ANÁLISIS DEL RETO

Pablo Yepes, 201923830, p.yepes@uniandes.edu.co

Jimmy Ceron, 202214971, ja.ceronm1@uniandes.edu.co

Estudiante 3, código 3, email 3

Requerimiento <<n>>

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

Descripción

Breve descripción de como abordaron la implementación del requerimiento

Entrada	Parámetros necesarios para resolver el requerimiento.
Salidas	Respuesta esperada del algoritmo.
Implementado (Sí/No)	Si se implementó y quien lo hizo.

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

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

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 <<1>>

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

Descripción

Breve descripción de como abordaron la implementación del requerimiento

```
lista_resultados=catalog["resultados"]
lista_parti=lt.newList("ARRAY_LIST")
total=resultadoSize(catalog)

for result in lt.iterator(lista_resultados):

    if condition == "home":
        if result["home_team"] == team:
            lt.addLast(lista_parti,result)

    elif condition == "visitor":
        if result["away_team"]== team:
            lt.addLast(lista_parti,result)

    elif condition == "neutral":
        if result["away_team"]== team or result["home_team"] == team:
            lt.addLast(lista_parti,result)
    lista_por_fechas = sort(lista_parti)
    if lt.size(lista_por_fechas) < number:
        return get_data(lista_por_fechas)
    sublist= lt.subList(lista_por_fechas,1,number)
return total,lista_por_fechas,sublist, lista_parti
```

Entrada	Numero de partidos de consulta, nombre del equipo, condición
Salidas	Número total de equipos, partidos totales del equipo, partidos del equipo según su condición, lista con los partidos de consulta
Implementado (Sí/No)	Pablo Yepes

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Paso 1: Asignar variables	O(1)
Paso 2: iterar sobre la lista	O(N)
Paso 3: asignar según condiciones y añadir a la lista	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).

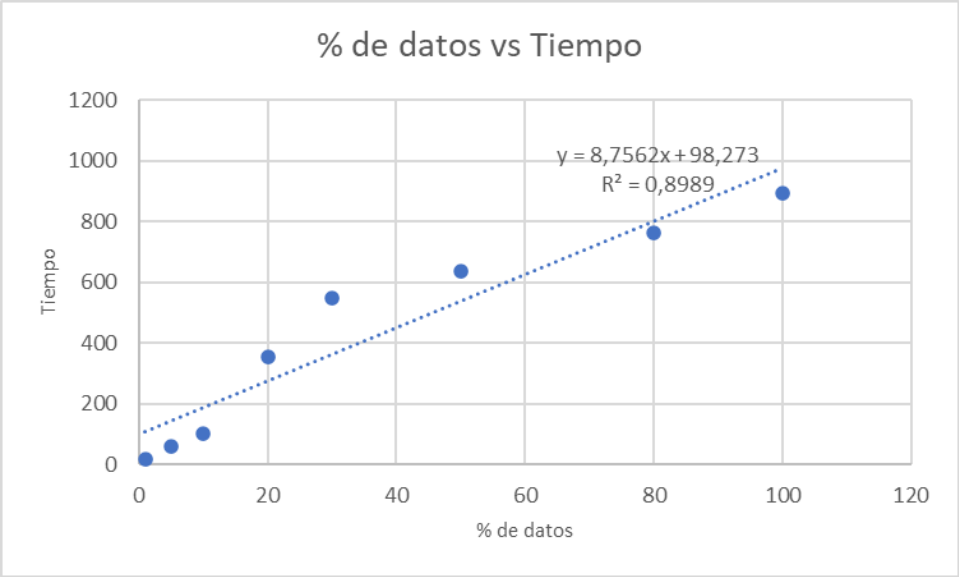
Entrada	**Tiempo (s)**
Small	20,51
5pct	58,52
10pct	103,5
20pct	354,5
30pct	549,48
50pct	635,05
80pct	762,7
Large	893,77

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.

