**Complexity analysis**

* **Time Complexity Analysis**

1. **Method addMatch()**

Texto

El contenido generado por IA puede ser incorrecto.

**Line-by-Line Analysis**:

* **Lines 2-3**: teams.obtain(homeTeam) and teams.obtain(awayTeam) are search operations in a hash table. In the **best case**, the complexity is **O(1)**. In the **worst case**, it can be **O(n)**, but we assume average hash table performance.
* **Lines 5-9**: The goal comparison and points update operations are constant, i.e., **O(1)**.
* **Lines 11-12**: The insertion into the matches hash table is **O(1)** in an efficient hash table.
* **Lines 14-20**: The goal comparison and point update operations for the teams are **O(1)**.
* **Line 22**: The actions.push() operation is **O(1)**.

**Overall Time Complexity for addMatch()**: **O(1)**, since all operations in the method are constant time and executed only once.

1. **Method matchSchedule()**

Texto

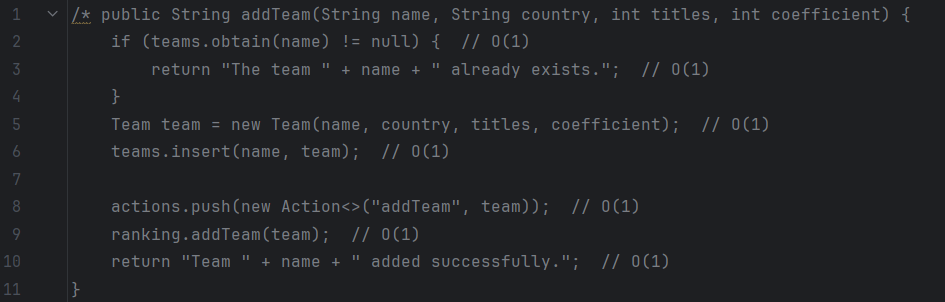
El contenido generado por IA puede ser incorrecto.

**Line-by-Line Analysis**:

* **Lines 2-3**: The matchQueue.isEmpty() operation and the return have **O(1)** complexity.
* **Line 6**: Creating a new queue is **O(1)**.
* **Lines 7-11**: The while (!matchQueue.isEmpty()) loop iterates through all elements in matchQueue. If there are **n** matches, this loop runs **n** times. Each operation inside the loop (dequeue(), schedule +=, enqueue()) is **O(1)**, so the loop has **O(n)** complexity.
* **Lines 12-13**: The second while (!queue.isEmpty()) loop also iterates through all matches in the temporary queue. Similar to the previous loop, it has **O(n)** complexity.

**Overall Time Complexity for matchSchedule()**: **O(n)**, where **n** is the number of matches in the matchQueue.

* **Space Complexity Analysis**
  1. **Method addTeam()**



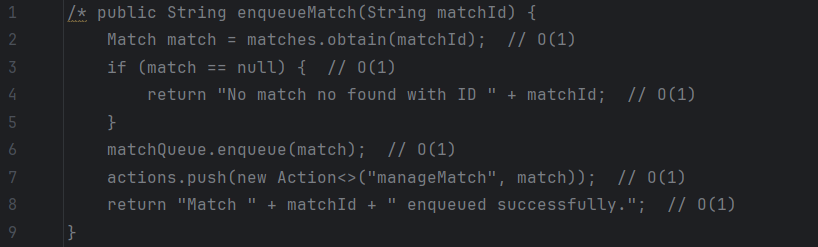
|  |  |  |  |
| --- | --- | --- | --- |
| **Tipo** | **Variable** | **Tamaño de un valor atómico** | **Cantidad de valores atómicos** |
| Entrada | **homeTeam** | **32 bits (String)** | **1** |
| Entrada | **awayTeam** | **32 bits (String)** | **1** |
| Entrada | **homeGoals** | **32 bits (int)** | **1** |
| Entrada | **awayGoals** | **32 bits (int)** | **1** |
| Entrada | **date** | **32 bits (String)** | **1** |
| Auxiliar | **home** | **32 bits (Object reference)** | **1** |
| Auxiliar | **away** | **32 bits (Object reference)** | **1** |
| Auxiliar | **matchId** | **32 bits (String)** | **1** |
| Auxiliar | **match** | **32 bits (Object reference)** | **1** |
| Auxiliar | **actions** | **32 bits (Object reference)** | **n (depends on number of actions)** |
| Salida | **result** | **32 bits (String)** | **1** |

**Line-by-Line Space Analysis**:

* **Lines 2-3**: Checking if the team already exists doesn't require additional space.
* **Line 5**: Creating a new Team object takes up space based on its attributes (name, country, titles, coefficient). This takes **O(1)** space.
* **Line 6**: The insertion into the teams hash table takes up space proportional to the number of teams, i.e., **O(n)**.
* **Line 8**: Pushing an action onto the actions stack requires space **O(n)**.
* **Line 9**: Adding the team to the ranking takes **O(n)** space.

**Overall Space Complexity for addTeam()**: **O(n)**, where **n** is the number of teams and the number of actions in the stack.

**Method enqueueMatch()**



|  |  |  |  |
| --- | --- | --- | --- |
| Tipo | Variable | Tamaño de un valor atómico | Cantidad de valores atómicos |
| Entrada | **matchQueue** | **32 bits (Queue)** | **n (depends on number of matches)** |
| Auxiliar | **schedule** | **32 bits (String)** | **1** |
| Auxiliar | **queue** | **32 bits (Queue)** | **n (depends on number of matches)** |
| Auxiliar | **match** | **32 bits (Object reference)** | **n (depends on number of matches)** |
| Auxiliar | **matchString** | **32 bits (String)** | **n (depends on number of matches)** |
| Salida | **schedule** | **32 bits (String)** | **1** |

**Line-by-Line Space Analysis**:

* **Lines 2**: The retrieval of a match from the matches hash table does not require additional space.
* **Lines 3-4**: Checking if the match exists doesn't require extra space.
* **Line 6**: The matchQueue.enqueue() operation adds the match to the queue. If there are **n** matches in the queue, this takes **O(n)** space.
* **Line 7**: Pushing an action onto the actions stack takes **O(n)** space.

**Overall Space Complexity for enqueueMatch()**: **O(n)**, where **n** is the number of matches in the queue and the number of actions in the stack.

**Summary of Complexities:**

**Time:**

1. **addMatch()**: **O(1)**, since all operations inside the method are constant time and executed only once.
2. **matchSchedule()**: **O(n)**, where **n** is the number of matches in the matchQueue.

**Space:**

1. **addTeam()**: **O(n)**, where **n** is the number of teams and the number of actions in the stack.
2. **enqueueMatch()**: **O(n)**, where **n** is the number of matches in the queue and the number of actions in the stack.