Problem 2936: Number of Equal Numbers Blocks

Problem Information

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

Problem Description

You are given a

0-indexed

array of integers,

nums

. The following property holds for

nums

:

All occurrences of a value are adjacent. In other words, if there are two indices

i < j

such that

nums[i] == nums[j]

, then for every index

k

that

$i < k < j$

,

$nums[k] == nums[i]$

.

Since

nums

is a very large array, you are given an instance of the class

BigArray

which has the following functions:

int at(long long index)

: Returns the value of

$nums[i]$

.

void size()

: Returns

$nums.length$

.

Let's partition the array into

maximal

blocks such that each block contains

equal values

. Return

the number of these blocks.

Note

that if you want to test your solution using a custom test, behavior for tests with

nums.length > 10

is undefined.

Example 1:

Input:

nums = [3,3,3,3,3]

Output:

1

Explanation:

There is only one block here which is the whole array (because all numbers are equal) and that is: [

3,3,3,3,3

]. So the answer would be 1.

Example 2:

Input:

nums = [1,1,1,3,9,9,9,2,10,10]

Output:

5

Explanation:

There are 5 blocks here: Block number 1: [

1,1,1

,3,9,9,9,2,10,10] Block number 2: [1,1,1,

3

,9,9,9,2,10,10] Block number 3: [1,1,1,3,

9,9,9

,2,10,10] Block number 4: [1,1,1,3,9,9,9,

2

,10,10] Block number 5: [1,1,1,3,9,9,9,2,

10,10

] So the answer would be 5.

Example 3:

Input:

nums = [1,2,3,4,5,6,7]

Output:

7

Explanation:

Since all numbers are distinct, there are 7 blocks here and each element representing one block. So the answer would be 7.

Constraints:

1 <= nums.length <= 10

15

1 <= nums[i] <= 10

9

The input is generated such that all equal values are adjacent.

The sum of the elements of

nums

is at most

10

15

.

Code Snippets

C++:

```
/**
* Definition for BigArray.
* class BigArray {
```

```
* public:
* BigArray(vector<int> elements);
* int at(long long index);
* long long size();
* };
*/
class Solution {
public:
int countBlocks(BigArray* nums) {

}
};
```

Java:

```
/**
* Definition for BigArray.
* class BigArray {
* public BigArray(int[] elements);
* public int at(long index);
* public long size();
* }
*/
class Solution {
public int countBlocks(BigArray nums) {

}
}
```

Python3:

```
# Definition for BigArray.
# class BigArray:
# def at(self, index: long) -> int:
# pass
# def size(self) -> long:
# pass
class Solution(object):
def countBlocks(self, nums: Optional['BigArray']) -> int:
```

Python:

```
# Definition for BigArray.
# class BigArray:
# def at(self, index):
# pass
# def size(self):
# pass
class Solution(object):
def countBlocks(self, nums):
"""
:type nums: BigArray
:rtype: int
"""
```

JavaScript:

```
/**
* Definition for BigArray.
* class BigArray {
* @param {number[]} elements
* constructor(elements);
*
* @param {number} index
* @return {number}
* at(index);
*
* @return {number}
* size();
* }
*/
/**
* @param {BigArray} nums
* @return {number}
*/
var countBlocks = function(nums) {

};
```

TypeScript:

```
/**
* Definition for BigArray.
* class BigArray {
* constructor(elements: number[]);
```

```
* public at(index: number): number;
* public size(): number;
* }
*/
function countBlocks(nums: BigArray | null): number {

};
```

C#:

```
/**
* Definition for BigArray.
* class BigArray {
* public BigArray(int[] elements);
* public int at(long index);
* public long size();
* }
*/
public class Solution {
public int CountBlocks(BigArray nums) {

}
}
```