La complejidad del requerimiento es lo suficientemente cercano a lo teórico para poder decir que es todo un éxito. Sin embargo, los datos parace que vayan a una tendencia no lineal.

Requerimiento <<2>>

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

Descripción

Breve descripción de como abordaron la implementación del requerimiento

```
def req_2(catalog, jugadorS, golesS):
    """
    Función que soluciona el requerimiento 2
    """
    # TODO: Realizar el requerimiento 2
    #lista_resultados= mp.valueSet(catalog['jugadores'])
    goleadoress = catalog['goleadores']
    contador = 0
    a= PrettyTable(hrules=ALL)
    a.field_names={"Fecha del gol", "Equipo local", "Equipo Visitante", "Equipo del jugador", "Minuto del gol", "Si fue por falta desde el penal", "Si fue autogol" }

    lista=lt.newList()
    pos = lt.isPresent(goleadoress, jugadorS)
    if pos > 0:
        jug_element = lt.getElement(goleadoress, pos)
        if jug_element is not None:
            for goleadores in lt.iterator(catalog["goalscorers"]):
                if jug_element["scorer"] == goleadores["scorer"]:
                    contador += 1
                    x=[goleadores["date"], goleadores["home_team"], goleadores["away_team"], goleadores["team"], goleadores["minute"], goleadores["penalty"], goleadores["own_goal"]]

                    a.add_row(x)
                    lt.addLast(lista,x)
                    golesS
    return a,contador
```

Entrada	Numero de goles de consulta, nombre jugador
Salidas	Número total de jugadores, goles totales del jugador, número de goles por penal, lista con los golesde consulta
Implementado (Sí/No)	Pablo Yepes

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Paso 1: Asignar variables	O(1)
Paso 2: iterar sobre la lista	O(N)
Paso 3: asignar según condiciones y añadir a la lista	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).

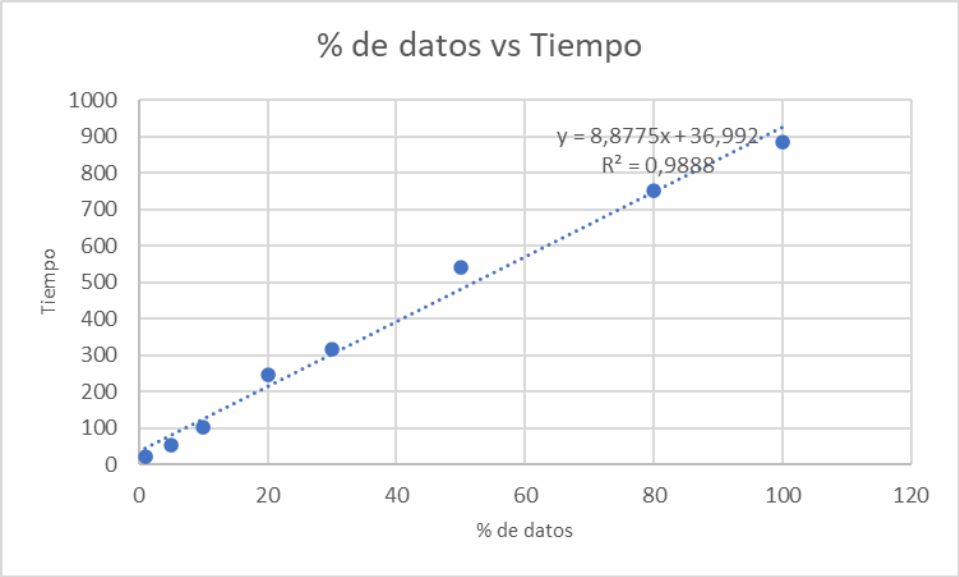
Entrada	**Tiempo (s)**
Small	22,2
5pct	55,66
10pct	104,52
20pct	246.85
30pct	316.84
50pct	542.74
80pct	750,9
Large	883,97

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.

Este Requerimiento si se acerca a la tendencia esperada, al tuilizar listas para la gran mayoría del código.

