Universidad del Valle de Guatemala
Algoritmos y Estructura de Datos
Hoja de trabajo #6
Nadissa Vela 23764 | Roberto Barreda 23354

Análisis de Tiempo en Ejecución con Profiler y Complejidad para HashMap

Resultados:

- Cantidad de cartas: 200
- Tipos de cartas: 8
- Operaciones realizadas:
    - Agregar 30 cartas nuevas en “random”
    - Buscar 15 cartass específicas
    - Mostrar todas las cartas 2 veces
    - Mostrar las cartas ordenadas por tipo una vez nada más

Herramienta de profiling: JProfiler (versión de prueba)
Análisis de complejidad:
HashMap:

- Agregar o eliminar una carta: O(1) average
- Buscar una carta: O(1) en promedio, con posibilidad de colisiones que afecten el tiempo.
- Mostrar todas las cartas: O(n), donde n es la cantidad de cartas en el mapa.
- Motrar las cartas ordenadas por tipo: O(n log n), ya que se requiere una ordenación interna del HashMap.

Resultados del profiler:

- Agregar cartas:
    - Promedio: 0.2 segundos por carta
    - Máximo: 0.35 segundos por carta (carta con nombre largo)

Comentario: El tiempo de agregar una carta, se mantiene dentro del rango esperado para un HashMap. El pico en el tiempo máximo puede estar relacionado con el tamaño del nombre de la carta.

- Buscar cartas:
    - Promedio: 0.07 segundos por carta
    - Máximo: 0.12 segundos por carta (posible colisión en la función hash)

Comentario: La búsqueda de cartas en promedio es rápida. La cosa es que se observa un pico en el tiempo máximo que podría indicar una colisión en la función hash. Se deberia, revisar y optimizar la función hash si las colisiones son frecuentes.

- Mostrar todas las cartas:
    - Primera ejecución: 1.5 segundos
    - Segunda ejecución: 1.2 segundos (aprovechó la caché)

Universidad del Valle de Guatemala
Algoritmos y Estructura de Datos
Hoja de trabajo #6
Nadissa Vela 23764 | Roberto Barreda 23354

Comentario: La primera ejecución toma más tiempo porque el JProfiler aún no ha almacennado las cartas en la caché. En la segunda ejecución, la caché se utiliza para acelerar el acceso a las cartas, lo que reduce el tiempo de ejecución.

- Mostrar cartas ordenadas por tipo:
    - 3.2 segundos

Comentario: La ordenación por tipo requiere una iteración adicional sobre el HashMap y la aplicación de un algoritmo de ordenación; así aumenta el tiempo de ejecución en comparación con las otras operaciones.