Problem 2239: Find Closest Number to Zero

Problem Information

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

Problem Description

Given an integer array

nums

of size

n

, return

the number with the value

closest

to

0

in

nums

. If there are multiple answers, return

the number with the

largest

value

.

Example 1:

Input:

nums = [-4,-2,1,4,8]

Output:

1

Explanation:

The distance from -4 to 0 is |-4| = 4. The distance from -2 to 0 is |-2| = 2. The distance from 1 to 0 is |1| = 1. The distance from 4 to 0 is |4| = 4. The distance from 8 to 0 is |8| = 8. Thus, the closest number to 0 in the array is 1.

Example 2:

Input:

nums = [2,-1,1]

Output:

1

Explanation:

1 and -1 are both the closest numbers to 0, so 1 being larger is returned.

Constraints:

1 <= n <= 1000

-10

5

<= nums[i] <= 10

5

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}

}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Find Closest Number to Zero
* 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 findClosestNumber(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Find Closest Number to Zero
* 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 findClosestNumber(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Find Closest Number to Zero
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 findClosestNumber(self, nums: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Find Closest Number to Zero
* 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 findClosestNumber = function(nums) {

};
```

TypeScript Solution:

```
/**
* Problem: Find Closest Number to Zero
* 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 findClosestNumber(nums: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Find Closest Number to Zero
* 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 FindClosestNumber(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Find Closest Number to Zero
* 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 findClosestNumber(int* nums, int numsSize) {
}
```

Go Solution:

```
// Problem: Find Closest Number to Zero
// 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 findClosestNumber(nums []int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Find Closest Number to Zero
// 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 find_closest_number(nums: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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



}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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