**ANÁLISIS DEL RETO**

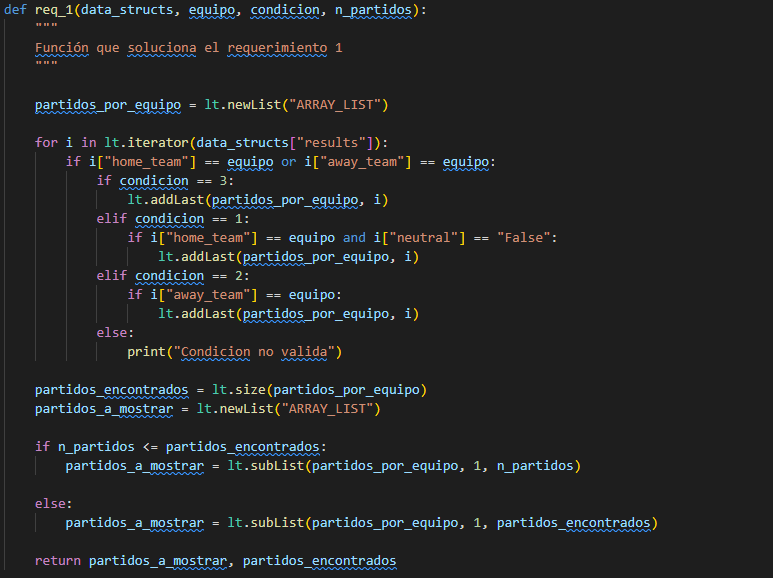
Andrés Leonardo Beltrán Contreras, 202014143, al.beltran

Jose Alejandro Rojas Zapata, 202313911, j.rojasz

Tomás Acosta Romero, 202310218, t.acostar

# **Requerimiento 1**

## **Descripción**



Este requerimento se encarga de retornar los partidos añadidos al array, en primer lugar crea la lista y luego itera sobre la estructura de datos y hace comparaciones para verificar que el dato coincida con las carrácteristicas ingresadas por parametro, en este caso el equipo, si es local, visitante o indiferente y finalmente buscar el numero de partidos que se jugaron. Para luego añadirlos a un nuevo array con los partidos disputados.

|  |  |
| --- | --- |
| **Entrada** | Data\_structs, equipo, condicion (home, away, indfirente), numero de partidos |
| **Salidas** | Los partidos que jugó un equipo según la condición (principal)  Datos del equipo (fecha, condicion de los equipos que disputaron, lugar (país), ciudad, marcador (local y visitante) |
| **Implementado (Sí/No)** | Todo el grupo :D |

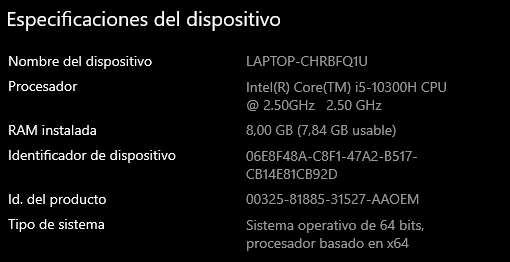
## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| 1.Creación de lista | O(1) |
| 2.Bucle for | O(n) |
| 3.Comparaciones y/o condicionales | O(1) |
| 4.Addlast | O(1) |
| 5. Creacion lista (partidos\_a\_mostrar) | 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).



|  |  |
| --- | --- |
| **Entrada** | **Tiempo (s)** |
| Small (país = perú) (n\_partidos=3) | 1.16001 ms |
| 5pct (pais = perú) (n\_partidos = 5) | 2.30410 ms |
| 10pct (pais = perú) (n\_partidos = 10) | 5.1573 ms |
| 20pct (pais = perú) (n\_partidos = 20) | 7.3402 ms |
| 30pct (pais = perú) (n\_partidos = 30) | 10.288 ms |
| 50pct (pais = perú) (n\_partidos = 50) | 17.025 ms |
| 80pct (pais = perú) (n\_partidos = 80) | 27.311 ms |
| Large (pais = perú) (n\_partidos = 100) | 27.850 ms |

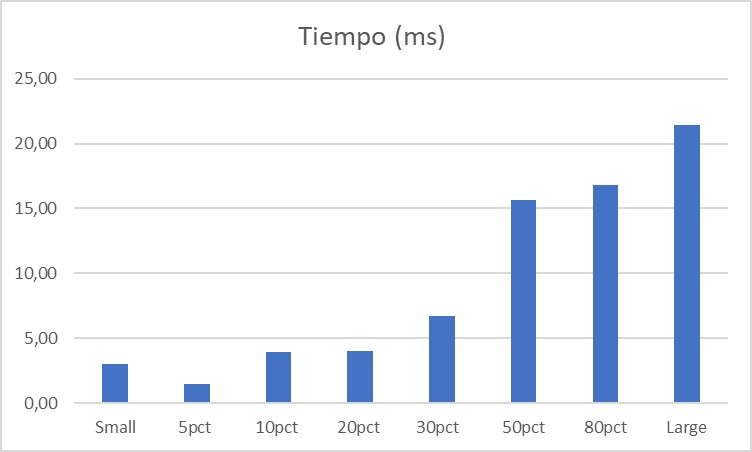
### **Tablas de datos**

Las tablas con la recopilación de datos de las pruebas.

