ANÁLISIS DEL RETO

Daniel Carrascal, 202111042, d.carrascal@uniandes.edu.co

Jerónimo Suárez Martínez, 202123775, J.suarezm2@uniande.edu.co

Geronimo Gamba, 202222214, G.gamba@uniandes.edu.co

Para las entradas y salidas, se usan las funciones del model

Requerimiento 1

Descripción

Para este requerimiento se utilizó el archivo de partidos, en el catalog. No se realizó copia porque este archivo solo iba a ser recorrido por un loop. Ya en el loop, se utilizó la condición de búsqueda deseada (visitante,local o indiferente) para saber que en que columna de dato debíamos buscar, away_team para visitante, home_team para local y cualquiera de los dos para indiferente. Ya sabiendo en que columna se debía buscar, se procedió a realizar una comparación. Sí el equipo en la condición deseada tenía el mismo nombre que el equipo ingresado por parámetro se agregaba a una lista nueva en la última posición, este proceso se realizó para todos los partidos almacenados en la base de datos de partidos. Después de recorrer todos los partidos de la base de datos resultados se organizaron los partidos con las condiciones dadas por el ejercicio y usando merg.

Entrada	Catalogo (lista de resultados), numero partidos requeridos, condición, equipo
Salidas	Sublista para tabular y tamaño (num partidos de partidos jugados)
Implementado (Sí/No)	Si

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Paso 1	O(n)
Paso 2	O(n)
Paso ….	O(n)
TOTAL	***O(n)***

Pruebas Realizadas

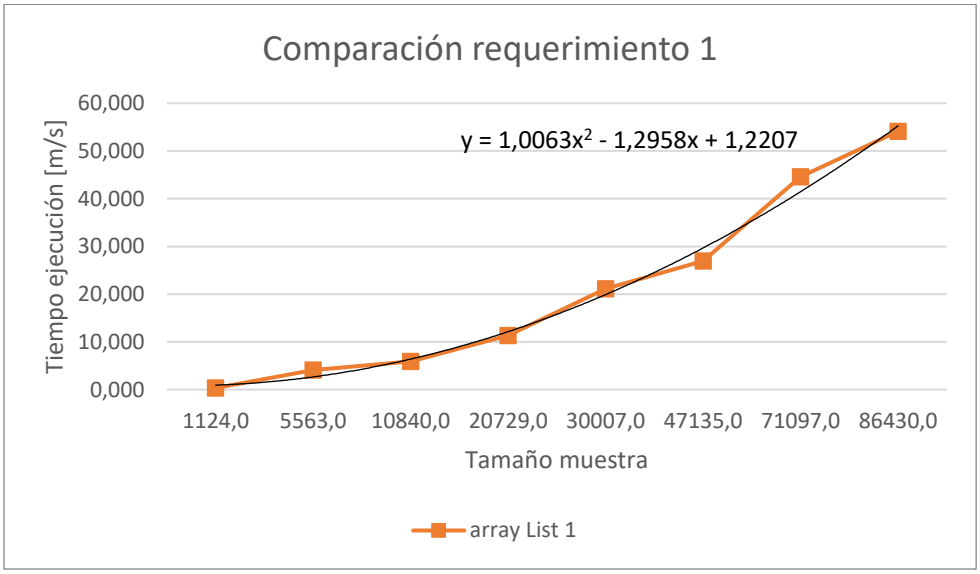
Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

Procesadores	Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz
Memoria RAM	16 GB
Sistema Operativo	Windows 11

Tablas de datos

MUESTRA	**ARRAY LIST**
SMALL	0,332
5 PCT	4,065
10 PCT	5,921
20 PCT	11,351
30 PCT	21,091
50 PCT	26,937
80 PCT	44,582
LARGE	54,125

Graficas



Análisis

Análisis de resultados de la implementación, tener cuenta las pruebas realizadas y el análisis de complejidad.

Requerimiento 2

Plantilla para el documentar y analizar cada uno de los requerimientos.

