Observaciones Laboratorio 6

Integrantes del grupo

Grupo 11: **Federico Melo Barrero.**
202021525.
f.melo@uniandes.edu.co

Juan Camilo Prieto Avella.
201814815.
jc.prietoa@uniandes.edu.co

Pruebas de tiempo y memoria de carga de los datos

Se toma el tiempo de crear los dos índices implementados, el que clasifica las obras de acuerdo al medio, **catalog['mediums']**, y el que las clasifica con respecto a las nacionalidades, **catalog['nationalities']**. En el archivo **view.py**, ambas cosas ocurren cuando se ejecuta la función **printloadData**, así que en esa es que se toman los tiempos.

Máquina utilizada:

	Máquina
Procesadores	1,6 GHz Intel Core i5 de dos núcleos,
Memoria RAM (GB)	8.00 GB.
Sistema Operativo	MacOS Big Sur 64-bits

Pruebas de cargar datos

Porcentaje de la muestra [pct]	**'CHAINING' α = 4.0**		**'PROBING' α = 0.5**	
	Tiempo (msec)	**RAM (GB)**	**Tiempo (msec)**	**RAM (GB)**
small	1691.73	De 1.99 a 2.01	1886.86	De 2.00 a 2.02
20%	205695.25	De 2.03 a 2.04	199827.376	De 2.00 a 2.05

Se prueba cambiando los factores de carga de separate-chaining:

Porcentaje de la muestra [pct]	**'CHAINING' α = 2.0**		**'CHAINING' α = 8.0**	
	Tiempo (msec)	**RAM (GB)**	**Tiempo (msec)**	**RAM (GB)**

small	1878.20	De 2.08 a 2.1	1710.205	De 2.04 a 2.05
20%	193397.375	De 2.01 a 2.04	205197.468	De 2.02 a 2.13

Se prueba cambiando los factores de carga de linear probing:

Porcentaje de la muestra [pct]	**'PROBING' α = 0.2**		**'PROBING' α = 0.8**	
	Tiempo (msec)	**RAM (GB)**	**Tiempo (msec)**	**RAM (GB)**
small	1950.59	De 2.04 a 2.05	2299.855	De 2.07 a 2.09
20%	192245.186	De 2.11 a 2.34	196458.576	De 2.27 a 2.32

a) Teniendo en cuenta cada uno de los requerimientos ¿Cuántos índices implementaría en el Reto? y ¿Por qué?

Para el requerimiento uno, que consiste en listar cronológicamente los artistas entonces realizaremos una tabla de hash que tenga como llave el año y valor un array list con los artistas de ese año. Por otro lado, usaríamos esta implementación porque llama en O(1) un año en especifico entonces se demoraría la cantidad de años que estén en el rango y esto seria mas rápido que una búsqueda binaria para el mejor caso y caso promedio pero en el peor caso se demoraría lo mismo que esta otra (O(n)).

Para el requerimiento dos, que consiste en listar cronológicamente las adquisiciones realizaremos un proceso similar al mencionado anteriormente donde usaremos una tabla de hash que tenga como llaves las fechas de las adquisiciones y como valor un array list de todas las adquisiciones que estén en esa fecha para que al llamar una fecha en especifico tenga un costo de tiempo de O(1) por lo que se demoraría la cantidad de años que estén en el rango y es mas rápido que una búsqueda binaria por lo mencionado anteriormente.

Para el requerimiento tres, que consiste en clasificar las obras de un artista por técnica haremos una tabla de hash que tenga como llave el nombre del artista y como valor otra tabla de hash que tenga como llave las técnicas usadas por ese artista y como valor un array list de las obras de ese artista para esa técnica en especifico, lo cual tendrá un costo en tiempo de O(2) para llamar a un artista en especifico con una tabla en especifico que seria mucho mas rápido que tenerlo todo con apuntadores en un array list ordenado.

Para el requerimiento cuatro, que consiste en clasificar las obras por la nacionalidad de sus creadores usaremos una tabla de hash con llaves la nacionalidad y como valor un array list con las obras hechas por artistas de esa nacionalidad, lo que nos dará un costo en tiempo de O(1) para llamar las obras especificas de una nacionalidad que será mucho mas rápido que usar los array list ordenadas con apuntadores para saber que obras pertenecen a que nacionalidad.

