Universidad de los Andes
Facultad de Ingeniería

Departamento de Ingeniería de Sistemas y Computación
Estructuras de Datos y Algoritmos
ISIS-1225

ABET Engineering Accreditation Commission

ANÁLISIS DEL RETO

Silvana Archila Gómez, 202220350, s.archilag@uniandes.edu.co

Juan Esteban Salamanca, 202221474, je.salamanca@uniandes.edu.co

Pablo Castellanos, 202220548, p.castellanosr@uniandes.edu.co

Requerimiento <<1>>

Descripción

Este algoritmo filtra los datos del archivo csv "results" teniendo en cuenta el equipo dado por parámetro y la condición (local o visitante). Luego escoge únicamente los primeros n partidos cargados, siendo en un número dado por parámetro. La función entrega la cantidad de partidos cargados y luego una tabla con la información de los últimos 6 partidos cargados.

Entrada	data_structs, n(# de partidos que se quieren revisar), equipo, condición (local o visitante).
Salidas	La cantidad de partidos jugados por el equipo escogido en la condición dada. Y una tabla con la información de los últimos 6 partidos procesados.
Implementado (Sí/No)	Se implementó por Silvana Archila

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Iterar por la lista "resultados" (model)	O(N)
Crear dos sublistas cada una con tres elementos (view)	O(N)
Iterar por las sublistas (view)	O(X), siendo X < N
Imprimir la información (view)	O(1)
TOTAL	***O(N)***

Pruebas Realizadas

Procesadores	**11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz, 2419 Mhz, 4 procesadores principales, 8 procesadores lógicos**

Memoria RAM	8 GB
Sistema Operativo	Microsoft Windows 11 Home Single Language

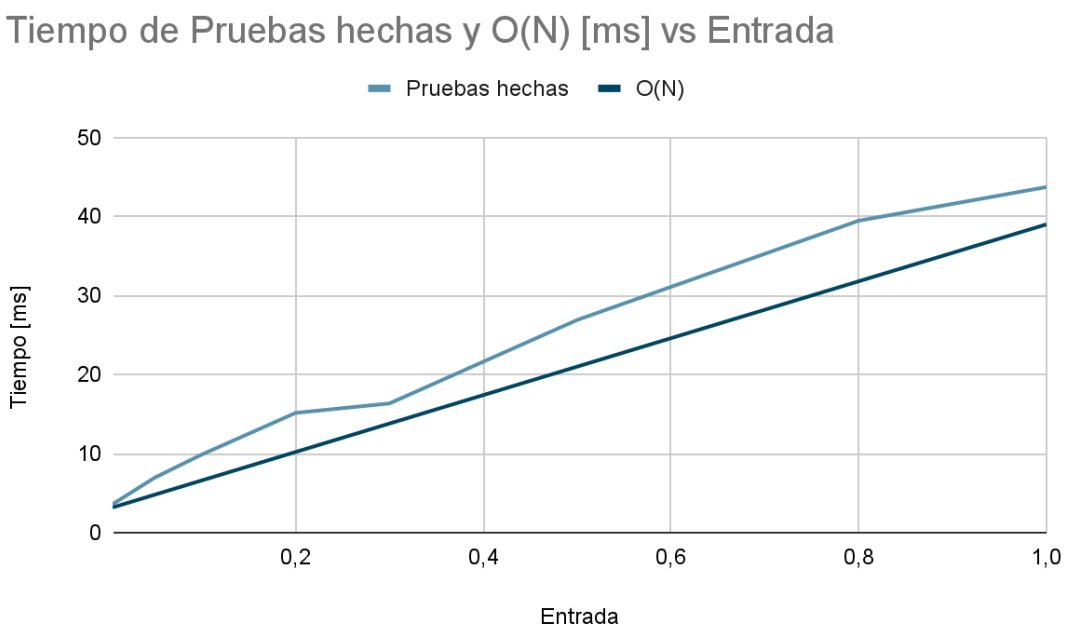
Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

Entrada	Tiempo (ms)
small	3.598
5 pct	6.962
10 pct	9.876
20 pct	15.145
30 pct	16.344
50 pct	26.904
80 pct	39.463
large	43.721

Gráficas

Las gráficas con la representación de las pruebas realizadas.



Análisis

Como se puede ver en la gráfica, el comportamiento de la complejidad temporal del requerimiento 1 es casi lineal, por lo tanto su complejidad es O(N). Este es un buen resultado ya que el tiempo de ejecución es relativamente corto para archivos con una gran cantidad de datos.

Requerimiento <<2>>

Descripción

Para este requerimiento se decidió recorrer el archivo de goalscorers y agregar a una lista auxiliar todos lo goles hechos por el jugador deseado, luego se organiza esta lista del tal forma que queden de primeros los partidos más viejos y de últimos los más modernos, dando de esta forma los últimos N goles de un jugador.

