Análisis del reto 2

Alumno: Pablo Pedreros Díaz – 202112491 - p.pedreros@uniandes.edu.co

Carga de datos.

Ya en el laboratorio anterior definimos varios de los índices que vamos a usar en este reto y las formas que usará cada uno para resolver las colisiones. En general tenemos que es más rápido hacer linear probing que separate chaining, pero también ocupará más espacio, por lo que para encontrar un equilibrio entre velocidad y espacio empleado, usaremos linear probing en los mapas con pocas llaves pero que tengamos que recorrer varias veces, por lo que ganaremos velocidad y no perderemos tanto espacio haciendo cupo para las llaves, mientras que usaremos separate chaining para mapas con muchas llaves en los que tengamos que hacer búsquedas específicas.

Al agregar varios índices a la carga de datos, el tiempo de la carga de datos aumentó considerablemente respecto al reto 1, pero todos los requerimientos fueron mucho más eficientes y tuvieron tiempos bastante más bajos que sin usar mapas.

La tabla con los tiempos de la carga de datos es la siguiente:

Porcentaje de datos (%)	Tiempo (ms)
5	734.375
10	1453.125
20	2703.125
30	3578.125
50	5781.25
80	8781.25
100	10625.0

Requerimiento 1. ($N = Número\ de\ artistas$)

Para el requerimiento uno lo más eficiente va a ser seguir usando la misma implementación con listas que en el reto 1. La implementación más eficiente será tomar la lista con todos los artistas que nos crea el catálogo, y ordenarla por medio de merge sort, para luego buscar, con una búsqueda binaria, los índices máximo y mínimo de los artistas

dentro del rango de años que necesitamos y sacar con ellos una sublista de artistas para el rango. Como este es el único requerimiento para el que tendremos que organizar todos los artistas, podemos organizarlos desde la carga de datos para no perder tiempo al momento de hacer el requerimiento, sino que solo haya que buscar los dos índices por búsqueda binaria y sacar la sublista para imprimir. Así el orden de complejidad de este requerimiento en notación Big O sería de $(O)logN$ de las búsquedas binarias sin contar la complejidad NlogN del ordenamiento que hacemos en la carga de datos. En este caso, lo único que reduce la complejidad del requerimiento respecto al $O(NlogN)$ del reto 1, es el hecho de hacer el ordenamiento desde la carga de datos para que el requerimiento sea más rápido.

*También se podría implementar el requerimiento construyendo en la carga de datos un mapa que tenga como índice cada años y como valores una lista con los artistas nacidos en ese año, más este sería de complejidad más alta que la implementación de ordenar una lista, pues además de construir el índice en la carga de datos, habría que mirar llave por llave si están dentro del rango de años para ver si agregar sus artistas a la sublista con artistas del rango, por lo que estaríamos teniendo una complejidad de O(M), siendo M el número de años en los que nació algún autor.

Quedándonos con la solución más eficiente, el promedio de tiempo luego de 3 intentos para el requerimiento por cada tamaño de archivo está dado por la siguiente tabla, usando como ejemplo un rango entre los años 1920 y 1985:

Porcentaje de datos (%)	Número de artistas	Tiempo (ms)
5	4996	0
10	6656	0
20	8724	0
30	10063	0
50	12137	0
80	14143	0
100	15223	0

Requerimiento 2. ($N = Número\ de\ obras$)

El mismo análisis del requerimiento 1 nos sirve para el requerimiento 2. Lo más eficiente será usar la lista de obras que nos entrega el catálogo, y como es el único requerimiento en el que tenemos que organizar todas las obras, podemos hacerlo desde la carga de datos para luego con dos búsquedas binarias sacar nuestra sublista igual que en el requerimiento 1, por lo que también tendremos una complejidad de $O(logN)$ luego de ordenar por merge sort en la carga de datos (complejidad $O(NlogN)$). Nuevamente, lo único que reduce la complejidad del requerimiento respecto al $O(NlogN)$ del reto 1, es el hecho de hacer el ordenamiento desde la carga de datos para que el requerimiento en sí sea más rápido.

*Igual que en el requerimiento 1, podríamos hacer la implementación por medio de un mapa con todas las fechas de adquisición como índices, pero nuevamente esto significaría una complejidad de O(M) donde M es el número de fechas de adquisición; mucho menos eficiente que las búsquedas binarias de la primera implementación.

