ANÁLISIS DEL RETO

Ximena Lopez, 202312848, ax.lopez@uniandes.edu.co

Juan Esteban Guzman, 202313337 je.guzman2@uniandes.edu.co

Sofia Losada, 202221008, s.losadam@uniandes.edu.co

Requerimiento 1

Descripción

En el requerimiento se implementa un algoritmo para conocer una cantidad N de partidos jugados recientemente, dada según su condición de encuentro. De esta manera se implementa un ciclo *for* , con un condicional anidado que recorre los datos y arroja el resultado.

Entrada	El número (N) de partidos de consulta, Nombre del equipo, Condición del equipo en los partidos consultados
Salidas	Se muestra el total de partidos según su condicion, junto con la fecha del partido, equipo local, equipo visitante, país de encuentro, ciudad donde se disputa el encuentro, goles del local, goles del visitante.
Implementado (Sí/No)	Si

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Paso 1: Se crea la lista ‘finales’ como una lista vacía.	O(1)
Paso 2: Bucle for cada in lt.iterator(lista). La complejidad es dependiendo de los elementos de ‘lista’.	O(N)
Paso 3: tres comprobaciones condicionales basadas en el valor de ‘condicion’. Estas comprobaciones condicionales son operaciones de tiempo constante.	O(1)
Paso 4: Al cumplirse una condición se agrega un elemento a la lista finales utilizando addData. La inserción en una lista vinculada generalmente es una operación de tiempo constante.	O(1)
Paso 5: ordenamiento con shell	O(N^3/2)
TOTAL	***O(N)***

Pruebas Realizadas

Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

Muestra	Salida	Tiempo (ms)
small	Dato1	1.652
5 pct	Dato2	3.480
10 pct	Dato3	9.195
20 pct	Dato4	11.508
30 pct	Dato5	15.942
50 pct	Dato6	17.940
80 pct	Dato7	47.085
large	Dato8	38.026

Requerimiento 2

Descripción

El requerimiento necesita implementar un algoritmo que permita conocer los primeros N goles de un jugador que entra como parámetro. De esta manera se implementa un ciclo *for* y un *if* anidado que recorre los datos bajo condición.

Entrada	El número (N) de goles de consulta, Nombre completo del jugador.
Salidas	Se debe mostrar: el total de anotaciones obtenidas por el jugador. El array debe incluir fecha, Equipo local, Equipo visitante, Equipo del jugador, Minuto en el que se marcó el gol, Tipo de anotación, si fue por falta desde el penal, Tipo de anotación, si fue autogol.
Implementado (Sí/No)	Si

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	Complejidad
Paso 1: Se crea la lista ‘finales’ como una lista vacía	O(1)

Paso 2: Se ordena con shell la lista que le entra como parámetro, goal_scorers.	O(N^3/2)
Paso 3: Se realiza una búsqueda iterativa con for in que recorre toda la lista.	O(N)
Paso 4: Si la condición se cumple, se agrega el elemento cada a la lista finales utilizando addData. La inserción en una lista vinculada generalmente es una operación de tiempo constante.	O(1)
TOTAL	***O(N)***

Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

Muestra	Salida	Tiempo (ms)
small	Dato1	12.497[ms]
5 pct	Dato2	76.249
10 pct	Dato3	145.699
20 pct	Dato4	
30 pct	Dato5	
50 pct	Dato6	
80 pct	Dato7	
large	Dato8	

Requerimiento 3

El requerimiento necesita implementar un algoritmo que permita consultar la cantidad N de partidos jugados por un equipo dado por parámetro en un periodo específico de tiempo, para esto se realiza un *for iterativo* con una búsqueda binaria anidada y y un ordenamiento tipo shell.

Entrada	Nombre del equipo, La fecha inicial del periodo a consultar, La fecha final del periodo a consultar
Salidas	Se debe retornar: el Número total de partidos disputados, Número total de partidos disputados como local, Número total de partidos disputados como visitante. El array debe incluír: Fecha del partido, marcador del equipo local, marcador del equipo visitante, equipo

	local, equipo visitante, país del encuentro, ciudad donde se disputa el encuentro, nombre del torneo asociado, anotación, si el partido presento goles por faltas desde el punto penal, anotación, si el partido presento autogoles.
Implementado (Sí/No)	Si / Ximena Lopez

