Reto No. 17 Curando y Explorando el MoMA (Modern Museum of Art)

INTEGRANTES

Estudiante 1 Mateo López Cespedes

Cod 202014481

Correo m.lopez23@uniandes.edu.co

Estudiante 2 Valentina Perea

Cod 202013095

Correo v.peream@uniandes.edu.co

Link repositorio:

EDA2021-2-SEC03-G17/Reto2---G17: Template Reto EDA (github.com)

REQUERIMIENTOS INDIVIDUALES:

-*REQ3:* Valentina Perea Márquez

-*REQ4:* Mateo López Cespedes

AMBIENTES DE PRUEBA

MAQUINA	**1: Mateo Lopez**	**2: Valentina Perea**
Procesadores	Intel(R) Core(TM) i5-9300H CPU @ 2.40GHz 2.40 GHz	AMD Ryzen 5 3400G with Radeon Vega Graphics 3.70
Memoria RAM (GB)	8,00 GB	16,0 GB
Sistema Operativo	Windows 10 Home	Windows 10 Pro 2004 64 bits

Tabla 1. Especificaciones de las máquinas para ejecutar las pruebas de rendimiento.

COMPLEJIDAD DE REQUERIMIENTOS:

REQUERIMIENTO	PROMEDIO	MEJOR CASO	PEOR CASO	ESPECIFICACION
1	O(n/k+6)	O(n) Si no hay ningún elemento entre las fechas.	O(2n+6)	La función hace uso de una iteración que comprende el rango de fechas. Y otras dos para obtener los primeros 3 elementos y los ultimo dos.
2	O(2n)	O(n) No hay fechas que estén dentro del rango	O(4n)	Se hacen 3 iteraciones sobre listas cada una con menor tamaño que la anterior. Hay 1 mp.get() y 1 mp.contains()
3	O(n/k+n)	O(1) El artista no se encuentra en el catálogo.	O(2n)	La función utiliza 2 contains, 2 get y 2getValue, además de una iteración de todos los elementos y una de todas sus llaves.
4	O(nlog(n))	O(1) La lista ya se encuentra ordenada	O(nlog(n)^ 2)	Se hace un recorrido sobre la totalidad de las llaves del mapa para luego compararlas. Se usa el shell sort.
5	O(3*nlogn*)	O(1) El departamento no se encuentra en el catálogo.	O(*3nlog²n)*	La funcion obtiene la pareja y valor en O(1), itera una vez pero usa el ordenamiento shell 2 veces.
6	:(	:(	:(	:(

Tabla 2. Complejidades de los requerimientos en notación BigO.

? Cambia algo en la complejidad y tiempo de ejecución de los requerimientos entre los TAD Lista y los Hash Maps? Justifique la respuesta.

REQUERIMIENTO	PROMEDIO	MEJOR CASO	PEOR CASO	ESPECIFICACION
1	O(n+n ^1.25)	O(n + n log(n))	O(n+n^3/2)	Se usa un iterador o(N), y una función Sort shell.
2	O(n+n ^1.25)	O(n + n log(n))	O(n+n^3/2)	La función usa un iterador o(n) y un Shell sort.
3	O(2n)	O(2n)	O(2n)	Se ITERA la lista 2 veces O(2N), no se organiza nada.
4				
5	O(n+2(n ^1.25))	O(n+2(n log(n)))	O(n+2(n ^3/2))	Una iteración a la lista (o(N)) y una adición a una lista con Addlast. Además hay 2

REQUERIMIENTO	PROMEDIO	MEJOR CASO	PEOR CASO	ESPECIFICACION
1	O(n/k+6)	O(n) Si no hay ningún elemento entre las fechas.	O(2n+6)	La función hace uso de una iteración que comprende el rango de fechas. Y otras dos para obtener los primeros 3 elementos y los ultimo dos.
2	O(2n)	O(n) No hay fechas que estén dentro del rango	O(4n)	Se hacen 3 iteraciones sobre listas cada una con menor tamaño que la anterior. Hay 1 mp.get() y 1 mp.contains()
3	O(n/k+n)	O(1) El artista no se encuentra en el catálogo.	O(2n)	La función utiliza 2 contains, 2 get y 2getValue, además de una iteración de todos los elementos y una de todas sus llaves.
4	O(nlog(n))	O(1) La lista ya se encuentra ordenada	O(nlog(n)^ 2)	Se hace un recorrido sobre la totalidad de las llaves del mapa para luego compararlas. Se usa el shell sort.
5	O(3nlogn)	O(1) El departamento no se encuentra en el catálogo.	O(3nlog²n)	La funcion obtiene la pareja y valor en O(1), itera una vez pero usa el ordenamiento shell 2 veces.

Si definitivamente la complejidad de los mismos requerimientos cambia. Se reducen las complejidades en los Tad Maps.