|  |  |
| --- | --- |
| **Entrada** | **Tiempo (s)** |
| Small (país = perú) (n\_partidos=3) | 1.16001 ms |
| 5pct (pais = perú) (n\_partidos = 5) | 2.30410 ms |
| 10pct (pais = perú) (n\_partidos = 10) | 5.1573 ms |
| 20pct (pais = perú) (n\_partidos = 20) | 7.3402 ms |
| 30pct (pais = perú) (n\_partidos = 30) | 10.288 ms |
| 50pct (pais = perú) (n\_partidos = 50) | 17.025 ms |
| 80pct (pais = perú) (n\_partidos = 80) | 27.311 ms |
| Large (pais = perú) (n\_partidos = 100) | 27.850 ms |

### **Graficas**

Las gráficas con la representación de las pruebas realizadas.



## **Análisis**

A pesar de que obtener un elemento en una lista enlazada, dada su posición, generalmente tiene una complejidad constante (O(1)), la implementación de este requerimiento tiene un orden lineal O(m). Esto se debe a que, en el peor de los casos, la función debe recorrer toda la lista de resultados para verificar si cada partido cumple con las condiciones dadas. La iteración a través de todos los resultados es lo que domina la complejidad temporal.

# **Requerimiento 2**

## **Descripción**



Este requerimento se encarga de retornar los goles anotados por un jugador y las carácteristicas atribuidas al gol. Debido a nuestra carga de datos creamos un stack para hacer uso del LIFO, en el stack se añaden los goles marcados por el jugador ingresado por parámetro y luego ingresan al array que se retorna.

|  |  |
| --- | --- |
| **Entrada** | Data\_estructures, numero de goles a buscar, jugador |
| **Salidas** | Lista con los goles anotados por el jugador entrado por parametro y sus condiciones (fecha, lugar, home team o away, equipo, minuto de gol y tipo de gol |
| **Implementado (Sí/No)** | Todo el grupo :D |

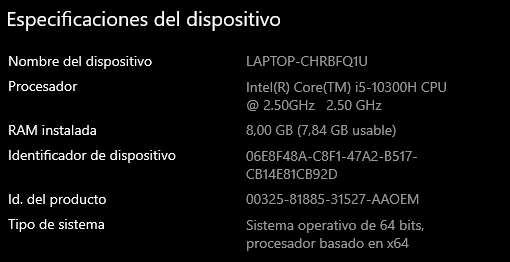
## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Creación de stack | O(1) |
| Recorrido de la estructura de datos | O(n) |
| Comparacion (i[scorer] == “jugador”) | O(1) |
| Push | O(1) |
| Creacion de lista (goles\_a\_mostrar) | O(1) |
| Comparacion | 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).



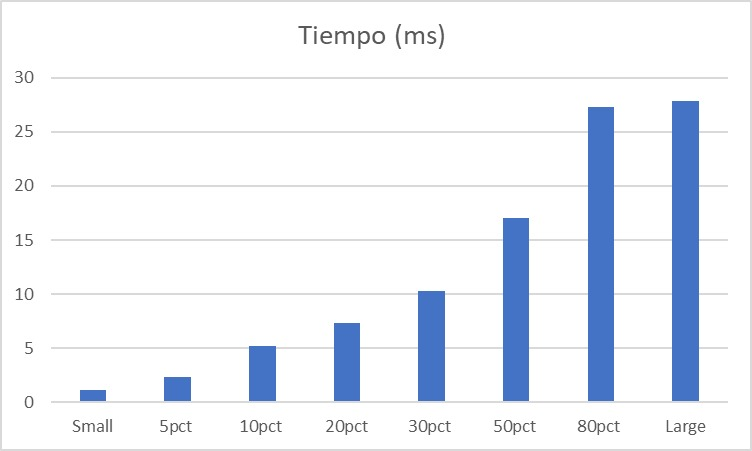
|  |  |
| --- | --- |
| **Entrada** | **Tiempo (s)** |
| Small (jugador = Carlos Scaron)  (n\_goles=3) | 3.0213 ms |
| 5pct (jugador = Carlos Scarone) (n\_goles=5) | 1.4958 ms |
| 10pct (jugador = Cristiano Ronaldo) (n\_goles=10) | 3.9 ms |
| 20pct (jugador = Cristiano Ronaldo) (n\_goles=20) | 4.0391 ms |
| 30pct (jugador = Cristiano Ronaldo) (n\_goles=30) | 6.7375 ms |
| 50pct (jugador = Crisitano Ronaldo) (n\_goles=50) | 15.665 ms |
| 80pct (jugador = Cristiano Ronaldo) (n\_goles=80) | 16.777 ms |
| Large (jugador = Crisitano Ronaldo) (n\_goles=100) | 21.408 ms |

