Universidad de los Andes
EDA Reto 1 Sección 1
26/09/23

Juan Diego Chaves 202221851 jd.chaves2@uniandes.edu.co

Santiago Baquero Villa - 202220396

Juan Diego Chaves Zambrano - 202221851

Akshaya Arunachalam - 202020637

Análisis del Reto 1

Requerimiento 1:

```python
def print_req_1(control):
    """
        Función que imprime la solución del Requerimiento 1 en consola
    """
    team_name = input('Por favor escriba el nombre del equipo a consultar: ')
    num_matches = int(input('Por favor digite el número de partidos a consultar: '))
    print('Por favor seleccione la condición de el equipo en los partidos a consultar: ')
    print('[1] para partidos HOME')
    print('[2] para partidos AWAY')
    print('[3] para todos los partidos jugados')
    team_condition = int(input('Digite el número seleccionado: '))
    if team_condition == 1:
        team_condition = 'HOME'
    elif team_condition == 2:
        team_condition = 'AWAY'
    elif team_condition == 3:
        team_condition == 'ANY'
    else:
        print('Seleccione una opción correcta')
        print_req_1(control)
    games_list, total_games, total_time = controller.req_1(control, num_matches, team_name, team_condition)
    if type(games_list) == type(None):
        print('Unknown')
    else:
        print("================================REQ 1================================")
        games_list_table = new_table(games_list, 'keys', 'grid')
        print (games_list_table)
        print('Se encontraron ' + str(total_games)+ ' partidos.')
        print('Se demoró ' + str(f"{total_time: .3f}") + ' milisegundos.')
```

view.py

```python
def req_1(control, num_matches, team_name, team_condition):
    """
    Retorna el resultado del requerimiento 1
    """
    tiempo_inicio = get_time()
    team_matches, total_matches = model.req_1(control['model'], num_matches, team_name, team_condition)
    tiempo_final = get_time()
    tiempo_total = delta_time(tiempo_inicio, tiempo_final)
    if total_matches == 0:
        return None
    if total_matches > 6:
        matches = get_sublist_6elements(team_matches, total_matches)
        return matches, total_matches, tiempo_total
    else:
        matches = team_matches['elements']
        return matches, total_matches, tiempo_total
```

controller.py

```python
def req_1(catalog, num_matches, team_name, team_condition):
    """
    ARGS: catalog con información de resultados ya ordenada por fechas de más reciente a más vieja, número de partidos
    a consultar, nombre del equipo de interés, condición del equipo en el partido.
    RET: LISTA ENCADENADA de la cantidad deseada de partidos
    """
    results = catalog['results']
    results_size = lt.size(results)
    team_matches = lt.newList('ARRAY_LIST')
    total_matches = 0
    for result_index in range(1, results_size + 1):
        match = lt.getElement(results, result_index)
        if team_condition == 'HOME':
            if match['home_team'] == team_name:
                total_matches += 1
                lt.addLast(team_matches, match)
        elif team_condition == 'AWAY':
            if match['away_team'] == team_name:
                total_matches += 1
                lt.addLast(team_matches, match)
        else:
            if (match['home_team'] == team_name) or (match['away_team'] == team_name):
                total_matches += 1
                lt.addLast(team_matches, match)
        if total_matches == num_matches:
            break
    return team_matches, total_matches
```