Entrada	data_structcs, jugador(#nombre completo jugador),ng(#número de goles que se desean del jugador)
Salidas	El total de anotaciones de un jugador y una tabla representando los mismos
Implementado (Sí/No)	Se implementó por Pablo Castellanos

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Iterar por la lista “goles” (model)	O(N)
Ordena de fecha menor a fecha mayor (invertir datos) con shell sort	O(N)^1.5
Iterar en el resultado y agrega los 3 primeros resultados y las 3 últimos (view)	O(X), siendo X < N
Imprimir la información (view)	O(1)
TOTAL	***O(N)^1.5***

Pruebas Realizadas

Procesadores	**CORE I5 10Th GEN**
Memoria RAM	8 GB
Sistema Operativo	Windows 11

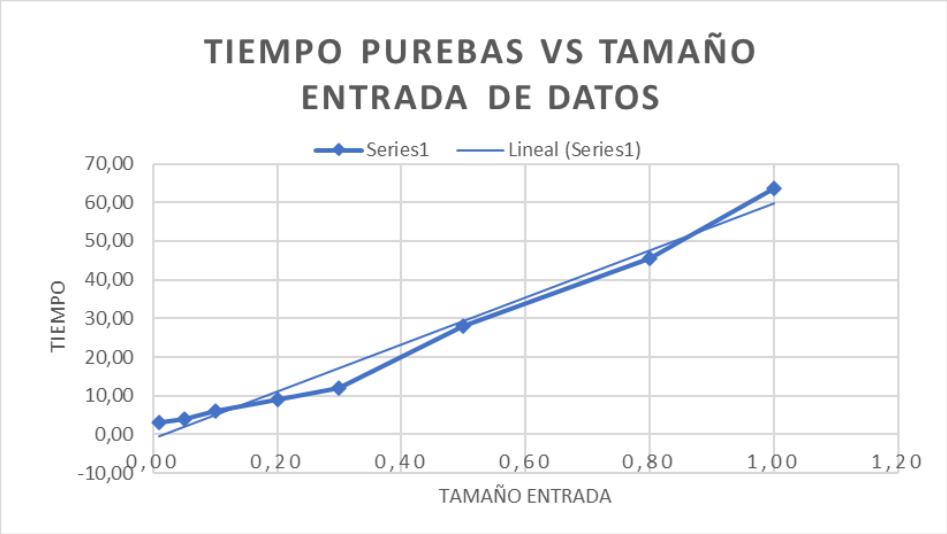
Tablas de datos

Entrada	**Tiempo (ms)**
small	3,021
5 pct	3,927
10 pct	6,115
20 pct	8,799

30 pct	12,002
50 pct	28,145
80 pct	45,506
large	63,638

Gráficas

Las gráficas con la representación de las pruebas realizadas.



Análisis

La función se parece mucho más a una función con complejidad O(N) que una función con complejidad O(N^1.5), esto se debe a que en el shell sort no se está en el peor de los casos y por ende la complejidad de la función baja a un O(N).

Requerimiento <<3>>

Descripción

Para resolver este requerimiento primero se filtra la información de los archivos csv “results” y “goalscorers” para luego filtrarlos por equipo y fecha. Después se organizan ambas listas en una sola y se organiza por fecha. Luego se une la información de “results” y de “goalscorers” que ocurrieron en la misma fecha. En view se organizan las tablas y se cuenta la cantidad de partidos como visitante y como local. Luego se imprime toda la información.

Entrada	data_structs, equipo que se quiere consultar, fecha inicial y fecha final.

Salidas	La cantidad de juegos que jugó el equipo consultado, la cantidad de juegos que jugó el equipo consultado como visitante y la cantidad de juegos que jugó el equipo consultado como local. Además entrega una tabla con la información de los últimos 6 partidos cargados.
Implementado (Sí/No)	Sí se implementó por Silvana Archila

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Iterar sobre la lista de resultados para filtrarla	O(N)
Iterar sobre la lista de goalscorers para filtrarla	O(N)
Iterar sobre ambas listas para agregarlas a una sola lista con toda la información	O(k) donde k < N
Organizar la lista con toda la información usando shell sort	O(NlogN)
Iterar en la lista organizada para unir la información de results y goalscorers.	O(k) donde k < N
TOTAL	***O(NlogN)***

