Problem 2740: Find the Value of the Partition

Problem Information

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

Problem Description

You are given a

positive

integer array

nums

.

Partition

nums

into two arrays,

nums1

and

nums2

, such that:

Each element of the array

nums

belongs to either the array

nums1

or the array

nums2

.

Both arrays are

non-empty

.

The value of the partition is

minimized

.

The value of the partition is

|max(nums1) - min(nums2)|

.

Here,

max(nums1)

denotes the maximum element of the array

nums1

, and

min(nums2)

denotes the minimum element of the array

nums2

.

Return

the integer denoting the value of such partition

.

Example 1:

Input:

nums = [1,3,2,4]

Output:

1

Explanation:

We can partition the array nums into nums1 = [1,2] and nums2 = [3,4]. - The maximum element of the array nums1 is equal to 2. - The minimum element of the array nums2 is equal to 3. The value of the partition is |2 - 3| = 1. It can be proven that 1 is the minimum value out of all partitions.

Example 2:

Input:

nums = [100,1,10]

Output:

9

Explanation:

We can partition the array nums into nums1 = [10] and nums2 = [100,1]. - The maximum element of the array nums1 is equal to 10. - The minimum element of the array nums2 is equal to 1. The value of the partition is |10 - 1| = 9. It can be proven that 9 is the minimum value out of all partitions.

Constraints:

2 <= nums.length <= 10

5

1 <= nums[i] <= 10

9

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $nums
* @return Integer
```

```
*/
function findValueOfPartition($nums) {

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (find-value-of-partition nums)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find the Value of the Partition
* Difficulty: Medium
* 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 findValueOfPartition(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Find the Value of the Partition
* Difficulty: Medium
* 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 findValueOfPartition(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Find the Value of the Partition
Difficulty: Medium
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 findValueOfPartition(self, nums: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Find the Value of the Partition
* Difficulty: Medium
* 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 findValueOfPartition = function(nums) {
```

```
};
```

TypeScript Solution:

```
/**
* Problem: Find the Value of the Partition
* Difficulty: Medium
* 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 findValueOfPartition(nums: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Find the Value of the Partition
* Difficulty: Medium
* 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 FindValueOfPartition(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Find the Value of the Partition
* Difficulty: Medium
```

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

}
```

Go Solution:

```
// Problem: Find the Value of the Partition
// Difficulty: Medium
// 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 findValueOfPartition(nums []int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Find the Value of the Partition
// Difficulty: Medium
// 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 find_value_of_partition(nums: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

```
}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (find-value-of-partition nums)
(-> (listof exact-integer?) exact-integer?)
)
```