# Problem 217: Contains Duplicate

## Problem Information

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

## Problem Description

Given an integer array

nums

, return

true

if any value appears

at least twice

in the array, and return

false

if every element is distinct.

Example 1:

Input:

nums = [1,2,3,1]

Output:

true

Explanation:

The element 1 occurs at the indices 0 and 3.

Example 2:

Input:

nums = [1,2,3,4]

Output:

false

Explanation:

All elements are distinct.

Example 3:

Input:

nums = [1,1,1,3,3,4,3,2,4,2]

Output:

true

Constraints:

1 <= nums.length <= 10

5

-10

9

<= nums[i] <= 10

9

## Code Snippets

**C++:**

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

}
};
```

**Java:**

```
class Solution {
public boolean containsDuplicate(int[] nums) {

}
}
```

**Python3:**

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

**Python:**

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

**JavaScript:**

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

};
```

**TypeScript:**

```
function containsDuplicate(nums: number[]): boolean {

};
```

**C#:**

```
public class Solution {
public bool ContainsDuplicate(int[] nums) {

}
}
```

**C:**

```
bool containsDuplicate(int* nums, int numsSize) {

}
```

**Go:**

```
func containsDuplicate(nums []int) bool {

}
```

**Kotlin:**

```
class Solution {
fun containsDuplicate(nums: IntArray): Boolean {

}
}
```

**Swift:**

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

}
}
```

**Rust:**

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

}
}
```

**Ruby:**

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

end
```

**PHP:**

```
class Solution {

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

}
}
```

**Dart:**

```
class Solution {
bool containsDuplicate(List<int> nums) {

}
}
```

### Scala:

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

}
}
```

### Elixir:

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

end
end
```

### Erlang:

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

### Racket:

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

## Solutions

### C++ Solution:

```
/*
* Problem: Contains Duplicate
* Difficulty: Easy
* 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:
bool containsDuplicate(vector<int>& nums) {

}
};
```

**Java Solution:**

```
/**
* Problem: Contains Duplicate
* Difficulty: Easy
* 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 boolean containsDuplicate(int[] nums) {

}
}
```

**Python3 Solution:**

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

```
pass
```

**Python Solution:**

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

**JavaScript Solution:**

```
/**
* Problem: Contains Duplicate
* Difficulty: Easy
* 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 {boolean}
*/
var containsDuplicate = function(nums) {

};
```

**TypeScript Solution:**

```
/**
* Problem: Contains Duplicate
* Difficulty: Easy
* 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 containsDuplicate(nums: number[]): boolean {

};
```

## C# Solution:

```
/*
* Problem: Contains Duplicate
* Difficulty: Easy
* 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 bool ContainsDuplicate(int[] nums) {

}
}
```

## C Solution:

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

bool containsDuplicate(int* nums, int numsSize) {

}
```

## Go Solution:

```
// Problem: Contains Duplicate
// Difficulty: Easy
// 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 containsDuplicate(nums []int) bool {

}
```

### Kotlin Solution:

```
class Solution {
fun containsDuplicate(nums: IntArray): Boolean {

}
}
```

### Swift Solution:

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

}
}
```

### Rust Solution:

```
// Problem: Contains Duplicate
// Difficulty: Easy
// 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 contains_duplicate(nums: Vec<i32>) -> bool {

}
}
```

**Ruby Solution:**

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

end
```

**PHP Solution:**

```
class Solution {

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

}
}
```

**Dart Solution:**

```
class Solution {
bool containsDuplicate(List<int> nums) {

}
}
```

**Scala Solution:**

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

}
}
```

**Elixir Solution:**

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

end
end
```

## Erlang Solution:

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

## Racket Solution:

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