### **Tablas de datos**

|  |  |
| --- | --- |
| **Entrada** | **Tiempo (s)** |
| Small (jugador = Carlos Scaron)  (n\_goles=3) | 3.0213 ms |
| 5pct (jugador = Carlos Scarone) (n\_goles=5) | 1.4958 ms |
| 10pct (jugador = Cristiano Ronaldo) (n\_goles=10) | 3.9 ms |
| 20pct (jugador = Cristiano Ronaldo) (n\_goles=20) | 4.0391 ms |
| 30pct (jugador = Cristiano Ronaldo) (n\_goles=30) | 6.7375 ms |
| 50pct (jugador = Crisitano Ronaldo) (n\_goles=50) | 15.665 ms |
| 80pct (jugador = Cristiano Ronaldo) (n\_goles=80) | 16.777 ms |
| Large (jugador = Crisitano Ronaldo) (n\_goles=100) | 21.408 ms |

### **Graficas**

Las gráficas con la representación de las pruebas realizadas.

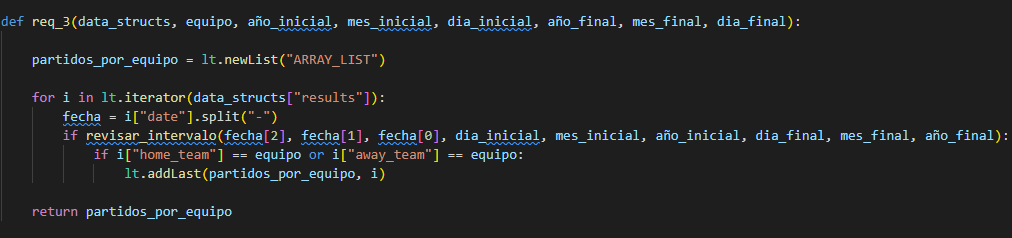


## **Análisis**

A pesar de que obtener un elemento en una pila, dada su posición, generalmente tiene una complejidad constante (O(1)), la implementación de este requerimiento tiene un orden lineal O(n). Esto se debe a que, en el peor de los casos, la función debe recorrer toda la lista de goles marcados para encontrar los goles del jugador especificado. La iteración a través de todos los goles marcados es lo que domina la complejidad temporal.

# **Requerimiento 3**

## **Descripción**



Este requerimento se encarga de retornas los partidos disputados por un equipo en un intérvalo de fechas, primero se crea un arraylist donde se van a almacenar los datos que concuerden con el intervalo y el nombre del equipo. Tomás creó funciones auxiliares para comparar las fechas de manera eficiente, luego de llamar esa función, se agrega el numero de partidos que jugó el equipo en el array partidos\_por\_equipo.

|  |  |
| --- | --- |
| **Entrada** | Data\_structs, fecha (descompuesta) y funciones auxiliares |
| **Salidas** | Partidos jugados por equipos en el intervalo mencionado |
| **Implementado (Sí/No)** | Tomás Acosta |

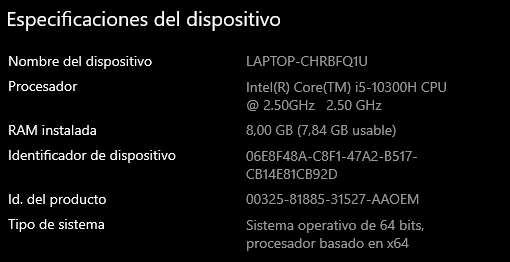
## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Creación de lista | O(1) |
| Recorrido de la estructura de datos | O(n) |
| Llamado a revisar intervalo | O(1) |
| Condición (i[“home\_team”]) o (i[“away\_team”]) | O(1) |
| addlast | 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).



|  |  |
| --- | --- |
| **Entrada** | **Tiempo (s)** |
| Small (fecha1 (01/01/00) fecha2 (30/12/2003) (pais = perú) | 0.1300 ms |
| 5pct (fecha1 (01/01/00) fecha2 (30/12/2005) | 8.9 ms |
| 10pct (fecha1 (01/01/00) fecha2 (30/12/2010) (pais = peru) | 14.055 ms |
| 20pct (fecha1 (01/01/00) fecha2 (30/12/2013) (pais = peru) | 30.9 ms |
| 30pct (fecha1 (01/01/00) fecha2 (30/12/2016) (pais = peru) | 38.902 ms |
| 50pct (fecha1 (01/01/00) fecha2 (30/12/2018) (pais = peru) | 59.211 ms |
| 80pct (fecha1 (01/01/00) fecha2 (30/12/2020)  (pais = peru) | 80.619 ms |
| Large (fecha1 (01/01/00) fecha2 (30/12/2022)  (pais = peru) | 105.5 ms |

