Problem 268: Missing Number

Problem Information

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

Problem Description

Given an array

nums

containing

n

distinct numbers in the range

[0, n]

, return

the only number in the range that is missing from the array.

Example 1:

Input:

nums = [3,0,1]

Output:

2

Explanation:

n = 3

since there are 3 numbers, so all numbers are in the range

[0,3]

. 2 is the missing number in the range since it does not appear in

nums

.

Example 2:

Input:

nums = [0,1]

Output:

2

Explanation:

n = 2

since there are 2 numbers, so all numbers are in the range

[0,2]

. 2 is the missing number in the range since it does not appear in

nums

.

Example 3:

Input:

nums = [9,6,4,2,3,5,7,0,1]

Output:

8

Explanation:

n = 9

since there are 9 numbers, so all numbers are in the range

[0,9]

. 8 is the missing number in the range since it does not appear in

nums

.

Constraints:

n == nums.length

1 <= n <= 10

4

0 <= nums[i] <= n

All the numbers of

nums

are

unique

.

Follow up:

Could you implement a solution using only

O(1)

extra space complexity and

O(n)

runtime complexity?

Code Snippets

C++:

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

}
};
```

Java:

```
class Solution {
public int missingNumber(int[] nums) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function missingNumber(nums: number[]): number {

};
```

C#:

```
public class Solution {
public int MissingNumber(int[] nums) {

}
}
```

C:

```
int missingNumber(int* nums, int numsSize) {

}
```

Go:

```
func missingNumber(nums []int) int {
}
```

Kotlin:

```
class Solution {
fun missingNumber(nums: IntArray): Int {

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

```
class Solution {
int missingNumber(List<int> nums) {

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (missing-number nums)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Missing Number
* Difficulty: Easy
* Tags: array, math, 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:
int missingNumber(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Missing Number
* Difficulty: Easy
* Tags: array, math, 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 int missingNumber(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Missing Number
Difficulty: Easy
Tags: array, math, 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 missingNumber(self, nums: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Missing Number
* Difficulty: Easy
* Tags: array, math, 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[]} nums
* @return {number}
*/
var missingNumber = function(nums) {
};
```

TypeScript Solution:

```
/**
* Problem: Missing Number
* Difficulty: Easy
* Tags: array, math, 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 missingNumber(nums: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Missing Number
* Difficulty: Easy
* Tags: array, math, 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 int MissingNumber(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Missing Number
* Difficulty: Easy
* Tags: array, math, 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
*/

int missingNumber(int* nums, int numsSize) {

}
```

Go Solution:

```go
// Problem: Missing Number
// Difficulty: Easy
// Tags: array, math, 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 missingNumber(nums []int) int {

}
```

Kotlin Solution:

```kotlin
class Solution {
fun missingNumber(nums: IntArray): Int {

}
}
```

Swift Solution:

```swift
class Solution {
func missingNumber(_ nums: [Int]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Missing Number
// Difficulty: Easy
// Tags: array, math, 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 missing_number(nums: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

```
class Solution {
int missingNumber(List<int> nums) {

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (missing-number nums)
(-> (listof exact-integer?) exact-integer?)
)
```