Requerimiento <<4>>

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

Descripción

Breve descripción de como abordaron la implementación del requerimiento

```python
def req_4(catalog,nombre, f_inicial,f_final):
    """
    Función que soluciona el requerimiento 4
    """
    # TODO: Realizar el requerimiento 4
    partidos=lt.newList("ARRAY_LIST")
    ciudades=lt.newList("ARRAY_LIST")
    paises=lt.newList("ARRAY_LIST")
    #lista_torneos= mp.valueSet(catalog['tournament'])

    penales=0
    f_inicial=time.strptime(f_inicial, "%Y-%m-%d")
    f_final=time.strptime(f_final, "%Y-%m-%d")
    print("a")

    for i in lt.iterator(catalog["resultados"]):
        print("a")
        fecha=time.strptime(i["date"], "%Y-%m-%d")
        if(i["tournament"]==nombre and fecha>=f_inicial and fecha<=f_final):
            lt.addLast(partidos,i)
            print("a")
    for j in lt.iterator(partidos):
        if (j["city"] not in ciudades):
            lt.addLast(ciudades,i["city"])
            print("b")
    for k in lt.iterator(partidos):
        if (k["country"] not in paises):
            lt.addLast(paises,i["country"])
            print("c")
    for l in lt.iterator(partidos):
        for m in lt.iterator(catalog['goleadores']):
            if (l["date"] == m["date"] and l["home_team"] == m["home_team"] and l["away_team"] == m["away_team"] and m["penalty"]==True):
                penales +=1
                print("d")
    num_partidos=lt.size(partidos)
    num_paises=lt.size(paises)
    num_c=lt.size(ciudades)
    return(num_partidos,num_paises,num_c,penales,partidos)
```

Entrada	Nombre del torneo, Fechas de consulta
Salidas	Numero de partidos/ciudades/países/penales y una lista con todos los partidos disputados.
Implementado (Sí/No)	Pablo Yepes

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Paso 1: Asignar variables	O(1)
Paso 2: iterar sobre la lista	O(N)
Paso 3: asignar según condiciones y añadir a la lista	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	20,83
5pct	55,66
10pct	104,52

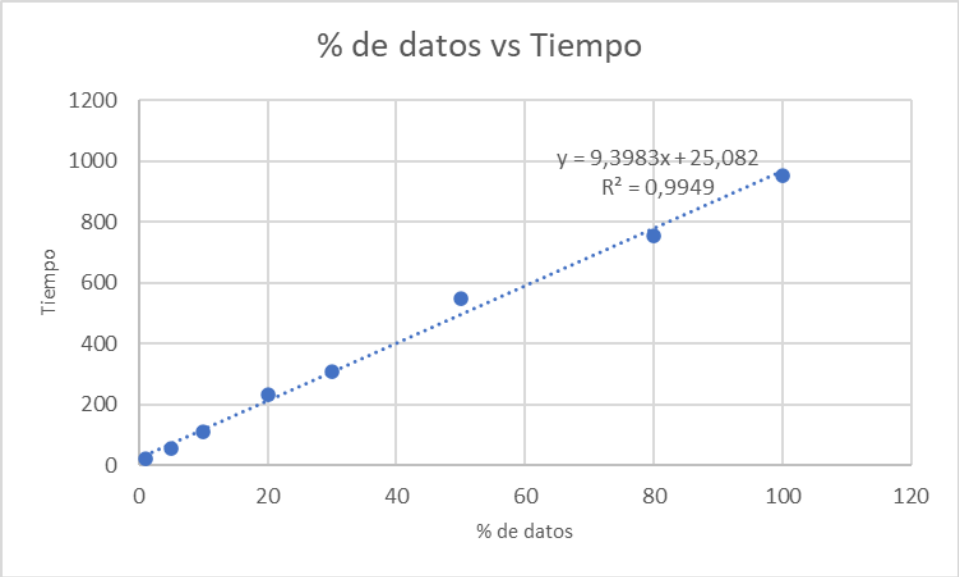
20pct	246.85
30pct	316.84
50pct	542.74
80pct	750,9
Large	883,97