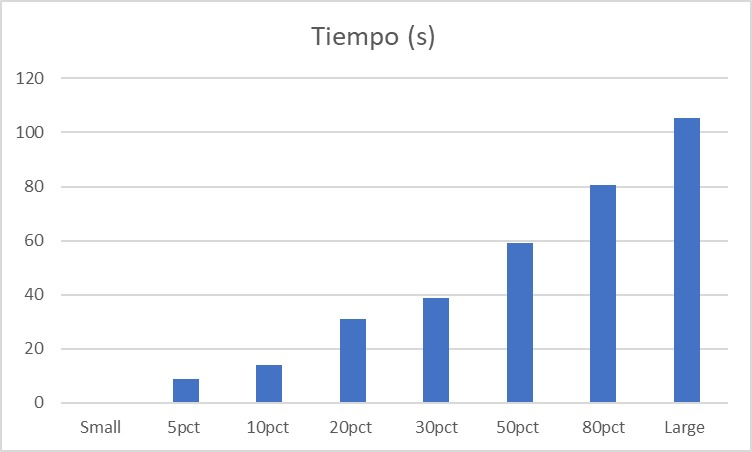
### **Tablas de datos**

Las tablas con la recopilación de datos de las pruebas.

|  |  |
| --- | --- |
| **Entrada** | **Tiempo (s)** |
| Small (fecha1 (01/01/00) fecha2 (30/12/2003) (pais = perú) | 0.1300 ms |
| 5pct (fecha1 (01/01/00) fecha2 (30/12/2005) | 8.9 ms |
| 10pct (fecha1 (01/01/00) fecha2 (30/12/2010) (pais = peru) | 14.055 ms |
| 20pct (fecha1 (01/01/00) fecha2 (30/12/2013) (pais = peru) | 30.9 ms |
| 30pct (fecha1 (01/01/00) fecha2 (30/12/2016) (pais = peru) | 38.902 ms |
| 50pct (fecha1 (01/01/00) fecha2 (30/12/2018) (pais = peru) | 59.211 ms |
| 80pct (fecha1 (01/01/00) fecha2 (30/12/2020)  (pais = peru) | 80.619 ms |
| Large (fecha1 (01/01/00) fecha2 (30/12/2022)  (pais = peru) | 105.5 ms |

### **Graficas**

Las gráficas con la representación de las pruebas realizadas.

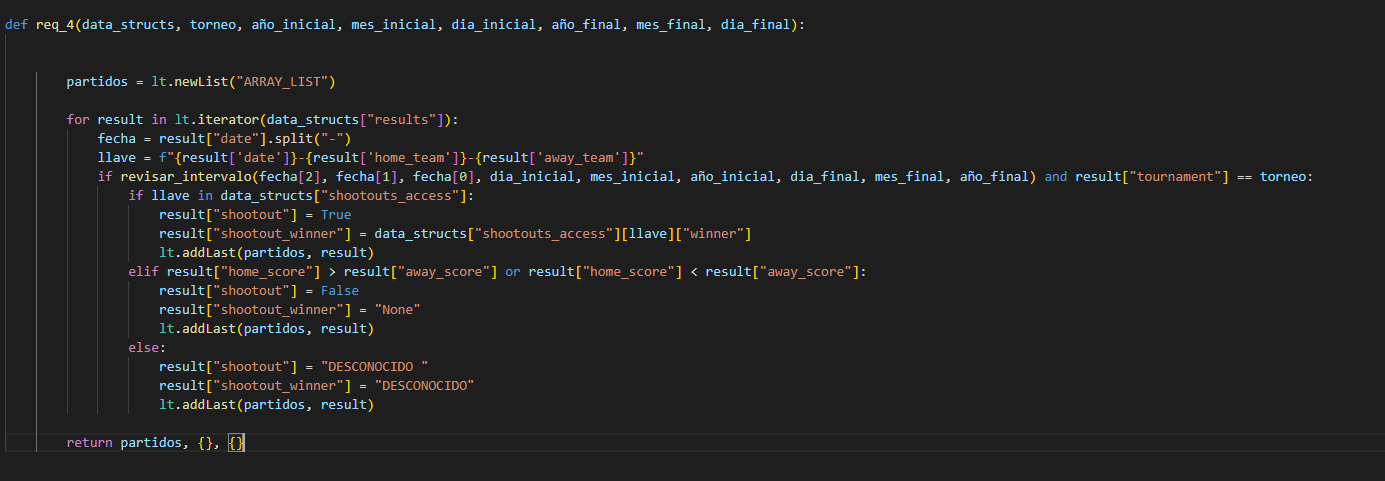


## **Análisis**

A pesar de que obtener un elemento en una lista enlazada, dada su posición, generalmente tiene una complejidad constante (O(1)), la implementación de este requerimiento tiene un orden lineal O(m). Esto se debe a que, en el peor de los casos, la función debe recorrer toda la lista de resultados para verificar si cada partido cumple con las condiciones dadas, y la iteración a través de todos los resultados domina la complejidad temporal.

