Problem 3371: Identify the Largest Outlier in an Array

Problem Information

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

Problem Description

You are given an integer array

nums

. This array contains

n

elements, where

exactly

n - 2

elements are

special

numbers

. One of the remaining

two

elements is the

sum

of these

special numbers

, and the other is an

outlier

.

An

outlier

is defined as a number that is

neither

one of the original special numbers

nor

the element representing the sum of those numbers.

Note

that special numbers, the sum element, and the outlier must have

distinct

indices, but

may

share the

same

value.

Return the

largest

potential

outlier

in

nums

.

Example 1:

Input:

nums = [2,3,5,10]

Output:

10

Explanation:

The special numbers could be 2 and 3, thus making their sum 5 and the outlier 10.

Example 2:

Input:

nums = [-2,-1,-3,-6,4]

Output:

4

Explanation:

The special numbers could be -2, -1, and -3, thus making their sum -6 and the outlier 4.

Example 3:

Input:

nums = [1,1,1,1,1,5,5]

Output:

5

Explanation:

The special numbers could be 1, 1, 1, 1, and 1, thus making their sum 5 and the other 5 as the outlier.

Constraints:

3 <= nums.length <= 10

5

-1000 <= nums[i] <= 1000

The input is generated such that at least

one

potential outlier exists in

nums

.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

```
};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
```

```
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (get-largest-outlier nums)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Identify the Largest Outlier in an Array
* Difficulty: Medium
* Tags: array, hash
*
* 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 getLargestOutlier(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Identify the Largest Outlier in an Array
* Difficulty: Medium
* Tags: array, hash
*
* 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 getLargestOutlier(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Identify the Largest Outlier in an Array
Difficulty: Medium
Tags: array, hash

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Identify the Largest Outlier in an Array
* Difficulty: Medium
* Tags: array, hash
*
* 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 getLargestOutlier = function(nums) {

};
```

TypeScript Solution:

```
/**
* Problem: Identify the Largest Outlier in an Array
* Difficulty: Medium
* Tags: array, hash
*
* 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 getLargestOutlier(nums: number[]): number {
```

```
};
```

C# Solution:

```
/*
* Problem: Identify the Largest Outlier in an Array
* Difficulty: Medium
* Tags: array, hash
*
* 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 GetLargestOutlier(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Identify the Largest Outlier in an Array
* Difficulty: Medium
* Tags: array, hash
*
* 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 getLargestOutlier(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Identify the Largest Outlier in an Array
// Difficulty: Medium
```

```
// Tags: array, hash
//
// 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 getLargestOutlier(nums []int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Identify the Largest Outlier in an Array
// Difficulty: Medium
// Tags: array, hash
//
// 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 get_largest_outlier(nums: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

```
end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (get-largest-outlier nums)
(-> (listof exact-integer?) exact-integer?)
)
```