Descripción

En el caso del req 2, se utilizaron los datos del archivo goleadores. Con este archivo, se fue recorriendo dato por dato mediante un for. Ya dentro del loop, se revisaba que el dato almacenado en la llave "scorer" fuera el nombre del jugador buscado, si era así, se almacenada el dato del gol en una lista nueva mediante lt.addlast. Posterior a recorrer todo el archivo de goleadores, se sacó el tamaño de la lista con los goles de los jugadores ya que este corresponde a la cantidad total de goles porque cada posición es un gol del jugador. Finalmente, se organizó el archivo con los criterios dados en el enunciado.

Entrada	Catalogo (lista con goleadores), número de goles requeridos, jugador
Salidas	Sublista para tabular, tamaño (número de goles hechos)
Implementado (Sí/No)	Si

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Paso 1	O(n)
Paso 2	O(n)
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).

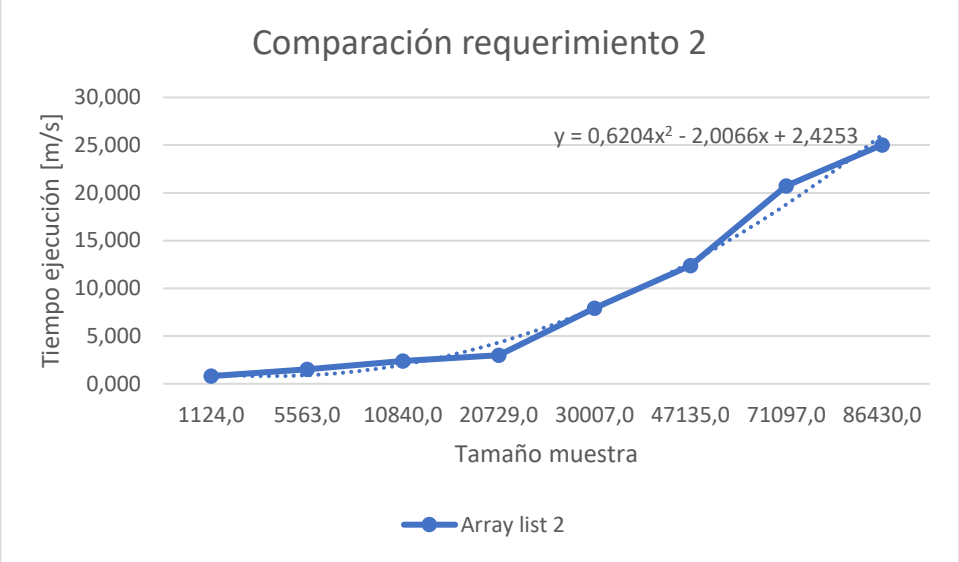
Procesadores	Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz
Memoria RAM	16 GB
Sistema Operativo	Windows 11
Procesadores	Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz

Tablas de datos

MUESTRA	**ARRAY LIST**
SMALL	0,804
5 PCT	1,510
10 PCT	2,395
20 PCT	2,986
30 PCT	7,905
50 PCT	12,387

80 PCT	20,713
LARGE	25,025

Graficas



Análisis

Análisis de resultados de la implementación, tener cuenta las pruebas realizadas y el analisis de complejidad.

Requerimiento 3

Plantilla para el documentar y analizar cada uno de los requerimientos.

Descripción

Para este requerimiento primero se utiliza los datos almacenados en goleadores, estos son filtrados dejando únicamente partidos donde haya jugado el equipo que se está consultando sin importar su condición. Posteriormente, se organizan los datos por las fechas porque luego sobre estos se realizará una búsqueda binaria.

Después de tener filtrados los datos de goleadores, se recorre partidos revisando que o el equipo local o el equipo visitante sea el equipo que se está buscando. Si el equipo si jugo en esa fecha, se procede a buscar la fecha en los goleadores que previamente habían sido filtrados y de este se obtiene la información de goles y penales, estos se agregan a los datos del partido. Posteriormente, dependiendo de si el equipo se encontró en “home_team” o “away_team” se le suma 1 al contador de partidos locales o visitantes respectivamente. Al final se utiliza merg para organizar los datos junto a los criterios

dados por el enunciado. También se suma la cantidad de partidos locales y visitantes para saber los partidos totales del equipo.

