Problem 414: Third Maximum Number

Problem Information

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

Problem Description

Given an integer array

nums

, return

the

third distinct maximum

number in this array. If the third maximum does not exist, return the

maximum

number

.

Example 1:

Input:

nums = [3,2,1]

Output:

1

Explanation:

The first distinct maximum is 3. The second distinct maximum is 2. The third distinct maximum is 1.

Example 2:

Input:

nums = [1,2]

Output:

2

Explanation:

The first distinct maximum is 2. The second distinct maximum is 1. The third distinct maximum does not exist, so the maximum (2) is returned instead.

Example 3:

Input:

nums = [2,2,3,1]

Output:

1

Explanation:

The first distinct maximum is 3. The second distinct maximum is 2 (both 2's are counted together since they have the same value). The third distinct maximum is 1.

Constraints:

1 <= nums.length <= 10

4

-2

31

<= nums[i] <= 2

31

- 1

Follow up:

Can you find an

O(n)

solution?

Code Snippets

C++:

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

}
};
```

Java:

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

}
```

```
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Third Maximum Number
* Difficulty: Easy
* Tags: array, sort
*
* 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 thirdMax(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Third Maximum Number
* Difficulty: Easy
* Tags: array, sort
*
* 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 thirdMax(int[] nums) {

}
```

```
}
```

Python3 Solution:

```
"""
Problem: Third Maximum Number
Difficulty: Easy
Tags: array, sort

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Third Maximum Number
* Difficulty: Easy
* Tags: array, sort
*
* 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 thirdMax = function(nums) {

};
```

TypeScript Solution:

```
/**
* Problem: Third Maximum Number
* Difficulty: Easy
* Tags: array, sort
*
* 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 thirdMax(nums: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Third Maximum Number
* Difficulty: Easy
* Tags: array, sort
*
* 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 ThirdMax(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Third Maximum Number
* Difficulty: Easy
* Tags: array, sort
*
* 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 thirdMax(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Third Maximum Number
// Difficulty: Easy
// Tags: array, sort
//
// 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 thirdMax(nums []int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

```
}
}
```

Rust Solution:

```
// Problem: Third Maximum Number
// Difficulty: Easy
// Tags: array, sort
//
// 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 third_max(nums: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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