model.py

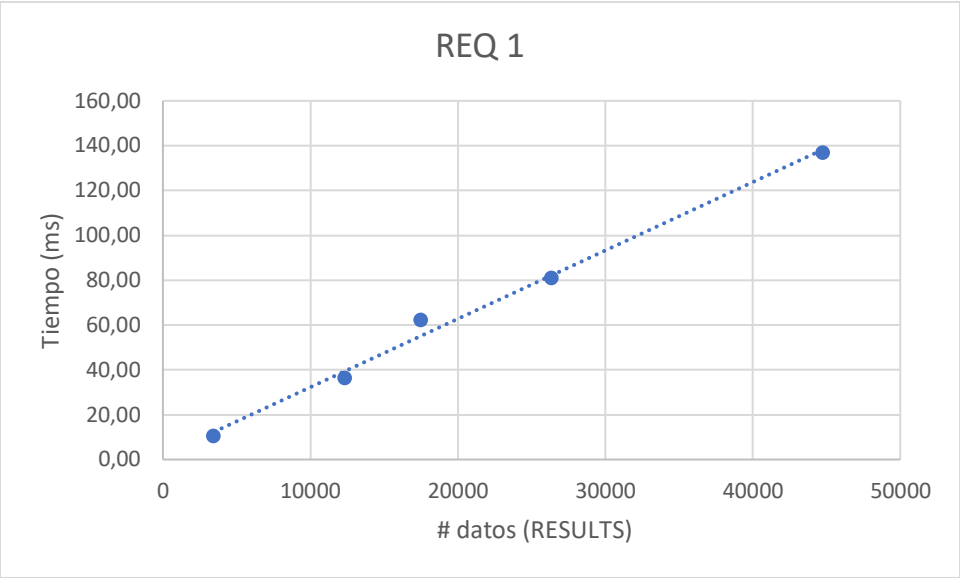
(comentario de retorno está errado)

El requerimiento 1 fue solucionado recorriendo la lista de 'results' de adelante para atrás dado que la lista estaba previamente ordenada con los partidos más recientes primero. Requiere de tres parámetros: el número de partidos, el nombre del equipo y la condición bajo la que se jugaba.

Paso	Notación	Descripción
Creación lista de partidos	O (1)	Inicialización de la lista de partidos de interés
Uso del for in range	O(n)	Recorre el archivo de 'results'. Se compara si el partido deseado tiene al equipo en la condición iniciada o es indiferente
Añadir a la lista de partidos	O(1)	Al ser Array_List, la complejidad de agregar un elemento es O(1)
Retornar lista al controller	O (1)	Se envía la lista al controller.py y si es el caso se unen dos sublistas cada una de los 3 elementos de los extremos
Imprimir resultado	O(1)	El view utiliza tabulate para imprimir la tabla de información
Total	O (n)	-

Rendimiento en prueba:

Cantidad de datos	Tiempo [ms]
3463	10,47
12364	36,28
17486	62,35
26341	81,11
44762	137,03



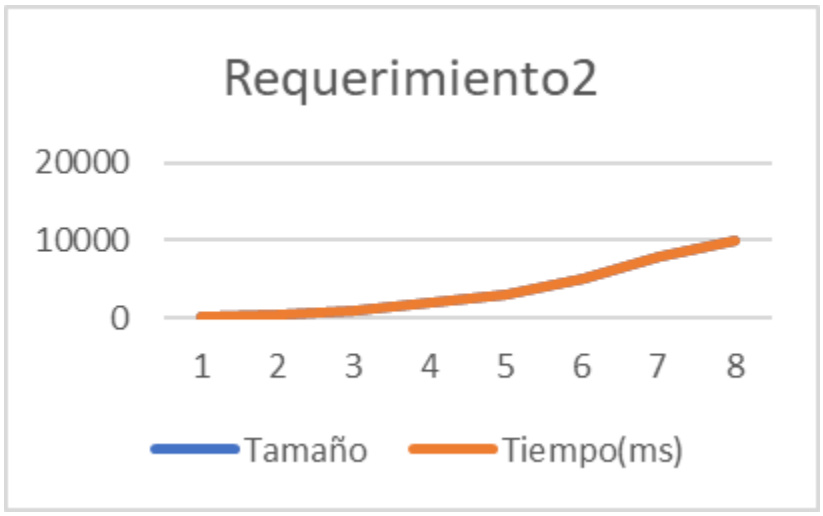
El comportamiento es el esperado; una tendencia lineal de crecimiento temporal.

