Problem 1133: Largest Unique Number

Problem Information

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

Problem Description

Given an integer array

nums

, return

the largest integer that only occurs once

. If no integer occurs once, return

-1

.

Example 1:

Input:

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

Output:

8

Explanation:

The maximum integer in the array is 9 but it is repeated. The number 8 occurs only once, so it is the answer.

Example 2:

Input:

nums = [9,9,8,8]

Output:

-1

Explanation:

There is no number that occurs only once.

Constraints:

1 <= nums.length <= 2000

0 <= nums[i] <= 1000

Code Snippets

C++:

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

}
};
```

Java:

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

```
}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (largest-unique-number nums)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Largest Unique Number
* 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 largestUniqueNumber(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Largest Unique Number
* 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 largestUniqueNumber(int[] nums) {

}
```

```
}
```

Python3 Solution:

```
"""
Problem: Largest Unique Number
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 largestUniqueNumber(self, nums: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Largest Unique Number
* 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 largestUniqueNumber = function(nums) {

};
```

TypeScript Solution:

```
/**
* Problem: Largest Unique Number
* 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 largestUniqueNumber(nums: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Largest Unique Number
* 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 LargestUniqueNumber(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Largest Unique Number
* 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 largestUniqueNumber(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Largest Unique Number
// 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 largestUniqueNumber(nums []int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

```
}
}
```

Rust Solution:

```
// Problem: Largest Unique Number
// 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 largest_unique_number(nums: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (largest-unique-number nums)
(-> (listof exact-integer?) exact-integer?)
)
```