# **Requerimiento 4**

# **Descripción**



Este requerimento se encarga de retornar los partidos que disputo un equipo en un torneo en específico, al mismo tiempo que las características de los resultados de los partidos, la más importante (si habían shootouts y quien los ganó) para ello se crea un diccionario de shootouts, luego de iterar sobre la estructura de datos, se almacenan los shootouts y las caracteristicas de los equipos que los jugaron como llave del diccionario. Luego se crea un array, luego de iterar sobre los resultados de los partidos, revisar los intervalos de fechas con las funciones auxiliarias implementadas, comparan las caracteristicas de los equipos en una fecha especifica con las almacenadas en el diccionario, en caso de coincidir se añade el resultado de los shootouts al array implementado, en caso de no coincidir, se dejan los resultados sin shoouts, y si no hubo shootouts ni winner el estado del partido es desconocido.

Al final se retornan los partidos, y los shootouts en caso de tener.

|  |  |
| --- | --- |
| **Entrada** | Data\_structs, fecha (descompuesta), torneo especifico |
| **Salidas** | Los partidos que jugó un equipo según la condición (principal)  Datos del equipo (fecha, condicion de los equipos que disputaron, lugar (país), ciudad, marcador (local y visitante) |
| **Implementado (Sí/No)** | Jose Rojas |

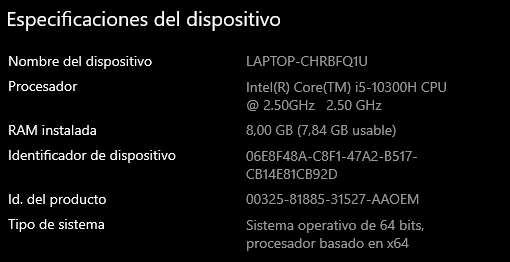
## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Creación de lista | O(1) |
| Recorrido de la estructura de datos | O(n) |
| Creación de la fecha | O(1) |
| Creación de llave | O(1) |
| Condicionales | O(1) |
| Retorno | 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).



|  |  |
| --- | --- |
| **Entrada** | **Tiempo (s)** |
| Small (fecha1 (01/01/00) fecha2 (30/12/2003) (torneo = FIFA World Cup) | 2.019 ms |
| 5pct (fecha1 (01/01/00) fecha2 (30/12/2005) (torneo = FIFA World Cup) | 9.573 ms |
| 10pct (fecha1 (01/01/00) fecha2 (30/12/2010) (torneo = FIFA World Cup) | 19.570 ms |
| 20pct (fecha1 (01/01/00) fecha2 (30/12/2013) (torneo = FIFA World Cup) | 32.3 ms |
| 30pct (fecha1 (01/01/00) fecha2 (30/12/2016) (torneo = FIFA World Cup) | 40.456 ms |
| 50pct (fecha1 (01/01/00) fecha2 (30/12/2018) (torneo = FIFA World Cup) | 64.341 ms |
| 80pct (fecha1 (01/01/00) fecha2 (30/12/2020)  (torneo = FIFA World Cup) | 85.9 ms |
| Large (fecha1 (01/01/00) fecha2 (30/12/2022)  (torneo = FIFA World Cup) | 105.9 ms |

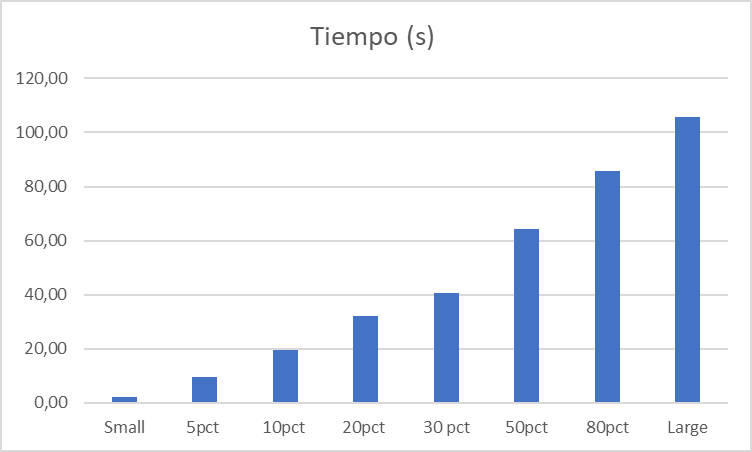
### **Tablas de datos**

Las tablas con la recopilación de datos de las pruebas.