Es importante aclarar, que la búsqueda binaria funciona en este ejercicio porque como primero se filtró los datos para que solo quedaran partidos donde jugo el equipo de interés no habrá datos repetidos en términos de fecha, porque un equipo solo juega un partido por día.

Entrada	Catálogo, equipo, fecha inferior, fecha superior
Salidas	Sublista para tabular, num partidos totales, num partidos visitante, num partidos locales
Implementado (Sí/No)	Si, Daniel Carrascal

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Paso 1	O(n)
Paso 2	O(n)
Paso 3	O(n)
Paso4	O(n)
Paso 5	O(n)
TOTAL	***O(n^2)***

Pruebas Realizadas

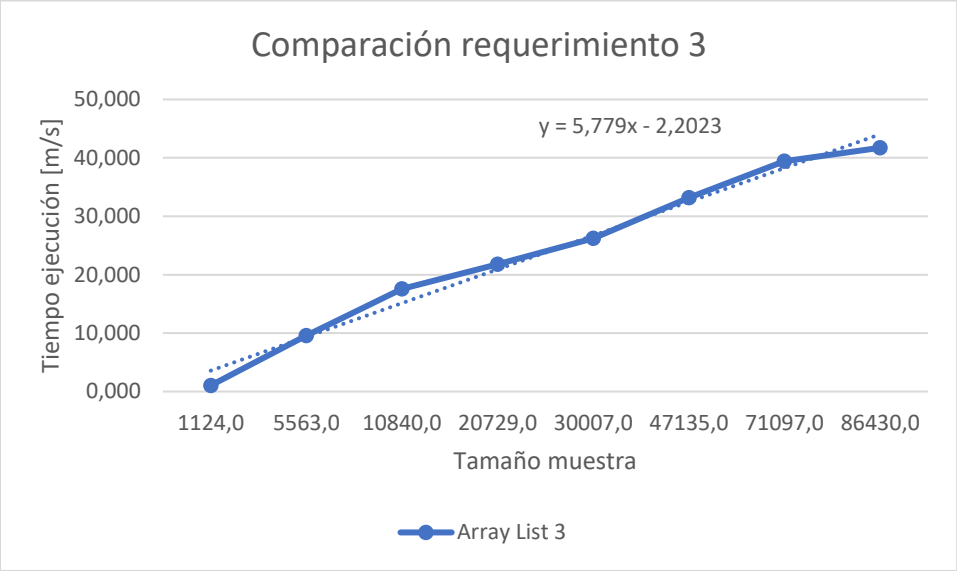
Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

Procesadores	Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz
Memoria RAM	16 GB
Sistema Operativo	Windows 11

Tablas de datos

MUESTRA	**ARRAY LIST**
SMALL	1,043
5 PCT	9,543
10 PCT	17,549
20 PCT	21,768
30 PCT	26,219
50 PCT	33,153
80 PCT	39,430
LARGE	41,721

Graficas



Análisis

Análisis de resultados de la implementación, tener cuenta las pruebas realizadas y el analisis de complejidad.

Requerimiento 4

Plantilla para el documentar y analizar cada uno de los requerimientos.

Descripción

Para este requerimiento, se hace uso de varias sub listas que permiten ir almacenando los datos importantes. Primero, se itera sobre los resultados y se va agregando a una de estas listas los partidos que cumplan con el torneo y la restricción de fechas. A partir de esto, y por cada iteración, si el país donde se juega y la ciudad donde se juega ya están en la lista correspondiente a países y en la correspondiente a ciudades, no se hace nada, pero si no pertenecen aún a las listas, se agrega. Finalmente, por cada partido agregado, se agrega a otra **lista especifica** los datos de la fecha, el equipo visitante y el equipo local para poder comparar, siendo estos los datos que hacen al partido único. Es importante recalcar que esta última lista mencionada y la lista de partidos incluidos por torneo y fecha no solo tienen la misma longitud, sino que sus elementos comparten la misma posición. Por lo tanto, podré usar la **lista especifica** con los datos específicos para comparar y la lista grande para ir organizando los datos.

