2005. Subtree Removal Game with Fibonacci Tree

Problem Description

In this problem, Alice and Bob are playing a game with a Fibonacci tree. A Fibonacci tree is a binary tree created using the order function `order(n)`. The game proceeds as follows:

1. Alice starts first.
2. On each turn, a player selects a node from the tree and removes that node along with its subtree.
3. The player is forced to delete the root node and loses the game.

Our task is to return `true` if Alice wins the game or `false` if Bob wins, assuming both players play optimally. A subtree of a binary tree consists of a node in the tree and all of this node's descendants. The tree itself could also be considered as a subtree of itself.

Example

Let's take `n = 7`.

According to the solution, when Alice starts first and plays optimally, the function `findGameWinner(7)` should return `true`, meaning Alice will win the game. The remainder of 7 % 6 is not 1, so the solution returns true for Alice's winning.

Solution Approach

The pattern in this problem is that if the number `n` is such that `n % 6 != 1`, then Alice will win. Otherwise, Bob wins. This pattern uses simple modulo arithmetic.

The solution to this problem is provided as follows using a simple Boolean function in a class called `Solution`.

Python Solution

```
python
class Solution:
    def findGameWinner(self, n: int) -> bool:
        return n % 6 != 1
```

Java Solution

```
java
public class Solution {
    public boolean findGameWinner(int n) {
        return n % 6 != 1;
    }
}
```

JavaScript Solution

```
javascript
class Solution {
    findGameWinner(n) {
        return n % 6 !== 1;
    }
}
```

C++ Solution

```
cpp
class Solution {
public:
    bool findGameWinner(int n) {
        return n % 6 != 1;
    }
};
```

C# Solution

```
csharp
public class Solution {
    public bool FindGameWinner(int n) {
        return n % 6 != 1;
    }
}
```

Conclusion