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Paso 1: Iniciamos con las operaciones de 'lis', 'total', 'local', y 'visi': Todas estas operaciones son de tiempo constante.	O(1)
Paso 2: for cada in lt.iterator(lista): Este bucle itera a través de todos los elementos en la lista de entrada 'lista'. La complejidad de este bucle depende del número de elementos en lista.	O(N)
Paso 3: Dentro del bucle, realizamos: • Comprobaciones condicionales: Las comparaciones de fechas y las comparaciones de cadenas (por ejemplo, home == equipo) son operaciones de tiempo constante y se pueden considerar O(1) en promedio. • Llamada a 'busqueda_binaria_scorers': Esto depende de la implementación de la función busqueda_binaria_scorers, pero si asumimos que es una búsqueda binaria típica, la complejidad sería O(log m), donde "m" es el tamaño de la lista scorers. • Las operaciones de agregar elementos a lis (por ejemplo, lt.addLast(lis, cada)) son operaciones de tiempo constante y se pueden considerar O(1) en promedio.	Comprobaciones condicionales: O(1) Llamada a 'busqueda_binaria_scorers': O(log m). Operaciones para agregar elementos: O(1)
Paso 4: Llevar un conteo de 'local' y 'visi' es una operación de tiempo constante.	O(1)
TOTAL	***O(n*logm)***

Pruebas Realizadas

Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

Procesadores	**Intel PENTIUM GOLD**
Memoria RAM	8 GB
Sistema Operativo	Windows 10

Entrada	**Tiempo (s)**
small	6.216
5pct	74.181
10pct	152.832
20pct	336.786
30pct	521.876
50pct	928.480
80pct	1699.022
large	2289.232

Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

Muestra	**Salida**	**Tiempo (ms)**
small	Dato1	6.216
5 pct	Dato2	74.181
10 pct	Dato3	152.832
20 pct	Dato4	336.786
30 pct	Dato5	521.876
50 pct	Dato6	928.480
80 pct	Dato7	1699.022

large	Dato8	2289.232

Graficas

Las gráficas con la representación de las pruebas realizadas.

Análisis

Tiene una complejidad mas alta log n y a medida que aumenta el tamaño de los datos se ve un crecimiento mas grande, parecido a la gráfica exponencial

Requerimiento 4

Para el siguiente requerimiento se utiliza un algoritmo para consultar los partidos relacionados con un torneo utilizando su nombre y un periodo entre dos fechas especificadas. De esta manera, se implementa un *for* para recorrer el arreglo con los valores. Después el primer *if* anidado se utiliza un *for* anidado para recorrer los datos que respondan a la condición del rango de fechas.

Entrada	Nombre del torneo, la fecha inicial del periodo a consultar, la fecha final del periodo a consultar
Salidas	Se debe mostrar: el total de partidos relevantes al torneo, el total de países involucrados en el torneo, el total de ciudades donde se

	disputan los partidos del torneo, el total de partidos definidos por cobros de punto penal. Así mismo, el array debe contener:
Implementado (Sí/No)	Si / Juan Esteban

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Paso 1: se inician las estructuras de datos. ListTotalCountries , fechaini, fechaFin y answer son de tiempo constante.	O(1)
Paso 2: bucle principal que itera a traves de los resultados de los partdios . for i in range(lt.size(data_structs['model']['results'])). Su complejidad depende del número de elementos.	O(N)
Paso 3: bucle anidado. for x in range((lt.size(data_structs['model']['shootouts']))). Está anidado dentro del bucle principal y, por lo tanto, su complejidad es O(n*m)	O(N)
Paso 4: Añadir elementos de una lista. lt.addLast(listTotalCountries, countryResult), lt.addLast(listTotalCities, cityResult), y lt.addLast(answer, matchResult). Utilizan listas vinculadas para hacer la inserción final a la lista.	O(1)
Paso 5: Ordenamiento. En answer con se.sort se realiza en función de tres criterios y tiene una complejidad de O(n log n).	O(nlogn)
TOTAL	***O(n*m*log*n)***

Procesadores	**AMD Ryzen 7 5700G with Radeon Graphics**
Memoria RAM	8 GB
Sistema Operativo	Windows 11

Pruebas Realizadas

Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

Entrada	**Tiempo (s)**
small	35.528 [ms]
5pct	63.367[ms]
10pct	70.000[ms]

20pct	70.031[ms]
30pct	89.754[ms]
50pct	192.013[ms]
80pct	1133.131[ms]
large	3477.369[ms]

Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

Muestra	Salida	Tiempo (ms)
small	Dato1	11.043
5 pct	Dato2	74.431
10 pct	Dato3	89.346
20 pct	Dato4	145.872
30 pct	Dato5	209.115
50 pct	Dato6	609.341
80 pct	Dato7	1454.765
large	Dato8	3412.931

Graficas

Las gráficas con la representación de las pruebas realizadas.