Una tabla de símbolos cuyas llaves son los países y cuyos valores son las obras cuyos autores son de dicha nacionalidad.

Para el requerimiento cinco, que consiste en transportar las obras de un departamento especifico y para realizar esto haremos una tabla de hash con llaves los departamentos y como valor un array list con las obras de ese departamento por lo que llamar un departamento especifico para transportar sus obras tendrá un costo de O(1), en cambio si usáramos un array list tendríamos que buscar ubicación por ubicación que obras pertenecen a un departamento en especifico que costara siempre O(n).

Para el requerimiento seis o bono, que consiste en encontrar los artistas más prolíficos del museo usaremos una tabla de hash con llaves el nombre del artista y como valor un array, de forma que el tamaño del array es el número de obras de cada artista, teniendo una complejidad temporal de O(1). A partir de eso se realiza una comparación del número de obras de cada artista y se hallan los más prolíficos.

b) Según los índices propuestos ¿en qué caso usaría Linear Probing **o** Separate Chaining **en estos índices? y ¿Por qué?**
Según los índices propuestos usaremos linear probing en todos los índices porque puede llegar a ser mas costoso en memoria que separate chaining pero en este ejercicio en particular la memoria no es problema y al ocupar mas memoria va a tener un costo menor en tiempo que es lo que queremos optimizar en el reto.

c) Dado el número de elementos de los archivos MoMA, ¿Cuál sería el factor de carga para estos índices según su mecanismo de colisión?

Dado el numero de elemento de los archivos MoMa nuestro factor de carga para todos los índices será de 0.2 para que halla una menor probabilidad de colisión y nuestros algoritmos sean mas rápidos aunque se aumenta en costo en espacio pero es muy poco espacio para toda la memoria RAM que tenemos.

d) ¿Qué diferencias en el tiempo de ejecución notan al ejecutar la cargar los datos al cambiar la configuración de Linear Probing **a** Separate Chaining**?**

Al realizar las pruebas de ejecución, en las comparaciones se ve favorecido **Linear Probing**. Comparando las estructuras de datos de frente, se puede ver que

Linear Probing con α = 0.2 (192245.186) < Separate Chaining con α = 2.0 (193397.375)

Linear Probing con α = 0.5 (199827.376) < Separate Chaining con α = 4.0 (205695.25)

Linear Probing con α = 0.8 (196458.576) < Separate Chaining con α = 8.0 (205197.468)

Nótese sobre lo anterior que la comparación entre los dos métodos de manejo de colisiones se realiza con factores de carga disímiles. Se puede ver que para separate chaining se utilizan

factores de carga de 2.0, 4.0 y 8.0 mientras que para linear probing se emplean factores de carga de 0.2, 0.5 y 0.8. Esto por supuesto se debe a la naturaleza de cada una de las estructuras de datos y la forma en la que manejan las colisiones, sería insensato utilizar linear probing con factores de carga altos.

e) ¿Qué configuración de ADT Map escogería para el índice de técnicas o medios**?, especifique el mecanismo de colisión, el factor de carga y el número inicial de elementos.**

Se emplea un índice que usa el mecanismo de colisión linear probing, puesto que es la estructura que mas rápido encuentra un hit o miss. Se hace uso de un factor de carga de $\alpha = 0.2$.

El número inicial de elementos se elige como 21251, pues ese es el número de técnicas o medios **'Medium'** únicos que hay en el archivo **large**, en otras palabras, es el número máximo de parejas llave valor que se espera almacenar en el índice de técnicas o medios. Este número no es primo, pero el proceso de buscar un número primo para el tamaño de la tabla de hash lo hacen las funciones de la librería DISClib.

f) ¿Qué configuración de ADT Map escogería para el índice de nacionalidades**?, especifique el mecanismo de colisión, el factor de carga y el número inicial de elementos.**

Se emplea un índice que usa el mecanismo de colisión linear probing, puesto que es la estructura que mas rápido encuentra un hit o miss. Se hace uso de un factor de carga de $\alpha = 0.2$.

Se selecciona 119 como el número inicial de elementos porque ese es el número de nacionalidades **'Nationality'** únicas al interior del archivo **large**. Es decir, es el número máximo de parejas llave valor que se espera almacenar en el índice de nacionalidades. Al igual que en el punto anterior, este número no es primo, pero el proceso de buscar un número primo para el tamaño de la tabla de hash lo hacen las funciones de la librería DISClib.