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 <<6>>

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

Descripción

Breve descripción de como abordaron la implementación del requerimiento

```
# TODO: Realizar el requerimiento 6
fecha_inicial = time.strptime(f_inicial, "%Y-%m-%d")
fecha_final = time.strptime(f_final, "%Y-%m-%d")
equipos = lt.newList('ARRAY_LIST', cmpfunction=compare_teams)
paises = lt.newList('ARRAY_LIST', cmpfunction=compare_string)
ciudades = {}
n_partidos = 0
#lista_resultados= mp.valueSet(catalog['jugadores'])
#lista_goleadores= mp.valueSet(catalog['tournaments'])
for partido in lt.iterator(catalog['resultados']):
    if (fecha_inicial<partido['date']<fecha_final) and partido['tournament']==nombre:
        n_partidos +=1
        winner = winner(partido)
        local = lt.isPresent(equipos, partido['home_team'])
        visitante = lt.isPresent(equipos, partido['away_team'])
        if not local:
            equipo = {'team':partido['home_team'], 'points':0, 'score_dif':0,'n_matches':0,'penalties':0,'own_goals':0,'victories':0,'draws':0,'defeats':0,
            |   |   |   'scores':0,'scores_received':0, 'top_scorer':lt.newList('ARRAY_LIST')}
            lt.addLast(equipos, equipo)
            local=lt.size(equipos)
        if not visitante:
            equipo = {'team':partido['away_team'], 'points':0, 'score_dif':0,'n_matches':0,'penalties':0,'own_goals':0,'victories':0,'draws':0,'defeats':0,
            |   |   |   'scores':0,'scores_received':0, 'top_scorer':lt.newList('ARRAY_LIST')}
            lt.addLast(equipos, equipo)
            visitante=lt.size(equipos)

        if winner == 'home':
            cambio_l = lt.getElement(equipos, local)
            cambio_v = lt.getElement(equipos, visitante)
            cambio_l['points']+=3
            cambio_l['victories']+=1
            cambio_v['defeats']+=1
            lt.changeInfo(equipos, local, cambio_l)
            lt.changeInfo(equipos, visitante, cambio_v)
        elif winner =='draw':
            cambio_l = lt.getElement(equipos, local)
            cambio_v = lt.getElement(equipos, visitante)
            cambio_l['points']+=1
            cambio_l['draws']+=1
            cambio_v['draws']+=1
            cambio_v['points']+=1
        elif winner =='away':
            cambio_l = lt.getElement(equipos, local)
            cambio_v = lt.getElement(equipos, visitante)
            cambio_v['points']+=3
            cambio_v['victories']+=1
            cambio_l['defeats']+=1
        cambio_l = lt.getElement(equipos, local)
        cambio_v = lt.getElement(equipos, visitante)
        cambio_l['score_dif']+=(int(partido['home_score'])-int(partido['away_score']))
        cambio_v['score_dif']+=(int(partido['away_score'])-int(partido['home_score']))
        cambio_l['n_matches']+=1
        cambio_v['n_matches']+=1
```

```
        if lt.isEmpty(paises):
            lt.addLast(paises, partido['country'])
        elif not lt.isPresent(paises, partido['country']):
            lt.addLast(paises, partido['country'])

        if partido['city'] not in ciudades:
            ciudades[partido['city']]=1
        else:
            ciudades[partido['city']]+=1
        lt.changeInfo(equipos, local, cambio_l)
        lt.changeInfo(equipos, visitante, cambio_v)
    for gol in lt.iterator(catalog['goleadores']):
        local = lt.isPresent(equipos, gol['home_team'])
        visitante = lt.isPresent(equipos, gol['away_team'])
