1688. Count of Matches in Tournament

Easy Math Simulation

Leetcode Link

Problem Description

In this problem, we are given an integer `n` which represents the number of teams participating in a tournament. The tournament follows a particular set of rules for its matches that depend on whether the number of teams is even or odd:

- If the number of teams is even, each team is paired with another team for a match, so there are `n / 2` matches, and consequently, `n / 2` teams advance to the next round.
- If the number of teams is odd, there is one team that randomly advances without playing, and the remaining teams are paired off. This results in `(n - 1) / 2` matches, and `(n - 1) / 2 + 1` teams advance to the next round, considering the one that advanced without playing.

The goal is to determine the total number of matches played by the time we have a winner for the tournament.

Intuition

The intuition for arriving at the solution relies on understanding the process of elimination in the tournament. Realize that in each match, one team is eliminated until there is only one winner. No matter the current number of teams, each match reduces the number of teams by one. Therefore, to know how many matches are played in the entire tournament, we need to count the number of times we can reduce the number of teams from `n` to just one team remaining.

Since every match eliminates one team, and we start with `n` teams, we will have to play `n - 1` matches in total for there to be a single winner left standing. This holds true regardless of whether the number of teams in the current round was even or odd. In other words, the solution simply relies on the observation that the match count equals the number of eliminations necessary to arrive at a sole winner. No matter the round or the parity of the team count, one winner requires `n - 1` eliminations/matches.

Solution Approach

The problem states that in each round of the tournament, half of the teams (or half rounded down if odd) are eliminated through matches. Since the goal is to determine the number of matches rather than simulating the entire tournament, we aim to find a more efficient approach than literally pairing teams and counting matches each round.

The solution approach is straightforward and hinges on the fact that each match results in exactly one team being eliminated, leading to one less team in the next round. Given that the tournament starts with `n` teams and ends with 1 winner, we need a total of `n - 1` eliminations to declare the winner. Since one match results in one elimination, the total number of matches will also be `n - 1`.

From a mathematical perspective, this is an arithmetic progression where the difference between consecutive elements (the number of teams in each round) is 1, making it possible to calculate the total number of elements directly, instead of iterating through the progression. This is based on the fact that no matter how teams are paired and how many teams get a bye (advance without playing), the tournament structure ensures one team is eliminated per match.

The implementation uses no additional data structures or complex algorithms. It exploits this arithmetic rule and returns the result in constant time with the Python expression:

```python
class Solution:
    def numberOfMatches(self, n: int) -> int:
        return n - 1
```

This expression simply subtracts 1 from the initial number of teams to provide the required number of matches, making it an elegant and efficient solution to the problem.

Example Walkthrough

Let's consider a small example to illustrate the solution approach. Suppose there are 7 teams participating in the tournament. Following the rules:

1. Since 7 is odd, one team gets a bye and advances without playing, and there will be `(7 - 1) / 2 = 3` matches in the first round, leaving us with 4 teams (3 winners + 1 bye).
2. Now 4 teams are even, so `4 / 2 = 2` matches occur in the second round, leaving us with 2 teams.
3. Finally, those 2 teams play one match in the third round to determine the winner.

If we count the total number of matches played, we have 3 (first round) + 2 (second round) + 1 (third round) = 6 matches.

According to our solution approach, instead of following each round step-by-step, we can directly calculate the total number of matches by subtracting 1 from the total number of teams: `7 - 1 = 6`. So, the total number of matches that will be played to determine the winner is 6, which corresponds with our step-by-step calculation.

This example corroborates our intuition and solution approach that regardless of the number of teams and without the need to simulate every round, we can determine that the total number of matches will always be one fewer than the total number of teams.

Python Solution

```python
class Solution:
    def numberOfMatches(self, teams: int) -> int:
        # In a tournament, every match eliminates one team,
        # hence the total number of matches will always be (number of teams - 1)
        # since the last remaining team doesn't need to play a match to be declared the winner.
        return teams - 1

```

Java Solution

```java
class Solution {
    // Method to determine the number of matches played in a tournament
    // where n teams are competing in a knockout format.
    public int numberOfMatches(int n) {
        // In a knockout tournament, each match eliminates one team,
        // hence the total number of matches that will be played is
        // always one less than the number of teams, because in the
        // end there will be only one winner, and no more matches can be played.
        return n - 1;
    }
}

```

C++ Solution

```cpp
class Solution {
public:
    // Function that calculates the number of matches played in a tournament
    // where each match eliminates one team, and there is exactly one winner.
    int numberOfMatches(int teams) {
        // Since in a knockout tournament, each match eliminates one team,
        // and there must be exactly one overall winner, the number of matches
        // played will always be one less than the number of teams, because
        // for each team to be eliminated, one match has to be played.
        // Thus, the total number of matches is teams - 1.
        return teams - 1;
    }
};

```

Typescript Solution

```typescript
/**
 * Calculates the total number of matches that will be played in a knockout tournament.
 *
 * @param {number} teams - The total number of teams participating in the tournament.
 * @return {number} The total number of matches played to determine a single winner.
 */
function numberOfMatches(teams: number): number {
    // In a knockout tournament, the total number of matches is always one less
    // than the number of teams because each match eliminates one team until one team remains.
    return teams - 1;
}

```

Time and Space Complexity

The time complexity of the `numberOfMatches` method is `O(1)` because it performs a single subtraction operation, regardless of the size of `n`.

The space complexity of the method is also `O(1)` because it uses a fixed amount of space; it only stores the result of the subtraction, not depending on the input size.