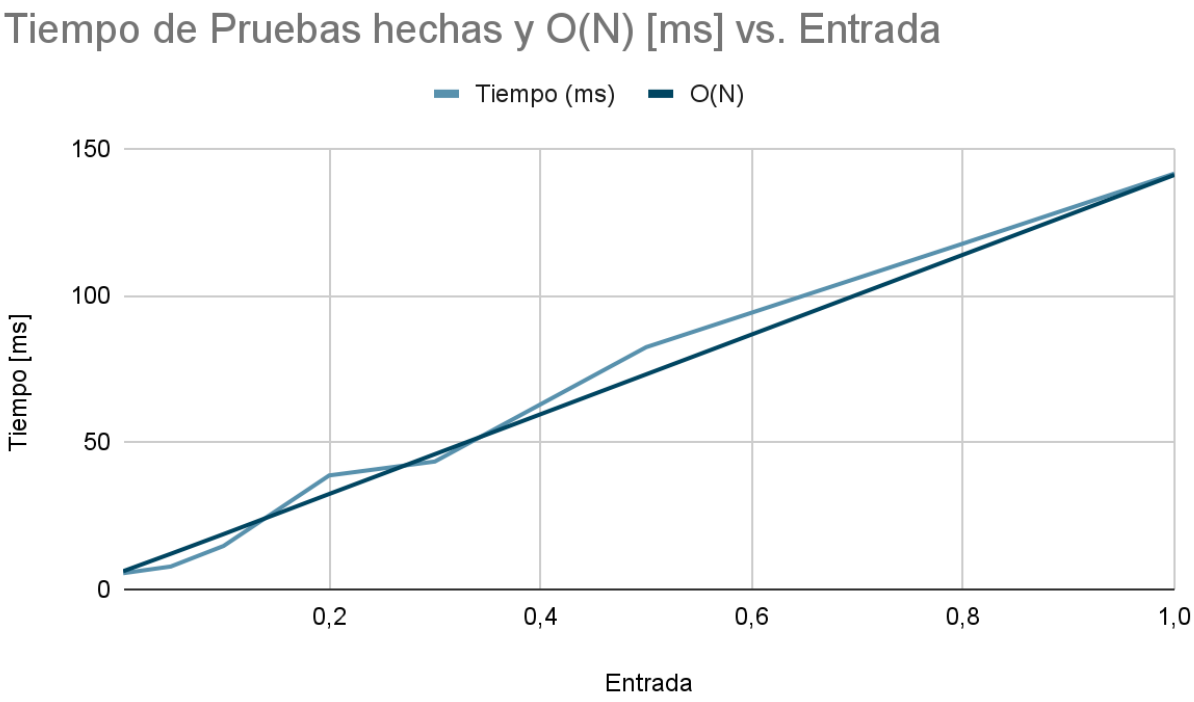
Pruebas Realizadas

Procesadores	**11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz, 2419 Mhz, 4 procesadores principales, 8 procesadores lógicos**
Memoria RAM	8 GB
Sistema Operativo	Microsoft Windows 11 Home Single Language

Tablas de datos

Entrada	**Tiempo (ms)**
small	5.508
5 pct	7.775
10 pct	14.772
20 pct	38.762
30 pct	43.498
50 pct	82.395
80 pct	117.646
large	141.444

Gráficas



Análisis

Como se puede ver en la gráfica, la complejidad temporal del algoritmo es casi perfectamente lineal y, por lo tanto, su complejidad es O(N). Esto es distinto a lo analizado en los pasos y se puede deber a que el shellsort no trabaja en el peor de los casos. Este es un buen resultado ya que, para archivos con muchos datos el algoritmo se demora relativamente poco.

Requerimiento <<4>>

Descripción

Se encarga de buscar los partidos relacionados a un torneo en cierto margen de tiempo. Para ello este requerimiento, filtra y evalúa los datos de los archivos CSV "results" con un bucle "for". Luego, buscamos por cada dato que cumple los requerimientos se busca al ganador de tiros desde el punto penal en los archivos "shootouts". Combinamos esta información en una lista, la ordenamos y mostramos los primeros y últimos 3 elementos si la lista tiene más de 6 elementos."

Entrada	data_strucs, torneo, numero partidos, fecha inicial, fecha final
Salidas	Devuelve el total de ciudades, países, partidos y penales relacionados al torneo, así como una gráfica con los primeros y

	últimos 3 resultados del partido ordenados por fecha(Mayor a menor)
Implementado (Sí/No)	Si se implementó por Pablo Castellanos

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Paso 1 recorrer “results” (Model)	O(N)
Paso 2 recorrer “shootouts” por cada result qué me sirve en búsqueda de un ganador (Model)	O(N*M) siendo M la cantidad de shoutouts
Paso 3 agregar de último toda la información que me sirve a una lista (Model)	O(1)
Paso 4 Sortear la lista del paso 3	O(N)^1.5
Paso 5 iteramos en la lista sorteada y si el tamaño es mayor a 6	O(N)
TOTAL	***O(N*M)***

Pruebas Realizadas

Las pruebas se realizaron en en un computador con las siguientes especificaciones y con los inputs:

