DOCUMENTO DE ANÁLISIS RETO 2

Nicholas Barake (n.barake) **Código** 202020664
Jesed Domínguez (j.dominguezp) **Código** 202011992

****NOTAS:** Las pruebas se realizaron con los parámetros que había en los ejemplos del PDF guía.
Las pruebas para todos los requerimientos se realizaron con una carga de datos de tipo PROBING ya que vimos que era más rápido.

CARGA DE DATOS

	PROBING	**CHAINING**
10pct	2187.5 ms	2421.87 ms
30pct	6046.87 ms	5531.25 ms
80pct	11625.0 ms	11562.5 ms
Large	12640.62 ms	13625.0 ms

REQUERIMIENTO 1

Complejidad algorítmica: O(2n) Esto se debe a que hay dos For in, por lo tanto, el número de líneas que hay se multiplica por n veces que se itera el ciclo For in.

Pruebas de tiempo:

- 10% 2953.125 ms
- 30% 9796.875 ms
- 80% 18218.75 ms
- Large 19953.12 ms

Comparación Reto 1: Comparado con el anterior reto, la complejidad algorítmica que presenta es la misma debido a que se usó el mismo método para llevar a cabo el requerimiento.

REQUERIMIENTO 2

Complejidad algorítmica: O(4n) Esto se debe a que se encuentras cuatro For in, algunos dentro de otros, esto hace que la complejidad algorítmica incremente considerablemente debido a que el número de líneas dentro del For in debe recorrerse n veces según el tamaño de la iteración.

Pruebas de tiempo:

- 10% 5812.5 ms
- 30% 15296.87 ms
- 80% 35609.37 ms
- Large 43296.875 ms

Comparación Reto 1: Los tiempos de ejecución para este requerimiento fueron bastante similares al del Reto 1, por lo que la solución requería del ordenamiento de las obras y la única manera de hacer eso (hasta ahora) es con una lista y los algoritmos de ordenamiento que se conocen. Debido a que los mapas son non-ordered tables, pierde el sentido de guardar datos ordenados en un mapa porque igualmente el mapa los va a desordenar.

REQUERIMIENTO 3 (Realizado por Nicholas Barake)

Complejidad algorítmica: O(n), n siendo el tamaño de la lista de las obras de un artista. Se nota que a pesar de tener tiempo lineal, el algoritmo se comporta muy rápido tanto en comparación con los otros requerimientos, como en cuanto a los archivos más grandes.

Pruebas de tiempo:

- 10% 0.0 ms
- 30% 15.625 ms
- 80% 46.875 ms
- Large 46.875 ms

Comparación Reto 1: La complejidad en el Reto 1 fue O(4n) y los tiempos de prueba llegaron hasta los 200 ms. Esta implementación del código fue mucho más eficiente debido a que los datos se sacaron de un mapa con tiempo constante y no de una lista iterando.

REQUERIMIENTO 4 (Realizado por Jesed Domínguez)

Complejidad algorítmica: O(1) Se demora un tiempo constante debido a que este requerimiento lo desarrollamos mediante buscar datos en mapas. Por lo tanto, gracias a la facilidad de los mapas para hallar información en tiempo constante, así mismo quedó la complejidad algorítmica del requerimiento.

Pruebas de tiempo:

- 10% 15.625 ms
- 30% 15.625 ms
- 80% 15.625 ms
- Large 15.625 ms

Comparación Reto 1: En comparación al reto anterior, la complejidad algorítmica mejoró bastante debido a que ahora no se está utilizando For in dentro de otro For in para comparar los ConstituentID del catálogo de artistas y obras de arte. En este caso, se están usando For para hallar el valor de respectivas llaves en los mapas (mp.get()).

REQUERIMIENTO 5

Complejidad algorítmica: O(2n) Esto se debe a que requiere de dos For in para acceder correctamente a la información. Siendo n las veces en que itera el For para alcanzar a tomar la información de todos los datos en el mapa y en las listas.

Pruebas de tiempo:

- 10% 703.125 ms
- 30% 2506.25 ms
- 80% 7078.12 ms
- Large 9031.25 ms

Comparación Reto 1: Los tiempos de ejecución se suponen rápidos en comparación con los otros requerimientos del Reto.