Máquina:

11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz 2.80 GHz/ 16 GB RAM

Requerimiento 2 (Akshaya):

Paso	Notación	Descripción
Creación de la lista de goles	O (1)	Inicialización de la lista de goles
Uso del for	O(n)	Recorre el archivo de 'goalscorers'. En el primer if, se compara el nombre del jugador con el que da el usuario, se van guardando los datos en la lista de goles.
Función sort ()	O (n log n)	Se ordenan los goles del jugador del más antiguo al más reciente.
Retornar lista y imprimir resultados	O (1)	Se envía la lista al view.py y se imprimen los resultados en pantalla.
Total	O (n log n)	-



La gráfica se comporta como O (n log n).

Requerimiento 3(Santiago Baquero)

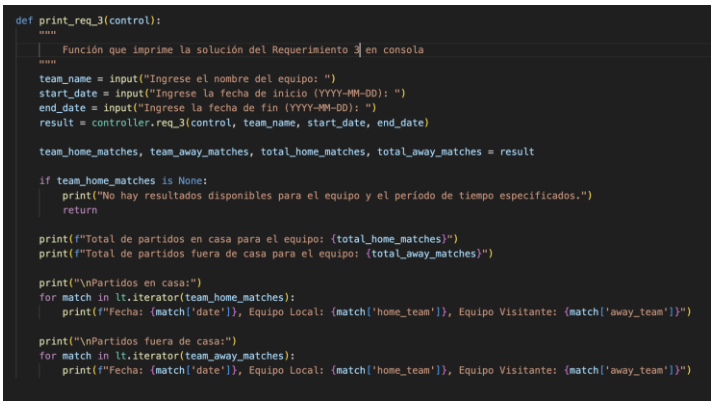
Model:

```
def req_3(catalog, team_name, start_date, end_date):
    """
    Función que soluciona el requerimiento 3
    """
    results = catalog.get('results')
    if results is None:
        return None

    results_size = lt.size(results)
    team_home_matches = lt.newList('ARRAY_LIST')
    team_away_matches = lt.newList('ARRAY_LIST')
    total_home_matches = 0
    total_away_matches = 0

    start_date = date.fromisoformat(start_date)
    end_date = date.fromisoformat(end_date)

    for result_index in range(1, results_size + 1):
        match = lt.getElement(results, result_index)
        match_date = date.fromisoformat(match['date'])

        if start_date <= match_date <= end_date:
            if match['home_team'] == team_name:
                total_home_matches += 1
                lt.addLast(team_home_matches, match)
            elif match['away_team'] == team_name:
                total_away_matches += 1
                lt.addLast(team_away_matches, match)

    def custom_sort(match1, match2):
        return match1['date'] - match2['date']

    sa.sort(team_home_matches, custom_sort)
    sa.sort(team_away_matches, custom_sort)

    return team_home_matches, team_away_matches, total_home_matches, total_away_matches
```

Controller:

```
def req_3(control, team_name, start_date, end_date):
    """
    Retorna el resultado del requerimiento 3
    """
    return model.req_3(control['model'], team_name, start_date, end_date)
```

View:



```
def print_req_3(control):
    """
        Función que imprime la solución del Requerimiento 3 en consola
    """
    team_name = input("Ingrese el nombre del equipo: ")
    start_date = input("Ingrese la fecha de inicio (YYYY-MM-DD): ")
    end_date = input("Ingrese la fecha de fin (YYYY-MM-DD): ")
    result = controller.req_3(control, team_name, start_date, end_date)

    team_home_matches, team_away_matches, total_home_matches, total_away_matches = result

    if team_home_matches is None:
        print("No hay resultados disponibles para el equipo y el período de tiempo especificados.")
        return

    print(f"Total de partidos en casa para el equipo: {total_home_matches}")
    print(f"Total de partidos fuera de casa para el equipo: {total_away_matches}")

    print("\nPartidos en casa:")
    for match in lt.iterator(team_home_matches):
        print(f"Fecha: {match['date']}, Equipo Local: {match['home_team']}, Equipo Visitante: {match['away_team']}")

    print("\nPartidos fuera de casa:")
    for match in lt.iterator(team_away_matches):
        print(f"Fecha: {match['date']}, Equipo Local: {match['home_team']}, Equipo Visitante: {match['away_team']}")
```

