Álvaro A. Bacca, Juan S. Hernández

Santiago Arteaga

ISIS 1225

30 de abril de 2022

Análisis de resultados

Información del equipo

Req. 3 - Juan Sebastián Hernández, 202011077, j.hernandezv@uniandes.edu.co

Req. 4 - Álvaro A. Bacca, 202121869, a.baccap@uniandes.edu.co

Carga de datos

Para la carga de datos se crean diferentes tipos de mapas ordenados y no ordenados. Para el caso de los ordenados, estos siempre correspondieron a RBTs, debido a que su búsqueda tenía una complejidad menor a la del BST (logarítmica y lineal, respectivamente).

Entonces, en la función crear maps se crean los mapas para todos los requerimientos. Estos tienen las complejidades, teniendo en cuenta sus diferentes formas de crear:

1. Se crea un mapa de llaves nombres de clubes y valores un RBT con los jugadores del club, por lo que la complejidad es $O(NBlog(B))$, siendo N el número de jugadores y B el número de jugadores de un club específico.
2. Se tiene una complejidad $O(4Nlog(N))$ siendo N el número de jugadores.
3. Se tiene una complejidad $O\big(Nlog(N)\big)$ siendo N el número de jugadores.
4. Se tiene una complejidad $O\big(NBlog(B)\big)$ siendo N el número de jugadores y B el número de jugadores con ese tag
5. Se tiene una complejidad $O\big(NBlog(B)\big)$.
6. Se tiene una complejidad $O\big(P * 2NBlog(B)\big)$.

De esa manera, de manera general, la complejidad de la carga de datos va a estar limitada por el requerimiento 6, debido que esta tiene que recorrer dos veces los jugadores multiplicado por el número de posiciones que se encuentren en el archivo de entrada.

Req. 1 – Cinco adquisiciones recientes

Para el requerimiento, por el mapa ya creado, se realiza únicamente una operación de om.keySet(), la cuál tiene una complejidad $O(\log(N))$ debido a que se usa un RBT. Todas las demás búsquedas son realizadas en $O(1)$ debido a que se usan mapas no ordenados. De esa

manera, el requerimiento tiene una complejidad $O(\log(N))$ siendo N el número de jugadores del club.

Req. 2 – Jugadores según posición, desempeño, potencial, salario

En primer lugar, se realizan búsquedas en un mapa no ordenado, operación $O(1)$. Posteriormente, se recorren las llaves de 4 mapas diferentes para contar los jugadores, por lo que al final el requerimiento posee una complejidad $O(A + B + C + D)$, cada letra correspondiendo con el número de jugadores de la posición en concreto y que están en los rangos de desempeño general, potencial, salario y los jugadores que tienen las tres características, que al final son los que se van a retornar. Por parte de la carga de datos, se tiene una complejidad de $O(1)$, esto debido a que siempre se imprime un número pequeño de jugadores independientemente del tamaño del archivo y todas las búsquedas se hacen en un mapa no ordenado. De esa manera, el requerimiento tiene una complejidad final $O(N)$, siendo N la suma de los números de jugadores que pertenecen a los rangos mencionados anteriormente.

Req. 3 – Jugadores con etiqueta en rango salarial

Con el mapa ordenado creado en la carga de datos, se recorre todo el mapa para determinar cuáles contienen el *tag* ingresado. De esa manera, tiene una complejidad de orden $O(N)$, siendo N el número de jugadores que tienen un sueldo en el rango ingresado por el usuario. Por parte de la carga de datos, se tiene una complejidad de $O(1)$, esto debido a que siempre se imprime un número pequeño de jugadores independientemente del tamaño del archivo y todas las búsquedas se hacen en un mapa no ordenado. De esa manera, el requerimiento tiene una complejidad $O(N)$, siendo N el número de jugadores en el rango de sueldos puesto por el usuario. Cabe aclarar, que este approach dado al requerimiento fue

teniendo en mente de reducir la complejidad de la carga de datos en la creación de los mapas del requerimiento.

Req. 4 – Jugadores con rasgo en periodo de tiempo

Primero, se consigue el mapa ordenado correspondiente a la característica deseada. Esto se logra a través de la hash table antes creada llamada **`p_traits`**. Esta operación es $O(1)$. Luego, se usa este mapa ordenado, siguiendo el ordenamiento de año de nacimiento, para encontrar los jugadores que corresponden a los topes de fechas dados. Cada una de estas operaciones es $O(\lg N)$ para N jugadores con esa característica (podrían ser todos). Por último, se usa **`.keys()`** para sacar el rango utilizando los dos topes. Gracias a que el mapa es un RBT, esta operación también se hace en $O(\lg N)$. Con respecto a la vista, la entrada de datos y la impresión son $O(1)$, la primera por obviedad y la segunda porque se imprime un pequeño número de jugadores de ese rango. En conclusión, la complejidad del requerimiento 4 es $O(\lg N)$ donde N es el número de jugadores asociados a esa característica, posiblemente todos los jugadores.

Req. 5 – Histograma de propiedad

Primero se consiguen los extremos para dividir las *bins* apropiadamente. Usando los valores guardados anteriormente bajo el nombre **`minmax_[prop]`**, donde **`prop`** es la propiedad en cuestión, esto se hace en $O(1)$. Luego, por cada *bin* se busca el rango de jugadores que entran usando **`.keys()`** y los límites del *bin*. En total, este ciclo es $O(B \lg N)$ para B *bins* y N jugadores en total. Con respecto a la vista, la entrada de datos y la impresión son $O(1)$

(revisar req. 4). En conclusión, la complejidad total de este requerimiento es de $O(B \lg N)$ para B *bins* y N jugadores en total.

Req. 6 – Posibles sustitutos

Para implementar la función se realizan las operaciones:

- om.get() → O(log(N)) siendo N el número de jugadores para una determinada posición (del orden de los miles y cientos para large).
- om.ceiling() → O(log(N)) siendo N el número de jugadores para una determinada posición (del orden de los miles y cientos para large).
- om.floor() → O(log(N)) siendo N el número de jugadores para una determinada posición (del orden de los miles y cientos para large).

Estas últimas dos operaciones se realizan durante un *while* un número m veces, siendo m el número de jugadores con un mismo VR. Por su parte, para la impresión en View se tiene una complejidad O(1) por las mismas razones explicadas en el req. 4. De esa manera, podemos concluir que el requerimiento tiene una complejidad de $O(\log(N))$, siendo N el número de jugadores que hacen parte de la posición del jugador a buscar.