Guatemala 13 de marzo de 2024
Mia Fuentes - 23775
Isabella Obando - 23074

Algoritmos y Estructura de Datos

Hoja de Trabajo no. 6

JProfiler

HashMap:

Esta implementación sirve para añadir o eliminar una carta, tiene un orden O(1). Al buscar la carta usa el mismo orden que añadir o eliminar, personal mostrar todas las cartas este tiene un orden O(n), donde “n” son todas las cartas en el mapa. Cuando se muestran las cartas ordenadas, usa un orden de $O(n \log_n)$

Resultados

Añadir cartas:

Promedio: 8683463400 nanosegundos

El tiempo de ejecución por el HashMap se mantiene en un rango adecuado para la implementación; cuando este tiene altos rangos de tiempo, puede depender del tamaño de la variable ingresada, en este caso sería el nombre de la carta.

Buscar cartas:

Promedio: 1485900 nanosegundos

La implementación del Hash al buscar las cartas es mucho más eficiente que añadir cartas, ya que es más sencillo realizar esta operación, hay tiempos de corrida mucho mayores que otras corridas, puede sugerir que hay más datos en los cuales busca el Hash o que hubo una colisión al ejecutarse.

Mostrar todas las cartas:

Promedio: 1355700 nanosegundos

En esta implementación ya interviene el JProfiler, en el cual ejecuta más rapido dependiendo los datos que vaya ingresando a la caché, por lo cual el tiempo de implementación es descendente.

Mostrar cartas en orden:

Promedio: 729840000 nanosegundos

El uso de Hash para esta implementación, tiene un tiempo de ejecución mucho más alto que los demás casos, ya que la ordenación usa una iteración más para el HashMap.