Paso	Notación
Se obtiene la clave results del catalogo	**O (1)**
Verificar si results existe, de no ser así retorna None	**O (1)**
Inicializar las diferentes variables	**O (1)**
Se convierte start_date y end_date a fechas	**O (1)**
Se iterar a través de aquello que se encuentre dentro del catálogo, todo lo que se encuentra dentro de esta función tiene notación 0(1)	**O (n)**
Ordena las listas team_home_matches y team_away_matches llamando a shell sort	**O (n log n)**
Retrono team_home_matches, team_away_matches, total_home_matches y total_away_matches	**O (1)**

<u>Requerimiento 4 (Juan Diego Chaves):</u>

```python
def print_req_4(control):
    """
        Función que imprime la solución del Requerimiento 4 en consola
    """
    tournament_name = input('Por favor escriba el nombre del torneo que quiere consultar: ')
    start = input('Por favor escriba la fecha de inicio de consulta en formato YYYY-MM-DD: ')
    end = input('por favor escriba la fecha final de consulta en formato YYYY-MM-DD: ')
    print("==================================REQ 4==================================")
    print('Torneo a consultar: ' + tournament_name)
    print('Fecha inicio: ' + start)
    print('Fecha fin: ' + end)
    tournament_matches, total_matches, total_shootouts, total_countries, total_cities, tiempo_total = controller.req_4(control, tournament_name, start, end)
    if tournament_matches == None:
        print('Unknown')
    else:
        tournament_games_table = new_table(tournament_matches, 'keys', 'grid')
        print('Total matches: ' +str(total_matches))
        print('Total shootouts: ' + str(total_shootouts))
        print('Total cities: ' + str(total_cities))
        print('Total countries: ' + str(total_countries))
        print(tournament_games_table)
        print('Se demoró: ' + str(f"{tiempo_total: .3f}") + ' milisegundos')
```

```python
def req_4(control, tournament_name, start, end):
    """
    Retorna el resultado del requerimiento 4
    """
    tiempo_inicio = get_time()
    tournament_matches, total_matches, total_shootouts, total_countries, total_cities = model.req_4(control['model'], tournament_name, start, end)
    tiempo_final = get_time()
    tiempo_total = delta_time(tiempo_inicio, tiempo_final)
    if tournament_matches == None:
        return tournament_matches, total_matches, total_shootouts, total_countries, total_cities, tiempo_total
    if total_matches > 6:
       top6 = get_sublist_6elements(tournament_matches, total_matches)
       return top6, total_matches, total_shootouts, total_countries, total_cities, tiempo_total
    return tournament_matches['elements'], total_matches, total_shootouts, total_countries, total_cities, tiempo_total
```

```python
def req_4(catalog, tournament_name, start, end):
    start_date = date.fromisoformat(start)
    end_date = date.fromisoformat(end)
    tournament_matches = lt.newList('ARRAY_LIST')
    cities = lt.newList('ARRAY_LIST')
    countries = lt.newList('ARRAY_LIST')
    results = catalog['results']
    shootouts = catalog['shootouts']
    shootout_start_index = 1
    total_shootouts = 0
    total_countries = 0
    total_cities = 0
    total_matches = 0
    winner = ['']

    for result_index in range(1, lt.size(results) + 1):
        result = lt.getElement(results, result_index)
        match_date = date.fromisoformat(result['date'])
        if (start_date <= match_date) and (end_date >= match_date):
            if result['tournament'] == tournament_name:
                total_matches += 1
                if result['home_score'] == result['away_score']:
                    for shootout_index in range(shootout_start_index, lt.size(shootouts) + 1):
                        shootout = lt.getElement(shootouts, shootout_index)
                        if date.fromisoformat(shootout['date']) == match_date and shootout['home_team'] == result['home_team']:
                            winner[0] = shootout['winner']
                            shootout_start_index = shootout_index
                            total_shootouts += 1
                            break
```

```
                else:
                    winner[0] = 'Unknown'
                tournament_match = new_match(result, winner[0])
                if lt.size(tournament_matches) == 0:
                    lt.addFirst(tournament_matches, tournament_match)
                    lt.addFirst(countries, tournament_match['country'])
                    lt.addFirst(cities, tournament_match['city'])
                else:
                    lt.addLast(tournament_matches, tournament_match)
                    if lt.isPresent(cities, tournament_match['city']) == 0:
                        lt.addFirst(cities, tournament_match['city'])
                        total_cities += 1
                    if lt.isPresent(countries, tournament_match['country']) == 0:
                        lt.addFirst(countries, tournament_match['country'])
                        total_countries += 1
    if total_matches == 0:
        return None, None, None, None, None
    tournament_matches = merg.sort(tournament_matches, compare_tournament_matches)
    return tournament_matches, total_matches, total_shootouts, total_countries, total_cities

def new_match(result, winner):
    match = {'date' : '','tournament' : '', 'country' : '', 'city' : '', 'home_team' : '' , 'away_team' : '' ,
             'home_score' : '', 'away_score' : '', 'winner' : ''}
    match['date'] = result['date']
    match['tournament'] = result['tournament']
    match['country'] = result['country']
    match['city'] = result['city']
    match['home_team'] = result['home_team']
    match['away_team'] = result['away_team']
    match['home_score'] = result['home_score']
    match['away_score'] = result['away_score']
    match['winner'] = winner
    return match
```

```
def compare_tournament_matches(t1, t2):
    if (compareByDate(t1, t2) == 0) and(type(compareByDate(t1, t2)) == type(0)):
        if (compareByCountry(t1, t2)  == 0) and (type(compareByCountry(t1, t2)) == type(0)):
            return compareByCity(t1, t2)
        else:
            return compareByCountry(t1, t2)
    else:
        return compareByDate(t1, t2)
```