Tournament name: Copa América

Start date In YYYY-mm-dd: 1955-06-0

End date In YYYY-mm-dd: 2022-06-30

Procesadores	**CORE I5 10Th GEN**
Memoria RAM	8 GB
Sistema Operativo	Windows 11

Tablas de datos

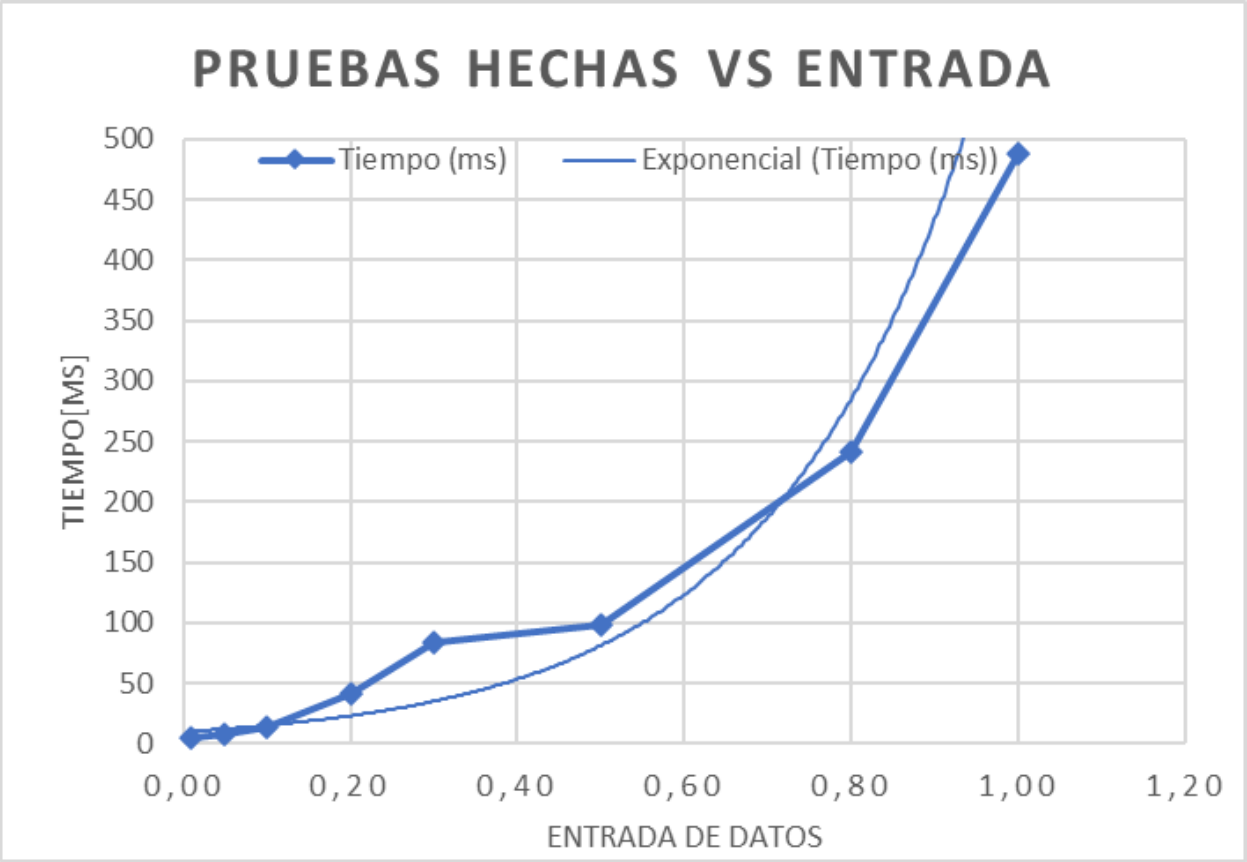
Las tablas con la recopilación de datos de las pruebas.

Entrada	**Tiempo (ms)**
small	5.584
5 pct	7.456
10 pct	13.434

20 pct	41.357
30 pct	83.656
50 pct	98.951
80 pct	241.806
large	488.689

Graficas

Las gráficas con la representación de las pruebas realizadas.



Análisis

Si bien es cierto que la idea de tener un ciclo for dentro de otro ciclo for no es muy conveniente, es pertinente aclarar que para este caso en específico es una opción válida puesto que el valor de M no supera los 600 elementos. Y así mismo solo se entra en este ciclo si el partido en results cumple con los requisitos no obstante este código con un volumen de datos mucho más grande no sería viable puesto que su complejidad sería O(N*M) como se ve en el gráfico.

Requerimiento <<5>>

Descripción

Este requerimiento plantea encontrar las anotaciones de un jugador dado a partir de su nombre y dos fechas. Esto se logra examinando la lista de jugadores que anotaron goles y a medida que van surgiendo coincidencias con el jugador buscado, cada ítem se va añadiendo a una serie de listas que simbolizan las columnas de la tabla. Ya cuando se tienen todas las anotaciones del jugador en dicha lista, se hace un segundo recorrido para encontrar los detalles en la base de datos de partidos con el fin de completar la información en las columnas previamente descritas. Al final, teniendo la información en listas que simbolizan las columnas, estas se cambian a listas nativas de Python con el fin de quedar en un formato en el que se les pueda hacer Tabulate

Entrada	Base de datos “data_structs”, nombre del jugador, fecha final e inicial.
Salidas	tabla con fecha, minuto, equipo local, visitante, equipo al que pertenece el jugador,torneo, si el gol fue de penal o si fue autogol, resultado local y visitante de todos los goles que anota el jugador en el periodo de tiempo establecido.
Implementado (Sí/No)	Si, Juan Esteban Salamanca

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Creación de listas que simbolizan columnas del resultado final	O(1)
Recorrido por “goalscorers” y adición a las listas creadas.	O(n)
Segundo recorrido por el arreglo de resultados que me sirven de “goalscorers” para encontrar su respectivo partido en “results”	O(mxa), donde a es menor a n y m es casi del mismo tamaño que n.
TOTAL	***O(mxa)***