Análisis

Requerimiento 5

En este requerimiento se utiliza un algoritmo para consultar las anotaciones de un jugador específicao en un periodo de tiempo determinado. Para esto se utiliza un recorrido *for* para recorrer el arreglo *goal_scorers*, pues la mayoría de los datos que se necesitan incluir en el arreglo de retorno se encuentran ahí. Dentro del *for* se filtran los datos de acuerdo a las condiciones de los *if* anidados. Finalmente se realiza un *for* anidado para recorrer el arreglo *results*, pues ahí se encuentra la información faltante que se debe retornar.

Entrada	Nombre del anotador, la fecha inicial del periodo a consultar, la fecha final del periodo a consultar
Salidas	Se debe mostrar: Número de anotaciones obtenidas por el jugador, numero de torneos en que anoto el jugador, número de

	anotaciones obtenidas desde el punto penal, número total de autogoles cometidos. El Array de retorno debe contener: Fecha del partido, minuto en el que se marcó el gol, equipo local, equipo visitante, equipo del jugador, marcador del equipo local, marcador del equipo visitante, nombre del torneo donde se marcó el gol, tipo de anotación, si fue por falta desde el penal, tipo de anotación, si fue autogol.
Implementado (Sí/No)	Si / Sofia Losada

Análisis de complejidad

Pasos	**Complejidad**
Paso 1 Asignar a resultados el array del csv results y a goal_scorers el array del csv goal_scorers. Crear la lista de retorno, las listas para guardar las apariciones.	O(1)
Paso 2 Objeto “goal” obtiene varios valores de la iteracion en goal_scorers	O(1)
Paso 3 Se realiza la comparación de nombre.lower() y nombre_jugador. Dado que j es la longitud de las cadenas	O(J)
Paso 4 En la segunda comparación, se utiliza una operación constante que determina que la fecha pertenezca al rango dado por parámetro	O(1)
Paso 5 En el ciclo for sobre results, se obtiene fecha 2, se hace la comparación con fecha y se obtiene torneo	O(1)
Paso 6 La implementación de la funcion isPresent. Que determina si el torneo ya está incluída. La cantidad de torneos es k	O(K)
Paso 7 El condicional que agrega o no el torneo	O(1)
TOTAL	***O(N * k + nlogn)***

Pruebas Realizadas

Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

Procesadores	**Intel PENTIUM GOLD**
Memoria RAM	8 GB
Sistema Operativo	IOS

Entrada	
small	10.545 [ms]
5pct	81.122 [ms]
10pct	40.696 [ms]
20pct	94.966 [ms]
30pct	269.580 [ms]
50pct	729.837 [ms]
80pct	1585.216 [ms]
large	3281.788 [ms]

Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

Muestra	**Salida**	**Tiempo (ms)**
small	Dato1	10.545 [ms]
5 pct	Dato2	81.122 [ms]
10 pct	Dato3	40.696 [ms]
20 pct	Dato4	94.966 [ms]
30 pct	Dato5	269.580 [ms]
50 pct	Dato6	729.837 [ms]
80 pct	Dato7	1585.216 [ms]
large	Dato8	3281.788 [ms]

Graficas

Las gráficas con la representación de las pruebas realizadas.

Análisis

Requerimiento 6

En este requerimiento se busca aplicar un algoritmo que clasifique los N mejores equipos de un torneo o liga, dentro de un periodo de tiempo. De esta manera se implementó un for para recorrer la lista de datos y mediante condicionales anidados se filtró la búsqueda para añadir a la lista de retorno los elementos que cumplieran las características.

Entrada	El número (N) de equipos para consulta, nombre del torneo que se desea consultar, la fecha inicial del periodo a consultar, la fecha final del periodo a consultar.
Salidas	Se debe mostrar: El total de equipos involucrados en el torneo., el total de encuentros disputados en el periodo de tiempo, el total de países involucrados en el torneo, el total de ciudades involucradas en el torneo, el nombre de la ciudad donde más partidos se han disputado. El Array de retorno debe contener: El nombre del equipo, el total de puntos obtenidos, la diferencia de goles, el total de partidos disputados, el total de puntos obtenidos desde la línea penal, el total de puntos recibidos por autogol, el total de victorias, el total de empates, el total de derrotas, el total de goles obtenidos por sus jugadores, el total de goles recibidos por el equipo. Además: El jugador con más anotaciones en el equipo con la siguiente información: ▪ Nombre del jugador. ▪ El total de goles anotados. ▪ El total de partidos donde anoto un gol. ▪ El promedio de tiempo (en minutos) para anotar los goles.
Implementado (Sí/No)	Si

