Problem 1295: Find Numbers with Even Number of Digits

Problem Information

Difficulty: Easy
Acceptance Rate: 0.00%
Paid Only: No

Problem Description

Given an array

nums

of integers, return how many of them contain an

even number

of digits.

Example 1:

Input:

nums = [12,345,2,6,7896]

Output:

2

Explanation:

12 contains 2 digits (even number of digits). 345 contains 3 digits (odd number of digits). 2 contains 1 digit (odd number of digits). 6 contains 1 digit (odd number of digits). 7896 contains 4 digits (even number of digits). Therefore only 12 and 7896 contain an even

number of digits.

Example 2:

Input:

nums = [555,901,482,1771]

Output:

1

Explanation:

Only 1771 contains an even number of digits.

Constraints:

1 <= nums.length <= 500

1 <= nums[i] <= 10

5

Code Snippets

C++:

```
class Solution {
public:
int findNumbers(vector<int>& nums) {

}
};
```

Java:

```
class Solution {
public int findNumbers(int[] nums) {
}
}
```

Python3:

```
class Solution:
def findNumbers(self, nums: List[int]) -> int:
```

Python:

```
class Solution(object):
def findNumbers(self, nums):
"""
:type nums: List[int]
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number[]} nums
* @return {number}
*/
var findNumbers = function(nums) {

};
```

TypeScript:

```
function findNumbers(nums: number[]): number {

};
```

C#:

```
public class Solution {
public int FindNumbers(int[] nums) {

}
}
```

C:

```
int findNumbers(int* nums, int numsSize) {

}
```

Go:

```
func findNumbers(nums []int) int {

}
```

Kotlin:

```
class Solution {
fun findNumbers(nums: IntArray): Int {

}
}
```

Swift:

```
class Solution {
func findNumbers(_ nums: [Int]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn find_numbers(nums: Vec<i32>) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[]} nums
# @return {Integer}
def find_numbers(nums)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $nums
* @return Integer
*/
function findNumbers($nums) {

}
}
```

Dart:

```
class Solution {
int findNumbers(List<int> nums) {

}
}
```

Scala:

```
object Solution {
def findNumbers(nums: Array[Int]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec find_numbers(nums :: [integer]) :: integer
def find_numbers(nums) do

end
end
```

Erlang:

```
-spec find_numbers(Nums :: [integer()]) -> integer().
find_numbers(Nums) ->
.
```

Racket:

```
(define/contract (find-numbers nums)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find Numbers with Even Number of Digits
* Difficulty: Easy
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
int findNumbers(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Find Numbers with Even Number of Digits
* Difficulty: Easy
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public int findNumbers(int[] nums) {
}
}
```

Python3 Solution:

```
"""
Problem: Find Numbers with Even Number of Digits
Difficulty: Easy
Tags: array, math

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def findNumbers(self, nums: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def findNumbers(self, nums):
"""
:type nums: List[int]
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Find Numbers with Even Number of Digits
* Difficulty: Easy
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/
```

```
/**
* @param {number[]} nums
* @return {number}
*/
var findNumbers = function(nums) {

};
```

TypeScript Solution:

```
/**
* Problem: Find Numbers with Even Number of Digits
* Difficulty: Easy
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function findNumbers(nums: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Find Numbers with Even Number of Digits
* Difficulty: Easy
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public int FindNumbers(int[] nums) {

}

}
```

C Solution:

```
/*
* Problem: Find Numbers with Even Number of Digits
* Difficulty: Easy
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

int findNumbers(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Find Numbers with Even Number of Digits
// Difficulty: Easy
// Tags: array, math
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func findNumbers(nums []int) int {

}
```

Kotlin Solution:

```
class Solution {
fun findNumbers(nums: IntArray): Int {

}
}
```

Swift Solution:

```
class Solution {
func findNumbers(_ nums: [Int]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Find Numbers with Even Number of Digits
// Difficulty: Easy
// Tags: array, math
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn find_numbers(nums: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} nums
# @return {Integer}
def find_numbers(nums)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $nums
* @return Integer
*/
function findNumbers($nums) {

}
}
```

Dart Solution:

```
class Solution {
int findNumbers(List<int> nums) {

}
}
```

Scala Solution:

```
object Solution {
def findNumbers(nums: Array[Int]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec find_numbers(nums :: [integer]) :: integer
def find_numbers(nums) do

end
end
```

Erlang Solution:

```
-spec find_numbers(Nums :: [integer()]) -> integer().
find_numbers(Nums) ->
.
```

Racket Solution:

```
(define/contract (find-numbers nums)
(-> (listof exact-integer?) exact-integer?)
)
```