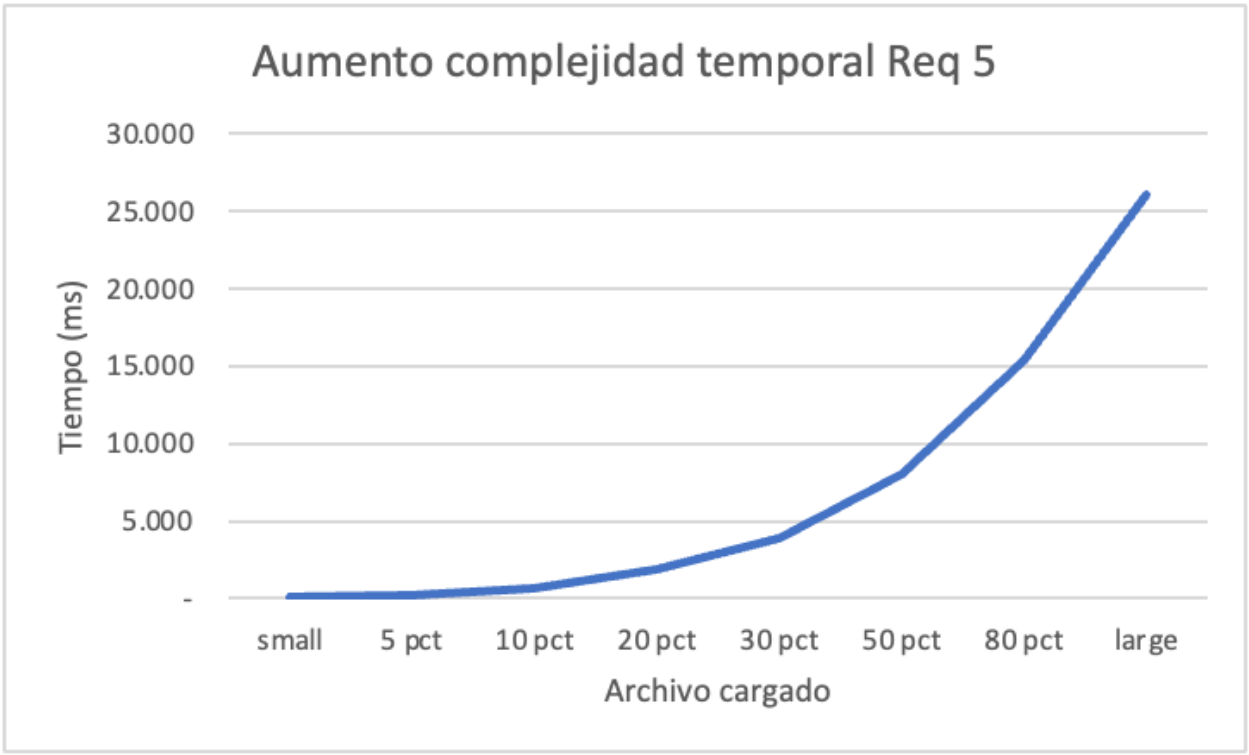
Pruebas Realizadas

Procesadores	**Apple M2 con 8 núcleos de CPU (4 de alto rendimiento y 4 de alta eficiencia), 10 núcleos de GPU, Neural Engine de 16 núcleos y 100 GB/s de ancho de banda de memoria**
Memoria RAM	8 GB
Sistema Operativo	macOS Ventura

Tablas de datos

Entrada	Tiempo (ms)
small	47.238
5 pct	219.704
10 pct	705.627
20 pct	1865.318
30 pct	3923.154
50 pct	8046.316
80 pct	15325.519
large	26039.529

Graficas



Análisis

A través de la gráfica se puede ver la complejidad cuadrática del requerimiento dado el orden de crecimiento mxa que se planteó en la anterior tabla. Esta idea se ve sustentada por el hecho que se recorre una lista m (casi idéntica a n) y se hace un doble recorrido con a (que es una sublista de n).

Requerimiento <<6>>

Busca el total de diferentes aspectos para cada equipo dentro de un torneo entre una fecha deseada. Para ello primero se hace un recorrido por results con el fin de sacar los partidos que cumplen el requisito así como una lista para cada aspecto requerido que no sea la tabla.Luego se corren los resultados por cada equipo con el fin de hallar el total de puntos, partidos, etc.

Descripción

Entrada	data_structs, torneo, eqp(top equipos que se desean ver), fecha inicial y fecha final
Salidas	El total de ciudades, países y equipos relacionados con el torneo, una tabla con los totales de cada equipo en el torneo
Implementado (Sí/No)	Si se implementó por Pablo Castellanos.

