Problem 2831: Find the Longest Equal Subarray

Problem Information

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

Problem Description

You are given a

0-indexed

integer array

nums

and an integer

k

.

A subarray is called

equal

if all of its elements are equal. Note that the empty subarray is an

equal

subarray.

Return

the length of the

longest

possible equal subarray after deleting

at most

k

elements from

nums

.

A

subarray

is a contiguous, possibly empty sequence of elements within an array.

Example 1:

Input:

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

Output:

3

Explanation:

It's optimal to delete the elements at index 2 and index 4. After deleting them, nums becomes equal to [1, 3, 3, 3]. The longest equal subarray starts at i = 1 and ends at j = 3 with length equal to 3. It can be proven that no longer equal subarrays can be created.

Example 2:

Input:

nums = [1,1,2,2,1,1], k = 2

Output:

4

Explanation:

It's optimal to delete the elements at index 2 and index 3. After deleting them, nums becomes equal to [1, 1, 1, 1]. The array itself is an equal subarray, so the answer is 4. It can be proven that no longer equal subarrays can be created.

Constraints:

1 <= nums.length <= 10

5

1 <= nums[i] <= nums.length

0 <= k <= nums.length

Code Snippets

C++:

```
class Solution {
public:
int longestEqualSubarray(vector<int>& nums, int k) {

}
};
```

Java:

```
class Solution {
public int longestEqualSubarray(List<Integer> nums, int k) {
```

```
}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function longestEqualSubarray(nums: number[], k: number): number {

};
```

C#:

```
public class Solution {
public int LongestEqualSubarray(IList<int> nums, int k) {

}
```

```
}
```

C:

```
int longestEqualSubarray(int* nums, int numsSize, int k) {

}
```

Go:

```
func longestEqualSubarray(nums []int, k int) int {

}
```

Kotlin:

```
class Solution {
fun longestEqualSubarray(nums: List<Int>, k: Int): Int {

}
}
```

Swift:

```
class Solution {
func longestEqualSubarray(_ nums: [Int], _ k: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn longest_equal_subarray(nums: Vec<i32>, k: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[]} nums
# @param {Integer} k
```

```
# @return {Integer}
def longest_equal_subarray(nums, k)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $nums
* @param Integer $k
* @return Integer
*/
function longestEqualSubarray($nums, $k) {

}
}
```

Dart:

```
class Solution {
int longestEqualSubarray(List<int> nums, int k) {

}
}
```

Scala:

```
object Solution {
def longestEqualSubarray(nums: List[Int], k: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec longest_equal_subarray(nums :: [integer], k :: integer) :: integer
def longest_equal_subarray(nums, k) do

end
```

```
end
```

Erlang:

```
-spec longest_equal_subarray(Nums :: [integer()], K :: integer()) ->
integer().
longest_equal_subarray(Nums, K) ->
.
```

Racket:

```
(define/contract (longest-equal-subarray nums k)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find the Longest Equal Subarray
* Difficulty: Medium
* Tags: array, hash, search
*
* 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 longestEqualSubarray(vector<int>& nums, int k) {

}
};
```

Java Solution:

```
/**
* Problem: Find the Longest Equal Subarray
```

```
* Difficulty: Medium
* Tags: array, hash, search
*
* 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 longestEqualSubarray(List<Integer> nums, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Find the Longest Equal Subarray
Difficulty: Medium
Tags: array, hash, search

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Find the Longest Equal Subarray
* Difficulty: Medium
* Tags: array, hash, search
*
* 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
* @param {number} k
* @return {number}
*/
var longestEqualSubarray = function(nums, k) {

};
```

TypeScript Solution:

```
/**
* Problem: Find the Longest Equal Subarray
* Difficulty: Medium
* Tags: array, hash, search
*
* 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 longestEqualSubarray(nums: number[], k: number): number {

};
```

C# Solution:

```
/*
* Problem: Find the Longest Equal Subarray
* Difficulty: Medium
* Tags: array, hash, search
```

```
*
* 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 LongestEqualSubarray(IList<int> nums, int k) {

}
}
```

C Solution:

```
/*
* Problem: Find the Longest Equal Subarray
* Difficulty: Medium
* Tags: array, hash, search
*
* 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 longestEqualSubarray(int* nums, int numsSize, int k) {

}
```

Go Solution:

```
// Problem: Find the Longest Equal Subarray
// Difficulty: Medium
// Tags: array, hash, search
//
// 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 longestEqualSubarray(nums []int, k int) int {

}
```

Kotlin Solution:

```
class Solution {
fun longestEqualSubarray(nums: List<Int>, k: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func longestEqualSubarray(_ nums: [Int], _ k: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Find the Longest Equal Subarray
// Difficulty: Medium
// Tags: array, hash, search
//
// 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 longest_equal_subarray(nums: Vec<i32>, k: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} nums
# @param {Integer} k
# @return {Integer}
def longest_equal_subarray(nums, k)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $nums
* @param Integer $k
* @return Integer
*/
function longestEqualSubarray($nums, $k) {

}
}
```

Dart Solution:

```
class Solution {
int longestEqualSubarray(List<int> nums, int k) {

}
}
```

Scala Solution:

```
object Solution {
def longestEqualSubarray(nums: List[Int], k: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec longest_equal_subarray(nums :: [integer], k :: integer) :: integer
def longest_equal_subarray(nums, k) do

end
end
```

Erlang Solution:

```
-spec longest_equal_subarray(Nums :: [integer()], K :: integer()) ->
integer().
longest_equal_subarray(Nums, K) ->
.
```

Racket Solution:

```
(define/contract (longest-equal-subarray nums k)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```