Reto 3:

Jose David Florez Ruiz Cod 202121507

Santiago Castro 202122158

Req. 3	Req. 4
Jose Florez	Santiago Castro

Análisis de complejidad:

Carga:

- **Complejidad:** O(N)
- **Justificación:** Esto se debe a que el peor proceso será el pasar por cada uno de los datos del csv y por ende este tiene una complejidad de O(N). Y el resto de los procesos su complejidad será mínima (menor que O(N)), por esto decidimos que esta es la complejidad de la carga. Esta no se podría reducir debido a que es necesario pasar por cada uno de los datos para añadirlo a nuestras estructuras de datos.

Requerimiento 1:

- **Complejidad:** O(M) - M siendo la cantidad de la cantidad de jugadores pertenecientes al club solicitado.
- **Justificación:** Esta complejidad se debe a que primero se hace un filtro con un hashmap en el cual solo recibimos los jugadores pertenecientes al club solicitado, este proceso teniendo una complejidad de *O(1). Después de esto, sigue la complejidad más alta que se evidencia al instante de crear el árbol, el cual tendrá una complejidad de O(M) siendo M la cantidad de jugadores. Esto lo hacemos debido a que esto reduciría la complejidad computacional al momento de cargar los datos, y así mismo no hacemos procesos innecesarios (crear árboles que no se usaran).

Requerimiento 2:

- **Complejidad:** O(A) - A siendo la cantidad de jugadores de la posición solicitada.
- **Justificación:** Al igual que en el requerimiento 1, apostamos a reducir la complejidad computacional al momento de la carga tratando de evitar procesos innecesarios haciendo solo creando los árboles que son solicitados por el usuario. Y en este

requerimiento la complejidad más alta seria O(A), A siendo la cantidad de jugadores de la posición solicitadas (generalmente bastante reducida previamente por el hashmap inicial). Esta se podría reducir creando los árboles previamente en la carga, pero esto conllevaría a mucho procesamiento computacional y pérdida de memoria innecesaria (porque se harán los árboles y se guardarán aun así no se usen).

Requerimiento 3:

- **Complejidad:** O(W) - W siendo la cantidad de jugadores en la lista del tag solicitado por el usuario.
- **Justificación:** Esta complejidad se debe a que ya que tenemos un hashmap que si los elementos son bastantes diversos en los tags (lo cual es lo más normal) la búsqueda de los elementos será O(1) y dejará una pequeña cantidad de elementos para filtrar la cual lo haremos en el árbol binario de filtro salario, el cual tendrá la complejidad más alta de O(W) cuando se va creando el mapa (La cual en datos cercanos en la realidad que son bastante variados este W nunca será demasiado grande).

Requerimiento 4:

- **Complejidad:** O(NlogN)
- **Justificación:** Esta complejidad se debe a que en el requerimiento se debe construir un árbol con una lista, que su complejidad es O(n). Además, que se debe buscar un rango de fechas, por lo que su complejidad es de O(logN * #de fechas que se quieran buscar).

Requerimiento 5:

- **Complejidad:** O(K)
- **Justificación:** En base a un map con propiedad como llave y el valor siendo una lista de jugadores que cumplen la propiedad dada, se recorre toda esta lista para agregar los datos de la lista a un árbol, esta lista está representada por la letra K. Después se busca en el árbol en base una propiedad dada por el usuario (logK).

Requerimiento 6:

- **Complejidad:** O(P) - P siendo la cantidad de jugadores de la posición solicitada por el usuario.
- **Justificación:** Se crean cuatros arboles binarios con la cantidad de jugadores de la posición solicitada por el usuario, lo cual tendrá una complejidad de O(P), y luego se buscan por los datos especificados por el usuario, para después crear un map con los valores representativos de los jugadores que pueden suplantar al jugador especificado por el usuario, por lo que su complejidad será O(log(P)).