Juan Sebastián Sánchez – 202121498 – js.sanchezl1@uniandes.edu.co - Req. 4

Alejandro Jaramillo Castellanos – 202111794 - a.jaramilloc2@uniandes.edu.co - Req. 3

Análisis con RBT, PROBING, ARRAY_LIST Y MERGE SORT

Cantidad Datos	Req. 1 Tiempo de Ejecución [ms]	Req. 2 Tiempo de Ejecución [ms]	Req. 3 Tiempo de Ejecución [ms]	Req. 4 Tiempo de Ejecución [ms]	Req. 5 Tiempo de Ejecución [ms]	Req. 6 Tiempo de Ejecución [ms]	Carga Tiempo de Ejecución [ms]
0.50%	0.13	3.97	0.37	1.45	1.58	3.35	92.12
5%	0.14	6.41	0.61	2	1.59	10.28	240.3
10%	0.21	9.53	1	2.18	1.68	18.65	432.12
20%	0.3	15.3	2.15	3.9	1.81	32.41	850.7
30%	0.36	21.71	4.48	4.38	1.9	44.6	1278.5
50%	0.71	33.68	6.13	6.52	1.95	71.4	2177.2
80%	1.34	51.07	6.24	10.55	1.98	113.23	3482.2
100%	2.67	65.84	8.03	10.66	2.88	133.53	4756.1

Carga O(N):

Análisis:

En la carga se llenan los árboles, mapas y listas, debido a que se hace jugador por jugador, la carga tiene una complejidad de O(N).

Requerimiento 1 O(N*Log(N)):

Análisis:

Primero se busca el equipo de fútbol en un árbol ordenado, que tiene por llaves el nombre de los equipos de futbol, lo cual tiene complejidad de log(N). Posteriormente, se extrae la lista que está en el valor del nodo, la cual contiene los jugadores del equipo. Esta se organiza con un merge sort, la cual tiene complejidad promedio de N*log(N). Lo cual implica que la complejidad del requerimiento es N*log(N).

Requerimiento 2 (O(M)):

Análisis:

Primero se trae una lista de los jugadores que juegan en la posición que ingresa el usuario(O(M)). Esto se encuentra en un mapa donde la llave es la posición y el valor es una lista de los jugadores que juegan en esa posición. Luego, se crea un árbol y se va llenando con la llave que sea el valor de su ‘overall’ y valor una lista de los jugadores con ese ‘overall’(O(M)). Después, se crea una lista con esos jugadores que están dentro del rango de ‘overall’ ingresado por el usuario (O(B)). Lo siguiente es en otro árbol organizar los

jugadores por su ‘potential’(O(B)). Creamos otro árbol organizando a los jugadores por su ‘wage_eur’ que también estén en el rango de ‘potential’ que ingresó el usuario (O(V)). Sacamos los jugadores dentro del rango de ‘wage_eur’ que ingresa el usuario y los añadimos a una lista (O(V)). Finalmente, retornamos esa lista con los jugadores que jueguen en esa posición y estén dentro de los rangos solicitados.

*M es un subconjunto de N

*B es un subconjunto de M

*V es un subconjunto de B

Requerimiento 3(Alejandro) (O(M)):

Análisis:

Primero traemos todos los jugadores del árbol de tags que cumplan con el tag requerido por el usuario (O(M))*. Luego creamos un mapa con llave el ‘wage_eur’ de los jugadores y valor una lista de los jugadores que tengan ese ‘wage_eur’(O(M)). Lo siguiente es traer todos los jugadores dentro de ese intervalo y organizarlos con merge (O(Blog(B)). Esta lista ya ordenada se retorna al usuario.

*M es un subconjunto de N

*B es un subconjunto de M

Requerimiento 4 (Juan Sebastián) (O(N)):

Análisis:

Primero traemos el árbol con llave los ‘traits’ de los jugadores y valor una lista de todos los jugadores que tengan ese ‘trait’ (O(N)). Luego sacamos el árbol que tiene como llave las fechas de nacimiento de los jugadores y valor los jugadores que cumplan con el ‘trait’ ingresado por el usuario (O(M)). Después, sacamos todos los jugadores dentro del rango de fechas ingresado por el usuario y los retornamos en una lista (O(B)). Esa lista se ordena con Merge sort y se retorna al usuario (O(Blog(B)).

*M es un subconjunto de N

*B es un subconjunto de M

Requerimiento 5 O(M):

Análisis:

Primero dependiendo de la selección del usuario se escoge el árbol con el que se hará el histograma. Luego se cuenta la cantidad de jugadores haciendo un ciclo en el value set del árbol, sumando los size de cada lista, esto es O(M), teniendo que M es un subconjunto de

todos los jugadores (N), porque como es el value set, son muchos menos datos, por lo que no es lineal de todos los datos, sino de muchos menos. Posteriormente se hacen varios ciclos los cuales no recorren listas de máximo de tamaño a la cantidad de intervalos que puso el usuario, por lo que el requerimiento es constante, ya que los ciclos se hacen en listas de tamaño muy reducido.

Requerimiento 6 (Bono) O(M):

Análisis:

Primero se traen todos los jugadores de la posición, estos jugadores están en un mapa de llave la posición y valor una lista con los jugadores. Esto trae un conjunto de datos M, siendo M un subconjunto de todos los datos (N). Posteriormente se arman los árboles iterando jugador por jugador. Lo cual implica que la complejidad del requerimiento es O(M).