciones en las que el tiempo de corrida puede ser mayor o menor para los diferentes algoritmos, este no es completamente consistente y lo más adecuado sería obtener un promedio de una cantidad más grande de corridas. Los valores mostrados fueron los valores que más frecuentemente se mostraron y se acercaban a la media en el tiempo de corrida.

Análisis de Complejidad:

```java
public void showAvailable(){
    //Toma el set de keys del map y para cada key imprime el nombre y el tipo
    long startTime = System.nanoTime();
    Set<String> keySet = availableCards.keySet();
    for (String key : keySet){
        System.out.println("Nombre: " + key + "Tipo: " + availableCards.get(key));
    }
    long endTime = System.nanoTime();
    long duration = endTime - startTime;
    System.out.println("Tiempo Tomado: " + duration);
}
```

Analizando el método línea por línea, availableCards.keySet() tiene complejidad O(1). Este únicamente retorna el objeto ya asociado con el HashMap.

El problema cae a la hora de obtener los Values de cada Key, en este set específico de datos de cartas ocurren colisiones dentro del HashMap. Esto sucede cuando diferentes Keys mappean al mismo valor dentro del HashMap, lo cuál causa que se deba recorrer una lista enlazada para obtener el valor deseado. En el peor de los casos, la complejidad sería O(n) para encontrar el valor deseado. No podría decir con certeza cuál es el caso, pero el HashMap probó ser la implementación menos eficiente, probablemente debido a este factor