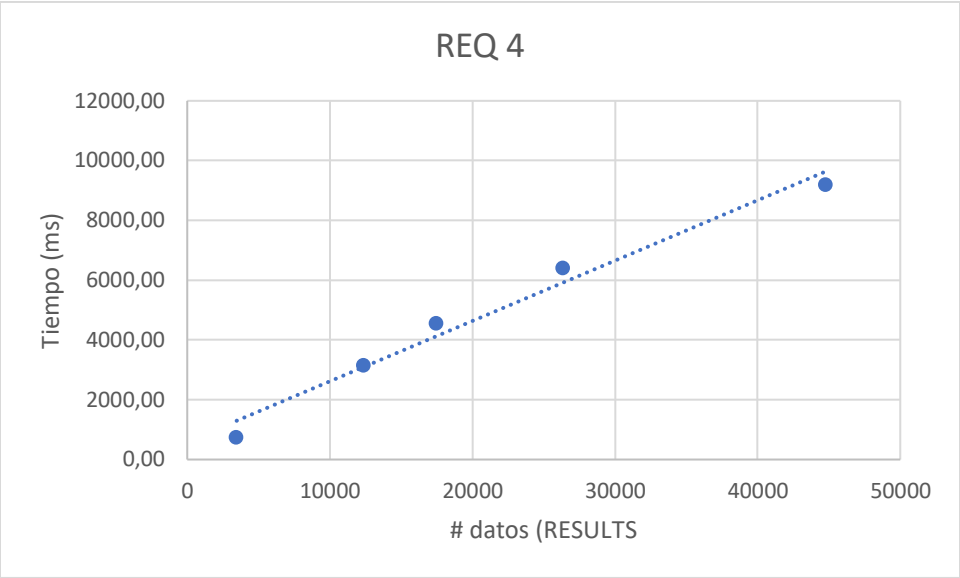
Se crean listas donde se va a guardar la información deseada y se recorre ‘results’ buscando partidos que cumplen con los criterios de torneo y fecha. Si los marcadores son iguales, se empieza a recorrer shootouts para encontrar el resultado de los penales. Al momento de encontrar un shootout, se rompe el for y se retorna el índice de ese shootout para empezar desde ahí la próxima vez que se empiece a recorrer shootouts dado que está organizado de más reciente a más viejo; al igual que results. Esto para economizar tiempo. Para organizar la información se crea un nuevo dato llamado tournament_match que contiene la información relevante para la solución. Se ordena la lista de tournament_matches y se retorna al controller.