        if fecha_inicial<gol['date']<fecha_final and local and visitante:
            cambio_l = lt.getElement(equipos,local)
            cambio_v=lt.getElement(equipos, visitante)
            if gol['own_goal']=='True':
                if gol['team'] =='home':
                    cambio_l['own_goals']+=1
                    cambio_l['scores']+=1
                    cambio_l['scores_received']+=1
                elif gol['team'] =='away':
                    cambio_v['own_goals']+=1
                    cambio_v['scores']+=1
                    cambio_v['scores_received']+=1
            elif gol['team'] == 'home':
                cambio_l['scores']+=1
                cambio_v['scores_received']+=1
            elif gol['team'] == 'away':
                cambio_v['scores']+=1
                cambio_l['scores_received']+=1
            if gol['penalty']=='True':
                if gol['team'] =='home':
                    cambio_l['penalties']+=1
                elif gol['team'] =='away':
                    cambio_v['penalties']+=1

            if gol['team'] == 'home':
                jugador = lt.isPresent(cambio_l['top_scorer'], gol)
                if not jugador:
                    nuevo_j = {'scorer':gol['scorer'],'goals':1,'matches':lt.newList('ARRAY_LIST',comparar_partidos),'avg_time':float(gol['minute'])}
                    lt.addLast(nuevo_j['matches'],{'date':gol['date'],'away_team':gol['away_team'],'home_team':gol['home_team']})
                    lt.addLast(cambio_l['top_scorer'], nuevo_j)
                    jugador=lt.size(cambio_l['top_scorer'])
```

```
            else:
                cambio_j = lt.getElement(cambio_v['top_scorer'], jugador)
                cambio_j['goals']+=1
                if not lt.isPresent(cambio_j['matches'], gol):
                    lt.addLast(cambio_j['matches'],{'date':gol['date'],'away_team':gol['away_team'],'home_team':gol['home_team']})
                cambio_j['avg_time']=((cambio_j['avg_time']*(cambio_j['goals']-1))+float(gol['minute']))/cambio_j['goals']
                lt.changeInfo(cambio_v['top_scorer'], jugador, cambio_j)
        elif gol['team'] == 'away':
            jugador = lt.isPresent(cambio_v['top_scorer'], gol)

            if not jugador:
                nuevo_j = {'scorer':gol['scorer'],'goals':1,'matches':lt.newList('ARRAY_LIST',comparar_partidos),'avg_time':float(gol['minute'])}
                lt.addLast(nuevo_j['matches'],{'date':gol['date'],'away_team':gol['away_team'],'home_team':gol['home_team']})
                lt.addLast(cambio_v['top_scorer'], nuevo_j)
                jugador=lt.size(cambio_v['top_scorer'])

            elif not lt.isEmpty(cambio_v['top_scorer']):

                cambio_j = lt.getElement(cambio_v['top_scorer'], jugador)
                cambio_j['goals']+=1
                if not lt.isPresent(cambio_j['matches'], gol):
                    lt.addLast(cambio_j['matches'],{'date':gol['date'],'away_team':gol['away_team'],'home_team':gol['home_team']})
                cambio_j['avg_time']+=float(gol['minute'])
                lt.changeInfo(cambio_v['top_scorer'], jugador, cambio_j)

        lt.changeInfo(equipos,local, cambio_l)
        lt.changeInfo(equipos, visitante, cambio_v)
    num_equipos=lt.size(equipos)
    num_paises=lt.size(paises)
    num_ciudades=len(ciudades)
    top_ciudad= max(ciudades,key=ciudades.get)
    return num_equipos, num_paises, num_ciudades, top_ciudad,equipos
```