Después de esto, se itera sobre los penales y a cada dato se le saca la misma lista que se mencionó anteriormente (lista con la fecha, el equipo de visitante y el de local), esta lista se compara con la **lista especifica** preguntando si el elemento pertenece a la lista. Si pertenece, me agregará a una lista toda la información del elemento y a otra **lista especifica 2** la información de este partido.

Por último, se itera sobre la **lista especifica** y la lista general de los partidos del torneo. En este paso estaré relacionando las listas específicas. Como ya se que todas las listas tienen las informaciones de los partidos del torneo, solo se debe organizar. Primero, si el dato del partido esta en la **lista especifica 2** es porque fue definido por penales y se agregará a "sublist" agregando uno al contador de penales. De lo contrario, agregará "desconocido" a la casilla correspondiente. Al terminar las iteraciones, solo se debe calcular la longitud de la lista de partidos en el torneo, países, ciudades y devolver estas junto al contador de penaltis y "sublist".

Entrada	Catálogo, torneo elegido, fecha inferior, fecha superior
Salidas	Total partidos jugados, total países involucrados, total ciudades jugadas, numero de partidos definidos por penaltis, Sublista para tabular
Implementado (Sí/No)	Sí, Jerónimo Suárez Martínez

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Paso 1	O(n)
Paso 2	O(n)
Paso 3	O(n^2)
Paso 4	O(n)
TOTAL	***O(n^2)***

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).

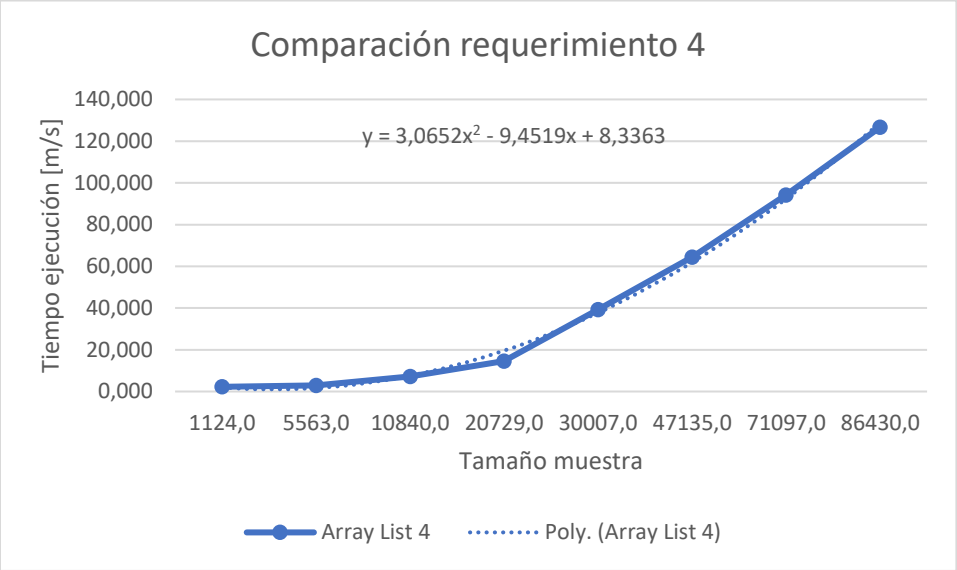
Procesadores	Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz
Memoria RAM	16 GB
Sistema Operativo	Windows 11

Tablas de datos

MUESTRA	**ARRAY LIST**
SMALL	2,258
5 PCT	2,986
10 PCT	7,220
20 PCT	14,629
30 PCT	39,307
50 PCT	64,441
80 PCT	94,231

LARGE	126,658

Graficas



Análisis

Análisis de resultados de la implementación, tener cuenta las pruebas realizadas y el análisis de complejidad.

Requerimiento 5

Plantilla para el documentar y analizar cada uno de los requerimientos.

Descripción

Al correr el código el programa busca inicializar un proceso de búsqueda y clasificación donde según los parámetros ingresados se busca un jugador y todas las participaciones de goles que tuvo en un rango de dos fechas. Entre la fecha inicial y la fecha final, el programa arroja; la cantidad de goles, la cantidad de penales, la cantidad de autogoles y en los torneos en los que se realizaron estas anotaciones. Todo esto presentado dentro del rango de tiempo ingresado por parámetro.