El promedio de tiempo con la solución más eficiente luego de 3 intentos para el requerimiento está dado por la siguiente tabla, usando como ejemplo un rango entre las fechas de 6 de junio de 1944 y 9 de noviembre de 1989:

Porcentaje de datos (%)	Número de obras	Tiempo (ms)
5	7572	156.25
10	15008	328.125
20	29489	718.75
30	43704	1062.5
50	71432	1859.375
80	111781	3078.125
100	138150	3828.125

Requerimiento 3. ($N = Número\ de\ obras;\ M = Número\ de\ obras\ del\ artista;\ Q = Número\ de\ medios\ utilizados\ por\ el\ artista;\ R = Número\ de\ artistas$)

Para este requerimiento crearemos un par de índices en la carga de datos del catálogo que nos agilizarán la manipulación de las obras. Lo primero será crear un índice por los nombres de los autores que tenga como valores sus diccionarios, de modo que apenas recibamos el nombre del autor a buscar, tengamos automáticamente su ID. Lo segundo será un índice por ID de los autores, que tenga como valor una lista con todas las obras de ese autor, de modo que apenas tengamos el ID del autor gracias al índice anterior, podamos extraer la lista de sus obras que serán con las que trabajaremos. Cuando ya tengamos una lista con las obras del autor que nos interesa, lo que haremos será crear un mapa solo con esas obras que tenga como llaves las diferentes técnicas utilizadas y como valores las listas con las obras correspondientes a esa técnica. Ya con este mapa podemos encontrar con el size del mapa cuántas obras se utilizaron y con los diferentes size de las listas en las llaves, cuál fue la técnica más utilizada. Así, la complejidad del requerimiento luego de la carga de datos será la de crear el mapa con las obras del autor ($O(M)$, siendo M el número de obras del artista) sumada a la de buscar dentro de las técnicas cuál es la más usada ($O(Q)$, siendo Q el número de técnicas o medios utilizados por el artista que no deberían ser más de unas 20), de forma que el requerimiento será de orden $O(M + Q)$.

La principal ventaja respecto a la implementación del reto 1 es que implementamos en la carga de datos los dos índices (uno de implementación de orden $O(N)$ y otro $O(R)$, siendo N el número de obras y R el número de artistas) por ID y por nombre del artista. Esto nos ahorrará la complejidad N adicional que teníamos que tener en el reto 1 para organizar todas las obras en una estructura a la hora del requerimiento, en cambio nos permitirá conseguir instantáneamente la lista de obras de un artista luego de recibir su nombre por parámetro.

El promedio de tiempo luego de 3 intentos para el requerimiento por cada tamaño de archivo está dado por la siguiente tabla, usando como ejemplo la artista Louise Bourgeois:

Porcentaje de datos (%)	Número de artistas	Número de obras	Número de obras del artista	Tiempo (ms)
5	4996	7572	159	0
10	6656	15008	317	0
20	8724	29489	650	15.625
30	10063	43704	982	15.625

50	12137	71432	1633	31.25
80	14143	111781	2654	31.25
100	15223	138150	3337	31.25

Requerimiento 5. ($N = Número\ de\ obras$; $M = Número\ de\ obras\ del\ departamento$)

El requerimiento 5 funcionará de forma muy parecida al reto 1. Lo que haremos será construir un índice en la carga de datos del catálogo que nos organice las obras por departamento, teniendo como valores las listas con las obras de cada departamento. Desde ese punto, el resto será igual que en la implementación del reto 1. Cuando generemos la lista de M elementos del departamento, a cada uno tendremos que asignarle su valor de transporte, que serían M iteraciones. Ya con nuestra lista con costos, para imprimir las obras más costosas y luego las obras más antiguas, tendremos que organizar por merge sort dos veces con diferentes parámetros nuestra lista de obras del departamento, lo que sería de complejidad 2MlogM (solo MlogM en notación Big O). Ya con la lista ordenada, imprimiremos los valores que pide el requerimiento, por lo que el orden de complejidad sería de M de encontrar los costos de cada obra más 2MlogM de ordenar la lista por costos y por antigüedad. En notación big O, el requerimiento será de complejidad $O(MlogM)$.

Parecido al requerimiento 3, la ventaja que nos dará esta implementación con mapas sobre la del reto 1 será la posibilidad de crear desde la carga de datos un índice (su implementación será de orden de complejidad O(N)) con las obras por departamento. Esto nos facilitará la lista con obras del departamento que buscamos para seguir con la implementación, ahorrándonos las N iteraciones en el requerimiento que se necesitaban para encontrar la lista de obras del departamento que buscamos.

El promedio de tiempo luego de 3 intentos para el requerimiento por cada tamaño de archivo está dado por la siguiente tabla, usando como ejemplo el departamento “Drawings & Prints:

Porcentaje de datos (%)	Número de obras	Número de obras del departamento	Tiempo (ms)
5	7572	4109	1468.75
10	15008	8133	2281.25
20	29489	15983	3546.875

30	43704	23709	5015.625
50	71432	38888	7062.5
80	111781	61397	10671.875
100	138150	76117	12937.5