Entrada	Número de equipos, nombre del torneo, Fechas de consulta
Salidas	• El total de años calendarios disponibles en el historial de partidos. • El total de torneos disputados en el año de consulta. • El total de equipos involucrados en el torneo. • El total de encuentros disputados en el año. • 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 listado de los equipos que conforman el torneo debe estar ordenado por el criterio compuesto de sus estadísticas. Donde cada uno de los equipos resultantes contendrán la siguiente información: o El nombre del equipo. o El total de puntos obtenidos o La diferencia de goles. o El total de partidos disputados. o El total de puntos obtenidos desde la línea penal. o El total de puntos recibidos por autogol. o El total de victorias. o El total de empates. o El total de derrotas. o El total de goles obtenidos por sus jugadores. o El total de goles recibidos por el equipo. o 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)	Pablo Yepes

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Paso 1: Asignar variables	O(1)
Paso 2: iterar sobre las listas (pasa para results y para goalscorers, pero por separado)	O(N)
Paso 3: asignar según condiciones y añadir a la lista	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).

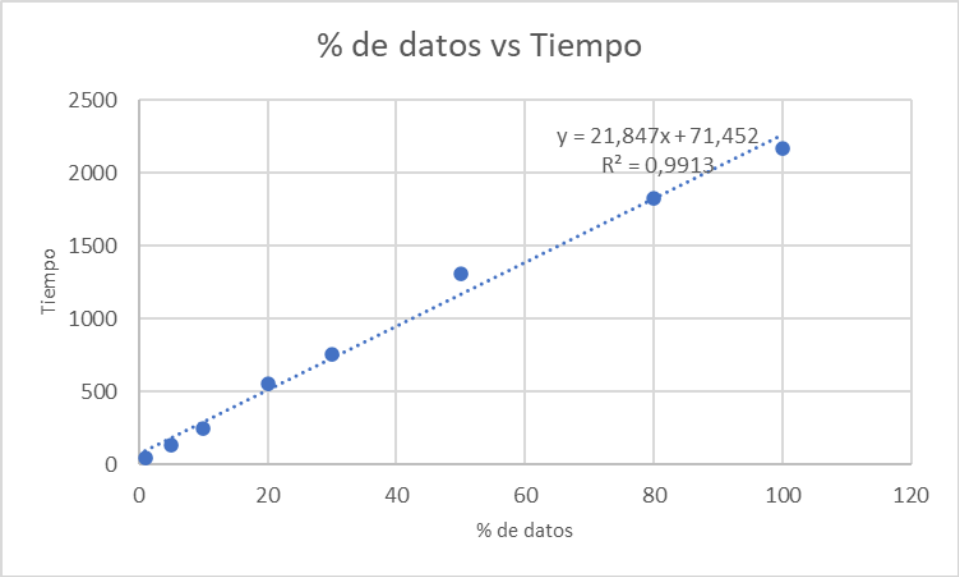
Entrada	**Tiempo (s)**
Small	44,23
5pct	135,82
10pct	245
20pct	555,16
30pct	753,96
50pct	1310,28
80pct	1829,25
Large	2164,62

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>>

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

Descripción

Breve descripción de como abordaron la implementación del requerimiento

Entrada	Nombre del torneo que se desea consultar (incluyendo amistosos). El puntaje (N) de los jugadores dentro del torneo (ej.: 3, 5, 10 o 20)
Salidas	El total de torneos disponibles para consultar. • El total de anotadores que participaron en el torneo. • El total de partidos dentro del torneo. • El total de anotaciones o goles obtenidos durante los partidos del torneo. • El total de goles por penal obtenidos en ese torneo • El total de autogoles en que incurrieron los anotadores en ese torneo • 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: o El nombre del anotador. o El puntaje que obtiene el jugador como anotador. o El total de goles anotados. o El total de goles anotados por penales. El total de autogoles anotados. o El tiempo promedio para anotar en minutos.

	o El total de torneos en que anotó el jugador. o El total de anotaciones obtenidos en una victoria. o El total de anotaciones obtenidos en un empate. o El total de anotaciones obtenidos en una derrota. o Ultimo 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)	Pablo Yepes

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Paso 1: Asignación de variables	O(1)
Paso 2 : iteraciones (pasa 3 veces, pero de manera separada)	O(N)
Paso ….	O(...)
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).