Paso	Notación	Descripción
Creación de las listas donde se guardará la información	O (1)	Inicialización de las listas de países, ciudades, y partidos del torneo
		Recorre el archivo de results. Si encuentra partidos que

Uso del for in range	O(n)	están en las fechas deseadas y son parte del torneo busca si son empates.
Uso del for in range	O (m)	Se recorre la lista de shootouts para encontrar el resultado del desempate. En caso de encontrarlo rompe el for y retorna posición.
Comparaciones de elementos y parámetros	O(1)	Se comparan los valores de los elementos encontrados con los parámetros con if
Uso del isPresent	O(x)	Se busca si la ciudad y país del partido ya había ocurrido en el torneo. En caso de que no se añaden a la lista
Uso del new_match	O(1)	Se genera un nuevo dato de match con la información del partido y la tanda de penalties
Merge sort	O(matches log matches)	Se ordenan los partidos con los criterios mencionados con un merge sort
Retornar al controller, hacer lista de 3 de arriba y 3 de abajo, imprimir	O(1)	Se hacen 2 sublistas si es necesario de 3 elementos; se usa tabulate para imprimir la info.
Total	O(n + m)	Recorre results y shootouts

Rendimiento en prueba:

Cantidad de datos	Tiempo [ms]
3463	738,94
12364	3162,35
17486	4563,92
26341	6401,35
44762	9203,40



El comportamiento es el esperado; crecimiento temporal de tendencia lineal

Máquina:

11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz 2.80 GHz/16 GB RAM

Requerimiento 6:

Model

```python
def req_6(catalog, tournament_name, start_date, end_date, num_teams):
    """
    Función que soluciona el requerimiento 6
    """
    results = catalog['results']
    shootouts = catalog['shootouts']

    filtered_results = []
    for result in results:
        result_date = date.fromisoformat(result['date'])
        if result['tournament'] == tournament_name and start_date <= result_date <= end_date:
            filtered_results.append(result)

    teams_stats = {}
    for result in filtered_results:
        home_team = result['home_team']
        away_team = result['away_team']
        home_score = int(result['home_score'])
        away_score = int(result['away_score'])

        if home_team not in teams_stats:
            teams_stats[home_team] = {'points': 0, 'goals_diff': 0, 'goals_scored': 0, 'matches_played': 0,
                                      'penalty_points': 0, 'auto_goal_points': 0, 'victories': 0, 'draws': 0, 'defeats': 0,
                                      'top_scorer': {'name': 'Unknown', 'total_goals': 0, 'matches_with_goals': 0, 'avg_minutes_per_goal': 0}}
        teams_stats[home_team]['matches_played'] += 1
        teams_stats[home_team]['goals_scored'] += home_score
        teams_stats[home_team]['goals_diff'] += (home_score - away_score)

    top_teams = []
    for _ in range(num_teams):
        max_points = -1
        top_team = None
        for team_name, stats in teams_stats.items():
            if stats['points'] > max_points and team_name not in top_teams:
                max_points = stats['points']
                top_team = team_name
        if top_team is not None:
            top_teams.append(top_team)

    results_dict = {team_name: teams_stats[team_name] for team_name in top_teams}
    return results_dict
```