C:

```
/**
* Definition for BigArray.
*
* YOU HAVE TO PASS THE OBJECT ITSELF AS THE FIRST PARAMETER
*
* struct BigArray {
* int (*at)(struct BigArray*, long long);
* long long (*size)(struct BigArray*);
* };
*/
int countBlocks(struct BigArray* nums){

}
```

Go:

```
/**
* Definition for BigArray.
* type BigArray interface {
* At(int64) int
* Size() int64
* }
*/
func countBlocks(nums BigArray) int {

}
```

Kotlin:

```
/**
* Definition for BigArray.
* class BigArray(elements: IntArray) {
* fun at(index: Long): Int
* fun size(): Long
* }
*/
class Solution {
fun countBlocks(nums: BigArray): Int {

}
}
```

Swift:

```
/**
* Definition for BigArray.
* class BigArray {
* init(elements: [Int]) {}
* func at(_ index: Int) -> Int {}
* func size() -> Int {}
* }
*/
class Solution {
func countBlocks(_ nums: BigArray) -> Int {

}
}
```

Rust:

```
/**
* Definition for BigArray.
* impl BigArray {
* pub fn new(elements: Vec<i32>) -> Self {}
* pub fn at(&self, usize) -> i32 {}
* pub fn size(&self) -> usize {}
* }
*/
impl Solution {
pub fn count_blocks(nums: BigArray) -> i32 {

}
}
```

Ruby:

```
# Definition for BigArray.
# class BigArray
# def initialize(elements)
# end
# def at(index)
# end
# def size
# end
# end
# @param {BigArray} nums
# @return {Integer}
def count_blocks(nums)

end
```

PHP:

```
/**
* Definition for BigArray.
* class BigArray {
* * @param Integer[] $elements
* function __construct($elements);
* * @param Integer $index
* * @return Integer
* function at($index);
* * @return Integer
* function size();
```

```
* }
*/
class Solution {

/**
* @param BigArray $nums
* @return Integer
*/
function countBlocks($nums) {

}
}
```

Dart:

```
/**
* Definition for BigArray.
* class BigArray {
* BigArray(List<int> elements);
* int at(int);
* int size();
* }
*/
class Solution {
int countBlocks(BigArray nums) {

}
}
```

Scala:

```
/**
* Definition for BigArray.
* class BigArray(elements: Array[Int]) {
* def at(Long): Int
* def size(): Long
* }
*/
object Solution {
def countBlocks(nums: BigArray): Int = {

}
```

```
}
```

Solutions

C++ Solution:

```
/*
* Problem: Number of Equal Numbers Blocks
* Difficulty: Medium
* Tags: array, search
*
* 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
*/

/**
* Definition for BigArray.
* class BigArray {
* public:
* BigArray(vector<int> elements);
* int at(long long index);
* long long size();
* };
*/
class Solution {
public:
int countBlocks(BigArray* nums) {

}
};
```

Java Solution:

```
/**
* Problem: Number of Equal Numbers Blocks
* Difficulty: Medium
* Tags: array, search
*
* 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
*/

/**
* Definition for BigArray.
* class BigArray {
* public BigArray(int[] elements);
* public int at(long index);
* public long size();
* }
*/
class Solution {
public int countBlocks(BigArray nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Number of Equal Numbers Blocks
Difficulty: Medium
Tags: array, search

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
"""

# Definition for BigArray.
# class BigArray:
# def at(self, index: long) -> int:
# pass
# def size(self) -> long:
# pass
class Solution(object):
def countBlocks(self, nums: Optional['BigArray']) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
# Definition for BigArray.
# class BigArray:
# def at(self, index):
# pass
# def size(self):
# pass
class Solution(object):
def countBlocks(self, nums):
"""
:type nums: BigArray
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Number of Equal Numbers Blocks
* Difficulty: Medium
* Tags: array, search
*
* 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
*/

/**
* Definition for BigArray.
* class BigArray {
* @param {number[]} elements
* constructor(elements);
*
* @param {number} index
* @return {number}
* at(index);
*
* @return {number}
* size();
* }
*/
/**
* @param {BigArray} nums
```

```
* @return {number}
*/
var countBlocks = function(nums) {

};
```