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Paso 1 encontrar los equipos dentro del torneo y los resultados que cumplen los requerimientos	O(N)
Paso 2 mirar si equipo, ciudad y país están presentes dentro de sus listas antes creadas y si no están las une con un addLast	O(X) donde X es el tamaño de cada lista donde se guarda solo una vez el valor deseado
Paso 3 encontrar los penales y autogoles dentro de goalscorer	O(N)
Paso 4 unir los penales y los results usando la lista de equipos, y de paso sacar los totales de cada equipo	O(N)
Paso 5 encontrar el mejor scorer de cada equipo	O(N)
Paso 6 unir el tamaño de todas las listas del paso 1 en un diccionario para darlo como return	O(1)
Paso 7 iterar dentro de los resultados para hacer un tabulate	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	**CORE I5 10Th GEN**
Memoria RAM	8 GB
Sistema Operativo	Windows 11

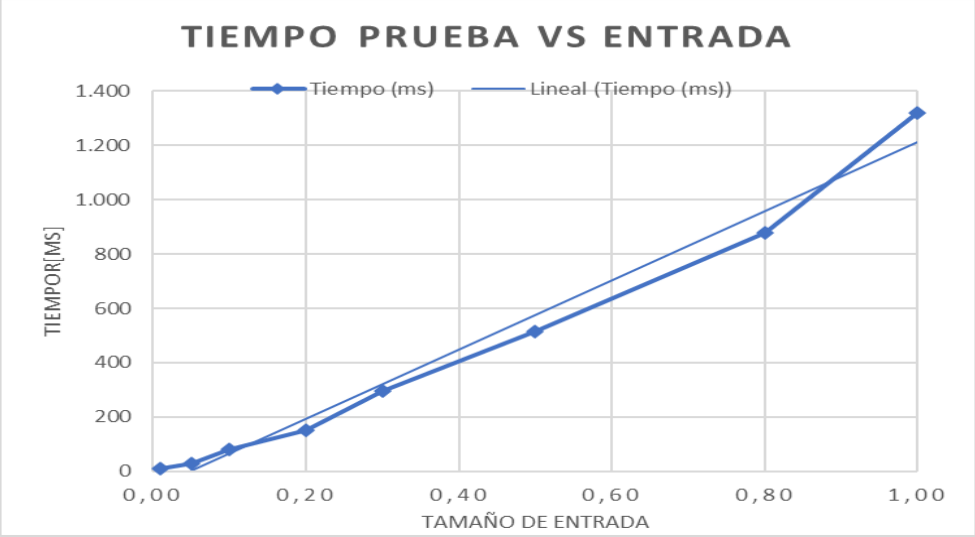
Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

Entrada	Tiempo (ms)
small	8.673
5 pct	26.866
10 pct	81.504
20 pct	150.086
30 pct	294.933
50 pct	513.306
80 pct	878.06
large	1319.879

Graficas

Las gráficas con la representación de las pruebas realizadas.



Análisis

Para la cantidad de datos y la complejidad del requerimiento los resultados de tiempo vistos son muy complacientes puesto que en el caso del archivo large se demoró menos de 2 segundos y se cumple con todo lo requerido

Requerimiento <<7>>

Descripción

Empezando por lo que se pedía, se tenía claro que se necesitaba obtener información de dos partes distintas de la carga de datos y encima se tenían que acceder a ellas de forma simultánea cada vez que se encontrara un jugador que hubiera metido un col. Por esta razón, primero se filtran los datos que se van a usar a lo largo del requerimiento para después recorrer sublistas con datos cuyas fechas concuerdan con las administradas como parámetro. De esta forma, se accede a la información total de cada jugador mediante un doble recorrido para que cada vez que se va encontrando un jugador del que se quiere recolectar la información, se acceda de una vez a la posición del mismo en ambos diccionarios de "data_structs", mientras que se va agregando dicha información a los contadores que se tienen que retornar.

Entrada	Base de datos "data_structs", número de jugadores, fecha final e inicial.
Salidas	Estadísticas generales de la búsqueda global, con partidos totales, goles totales, anotadores totales, torneos totales en donde se anotaron los goles y una lista con los jugadores que tienen los tres mejores y peores puntajes. Estos se obtienen sumando los goles totales que los jugadores anotan y restandoles los autogoles que tuvieron.
Implementado (Sí/No)	Si, Juan Esteban Salamanca

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Filtrar diccionario de resultados, llenando un arraylist con los resultados que estan dentro de las fechas estipuladas por el usuario.	O(n)
Filtrar diccionario de partidos, llenando un arraylist con los partidos que estan dentro de las fechas estipuladas por el usuario.	O(m)
Recorrer jugador por jugador en la lista filtrada de jugadores	O(a), donde a es menor (o igual en el peor caso) a m
Recorrer la lista de partidos para encontrar el partido en el que metió gol el jugador del que se está buscando la información en el recorrido anterior.	Log(b), donde b es menor (o igual en el peor caso) a n
Cuando se llega a la posición del jugador y el partido en el cual metió gol en ambas listas, se crea un arraylist si el jugador no ha sido añadido como parte a la respuesta final, o en caso de que haya sido añadido ya y se tenga un nuevo registro de gol, se actualizan sus datos.	O(1)
TOTAL	***O(aLog(b))***