|  |  |
| --- | --- |
| **Entrada** | **Tiempo (s)** |
| Small (fecha1 (01/01/00) fecha2 (30/12/2003) (torneo = FIFA World Cup) | 2.019 ms |
| 5pct (fecha1 (01/01/00) fecha2 (30/12/2005) (torneo = FIFA World Cup) | 9.573 ms |
| 10pct (fecha1 (01/01/00) fecha2 (30/12/2010) (torneo = FIFA World Cup) | 19.570 ms |
| 20pct (fecha1 (01/01/00) fecha2 (30/12/2013) (torneo = FIFA World Cup) | 32.3 ms |
| 30pct (fecha1 (01/01/00) fecha2 (30/12/2016) (torneo = FIFA World Cup) | 40.456 ms |
| 50pct (fecha1 (01/01/00) fecha2 (30/12/2018) (torneo = FIFA World Cup) | 64.341 ms |
| 80pct (fecha1 (01/01/00) fecha2 (30/12/2020)  (torneo = FIFA World Cup) | 85.9 ms |
| Large (fecha1 (01/01/00) fecha2 (30/12/2022)  (torneo = FIFA World Cup) | 105.9 ms |

### **Graficas**

Las gráficas con la representación de las pruebas realizadas.

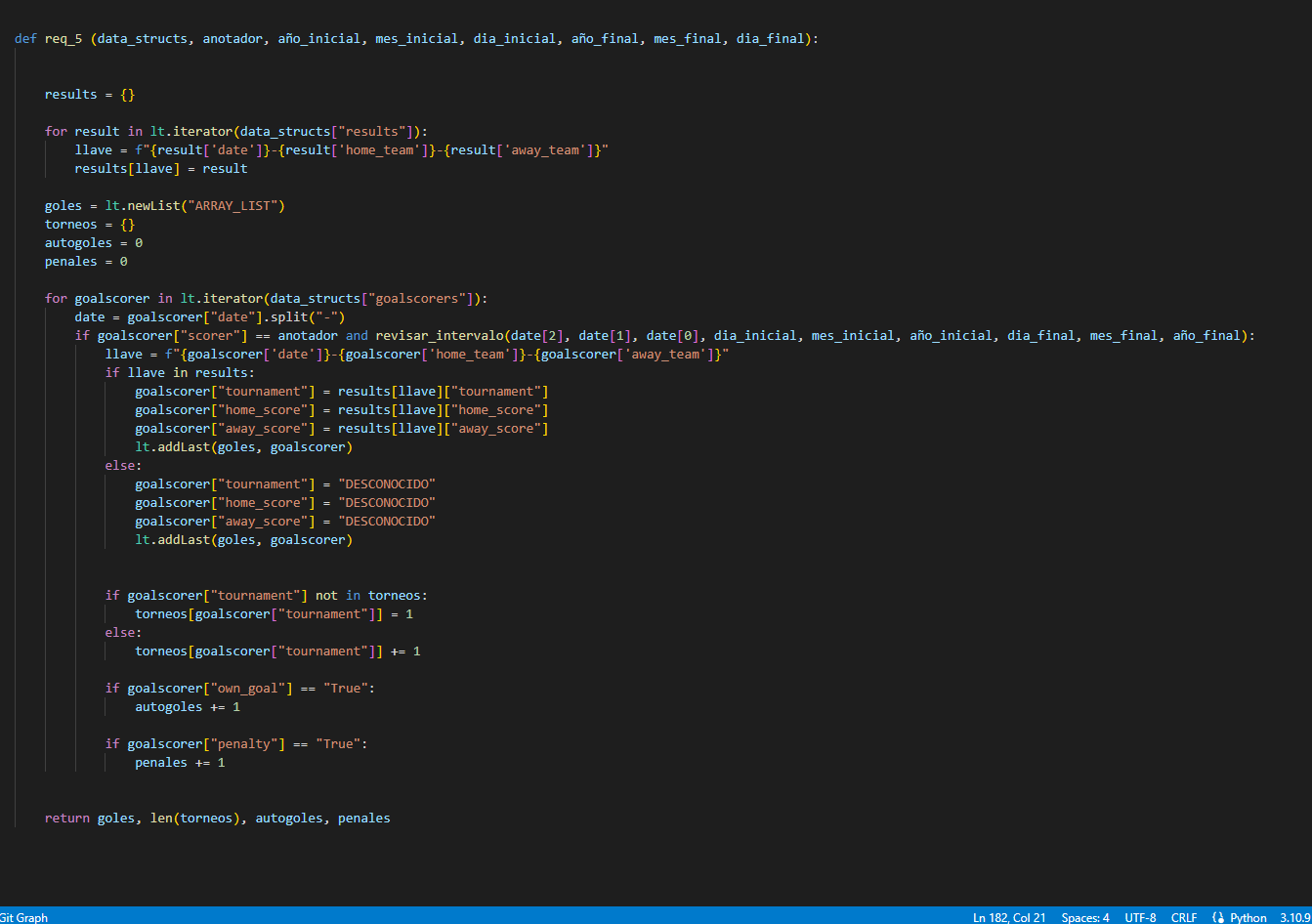


## **Análisis**

A pesar de que obtener un elemento en una lista enlazada, dada su posición, generalmente tiene una complejidad constante (O(1)), la implementación de este requerimiento tiene un orden lineal O(m). Esto se debe a que, al buscar partidos que cumplan con ciertas condiciones en la lista de resultados, lo primero que se hace es verificar si el partido pertenece a la lista. En el peor de los casos, la función debe recorrer toda la lista de resultados, lo que resulta en una complejidad lineal.

# **Requerimiento 5**

## **Descripción**



Este requerimento se encarga de de retornar los goles marcados por un jugador, primero se crea un diccionario, se iteran los resultados de la estructura de datos y se genera una llave basada en la fecha y en si el equipo es home o away y se almacena en el diccionario. Se crea una lista de goles y un diccionario llamado torneos al mismo tiempo que se generan dos contadores, se iteran los datos de los