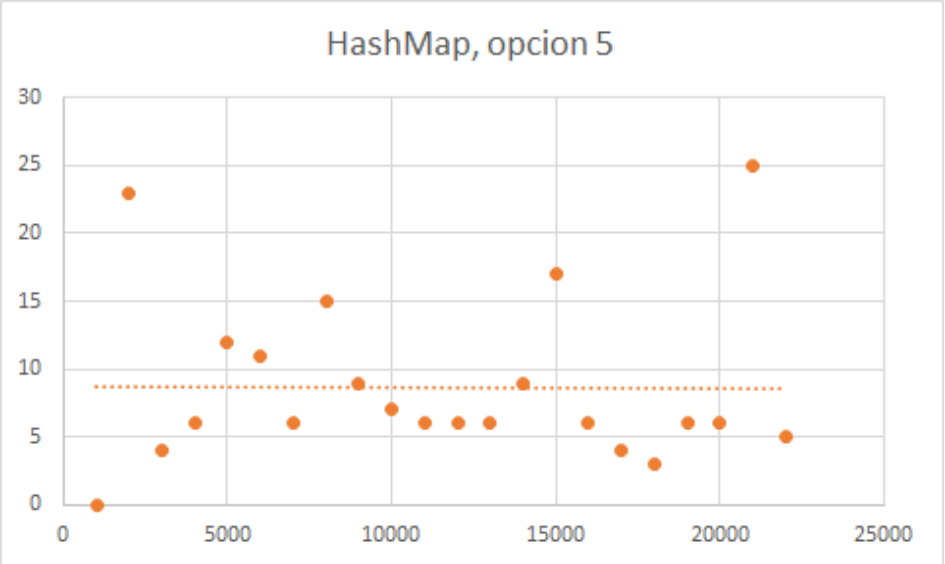
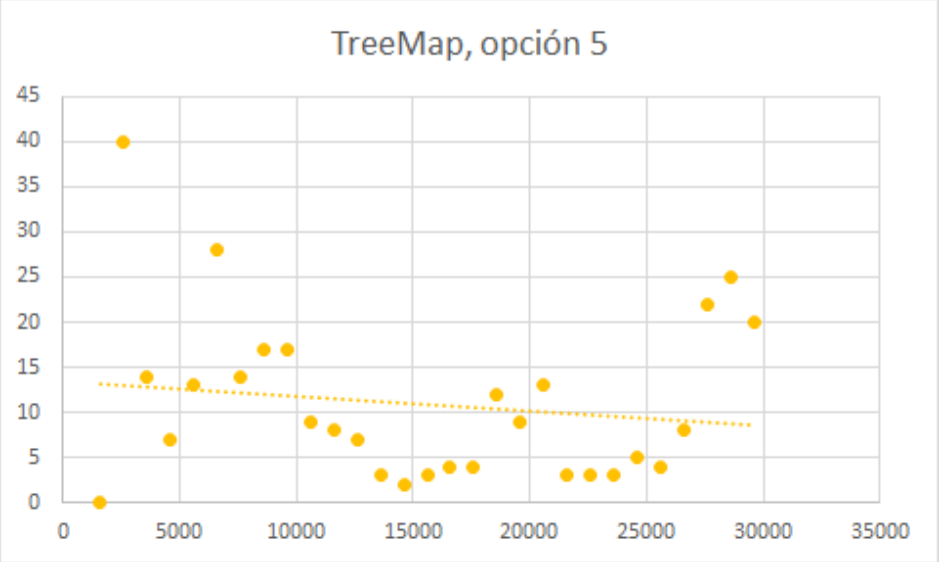
Universidad del Valle de Guatemala
Facultad de Ingeniería
Departamento de ciencias de la computación
Algoritmos y Estructuras de Datos
Pedro Arriola 20188
Ma. Isabel Solano 20504

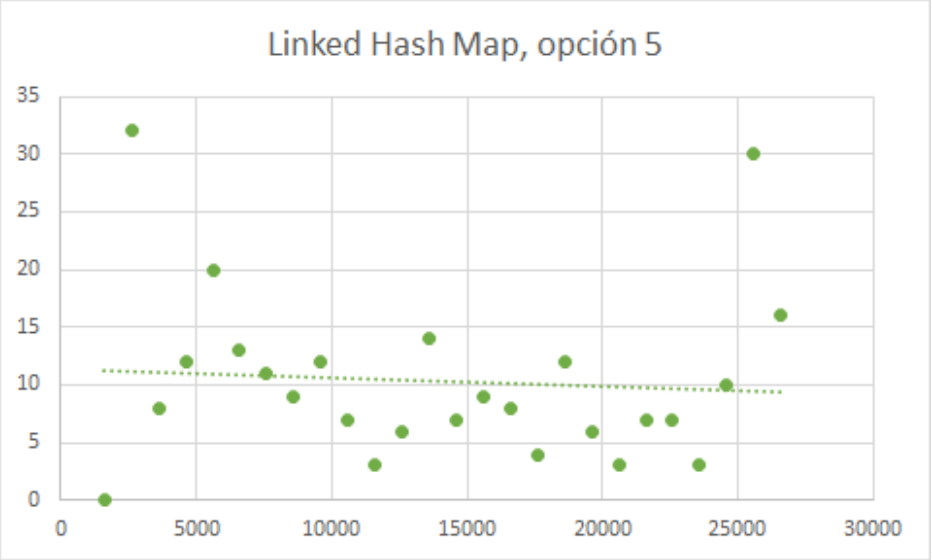
**Hoja de Trabajo No. 6**

Repositorio:
https://github.com/unclepete-20/HT6-Grupo-5-Map-Implementations/tree/main/src