Pruebas Realizadas

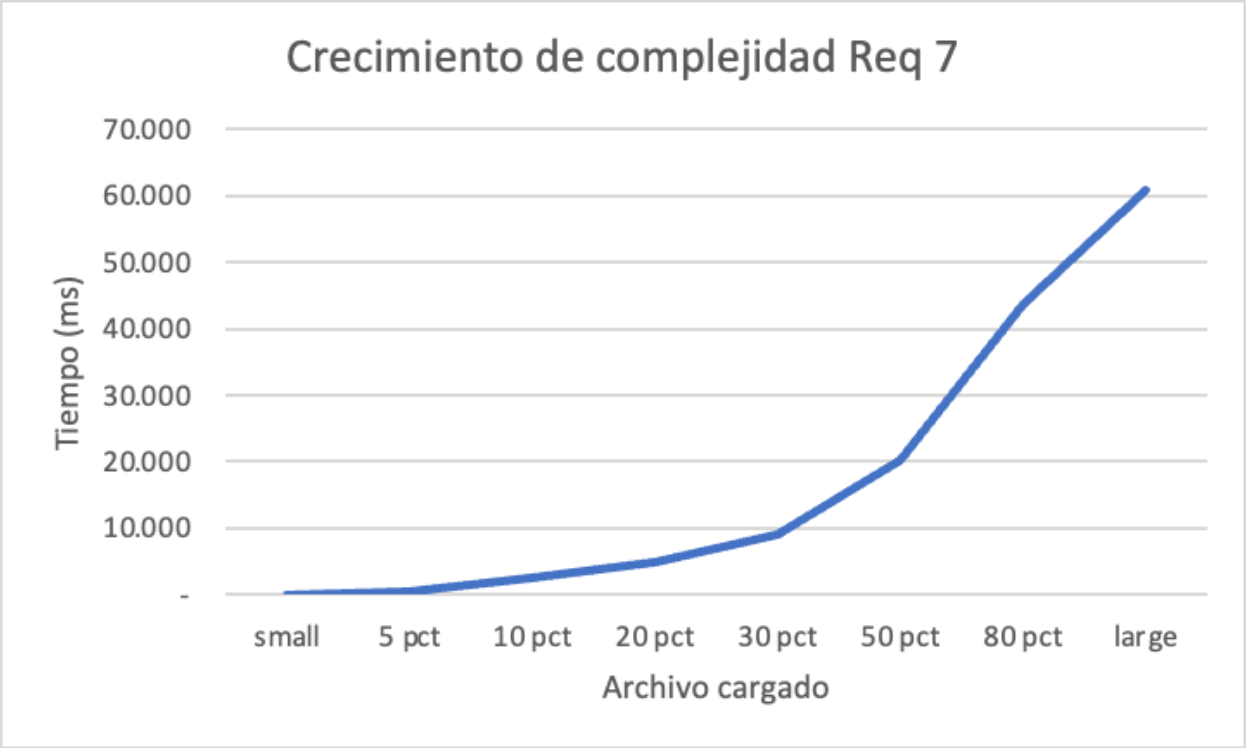
Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

Procesadores	**Apple M2 con 8 núcleos de CPU (4 de alto rendimiento y 4 de alta eficiencia), 10 núcleos de GPU, Neural Engine de 16 núcleos y 100 GB/s de ancho de banda de memoria**
Memoria RAM	8 GB
Sistema Operativo	macOS Ventura

Tablas de datos

Entrada	Tiempo (ms)
small	54.233
5 pct	594.390
10 pct	2677.191
20 pct	4925.952
30 pct	9154.974
50 pct	20351.864
80 pct	43635.118
large	60835.861

Graficas



Análisis

Los resultados que expone la tabla muestran el comportamiento linearítmico que se pretendía con la implementación del requerimiento. Esto se da cuando la magnitud de los datos aumenta significativamente como ocurre con los archivos a partir del 50%. Asimismo, se encuentra bastante eficiencia al tener cantidades pequeñas de datos y se puede concluir que el algoritmo es competente ya que se pusieron fechas que abarcaban casi la totalidad de los datos en lso parámetros de la función, lo que hace que se tuvieran que cubrir casi todos los datos de jugadores originales al iterar sobre “data_structs”.

Requerimiento <<8>>

Descripción

El requerimiento se abordó en dos funciones distintas en model. La primera guarda los datos de results y goalscorers en listas y las filtra por equipo y fecha. Luego se junta la información de ambas listas en una lista más grande y se organiza por fecha. Luego se crea una lista en donde se guarda la cantidad de goles metidos por cada jugador cada año. Después se itera sobre esta lista y se guarda en una nueva lista solamente el jugador que haya metido más goles en el año. Después se vuelve a iterar para encontrar en cuántos partidos el mejor jugador metió goles y se guarda en una nueva lista. Finalmente, se crea otra lista en donde se guarda el minuto promedio en el que el mejor jugador metió goles en un año. Por

último, se organiza toda esta información en una nueva lista que incluye el año, el mejor jugador, cuántos goles metió, en cuántos partidos metió esos goles y el minuto promedio en el que metió los goles. Luego se entrega esto a controller que lo pasa a view. View luego imprime todos los datos organizados. Además, llama a esta función dos veces para que haga el mismo proceso para cada equipo.

La segunda parte se hace en otra función. En esta función se guarda la información de results en una lista y se filtra fecha y por equipos. Luego se itera sobre esta lista filtrada para contar la cantidad de victorias y pérdidas de cada equipo y la cantidad de empates entre los dos. Por último, se carga la información de goalscorers y se guarda en una nueva lista solo la información de goalscorers del último partido cargado antes. Se entrega toda esta información a controller, quien lo entrega a view. En view se organiza la información y se imprime para que el usuario la vea.