Para el funcionamiento de este requerimiento se selecciona un jugador del catálogo y mediante las listas de partidos y anotaciones se inicializa un contador donde se van contando los respectivos goles anotados por este jugador. Para esto se crea un bucle donde se compara el nombre del jugador ingresado por parámetro y si este coincide con el autor del gol y con las fechas ingresadas se agrega al contador de anotaciones. Mediante un segundo bucle se filtran la información y se verifica por clasificación de; torneo, penales y autogoles y al finalizar se devuelve una tupla con anotaciones1, torneos1, penales1, autogoles1 y anotaciones_jugador. Todo esto solo es válido si se encuentra dentro de la fecha ingresada por parámetro. (Ffinal, Finicial).

Entrada	Catálogo, Jugador, Finicial, Ffinal.
Salidas	En una lista tabulada el algoritmo presenta. Anotaciones1, autogoles1, torneos1,pelanes1 y anotaciones_jugador.
Implementado (Sí/No)	Si. Geronimo Gamba.

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Paso 1	O(n)
Paso 2	O(n)
Paso 3	O(n)
TOTAL	***O(n^2)***

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(R) Core(TM) i7-10750H CPU @ 2.60GHz
Memoria RAM	8 GB
Sistema Operativo	Windows 11

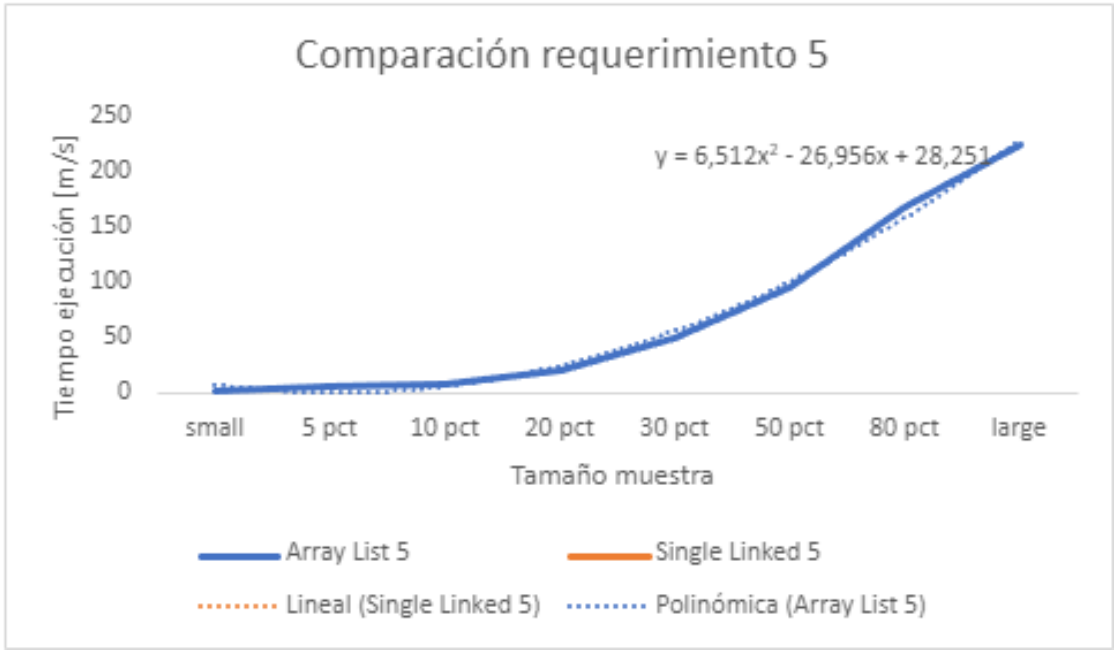
Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

Muestra	**Array List**
small	3,292
5 pct	6,651
10 pct	9,343
20 pct	22,439
30 pct	51,344
50 pct	96,126
80 pct	169,043
large	225,801

Graficas

Las gráficas con la representación de las pruebas realizadas.