goalscorers y se revisa si la fecha esta en el intervalo con las funciones auxiliares encargadas de revisar los intervalos, se genera una llave igual a la del primer diccionario pero en este caso para actualizar el diccionario de torneos, para con el verificar si los goles son penales o autogoles.

|  |  |
| --- | --- |
| **Entrada** | Data\_structs, anotador, fecha (descompuesta) |
| **Salidas** | Los partidos que jugó un equipo según la condición (principal) |
| **Implementado (Sí/No)** | Leonardo Beltrán |

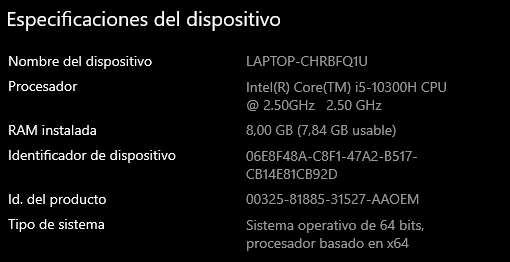
## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Creación de diccionario | O(1) |
| Recorrido de la estructura de datos | O(1) |
| Asignanción de variable (futura llave diccionario) | O(1) |
| Creación de lista | O(1) |
| Recorrido de estructura de datos | O(n) |
| Condicionales | O(1) |
| Retorno | 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).



|  |  |
| --- | --- |
| **Entrada** | **Tiempo (s)** |
| Small (fecha1 (01/01/00) fecha2 (30/12/2003) (jugador = Cristiano Ronaldo) | 0,732 ms |
| 5pct (fecha1 (01/01/00) fecha2 (30/12/2005) (jugador = Cristiano Ronaldo) | 7.6207 ms |
| 10pct (fecha1 (01/01/00) fecha2 (30/12/2010) (jugador = Cristiano Ronaldo) | 9,7051 ms |
| 20pct (fecha1 (01/01/00) fecha2 (30/12/2013) (jugador = Cristiano Ronaldo) | 9.6206 m |
| 30pct (fecha1 (01/01/00) fecha2 (30/12/2016) (jugador = Cristiano Ronaldo) | 18,847 ms |
| 50pct (fecha1 (01/01/00) fecha2 (30/12/2018) (jugador = Cristiano Ronaldo) | 29,808 ms |
| 80pct (fecha1 (01/01/00) fecha2 (30/12/2020)  (jugador = Cristiano Ronaldo) | 69.9 ms |
| Large (fecha1 (01/01/00) fecha2 (30/12/2022)  (jugador = Cristiano Ronaldo) | 82.660 ms |

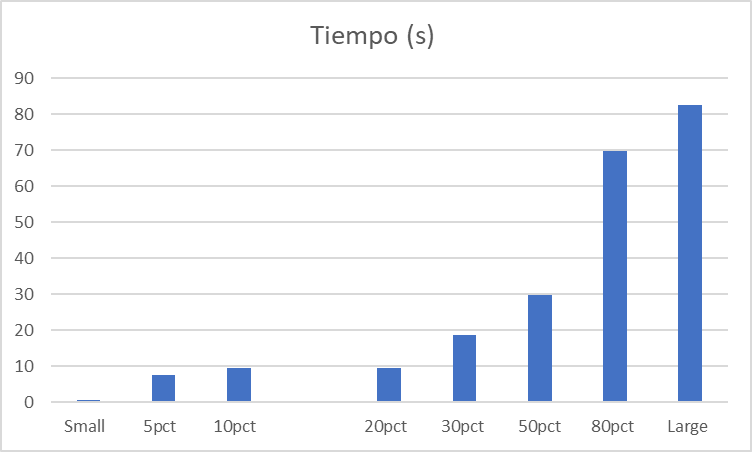
### **Tablas de datos**

Las tablas con la recopilación de datos de las pruebas.

