Problem 487: Max Consecutive Ones II

Problem Information

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

Problem Description

Given a binary array

nums

, return

the maximum number of consecutive

1

's in the array if you can flip at most one

0

.

Example 1:

Input:

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

Output:

4

Explanation:

- If we flip the first zero, nums becomes [1,1,1,1,0] and we have 4 consecutive ones. - If we flip the second zero, nums becomes [1,0,1,1,1] and we have 3 consecutive ones. The max number of consecutive ones is 4.

Example 2:

Input:

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

Output:

4

Explanation:

- If we flip the first zero, nums becomes [1,1,1,1,0,1] and we have 4 consecutive ones. - If we flip the second zero, nums becomes [1,0,1,1,1,1] and we have 4 consecutive ones. The max number of consecutive ones is 4.

Constraints:

1 <= nums.length <= 10

5

nums[i]

is either

0

or

1

.

Follow up:

What if the input numbers come in one by one as an infinite stream? In other words, you can't store all numbers coming from the stream as it's too large to hold in memory. Could you solve it efficiently?

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

```
func findMaxConsecutiveOnes(nums []int) int {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (find-max-consecutive-ones nums)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Max Consecutive Ones II
* Difficulty: Medium
* Tags: array, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/
```

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

}
};
```

Java Solution:

```
/**
* Problem: Max Consecutive Ones II
* Difficulty: Medium
* Tags: array, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public int findMaxConsecutiveOnes(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Max Consecutive Ones II
Difficulty: Medium
Tags: array, dp

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) or O(n * m) for DP table
"""

class Solution:
def findMaxConsecutiveOnes(self, nums: List[int]) -> int:
# TODO: Implement optimized solution
```

```
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Max Consecutive Ones II
* Difficulty: Medium
* Tags: array, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

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

};
```

TypeScript Solution:

```
/**
* Problem: Max Consecutive Ones II
* Difficulty: Medium
* Tags: array, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
```

```
*/

function findMaxConsecutiveOnes(nums: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Max Consecutive Ones II
* Difficulty: Medium
* Tags: array, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

public class Solution {
public int FindMaxConsecutiveOnes(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Max Consecutive Ones II
* Difficulty: Medium
* Tags: array, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

int findMaxConsecutiveOnes(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Max Consecutive Ones II
// Difficulty: Medium
// Tags: array, dp
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

func findMaxConsecutiveOnes(nums []int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Max Consecutive Ones II
// Difficulty: Medium
// Tags: array, dp
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

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

}
```

```
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (find-max-consecutive-ones nums)
(-> (listof exact-integer?) exact-integer?)
)
```