Análisis

Debido a que ambos bucles se ejecutan en secuencia y son bucles anidados, la complejidad es cuadrática ***O(n^2),*** pues al estar un bucle dentro de otro se tiene en consideración que si se ejecuta n veces en el bucle mas chiquito pues habrá una operación de O(n) *O (n) lo cual ayuda a entender la complejidad cuadrática del código. Se entiende que mientras un bucle itera los partidos el otro itera las anotaciones y que dentro de este segundo bucle hay ciertas verificaciones de coincidencia.

Esto se justifica al revisar la grafica pues se puede evidenciar como se presenta la tendencia de los datos y como la línea de mejor ajuste coincide con ser una ecuación cuadrática en relación al tiempo y a la dispersión de datos.

Requerimiento 6

Plantilla para el documentar y analizar cada uno de los requerimientos.

Descripción

Primero se crear listas auxiliares las cuales ayudaran a encontrar posiciones de datos más fácilmente, esto se explicará más adelante.

Iniciando el programa, se comienza a recorrer la información almacenada en resultados, y se revisa que el partido este dentro de las fechas establecidas como rangos y sea parte del torneo de interés. Si el partido cumple con estas condiciones, se revisa que la ciudad del partido, si esta no había sido almacenada en una lista con todos los nombres de la ciudad "ciudadesnombres" se procede agregar el nombre al final de lista y a su vez se genera un diccionario con el nombre de la ciudad como llave y el valor de 1, el cual es almacenado en otra lista que contiene las ciudades con su respectivo número de apariciones "ciudades", finalmente se suma uno en la cantidad de ciudades. Por otro lado, si ya había sido almacenada, se busca el nombre de la ciudad en la lista "ciudadesnombres" y con el índice de esta se ingresa a su posición en la lista "ciudades" y se le suma una aparición. Esto funciona porque como cuando se agrega un nombre a la primera lista al mismo tiempo se crea su dato en la lista de "ciudadesnombres" por lo que comparten el mismo índice.

Después se realiza el mismo proceso con países, pero en este caso, solo se utiliza una lista con el nombre de los países.

Después de esto, se comienza a analizar por equipo sus resultados, para esto revisa que el equipo a agregar sea nuevo (esto lo hace con el mismo proceso que el utilizado en ciudades) si lo es, almacena los datos de ese partido como los primero de ese equipo, si no es el primero agrega los datos al equipo, teniendo encuentra si gano, perdió o empato, después agrega los goles totales, los encuentra y la diferencia de estos. También revisa si hubo penal y autogol, si le suma una unidad a su respectiva llave. También se le agrega una llave para ver el goleador del equipo, esto se modifica después, por ahora será 0. Si el equipo ya existía se les agregan los datos correspondientes a los resultados del equipo.

Ya con casi todos los datos listos, se procede a buscar el goleador por equipo. Para esto se usan los datos de goleadores y se va iterando por jugador. Primero se organizan los datos de cada jugador, guardando su cantidad de goles, cantidad de partidos donde anoto, promedio de gol y su nombre. Después con los datos de cada jugador se mira a que equipo pertenece, después a ese equipo se le revisa si este tiene su goleador en 0, si es así, los datos del jugador se guardan como los del goleador del equipo. Por otro lado, si el goleador del equipo ya tiene información previa se revisa cuál de los jugadores tiene más goles y se deja el que tenga más.

Entrada	Torneo requerido, fecha inferior, fecha superior, numero de equipos, catalogo
Salidas	Lista para tabular, cantidad partidos, ciudades involucradas, países involucrados, ciudad donde más se jugó.
Implementado (Sí/No)	Sí, Daniel Carrascal.

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Paso 1	O(n)
Paso 2	O(n)
Paso 3	O(n)
Paso 4	O(1)
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).

Procesadores	Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz
Memoria RAM	16 GB
Sistema Operativo	Windows 11

Tablas de datos

MUESTRA	**ARRAY LIST**
SMALL	8,221
5 PCT	112,181
10 PCT	377,798
20 PCT	1283,439
30 PCT	2524,789
50 PCT	5686,434
80 PCT	10599,668
LARGE	14386,578

Graficas

Análisis
Análisis de resultados de la implementación, tener cuenta las pruebas realizadas y el análisis de complejidad.

