Problem 2057: Smallest Index With Equal Value

Problem Information

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

Problem Description

Given a

0-indexed

integer array

nums

, return

the

smallest

index

i

of

nums

such that

i mod 10 == nums[i]

, or

-1

if such index does not exist

.

x mod y

denotes the

remainder

when

x

is divided by

y

.

Example 1:

Input:

nums = [0,1,2]

Output:

0

Explanation:

i=0: 0 mod 10 = 0 == nums[0]. i=1: 1 mod 10 = 1 == nums[1]. i=2: 2 mod 10 = 2 == nums[2]. All indices have i mod 10 == nums[i], so we return the smallest index 0.

Example 2:

Input:

nums = [4,3,2,1]

Output:

2

Explanation:

i=0: 0 mod 10 = 0 != nums[0]. i=1: 1 mod 10 = 1 != nums[1]. i=2: 2 mod 10 = 2 == nums[2]. i=3: 3 mod 10 = 3 != nums[3]. 2 is the only index which has i mod 10 == nums[i].

Example 3:

Input:

nums = [1,2,3,4,5,6,7,8,9,0]

Output:

-1

Explanation:

No index satisfies i mod 10 == nums[i].

Constraints:

1 <= nums.length <= 100

0 <= nums[i] <= 9

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

```
};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

```
}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (smallest-equal nums)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Smallest Index With Equal Value
* Difficulty: Easy
* Tags: array
*
* 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 smallestEqual(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Smallest Index With Equal Value
* Difficulty: Easy
* Tags: array
*
* 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 smallestEqual(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Smallest Index With Equal Value
Difficulty: Easy
Tags: array

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 smallestEqual(self, nums: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Smallest Index With Equal Value
* Difficulty: Easy
* Tags: array
*
* 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 smallestEqual = function(nums) {

};
```

TypeScript Solution:

```
/**
* Problem: Smallest Index With Equal Value
* Difficulty: Easy
* Tags: array
*
* 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 smallestEqual(nums: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Smallest Index With Equal Value
* Difficulty: Easy
* Tags: array
*
```

```
* 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 SmallestEqual(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Smallest Index With Equal Value
* Difficulty: Easy
* Tags: array
*
* 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 smallestEqual(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Smallest Index With Equal Value
// Difficulty: Easy
// Tags: array
//
// 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 smallestEqual(nums []int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Smallest Index With Equal Value
// Difficulty: Easy
// Tags: array
//
// 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 smallest_equal(nums: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (smallest-equal nums)
(-> (listof exact-integer?) exact-integer?)
)
```