Problem 2441: Largest Positive Integer That Exists With Its Negative

Problem Information

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

Problem Description

Given an integer array

nums

that

does not contain

any zeros, find

the largest positive

integer

k

such that

-k

also exists in the array.

Return

the positive integer

k

. If there is no such integer, return

-1

.

Example 1:

Input:

nums = [-1,2,-3,3]

Output:

3

Explanation:

3 is the only valid k we can find in the array.

Example 2:

Input:

nums = [-1,10,6,7,-7,1]

Output:

7

Explanation:

Both 1 and 7 have their corresponding negative values in the array. 7 has a larger value.

Example 3:

Input:

nums = [-10,8,6,7,-2,-3]

Output:

-1

Explanation:

There is no a single valid k, we return -1.

Constraints:

1 <= nums.length <= 1000

-1000 <= nums[i] <= 1000

nums[i] != 0

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

/**
```

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Largest Positive Integer That Exists With Its Negative
* Difficulty: Easy
* Tags: array, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

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

}
};
```

Java Solution:

```
/**
* Problem: Largest Positive Integer That Exists With Its Negative
* Difficulty: Easy
* Tags: array, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

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

}
```

```
}
```

Python3 Solution:

```
"""
Problem: Largest Positive Integer That Exists With Its Negative
Difficulty: Easy
Tags: array, hash, sort

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Largest Positive Integer That Exists With Its Negative
* Difficulty: Easy
* Tags: array, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

/**
```

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

};
```

TypeScript Solution:

```
/**
* Problem: Largest Positive Integer That Exists With Its Negative
* Difficulty: Easy
* Tags: array, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

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

};
```

C# Solution:

```
/*
* Problem: Largest Positive Integer That Exists With Its Negative
* Difficulty: Easy
* Tags: array, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

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

}
}
```

C Solution:

```
/*
* Problem: Largest Positive Integer That Exists With Its Negative
* Difficulty: Easy
* Tags: array, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

int findMaxK(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Largest Positive Integer That Exists With Its Negative
// Difficulty: Easy
// Tags: array, hash, sort
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func findMaxK(nums []int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

```
}
}
```

Rust Solution:

```
// Problem: Largest Positive Integer That Exists With Its Negative
// Difficulty: Easy
// Tags: array, hash, sort
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

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

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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