Comparando las estructuras de datos de Tad lista y Hash Maps podemos empezar a notar grandes diferencias en el Mejor Caso donde el Hash Maps le toma ventaja a la lista con complejidades de O(1) y O(n) que son permitidas por algoritmos de Get y Contains, en

vez de iteraciones a toda la lista, también el método de búsqueda en el caso promedio permite reducir el número de elementos n a un n/k. Incluso las llaves permiten evitar el uso de funciones de ordenamiento puesto que con un iterador pueden llamar en orden las llaves.

Teniendo en cuenta estas complejidades, pensamos que los Hash Maps son mucho más eficientes al momento de catalogar información y acceder a ella, además de que ahorra tiempo y recursos en caso de búsqueda de elementos.

PRUEBAS DE TIEMPO DE CARGA DE DATOS:

REQUERIMIENTO	ARCHIVO Y ELEMENTOS	TIPO DE MAPA Y LOAD	TIEMPO MAQUINA 1	TIEMPO MAQUINA 2
Carga de datos	small	2 PROBING Y 5 CHAINING EN LOAD FACTOR 0.4 Y 4.0	156.25 milisegundos	109.375 milisegundos
Carga de datos	5pct	2 PROBING Y 5 CHAINING EN LOAD FACTOR 0.4 Y 4.0	703.125 milisegundos	453.125 milisegundos
Carga de datos	10pct	2 PROBING Y 5 CHAINING EN LOAD FACTOR 0.4 Y 4.0	1328.125 milisegundos	875.0 milisegundos
Carga de datos	20pct	2 PROBING Y 5 CHAINING EN LOAD FACTOR 0.4 Y 4.0	2656.25 milisegundos	2062.5 milisegundos
Carga de datos	30pct	2 PROBING Y 5 CHAINING EN LOAD FACTOR 0.4 Y 4.0	557359.375 milisegundos	569734.375 milisegundos
Carga de datos	50pct	2 PROBING Y 5 CHAINING EN LOAD FACTOR 0.4 Y 4.0	Mas de 15 min	Mas de 15 min
Carga de datos	80pct	2 PROBING Y 5 CHAINING EN LOAD FACTOR 0.4 Y 4.0	Mas de 15 min. 3102834.821 milisegundos	Mas de 15 min. 2171984.375 milisegundos

Tabla 3. Tabla de tiempo carga de datos en maquina 1 y 2.

? Cambia algo en la carga de datos y tiempo entre los TAD Lista y los Hash Maps? Justifique la respuesta.

Si cambia mucho la manera de cargar los datos y el tiempo. Comparativamente hablando los Tad Listas eran estructuras muy sencillas que tenían una carga lineal y se iban agregando a la misma lista, por este motivo la carga de los datos no llevaba mucho tiempo, sin embargo, al momento de implementar los Mapas estos permiten crear varios índices con la misma información, pero con diferente forma de acceder a ellos , por lo que se ocupa una mayor cantidad de memoria,

PRUEBAS DE TIEMPO DE EJECUCION REQUERIMIETOS:

REQUERIMIENTO	ARCHIVO	TIEMPO (MILISEGUNDOS) MAQUINA 1	TIEMPO (MILISEGUNDOS) MAQUINA 2
1	small	0.0 milisegundos	0.0 milisegundos
2	small	265.625 milisegundos	218.75 milisegundos
3	small	0.0 milisegundos	0.0 milisegundos
4	small	31.25 milisegundos	15.625 milisegundos
5	small	15.625 milisegundos	31.25 milisegundos
6	small	x	x

Tabla 3. Tabla de tiempo de ejecución de requerimientos archivo small en maquina 1 y 2

REQUERIMIENTO	ARCHIVO	TIEMPO (MILISEGUNDOS) MAQUINA 1	TIEMPO (MILISEGUNDOS) MAQUINA 2
1	30 pct	0.0 milisegundos	0.0 milisegundos
2	30 pct	5497.25	4843.75 milisegundos
3	30 pct	24.35	15.625 milisegundos
4	30 pct	1465.145	993.69 milisegundos
5	30 pct	6298.5	6187.5 milisegundos
6	30 pct	x	x

Tabla 4. Tabla de tiempo de ejecución de requerimientos archivo 30pct en maquina 1 y 2

Analisis:

Fue evidente que el uso de tablas de hash permitio una mejora en el tiempo de carga y ejecucion de los requerimientos a comparación del uso de TAD listas del Reto 1, pues al hacer uso de llaves era posible acceder a una mayor cantidad de datos que compartian una misma caracteristica, lo que hacia las labores de comparacion y recorrido mas cortas y amigables con los recursos de la maquina. Por el otro lado, las listas no ofrecian esta ventaja y en estas es casi obligatorio hacer un recorrido entero cada vez que se desee recolectar un dato de estas