Requerimiento 7
Plantilla para el documentar y analizar cada uno de los requerimientos.

Descripción
Primero se filtran los datos en resultados para eliminar todos los partidos amistosos, después se comienza a recorrer los goleadores en goleadores y se revisa que el partido este en la lista filtrada lo que quiere decir que no es amistoso. Después se revisa que el jugador se haya guardado antes, si es así, se agregar los datos que se piden como la cantidad de goles, goles en propia puerta, penales, puntos totales. También se revisa que el torneo en el que está el jugador para ver si es necesario agregar un torneo o si por lo contrario ya había al menos un partido jugado en ese torneo y no se le debe sumar partidos. Para saber la información del torneo, si gano, perdió, o empato en ese partido se busca el partido en los resultados y con eso se agrega la información a las correspondientes llaves. Por último, se revisa que el gol actual sea más viejo que el que esta guardo como gol más reciente, si no es asi, se cambia. Para esto se tiene desde que se crea la información del jugador una llave “last_gol” que inicia en cero, pero va siendo reemplazada por el gol con la fecha más reciente y si tiene la misma fecha se revisa el minuto más alto.

Entrada	Fecha inferior, fecha superior, numero equipos (número de jugadores), catalogo.
Salidas	Lista de información de jugadores, cantidad de jugadores, cantidad autogoles, cantidad penales, cantidad goles, cantidad partidos.
Implementado (Sí/No)	Sí.

Análisis de complejidad
Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Paso 1	O(n)
Paso 2	O(n)
Paso 3	O(1)
Paso 4	O(n*log(n))
TOTAL	***O(n^2)***

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).

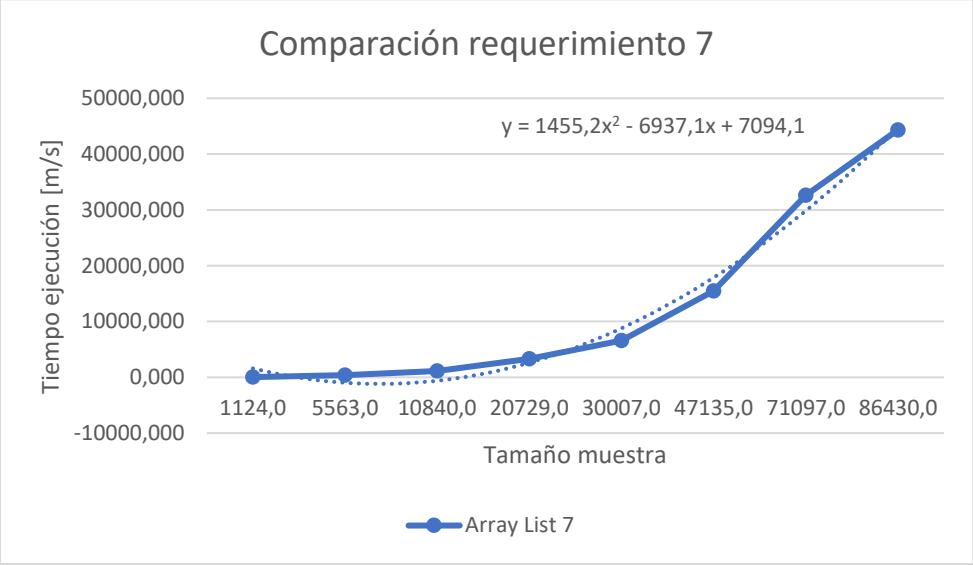
Procesadores	Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz

Memoria RAM	16 GB
Sistema Operativo	Windows 11
Procesadores	Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz

Tablas de datos

MUESTRA	**ARRAY LIST**
SMALL	39,944
5 PCT	398,816
10 PCT	1143,356
20 PCT	3312,205
30 PCT	6558,205
50 PCT	15461,094
80 PCT	32634,299
LARGE	44332,710

Graficas



Análisis

Análisis de resultados de la implementación, tener cuenta las pruebas realizadas y el analisis de complejidad.