Entrada	data_structs, equipo 1, equipo 2, fecha inicial y fecha final
Salidas	El algoritmo tiene dos partes, la primera entrega la diferencia de años entre el primer y último partido cargado, entrega la cantidad de partidos cargados como local y como visitante y una tabla con la información del último partido cargado para cada equipo. Además, entrega una tabla más grande con diferentes estadísticas por año para cada equipo. La segunda parte del algoritmo entrega las estadísticas de los partidos jugados entre los dos equipos dados por parámetro. Esto incluye la cantidad de partidos jugados entre los dos equipos, la cantidad de victorias y pérdidas de ambos equipos y la cantidad de empates. Además entrega una tabla con la información del último partido disputado entre los dos equipos en el rango de fecha dada y la información de cada gol que se marcó en dicho partido.
Implementado (Sí/No)	Sí se implementó por Silvana Archila

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Iterar sobre results para filtrar la información	O(N)
Iterar sobre goalscorers para filtrar la información	O(N)
Guardar toda la información en una lista y organizarla usando shell	O(N^1.5)
Buscar el mejor jugador	O(N)
Contar la cantidad de partidos y el promedio de minuto	O(N)
Guardar toda la información cargada en una lista usando addLast()	O(1)
Iterar sobre la lista de toda la información organizada.y buscar el último elemento usando lastELement() para editarlo.	O(N)
TOTAL	***O(N^1.5)***

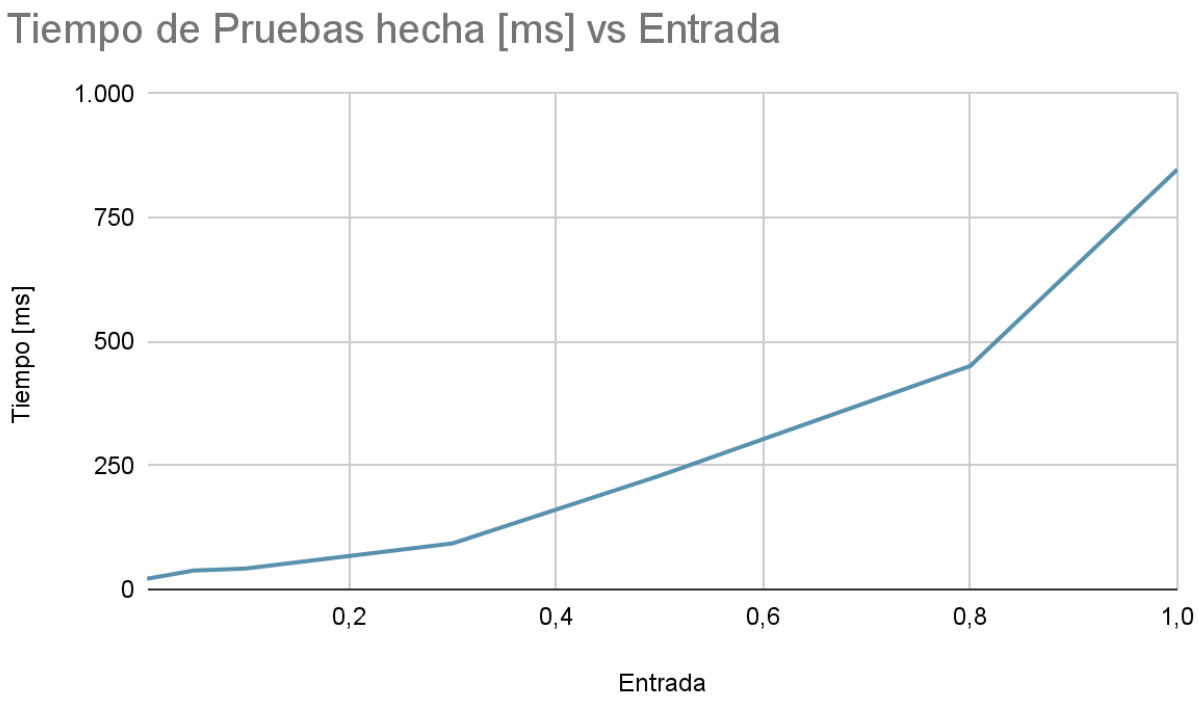
Pruebas Realizadas

Procesadores	**11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz, 2419 Mhz, 4 procesadores principales, 8 procesadores lógicos**
Memoria RAM	8 GB
Sistema Operativo	Microsoft Windows 11 Home Single Language

Tablas de datos

Entrada	Tiempo (ms)
small	21.448
5 pct	37.992
10 pct	42.005
20 pct	67.301
30 pct	92.687
50 pct	229.260
80 pct	450.765
large	847.197

Graficas



Análisis

En la gráfica podemos ver que la complejidad temporal es un poco mayor a O(N) como se analizó en los pasos, es más cercano a O(N^1.5). Este es un buen resultado ya que el algoritmo debe hacer muchas cosas y el tiempo es relativamente corto para archivos grandes.