Análisis de complejidad

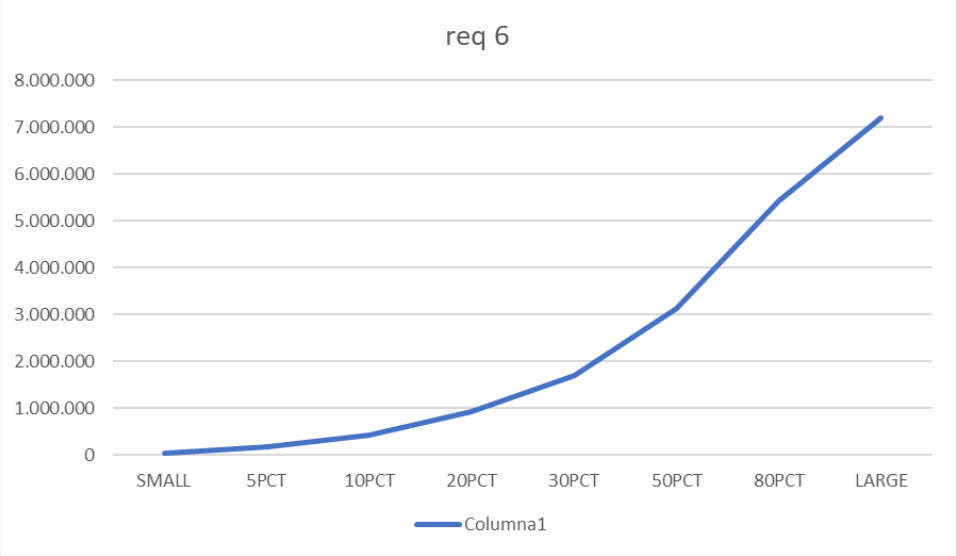
Pasos	**Complejidad**
Paso 1 Se invoca el requerimiento 4 de manera que se incorpora a su vez su complejidad	O(***O(m*nlogn)***
Paso 2 Se implementa un for iterativo	O(N)
Paso 3 Se establecen los condicionales anidados dentro de la función	O(N * nlogn)

Paso 4 Se implementan dos ciclos for, no anidados	O(2N)
TOTAL	***O(m*nlogn)***

Pruebas Realizadas

Procesadores	**Intel PENTIUM GOLD**
Memoria RAM	8 GB
Sistema Operativo	Windows 10

Entrada	**Tiempo (s)**
small	33.549
5pct	162.115
10pct	412.248
20pct	918.646
30pct	1702.207
50pct	3135.716
80pct	5426.113
large	



Requerimiento 7

Para este requerimiento se utiliza un algoritmo que clasifique los N mejores jugadores que participan en partidos de competencia oficial (excluyendo los juegos amistosos) dentro de un periodo de tiempo y esto puede entenderse como el TOP ranking TOP de ciertos jugadores en partidos oficiales.

Entrada	El número (N) de jugadores para consulta. la fecha inicial del periodo a consultar la fecha final del periodo a consultar (las fechas en formato "%Y-%m-%d")
Salidas	El total de anotadores que se encontraron en la consulta, el total de partidos o encuentros en que participaron los anotadores, el total de torneos donde participaron los anotadores en ese periodo, el total de anotaciones o goles obtenidos durante los partidos de ese periodo, el total de goles por penal obtenidos en ese periodo, el total de autogoles en que incurrieron los anotadores en ese periodo, el listado de anotadores debe estar ordenado por el criterio compuesto de sus estadísticas. Donde cada uno de los jugadores resultantes contendrán la siguiente información: El nombre del anotador, el puntaje que obtiene el jugador como anotador, el total de goles anotados, el total de goles anotados por penales. o El total de autogoles anotados, el tiempo promedio para anotar en minutos, el total de torneos en que anotó el jugador, el total de anotaciones obtenidos en una victoria, el total de anotaciones obtenidos en un empate, el total de anotaciones obtenidos en una derrota, último gol anotado por el jugador con la siguiente información: ▪ Fecha del encuentro. ▪ Nombres de los equipos local y visitante. ▪ Puntaje de los equipos local y visitante. ▪ Minuto en que anotó el gol. ▪ Detalles técnicos del gol (si fue por falta desde el punto penal o autogol).
Implementado (Sí/No)	Si

Análisis de complejidad

Pasos	**Complejidad**
Paso 1	
Paso 2	
Paso 3	
Paso 4:	
TOTAL	

Pruebas Realizadas

Entrada	**Tiempo (s)**
small	
5pct	
10pct	
20pct	
30pct	
50pct	
80pct	
large	