Requerimiento Ejemplo

Descripción

```python
def get_data(data_structs, id):
    """
    Retorna un dato a partir de su ID
    """
    pos_data = lt.isPresent(data_structs["data"], id)
    if pos_data > 0:
        data = lt.getElement(data_structs["data"], pos_data)
        return data
    return None
```

Este requerimiento se encarga de retornar un dato de una lista dado su ID. Lo primero que hace es verificar si el elemento existe. Dado el caso que exista, retorna su posición, lo busca en la lista y lo retorna. De lo contrario, retorna None.

Entrada	Estructuras de datos del modelo, ID.
Salidas	El elemento con el ID dado, si no existe se retorna None
Implementado (Sí/No)	Si. Implementado por Juan Andrés Ariza

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Buscar si el elemento existe (isPresent)	O(n)
Obtener el elemento (getElement)	O(1)
TOTAL	***O(n)***

Pruebas Realizadas

Las pruebas realizadas fueron realizadas en una maquina con las siguientes especificaciones. Los datos de entrada fueron el ID 1.

Procesadores	**AMD Ryzen 7 4800HS with Radeon Graphics**
Memoria RAM	8 GB
Sistema Operativo	Windows 10

Entrada	**Tiempo (ms)**
small	0.05
5 pct	0.33
10 pct	1.28
20 pct	2.54
30 pct	4.98
50 pct	7.51
80 pct	13.81
large	25.97

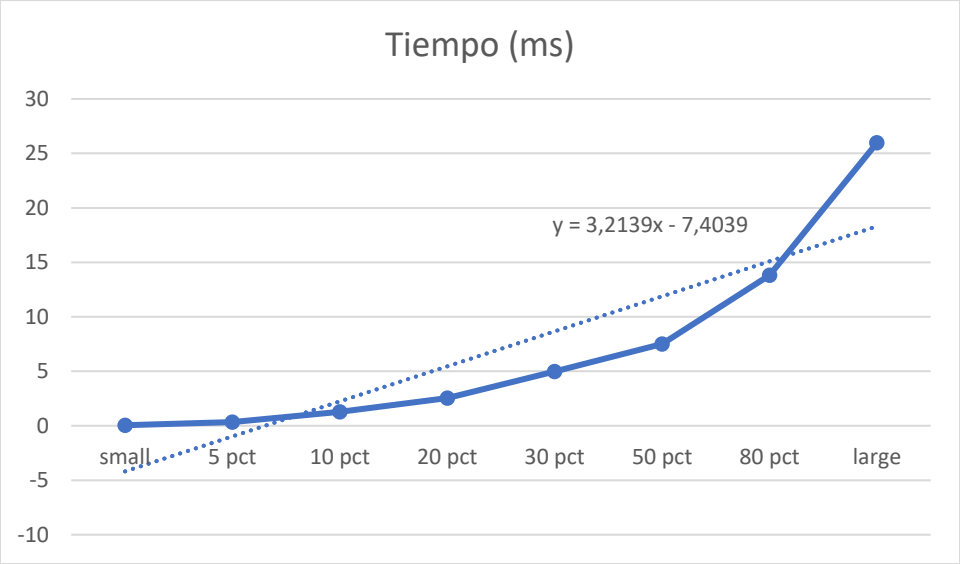
Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

Muestra	Salida	Tiempo (ms)
small	Dato1	0.05
5 pct	Dato2	0.33
10 pct	Dato3	1.28
20 pct	Dato4	2.54
30 pct	Dato5	4.98
50 pct	Dato6	7.51
80 pct	Dato7	13.81
large	Dato8	25.97

Graficas

Las gráficas con la representación de las pruebas realizadas.



Análisis

A pesar de que obtener un elemento en un *ArrayList,* dada su posición, tiene complejidad constante, la implementación de este requerimiento tiene un orden lineal O(n). Esto debido a que, lo primero que se hace es verificar si el elemento hace parte de la lista. Específicamente, a la hora de buscar un elemento en una lista, en el peor de los casos es necesario recorrer toda la lista, es decir, complejidad lineal.

Este comportamiento se puede evidenciar experimentalmente en la gráfica. Ya que, gracias a que los datos no se encuentran tan dispersos con respecto a la línea de tendencia, la curva coincide con el comportamiento lineal esperado.