Controller

```python
def req_6(control):
    """
    Retorna el resultado del requerimiento 6.
    """
    catalog = control['catalog']
    tournament_name = control['tournament_name']
    start_date = control['start_date']
    end_date = control['end_date']
    num_teams = control['num_teams']
    result = req_6(catalog, tournament_name, start_date, end_date, num_teams)

    return result
```

View

```python
def print_req_6(control):
    """
    Función que imprime la solución del Requerimiento 6 en consola.
    """
    tournament_name = input("Ingrese el nombre del torneo: ")
    start_date = input("Ingrese la fecha de inicio (en formato 'YYYY-MM-DD'): ")
    end_date = input("Ingrese la fecha de fin (en formato 'YYYY-MM-DD'): ")

    print(f"Resultado del Requerimiento 6 para el torneo '{tournament_name}' en el período del {start_date} al {end_date}:")
```

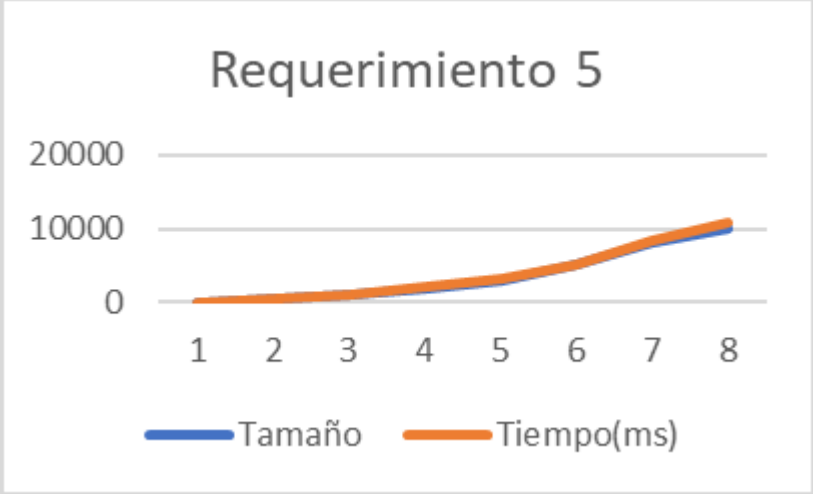
Paso	Notación
Obtengo las listas de los resultados y los shootouts	O (1)
Filtro los resultados por torneo y rango de fechas, las funciones dentro de ella son de notación O (1) estas son: la conversión de la fecha del resultado a objeto de fecha, verificar si el resultado pertenece al torneo y se encuentra en el rango dado y agregar el filtrado a la lista 'filtered results'	O (n)
Calculo las estadiscitcas de los equipos a partir de los resultados filtrados, para sus funciones internas: la extracción de la información del equipo local y visitante y sus puntajes y la actualización de las estadísticas para el equipo local la notación es O (1)	O (n)
Encuentro los equipos mejor clasificados, para sus funciones internas: repetir num_teams para encontrar los equipos con más puntos, la notación es O (n) y la iteración a través de los equipos para hallar el equipo con más puntos es O(n) igualmente	O (n * m)
Creo un diccionario para los equipos mejor clasificados	O (n)

Retrono el diccionario	O (1)

Requerimiento 5 (Akshaya):