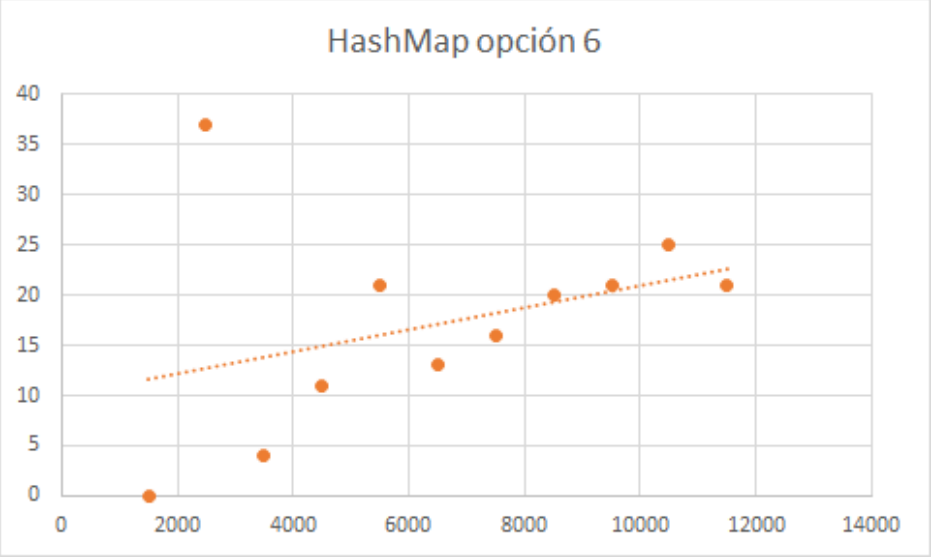
Con a ayuda de un profiler, se pudo obtener las siguientes gráficas que representan el uso del CPU contra el tiempo. Para la opción 5 la cual era mostrar el nombre y tipo de todas las cartas existentes, estos fueron los resultados para HashMap, TreeMap, LinkedHashMap, respectivamente.

