1295. Find Numbers with Even Number of Digits

Easy Array

Problem Description

The goal of this problem is to find out how many numbers in a given array `nums` have an even number of digits. To clarify, a number like 1234 has 4 digits, which is even, while a number like 123 has 3 digits, which is odd. So in the list [12, 123, 1234, 12345], there would be two numbers (12 and 1234) that have an even number of digits.

Intuition

The intuition behind the solution is based on understanding that the number of digits in a number can be determined by converting the number into a string and counting the length of that string. Once the number is converted to a string, it's straightforward to check if the length is even by using the modulo operator `%`. The modulo operation `len(str(v)) % 2` will be equal to 0 for numbers with an even number of digits, as any even number divided by 2 leaves a remainder of 0.

Solution Approach

The solution involves a simple and efficient approach that relies on Python's list comprehension and built-in functions. Here is a breakdown of the implementation:

1. Iterate through each number in the input list `nums` with a for loop embedded in a list comprehension.
2. Convert each number to a string by using the `str()` constructor. This is because the length of a number (how many digits it has) can be easily obtained from its string representation.
3. Use the `len()` function to count the number of characters, which represent the digits, in the string.
4. Check if the number of digits is even by applying the `%` (modulus) operator to `len(str(v))`. If the length modulo 2 is equal to 0, it means the number has an even number of digits.
5. The result of this check (`len(str(v)) % 2 == 0`) will be either `True` or `False`. In Python, `True` is equivalent to the integer 1, and `False` is equivalent to 0.
6. Use `sum()` to add up all the `True` values from the list comprehension, which corresponds to counting the numbers with an even number of digits.

No additional data structures or complex algorithms are necessary for this solution, as it's a straightforward application of string conversion and arithmetic operations to check the digit count.

Here is the single line of code implementing this approach:

```
return sum(len(str(v)) % 2 == 0 for v in nums)
```

This single line performs all the above steps, elegantly returning the count of numbers with an even number of digits in the input list `nums`.

Example Walkthrough

Let's illustrate the solution approach using a smaller example. Consider the array `nums` with the following numbers: `[555, 8001, 22, 3]`.

We want to count how many of these numbers have an even number of digits.

1. Let's start by looking at the first number in the array, which is `555`. Converting `555` to a string gives us `'555'`, which has 3 characters. The length of this string is odd because 3 % 2 is equal to 1.
2. We then look at the second number, `8001`. As a string, it's `'8001'`, which has 4 characters. The length of this string is even because 4 % 2 equals 0.
3. Moving to the third number, `22`, we convert it to a string to get `'22'`. This string has 2 characters, which is even since 2 % 2 equals 0.
4. Lastly, we have the number `3`. This becomes `'3'` as a string, which consists of only 1 character. The length is odd here because 1 % 2 equals 1.

Now, let's apply the short line of Python code from the given solution:

```
return sum(len(str(v)) % 2 == 0 for v in nums)
```

5. For each number in `nums`, we convert it to a string, count the length, and check if the length is even using the condition `len(str(v)) % 2 == 0`.
6. With our example array, the condition yields `False` for `555` and `3`, and `True` for `8001` and `22`.
7. When we sum up these boolean values (`False` being `0` and `True` being `1`), we get 0 (for `555`) + 1 (for `8001`) + 1 (for `22`) + 0 (for `3`), which equals 2.

Therefore, the function will return `2`, indicating that there are two numbers in the array `[555, 8001, 22, 3]` that have an even number of digits.

Solution Implementation

Python

```python
from typing import List

class Solution:
    def findNumbers(self, nums: List[int]) -> int:
        # Initialize a variable to count the even number of digits
        even_digit_count = 0

        # Iterate over each number in the nums list
        for number in nums:
            # Convert the current number to string to count digits
            digit_str = str(number)

            # Check if the length of digit_str is even
            if len(digit_str) % 2 == 0:
                # If it is even, increment the even_digit_count
                even_digit_count += 1

        # Return the total count of numbers with even number of digits
        return even_digit_count
```

Java

```java
class Solution {

    // Function to count numbers with even number of digits
    public int findNumbers(int[] nums) {
        int countEvenDigits = 0; // Counter for numbers with even number of digits

        // Iterate through each number in the array
        for (int num : nums) {
            // Check if the length of the number's string representation is even
            if (String.valueOf(num).length() % 2 == 0) {
                countEvenDigits++; // If even, increment the counter
            }
        }

        // Return the total count of numbers with even number of digits
        return countEvenDigits;
    }
}
```

C++

```cpp
#include <vector>
#include <string> // Include string library to use to_string function

class Solution {
public:
    // Function to find the count of numbers with an even number of digits
    int findNumbers(vector<int>& nums) {
        int evenDigitCount = 0; // Initialize the count of even digit numbers to 0

        // Loop through each number in the vector
        for (int num : nums) {
            // Convert the current number to a string
            string numAsString = to_string(num);
            // Check if the length of the string (number of digits) is even
            if (numAsString.size() % 2 == 0) {
                // If even, increment the count of even digit numbers
                ++evenDigitCount;
            }
        }

        // Return the total count of numbers with an even number of digits
        return evenDigitCount;
    }
};
```

TypeScript

```typescript
/**
 * Finds the number of elements in an array that have an even number of digits.
 * @param {number[]} nums - The array of numbers to check.
 * @return {number} The count of elements with an even number of digits.
 */
const findNumbers = (nums: number[]): number => {
    let countEvenDigits = 0; // Initialize count of numbers with even digits

    // Iterate through each number in the provided array
    for (const num of nums) {
        // Convert the number to a string and check if its length is even
        if (String(num).length % 2 === 0) {
            countEvenDigits++; // Increment the count if the number has even digits
        }
    }

    // Return the total count of numbers with even digits
    return countEvenDigits;
};
```

```python
from typing import List

class Solution:
    def findNumbers(self, nums: List[int]) -> int:
        # Initialize a variable to count the even number of digits
        even_digit_count = 0

        # Iterate over each number in the nums list
        for number in nums:
            # Convert the current number to string to count digits
            digit_str = str(number)

            # Check if the length of digit_str is even
            if len(digit_str) % 2 == 0:
                # If it is even, increment the even_digit_count
                even_digit_count += 1

        # Return the total count of numbers with even number of digits
        return even_digit_count
```

Time and Space Complexity

Time Complexity:

The time complexity of the function is `O(n*k)`, where `n` is the number of elements in the `nums` list and `k` is the average number of digits in the numbers. The reason for this is that for each number in the array, we convert it to a string (to count the number of digits). The string conversion operation is dependent on the number of digits in the number, hence the time complexity is a product of these two factors.

Space Complexity:

The space complexity is `O(1)`. This is because we only use a fixed amount of extra space: the space for the counter that the `sum()` function keeps accumulating. There are no additional data structures used that grow with the input size. The string representations of the numbers are temporary and not stored, so they do not add to the space complexity.