Problem 1346: Check If N and Its Double Exist

Problem Information

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

Problem Description

Given an array

arr

of integers, check if there exist two indices

i

and

j

such that :

i != j

0 <= i, j < arr.length

arr[i] == 2 * arr[j]

Example 1:

Input:

arr = [10,2,5,3]

Output:

true

Explanation:

For i = 0 and j = 2, arr[i] == 10 == 2 * 5 == 2 * arr[j]

Example 2:

Input:

arr = [3,1,7,11]

Output:

false

Explanation:

There is no i and j that satisfy the conditions.

Constraints:

2 <= arr.length <= 500

-10

3

<= arr[i] <= 10

3

Code Snippets

C++:

```
class Solution {
public:
bool checkIfExist(vector<int>& arr) {

}
};
```

Java:

```
class Solution {
public boolean checkIfExist(int[] arr) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function checkIfExist(arr: number[]): boolean {
```

```
};
```

C#:

```
public class Solution {
public bool CheckIfExist(int[] arr) {

}
}
```

C:

```
bool checkIfExist(int* arr, int arrSize) {

}
```

Go:

```
func checkIfExist(arr []int) bool {

}
```

Kotlin:

```
class Solution {
fun checkIfExist(arr: IntArray): Boolean {

}
}
```

Swift:

```
class Solution {
func checkIfExist(_ arr: [Int]) -> Bool {

}
}
```

Rust:

```
impl Solution {
pub fn check_if_exist(arr: Vec<i32>) -> bool {
```

```
}
}
```

Ruby:

```
# @param {Integer[]} arr
# @return {Boolean}
def check_if_exist(arr)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $arr
* @return Boolean
*/
function checkIfExist($arr) {

}
}
```

Dart:

```
class Solution {
bool checkIfExist(List<int> arr) {

}
}
```

Scala:

```
object Solution {
def checkIfExist(arr: Array[Int]): Boolean = {

}
}
```

Elixir:

```
defmodule Solution do
@spec check_if_exist(arr :: [integer]) :: boolean
def check_if_exist(arr) do

end
end
```

Erlang:

```
-spec check_if_exist(Arr :: [integer()]) -> boolean().
check_if_exist(Arr) ->
.
```

Racket:

```
(define/contract (check-if-exist arr)
(-> (listof exact-integer?) boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Check If N and Its Double Exist
* Difficulty: Easy
* Tags: array, hash, sort, 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:
bool checkIfExist(vector<int>& arr) {

}
};
```

Java Solution:

```
/**
* Problem: Check If N and Its Double Exist
* Difficulty: Easy
* Tags: array, hash, sort, 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 boolean checkIfExist(int[] arr) {

}
}
```

Python3 Solution:

```
"""
Problem: Check If N and Its Double Exist
Difficulty: Easy
Tags: array, hash, sort, 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 checkIfExist(self, arr: List[int]) -> bool:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Check If N and Its Double Exist
* Difficulty: Easy
* Tags: array, hash, sort, 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[]} arr
* @return {boolean}
*/
var checkIfExist = function(arr) {

};
```

TypeScript Solution:

```
/**
* Problem: Check If N and Its Double Exist
* Difficulty: Easy
* Tags: array, hash, sort, 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 checkIfExist(arr: number[]): boolean {

};
```

C# Solution:

```
/*
* Problem: Check If N and Its Double Exist
* Difficulty: Easy
* Tags: array, hash, sort, 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 bool CheckIfExist(int[] arr) {

}
}
```

C Solution:

```
/*
* Problem: Check If N and Its Double Exist
* Difficulty: Easy
* Tags: array, hash, sort, 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
*/

bool checkIfExist(int* arr, int arrSize) {

}
```

Go Solution:

```
// Problem: Check If N and Its Double Exist
// Difficulty: Easy
// Tags: array, hash, sort, 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 checkIfExist(arr []int) bool {

}
```

Kotlin Solution:

```
class Solution {
fun checkIfExist(arr: IntArray): Boolean {

}
}
```

Swift Solution:

```
class Solution {
func checkIfExist(_ arr: [Int]) -> Bool {

}
}
```

Rust Solution:

```
// Problem: Check If N and Its Double Exist
// Difficulty: Easy
// Tags: array, hash, sort, 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 check_if_exist(arr: Vec<i32>) -> bool {

}
}
```

Ruby Solution:

```
# @param {Integer[]} arr
# @return {Boolean}
def check_if_exist(arr)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $arr
* @return Boolean
*/
function checkIfExist($arr) {

}
}
```

Dart Solution:

```
class Solution {
bool checkIfExist(List<int> arr) {

}
}
```

Scala Solution:

```
object Solution {
def checkIfExist(arr: Array[Int]): Boolean = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec check_if_exist(arr :: [integer]) :: boolean
def check_if_exist(arr) do

end
end
```

Erlang Solution:

```
-spec check_if_exist(Arr :: [integer()]) -> boolean().
check_if_exist(Arr) ->
.
```

Racket Solution:

```
(define/contract (check-if-exist arr)
(-> (listof exact-integer?) boolean?)
)
```