Problem 3550: Smallest Index With Digit Sum Equal to Index

Problem Information

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

Problem Description

You are given an integer array

nums

.

Return the

smallest

index

i

such that the sum of the digits of

nums[i]

is equal to

i

.

If no such index exists, return

-1

.

Example 1:

Input:

nums = [1,3,2]

Output:

2

Explanation:

For

nums[2] = 2

, the sum of digits is 2, which is equal to index

i = 2

. Thus, the output is 2.

Example 2:

Input:

nums = [1,10,11]

Output:

1

Explanation:

For

nums[1] = 10

, the sum of digits is

1 + 0 = 1

, which is equal to index

i = 1

.

For

nums[2] = 11

, the sum of digits is

1 + 1 = 2

, which is equal to index

i = 2

.

Since index 1 is the smallest, the output is 1.

Example 3:

Input:

nums = [1,2,3]

Output:

-1

Explanation:

Since no index satisfies the condition, the output is -1.

Constraints:

1 <= nums.length <= 100

0 <= nums[i] <= 1000

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

```
:rtype: int
"""
```

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Smallest Index With Digit Sum Equal to Index
* 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 smallestIndex(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Smallest Index With Digit Sum Equal to Index
* 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 smallestIndex(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Smallest Index With Digit Sum Equal to Index
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 smallestIndex(self, nums: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Smallest Index With Digit Sum Equal to Index
* 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 smallestIndex = function(nums) {

};
```

TypeScript Solution:

```
/**
* Problem: Smallest Index With Digit Sum Equal to Index
* 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 smallestIndex(nums: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Smallest Index With Digit Sum Equal to Index
* 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 SmallestIndex(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Smallest Index With Digit Sum Equal to Index
* 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 smallestIndex(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Smallest Index With Digit Sum Equal to Index
// 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 smallestIndex(nums []int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Smallest Index With Digit Sum Equal to Index
// 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 smallest_index(nums: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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