

Problem 2780: Minimum Index of a Valid Split

Problem Information

Difficulty: **Medium**

Acceptance Rate: 0.00%

Paid Only: No

Problem Description

An element

x

of an integer array

`arr`

of length

m

is

dominant

if

more than half

the elements of

`arr`

have a value of

x

.

You are given a

0-indexed

integer array

nums

of length

n

with one

dominant

element.

You can split

nums

at an index

i

into two arrays

nums[0, ..., i]

and

nums[i + 1, ..., n - 1]

, but the split is only

valid

if:

$0 \leq i < n - 1$

`nums[0, ..., i]`

, and

`nums[i + 1, ..., n - 1]`

have the same dominant element.

Here,

`nums[i, ..., j]`

denotes the subarray of

`nums`

starting at index

`i`

and ending at index

`j`

, both ends being inclusive. Particularly, if

$j < i$

then

`nums[i, ..., j]`

denotes an empty subarray.

Return

the

minimum

index of a

valid split

. If no valid split exists, return

-1

.

Example 1:

Input:

nums = [1,2,2,2]

Output:

2

Explanation:

We can split the array at index 2 to obtain arrays [1,2,2] and [2]. In array [1,2,2], element 2 is dominant since it occurs twice in the array and $2 * 2 > 3$. In array [2], element 2 is dominant since it occurs once in the array and $1 * 2 > 1$. Both [1,2,2] and [2] have the same dominant element as nums, so this is a valid split. It can be shown that index 2 is the minimum index of a valid split.

Example 2:

Input:

nums = [2,1,3,1,1,1,7,1,2,1]

Output:

4

Explanation:

We can split the array at index 4 to obtain arrays [2,1,3,1,1] and [1,7,1,2,1]. In array [2,1,3,1,1], element 1 is dominant since it occurs thrice in the array and $3 * 2 > 5$. In array [1,7,1,2,1], element 1 is dominant since it occurs thrice in the array and $3 * 2 > 5$. Both [2,1,3,1,1] and [1,7,1,2,1] have the same dominant element as nums, so this is a valid split. It can be shown that index 4 is the minimum index of a valid split.

Example 3:

Input:

nums = [3,3,3,3,7,2,2]

Output:

-1

Explanation:

It can be shown that there is no valid split.

Constraints:

$1 \leq \text{nums.length} \leq 10$

5

$1 \leq \text{nums}[i] \leq 10$

9

nums

has exactly one dominant element.

Code Snippets

C++:

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

    }
};
```

Java:

```
class Solution {
    public int minimumIndex(List<Integer> nums) {

    }
}
```

Python3:

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

Python:

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

JavaScript:

```
/**
 * @param {number[]} nums
 * @return {number}
 */
```

```
var minimumIndex = function(nums) {  
  
};
```

TypeScript:

```
function minimumIndex(nums: number[]): number {  
  
};
```

C#:

```
public class Solution {  
    public int MinimumIndex(IList<int> nums) {  
  
    }  
}
```

C:

```
int minimumIndex(int* nums, int numsSize) {  
  
}
```

Go:

```
func minimumIndex(nums []int) int {  
  
}
```

Kotlin:

```
class Solution {  
    fun minimumIndex(nums: List<Int>): Int {  
  
    }  
}
```

Swift:

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

```
}  
}
```

Rust:

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

Ruby:

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

PHP:

```
class Solution {  
  
    /**  
     * @param Integer[] $nums  
     * @return Integer  
     */  
    function minimumIndex($nums) {  
  
    }  
}
```

Dart:

```
class Solution {  
    int minimumIndex(List<int> nums) {  
  
    }  
}
```

Scala:


```

object Solution {
  def minimumIndex(nums: List[Int]): Int = {

  }
}

```

Elixir:

```

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

  end
end

```

Erlang:

```

-spec minimum_index(Nums :: [integer()]) -> integer().
minimum_index(Nums) ->
.

```

Racket:

```

(define/contract (minimum-index nums)
  (-> (listof exact-integer?) exact-integer?)
  )

```

Solutions

C++ Solution:

```

/*
 * Problem: Minimum Index of a Valid Split
 * Difficulty: Medium
 * 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 minimumIndex(vector<int>& nums) {

    }

};

```

Java Solution:

```

/**
 * Problem: Minimum Index of a Valid Split
 * Difficulty: Medium
 * 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 minimumIndex(List<Integer> nums) {

    }

}

```

Python3 Solution:

```

"""
Problem: Minimum Index of a Valid Split
Difficulty: Medium
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 minimumIndex(self, nums: List[int]) -> int:
        # TODO: Implement optimized solution
        pass

```

Python Solution:

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

JavaScript Solution:

```
/**
 * Problem: Minimum Index of a Valid Split
 * Difficulty: Medium
 * 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 minimumIndex = function(nums) {

};
```

TypeScript Solution:

```
/**
 * Problem: Minimum Index of a Valid Split
 * Difficulty: Medium
 * 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 minimumIndex(nums: number[]): number {
```

```
};
```

C# Solution:

```
/*
 * Problem: Minimum Index of a Valid Split
 * Difficulty: Medium
 * 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 MinimumIndex(IList<int> nums) {

    }
}
```

C Solution:

```
/*
 * Problem: Minimum Index of a Valid Split
 * Difficulty: Medium
 * 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 minimumIndex(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Minimum Index of a Valid Split
// Difficulty: Medium
```

```

// 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 minimumIndex(nums []int) int {

}

```

Kotlin Solution:

```

class Solution {
    fun minimumIndex(nums: List<Int>): Int {

    }
}

```

Swift Solution:

```

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

    }
}

```

Rust Solution:

```

// Problem: Minimum Index of a Valid Split
// Difficulty: Medium
// 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 minimum_index(nums: Vec<i32>) -> i32 {

    }
}

```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

    }

}
```

Dart Solution:

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

  }
}
```

Scala Solution:

```
object Solution {
  def minimumIndex(nums: List[Int]): Int = {

  }
}
```

Elixir Solution:

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

```
end  
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (minimum-index nums)  
  (-> (listof exact-integer?) exact-integer?)  
  )
```