Problem 2654: Minimum Number of Operations to Make All Array Elements Equal to 1

Problem Information

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

Problem Description

You are given a

0-indexed

array

nums

consisting of

positive

integers. You can do the following operation on the array

any

number of times:

Select an index

i

such that

0 <= i < n - 1

and replace either of

nums[i]

or

nums[i+1]

with their gcd value.

Return

the

minimum

number of operations to make all elements of

nums

equal to

1

. If it is impossible, return

-1

.

The gcd of two integers is the greatest common divisor of the two integers.

Example 1:

Input:

nums = [2,6,3,4]

Output:

4

Explanation:

We can do the following operations: - Choose index i = 2 and replace nums[2] with gcd(3,4) = 1. Now we have nums = [2,6,1,4]. - Choose index i = 1 and replace nums[1] with gcd(6,1) = 1. Now we have nums = [2,1,1,4]. - Choose index i = 0 and replace nums[0] with gcd(2,1) = 1. Now we have nums = [1,1,1,4]. - Choose index i = 2 and replace nums[3] with gcd(1,4) = 1. Now we have nums = [1,1,1,1].

Example 2:

Input:

nums = [2,10,6,14]

Output:

-1

Explanation:

It can be shown that it is impossible to make all the elements equal to 1.

Constraints:

2 <= nums.length <= 50

1 <= nums[i] <= 10

6

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

```
};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

```
}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (min-operations nums)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Minimum Number of Operations to Make All Array Elements Equal to 1
* Difficulty: Medium
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

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

}
};
```

Java Solution:

```
/**
* Problem: Minimum Number of Operations to Make All Array Elements Equal to 1
* Difficulty: Medium
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

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

}
}
```

Python3 Solution:

```
"""
Problem: Minimum Number of Operations to Make All Array Elements Equal to 1
Difficulty: Medium
Tags: array, math

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Minimum Number of Operations to Make All Array Elements Equal to 1
* Difficulty: Medium
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

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

};
```

TypeScript Solution:

```
/**
* Problem: Minimum Number of Operations to Make All Array Elements Equal to 1
* Difficulty: Medium
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

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

};
```

C# Solution:

```
/*
* Problem: Minimum Number of Operations to Make All Array Elements Equal to 1
* Difficulty: Medium
* Tags: array, math
*
```

```
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

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

}
}
```

C Solution:

```
/*
* Problem: Minimum Number of Operations to Make All Array Elements Equal to 1
* Difficulty: Medium
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

int minOperations(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Minimum Number of Operations to Make All Array Elements Equal to
1
// Difficulty: Medium
// Tags: array, math
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func minOperations(nums []int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Minimum Number of Operations to Make All Array Elements Equal to
1
// Difficulty: Medium
// Tags: array, math
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

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

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

```
.
```

Racket Solution:

```
(define/contract (min-operations nums)
(-> (listof exact-integer?) exact-integer?)
)
```