|  |  |
| --- | --- |
| **Entrada** | **Tiempo (s)** |
| Small (fecha1 (01/01/00) fecha2 (30/12/2003) (jugador = Cristiano Ronaldo) | 0,732 ms |
| 5pct (fecha1 (01/01/00) fecha2 (30/12/2005) (jugador = Cristiano Ronaldo) | 7.6207 ms |
| 10pct (fecha1 (01/01/00) fecha2 (30/12/2010) (jugador = Cristiano Ronaldo) | 9,7051 ms |
| 20pct (fecha1 (01/01/00) fecha2 (30/12/2013) (jugador = Cristiano Ronaldo) | 9.6206 m |
| 30pct (fecha1 (01/01/00) fecha2 (30/12/2016) (jugador = Cristiano Ronaldo) | 18,847 ms |
| 50pct (fecha1 (01/01/00) fecha2 (30/12/2018) (jugador = Cristiano Ronaldo) | 29,808 ms |
| 80pct (fecha1 (01/01/00) fecha2 (30/12/2020)  (jugador = Cristiano Ronaldo) | 69.9 ms |
| Large (fecha1 (01/01/00) fecha2 (30/12/2022)  (jugador = Cristiano Ronaldo) | 82.660 ms |

### **Graficas**

Las gráficas con la representación de las pruebas realizadas.



## **Análisis**

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

# **Requerimiento 6**

## **Descripción**

Breve descripción de como abordaron la implementación de

|  |  |
| --- | --- |
| **Entrada** | Data\_structs, equipo, condicion (home, away, indfirente), numero de partidos |
| **Salidas** | Los partidos que jugó un equipo según la condición (principal)  Datos del equipo (fecha, condicion de los equipos que disputaron, lugar (país), ciudad, marcador (local y visitante) |
| **Implementado (Sí/No)** | Todo el grupo :D |

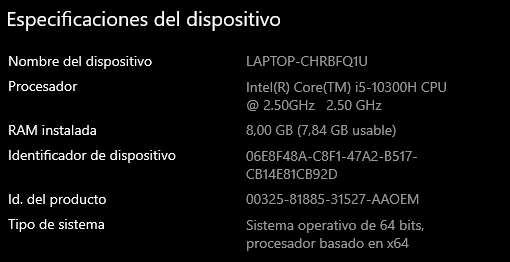
## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Creación de lista | O(...) |
| Recorrido de la estructura de datos | O(...) |
|  | O(...) |
| ***TOTAL*** | ***O(...)*** |

## **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 |  |
| 5pct |  |
| 10pct |  |
| 20pct |  |
| 30pct |  |
| 50pct |  |
| 80pct |  |
| large |  |

### **Tablas de datos**

Las tablas con la recopilación de datos de las pruebas.

### **Graficas**

Las gráficas con la representación de las pruebas realizadas.

## **Análisis**

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

# **Requerimiento 7**

## **Descripción**

Breve descripción de como abordaron la implementación de

|  |  |
| --- | --- |
| **Entrada** | Data\_structs, equipo, condicion (home, away, indfirente), numero de partidos |
| **Salidas** | Los partidos que jugó un equipo según la condición (principal)  Datos del equipo (fecha, condicion de los equipos que disputaron, lugar (país), ciudad, marcador (local y visitante) |
| **Implementado (Sí/No)** | Todo el grupo :D |

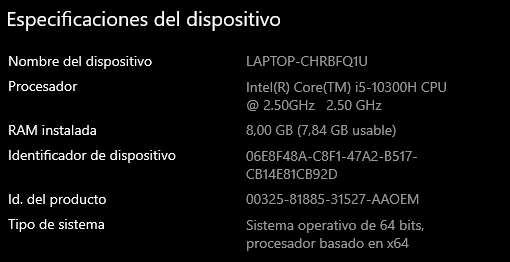
## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Creación de diccionario | O(1) |
| Iteracion en estructura de datos | O(n) |
| Creacion de lista | O(1) |
| ***TOTAL*** | ***O(...)*** |

## **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.

### **Graficas**

Las gráficas con la representación de las pruebas realizadas.

## **Análisis**

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

# **Requerimiento Ejemplo**

## **Descripción**



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.