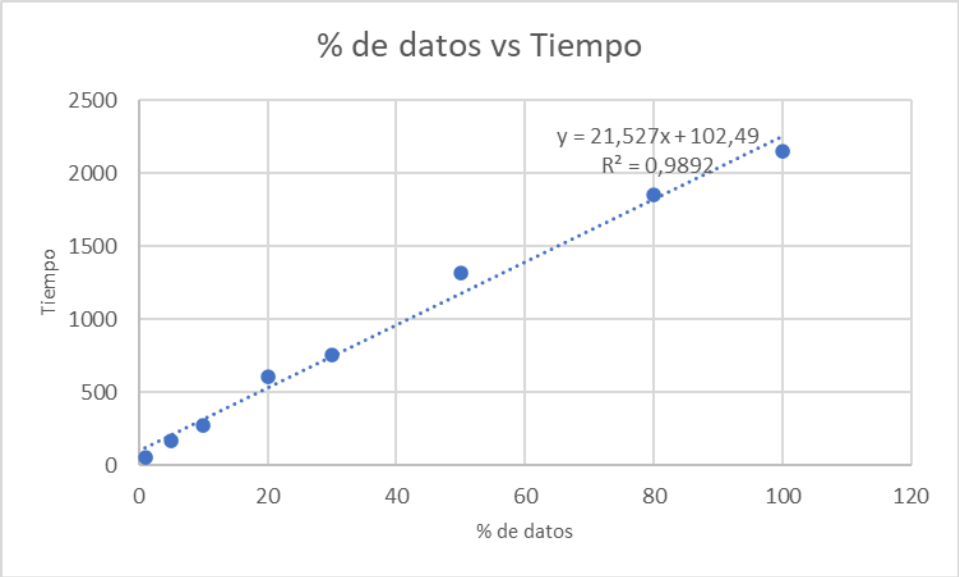
Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

Entrada	**Tiempo (s)**
Small	53,72
5pct	170,48
10pct	272,36
20pct	611,23
30pct	760,55
50pct	1320,95
80pct	1854,19
Large	2148,39

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.

Se nota claramente una relación lineal, lo cual era lo esperado en la literatura.

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

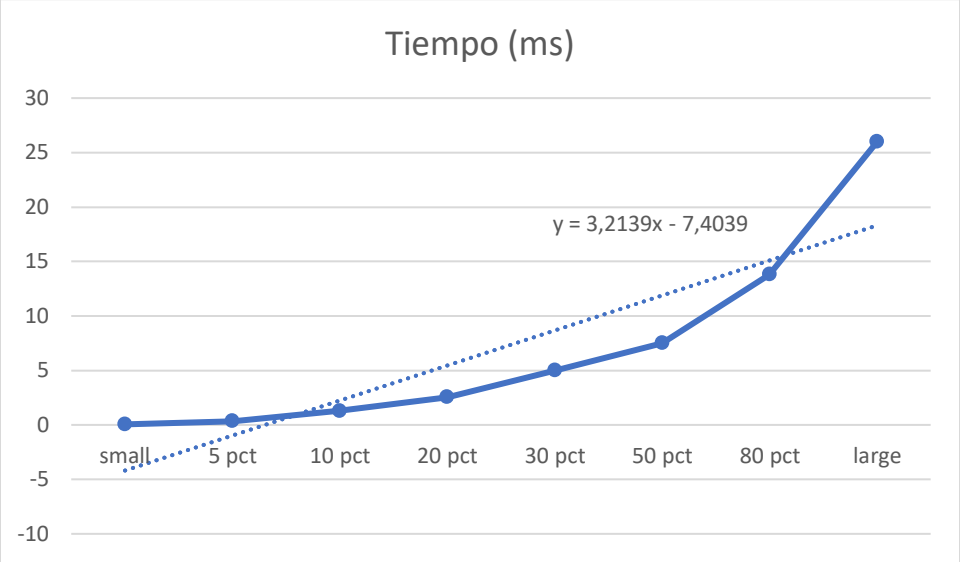
Tablas de datos

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.