Paso	Notación	Descripción
Inicialización de datos	O (1)	Solicitud de datos al usuario y inicialización de variables necesarias.
Primer for	O(n)	Se recorre el archivo de 'goalscorers' y se filtran los goles ocurridos dentro de la ventana de tiempo iniciada por el usuario. En el primer if, se compara el nombre del jugador del archivo con el que da el usuario y lo mismo con la fecha comparado con la fecha inicial y la fecha final, y se va agregando. En el segundo y tercer if respectivamente, se va agregando el penal y el auto en forma cíclica. El if que está por fuera es cuando no se encuentra la lista de goles.
Función sort ()	O (n log n)	Ordena goles del más reciente al más antiguo.
Segundo for	O(n^2)	Para cada gol en la lista de goles se busca el partido donde dicho gol fue anotado para crear lista de encuentros En el primer for se recorre la lista de goles, mientras que dentro de está en el segundo for se recorre la lista de resultados. Luego en los dos primeros if, se tiene que hacer las comparaciones entre los archivos goles y resultados en

		donde se encuentra el valor para ‘date’, ‘home_team’ y ‘away_team’. Y con esto, en el tercer if se mira si el elemento existe y luego se va guardando.
Tercer for	O(n)	Se recorre la lista de encuentros para generar la lista de torneos. En el primer if de adentro se busca y se agrega si el torneo está presente. Y en el if de afuera, se mira que no existan partidos amistosos que se eliminan.
Total	O (1) + O(n) + O (n log n) + O(n^2) + O(n) = O(n^2)	-

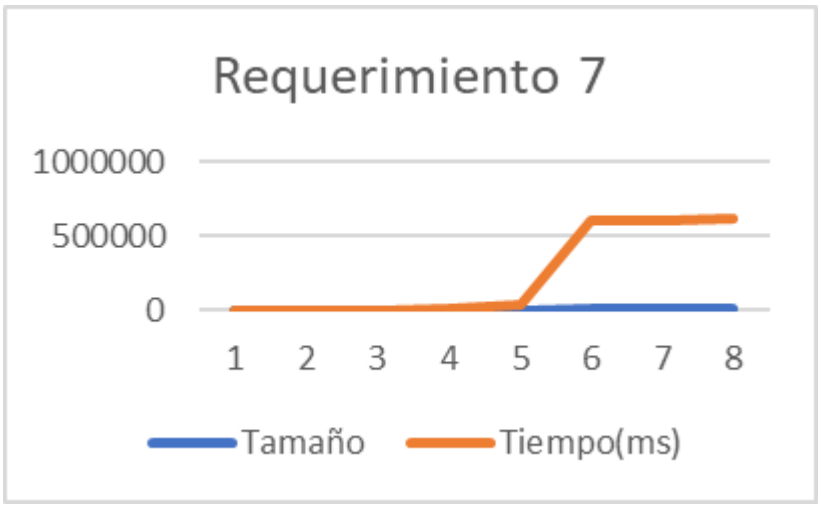


La gráfica se comporta cómo O(n^2).

Requerimiento 7(Akshaya):

Paso	Notación	Descripción
Inicialización de datos	O (1)	Solicitud de datos al usuario y inicialización de variables necesarias.
Primer for	O(n^2)	Obtención de la lista de goles anotados en la ventana de tiempo del usuario, obtención del partido donde cada gol ocurre y filtrado para

		descartar goles y partidos amistosos.
Segundo for	O(n^2)	Crea la lista de jugadores que anotaron gol en la ventana de tiempo, se recorre la lista de goles encontrada en el paso anterior y para cada gol se crea un diccionario para el jugador con las estadísticas necesarias para el requerimiento (ver función newPlayer ()). En caso, de que el jugador ya fuera creado previamente sólo se actualizan las estadísticas. Adicionalmente, para cada gol se recorre la lista de partidos para determinar si el gol fue anotado en una derrota, victoria o empate, y se guardan esos valores en diccionario de jugadores.
Tercer for	O(n^2)	Una vez obtenida la lista de jugadores del paso anterior se procede al cálculo de la estadística 'puntaje' y la estadística 'promedio'. Simultáneamente, se busca el partido correspondiente al último gol encontrado en la fase anterior.
Función sort ()	O (n log n)	Se ordena la lista de jugadores de acuerdo con el criterio del puntaje.
Total	O (1) + 3 * O(n^2) + O (n log n) = O(n^2)	-



La gráfica no se comporta cómo O(n^2).