Problem 219: Contains Duplicate II

Problem Information

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

Problem Description

Given an integer array

nums

and an integer

k

, return

true

if there are two

distinct indices

i

and

j

in the array such that

nums[i] == nums[j]

and

abs(i - j) <= k

.

Example 1:

Input:

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

Output:

true

Example 2:

Input:

nums = [1,0,1,1], k = 1

Output:

true

Example 3:

Input:

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

Output:

false

Constraints:

1 <= nums.length <= 10

5

-10

9

<= nums[i] <= 10

9

0 <= k <= 10

5

Code Snippets

C++:

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

}
};
```

Java:

```
class Solution {
public boolean containsNearbyDuplicate(int[] nums, int k) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

```
public class Solution {
public bool ContainsNearbyDuplicate(int[] nums, int k) {

}
}
```

C:

```
bool containsNearbyDuplicate(int* nums, int numsSize, int k) {

}
```

Go:

```
func containsNearbyDuplicate(nums []int, k int) bool {
}
```

Kotlin:

```
class Solution {
fun containsNearbyDuplicate(nums: IntArray, k: Int): Boolean {

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

```
object Solution {
def containsNearbyDuplicate(nums: Array[Int], k: Int): Boolean = {

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (contains-nearby-duplicate nums k)
(-> (listof exact-integer?) exact-integer? boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Contains Duplicate II
* Difficulty: Easy
* 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:
bool containsNearbyDuplicate(vector<int>& nums, int k) {

}
};
```

Java Solution:

```
/**
* Problem: Contains Duplicate II
* Difficulty: Easy
* 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 boolean containsNearbyDuplicate(int[] nums, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Contains Duplicate II
Difficulty: Easy
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 containsNearbyDuplicate(self, nums: List[int], k: int) -> bool:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Contains Duplicate II
* Difficulty: Easy
* 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
* @param {number} k
* @return {boolean}
*/
var containsNearbyDuplicate = function(nums, k) {

};
```

TypeScript Solution:

```
/**
* Problem: Contains Duplicate II
* Difficulty: Easy
* 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 containsNearbyDuplicate(nums: number[], k: number): boolean {

};
```

C# Solution:

```
/*
* Problem: Contains Duplicate II
* Difficulty: Easy
* 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 bool ContainsNearbyDuplicate(int[] nums, int k) {

}

}
```

C Solution:

```
/*
* Problem: Contains Duplicate II
* Difficulty: Easy
* 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
*/

bool containsNearbyDuplicate(int* nums, int numsSize, int k) {

}
```

Go Solution:

```
// Problem: Contains Duplicate II
// Difficulty: Easy
// 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 containsNearbyDuplicate(nums []int, k int) bool {

}
```

Kotlin Solution:

```
class Solution {
fun containsNearbyDuplicate(nums: IntArray, k: Int): Boolean {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Contains Duplicate II
// Difficulty: Easy
// 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 contains_nearby_duplicate(nums: Vec<i32>, k: i32) -> bool {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $nums
* @param Integer $k
* @return Boolean
*/
function containsNearbyDuplicate($nums, $k) {
}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

```
object Solution {
def containsNearbyDuplicate(nums: Array[Int], k: Int): Boolean = {

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (contains-nearby-duplicate nums k)
(-> (listof exact-integer?) exact-integer? boolean?)
)
```