ESTRUCTURAS DE DATOS Y ALGORITMOS

Reto 3

Santiago Usme - 201822047

Jesús Ospino – 201915195

Reflexión:

Para ordenar los datos se utilizó la fecha de cada accidente convirtiendo su formato de fecha a un número de segundos. Este número representa la diferencia de la fecha del accidente con respecto al 1ero de enero de 1970. Decidimos hacer el ordenamiento de los datos de esta forma porque no es una buena estrategia usar strings como los del formato original del archivo como llaves. Además, de esta forma es más fácil de entender lógicamente ya que los números tienen un sentido creciente mucho más fácil de organizar.

La altura del árbol resultó ser de 830. Teniendo en cuenta que el número de accidentes del archivo pequeño es de 999 accidentes, una altura de 830 para una estructura de árbol es irrazonable (debería ser cercana a . Esto sucede porque la estructura de árbol usada (BST) no se balancea como la del RBT. Este es uno de los cambios que se van a realizar más adelante.

Las operaciones incluidas en esta parte del reto son de buscar un accidente dentro de los datos y de consultar cuántos accidentes ocurrieron antes que una fecha ingresada. Ya que el árbol esta totalmente desbalanceado, en este caso la complejidad del código es cercana a O(n).

Otro punto para tener en cuenta es que cuando se cargan los datos al mapa y se devuelve el *size*, hay menos datos que en la lista con todos los datos. Esto significa que se pierden datos por alguna razón. Creemos que hay datos de accidentes en una misma fecha con misma exactitud hasta de segundos y se sobrescriben estos datos en el mapa.

**ACTUALIZACIÓN**

Se logró arreglar los errores de la versión anterior. Esto incluye el balance del árbol usando un árbol RBT y la pérdida de datos debido a la sobreescritura de datos por repetición de fechas. Para esto último cambiamos un poco las llaves (en este caso, las fechas) de los datos repetidos (en términos de segundos, por lo que no cambia mucho) para que no se sobrescribieran datos repetidos en el árbol.

Con respecto a la anterior versión, ahora la complejidad temporal de la carga de datos es mayor debido a que ahora el computador debe organizar los datos por que es un árbol RBT. Sin embargo, debido a que ahora el árbol está balanceado ahora, no se demora tanto buscando datos (se demora mucho menos). Esto se nota sin hacer ningún cálculo o prueba ya que antes el árbol tenía una altura de 830 para 999 datos, por lo que estaba extremadamente desbalanceado. Ahora el árbol tiene una altura de 10 lo que representa el valor más cercano a .

**ACTUALIZACIÓN # 2**

La complejidad del requerimiento 2 es menor que del requerimiento 3 dado que en el primero se hace un recorrido simple; para este caso en necesario hacer un recorrido doble.

No se puede buscar un rango de llaves en un Hashmaps, dado que no hay noción de orden.

1. ¿Cómo se utilizaron las estructuras de datos vistas en clase?

Se usaron un árbol RBT para en que cada key tenía como value tablas de Hash con las Severidad, Estado y Ciudad, asimismo, una lista con los Id’s de todos los accidentes para esa fecha.

2. ¿Cuáles campos seleccionaron para definir la llave de los árboles, y como generaliza esta solución a otros tipos de problemas similares?

La llave del árbol creado se definió como la primera parte “YY-MM-DD” de las fechas en el archivo de datos.

3. ¿Qué ventajas ofrece el uso de árboles sobre las estructuras de datos vistas anteriormente?

La noción de orden que no teníamos en los HashMaps aparece en los árboles. Además, en comparación con las listas, los árboles tienen una complejidad temporal menor que estas, por lo cual permite respuestas al usuario mucho más rápidas cuando el tamaño de datos tiende a ser más grande.

4. ¿En qué escenarios recomendaría utilizar árboles BST o árboles Red-Black?

Cuando sabemos que los datos vienen balanceados es mejor usar BST que los RBT, ya que el tiempo de construcción será menor que el RBT y los tiempos de respuesta serán muy similares.

5. ¿A qué conclusiones llegan a partir de analizar los resultados de complejidad temporal entre árboles binarios ordenados (BST) y Red-black ?

Los RBT tienen una complejidad mayor de estructuración, pero menor de respuesta, los BST son inversos a los RBT, menor en estructuración, mayor en respuesta.

Sí uso Bono del requerimiento 6. Con el archivo grande de 1 Gb Us\_Accidents\_Dec19.

El tiempo de carga fue en promedio 464.921875 segundos.

Para el primer requerimiento buscando con la entrada 2017-02-08 el resultado fue 482257.

Para el segundo requerimiento buscando con la entrada 2017-02-08 el resultado fue 2551.

Severidad 1: 3

Severidad 2: 1788

Severidad 3: 698

Severidad 4: 62

Para el tercer requerimiento buscando con la entrada 2017-02-08 el resultado fue 482257.

Para el cuarto requerimiento buscando con la entrada 2017-02-08 el resultado fue 2551.