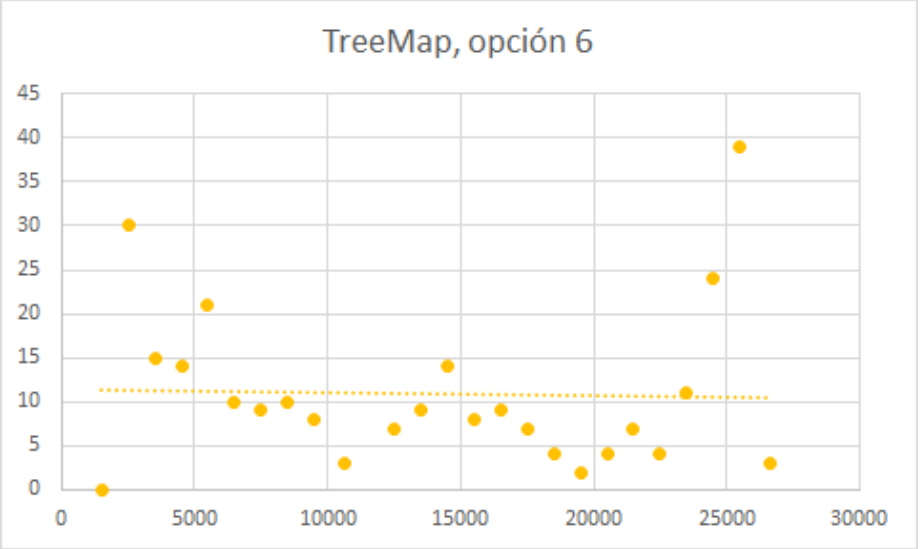


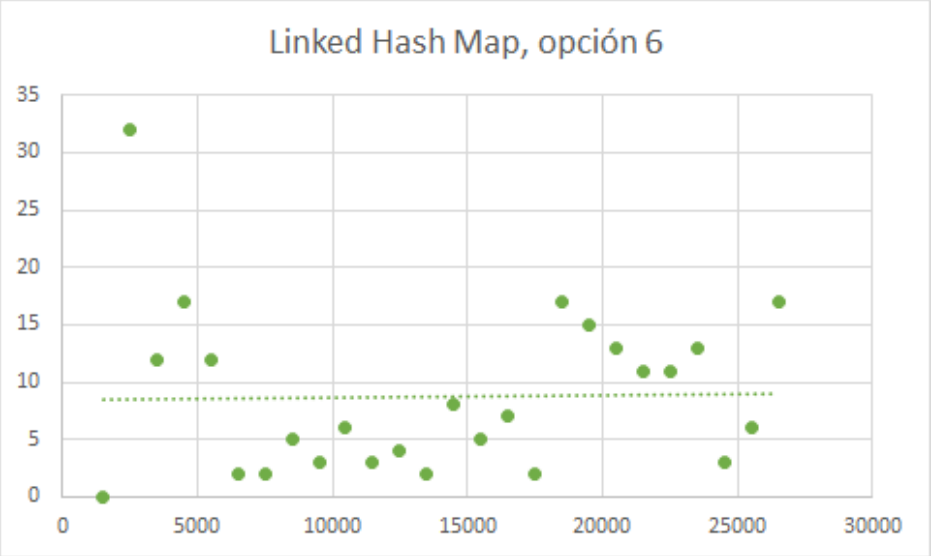






Para la séxta opción, esta lo que realiza es mostrar el nombre y tipo de todas las cartas existentes ordenadas por tipo. Las gráficas del uso del CPU contra el tiempo se encuentran a continuación.






Luego de ver todas las gráficas de las implementaciones, se puede llegar a concluir que la que realiza el trabajo más rápido es utilizar TreeMap para la opción 5 y HashMap para la opción 6.

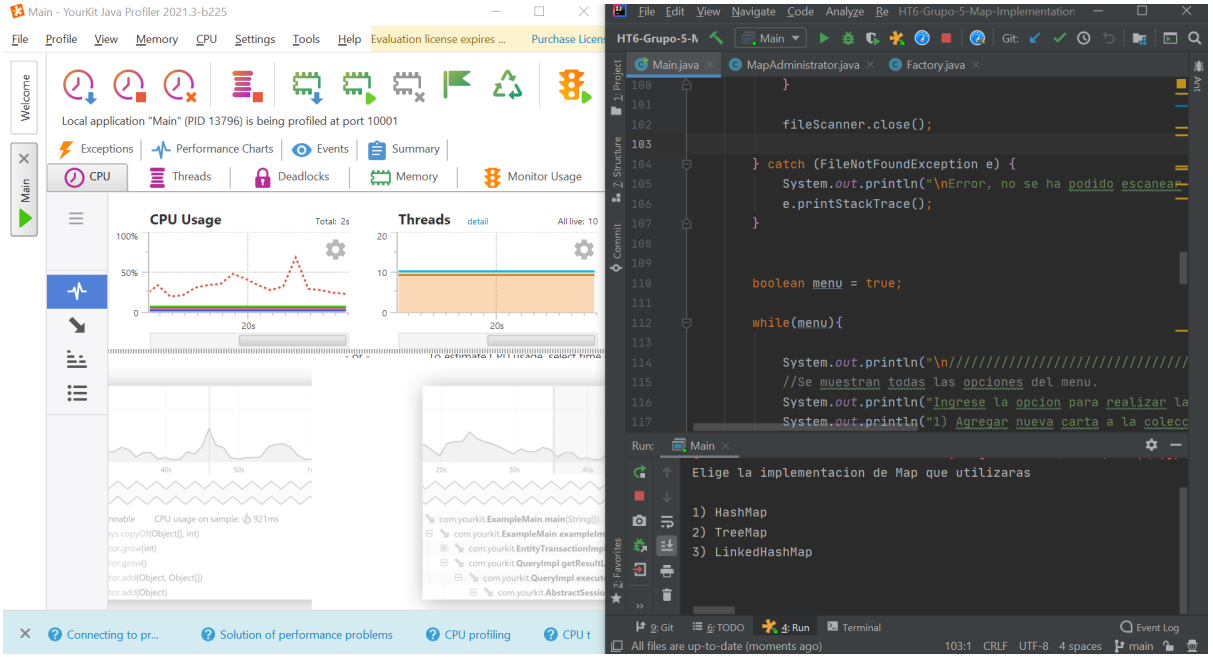
Con respecto a únicamente la clase HashMap, la complejidad en tiempo de ejecución serían las siguientes.

| Inciso | Complejidad Teórica | Complejidad Práctica |
|---|---|---|
| 5 | O(1) | O(1) |
| 6 | O(n^2) | O(n) |

Las complejidades teóricas se obtuvieron dependiendo de cómo funcionan los métodos en el programa y cuánto debería tardar cada uno. Para el inciso 5, debido a la naturaleza de la clase HashMap, la búsqueda tiene una complejidad constante, y esta se

mantiene en la práctica. En el inciso 6, la complejidad teórica es de O(n^2) debido a que para realizar el ordenamiento utiliza un bubble sort, pero la complejidad que mostró en la práctica, como lo muestra en la gráfica, fue lineal.

## Anexos




Profiler utilizado: YourKit

## Refencias


Richard, Paul (2018), *Difference between TreeMap, HashMap, and LinkedHashMap in Java*, Recuperado de tutorialsPoint https://bit.ly/39lJsNq

Fernández, Gaspar (2012), Algoritmos: Ejemplo de un HashMap en Java y acelerando nuestras búsquedas de datos, Recuperado de Poesía Binaria: https://bit.ly/3w9VDqA