TypeScript Solution:

```
/**
* Problem: Number of Equal Numbers Blocks
* Difficulty: Medium
* Tags: array, search
*
* 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
*/

/**
* Definition for BigArray.
* class BigArray {
* constructor(elements: number[]);
* public at(index: number): number;
* public size(): number;
* }
*/
function countBlocks(nums: BigArray | null): number {

};
```

C# Solution:

```
/*
* Problem: Number of Equal Numbers Blocks
* Difficulty: Medium
* Tags: array, search
*
* 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
*/
```

```
/**
* Definition for BigArray.
* class BigArray {
* public BigArray(int[] elements);
* public int at(long index);
* public long size();
* }
*/
public class Solution {
public int CountBlocks(BigArray nums) {

}
}
```

C Solution:

```
/*
* Problem: Number of Equal Numbers Blocks
* Difficulty: Medium
* Tags: array, search
*
* 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
*/

/**
* Definition for BigArray.
*
* YOU HAVE TO PASS THE OBJECT ITSELF AS THE FIRST PARAMETER
*
* struct BigArray {
* int (*at)(struct BigArray*, long long);
* long long (*size)(struct BigArray*);
* };
*/
int countBlocks(struct BigArray* nums){

}
```

Go Solution:

```
// Problem: Number of Equal Numbers Blocks
// Difficulty: Medium
// Tags: array, search
//
// 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

/**
* Definition for BigArray.
* type BigArray interface {
* At(int64) int
* Size() int64
* }
*/
func countBlocks(nums BigArray) int {

}
```

Kotlin Solution:

```
/**
* Definition for BigArray.
* class BigArray(elements: IntArray) {
* fun at(index: Long): Int
* fun size(): Long
* }
*/
class Solution {
fun countBlocks(nums: BigArray): Int {

}
}
```

Swift Solution:

```
/**
* Definition for BigArray.
* class BigArray {
* init(elements: [Int]) {}
* func at(_ index: Int) -> Int {}
```

```
* func size() -> Int {}
* }
*/
class Solution {
func countBlocks(_ nums: BigArray) -> Int {

}
}
```

Rust Solution:

```
// Problem: Number of Equal Numbers Blocks
// Difficulty: Medium
// Tags: array, search
//
// 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

/**
* Definition for BigArray.
* impl BigArray {
* pub fn new(elements: Vec<i32>) -> Self {}
* pub fn at(&self, usize) -> i32 {}
* pub fn size(&self) -> usize {}
* }
*/
impl Solution {
pub fn count_blocks(nums: BigArray) -> i32 {

}
}
```

Ruby Solution:

```
# Definition for BigArray.
# class BigArray
# def initialize(elements)
# end
# def at(index)
# end
```

```
# def size
# end
# end
# @param {BigArray} nums
# @return {Integer}
def count_blocks(nums)

end
```

PHP Solution:

```
/**
* Definition for BigArray.
* class BigArray {
* * @param Integer[] $elements
* function __construct($elements);
* * @param Integer $index
* * @return Integer
* function at($index);
* * @return Integer
* function size();
* }
*/
class Solution {

/**
* @param BigArray $nums
* @return Integer
*/
function countBlocks($nums) {

}
}
```

Dart Solution:

```
/**
* Definition for BigArray.
* class BigArray {
* BigArray(List<int> elements);
* int at(int);
```

```
* int size();
* }
*/
class Solution {
int countBlocks(BigArray nums) {

}
}
```

Scala Solution:

```
/**
* Definition for BigArray.
* class BigArray(elements: Array[Int]) {
* def at(Long): Int
* def size(): Long
* }
*/
object Solution {
def countBlocks(nums: BigArray): Int = {

}
}
```