Problem 2366: Minimum Replacements to Sort the Array

Problem Information

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

Problem Description

You are given a

0-indexed

integer array

nums

. In one operation you can replace any element of the array with

any two

elements that

sum

to it.

For example, consider

nums = [5,6,7]

. In one operation, we can replace

nums[1]

with

2

and

4

and convert

nums

to

[5,2,4,7]

.

Return

the minimum number of operations to make an array that is sorted in

non-decreasing

order

.

Example 1:

Input:

nums = [3,9,3]

Output:

2

Explanation:

Here are the steps to sort the array in non-decreasing order: - From [3,9,3], replace the 9 with 3 and 6 so the array becomes [3,3,6,3] - From [3,3,6,3], replace the 6 with 3 and 3 so the array becomes [3,3,3,3,3] There are 2 steps to sort the array in non-decreasing order. Therefore, we return 2.

Example 2:

Input:

nums = [1,2,3,4,5]

Output:

0

Explanation:

The array is already in non-decreasing order. Therefore, we return 0.

Constraints:

1 <= nums.length <= 10

5

1 <= nums[i] <= 10

9

Code Snippets

C++:

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

```
}
};
```

Java:

```
class Solution {
public long minimumReplacement(int[] nums) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

```
public class Solution {
public long MinimumReplacement(int[] nums) {

}
}
```

C:

```
long long minimumReplacement(int* nums, int numsSize) {

}
```

Go:

```
func minimumReplacement(nums []int) int64 {

}
```

Kotlin:

```
class Solution {
fun minimumReplacement(nums: IntArray): Long {

}
}
```

Swift:

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

}
}
```

Rust:

```
impl Solution {
pub fn minimum_replacement(nums: Vec<i32>) -> i64 {

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (minimum-replacement nums)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Minimum Replacements to Sort the Array
* Difficulty: Hard
* Tags: array, greedy, math, sort
*
* 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:
long long minimumReplacement(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Minimum Replacements to Sort the Array
* Difficulty: Hard
* Tags: array, greedy, math, sort
*
* 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 long minimumReplacement(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Minimum Replacements to Sort the Array
Difficulty: Hard
Tags: array, greedy, math, sort

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 minimumReplacement(self, nums: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Minimum Replacements to Sort the Array
* Difficulty: Hard
```

```
* Tags: array, greedy, math, sort
*
* 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 minimumReplacement = function(nums) {

};
```

TypeScript Solution:

```
/**
* Problem: Minimum Replacements to Sort the Array
* Difficulty: Hard
* Tags: array, greedy, math, sort
*
* 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 minimumReplacement(nums: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Minimum Replacements to Sort the Array
* Difficulty: Hard
* Tags: array, greedy, math, sort
*
* 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 long MinimumReplacement(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Minimum Replacements to Sort the Array
* Difficulty: Hard
* Tags: array, greedy, math, sort
*
* 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
*/

long long minimumReplacement(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Minimum Replacements to Sort the Array
// Difficulty: Hard
// Tags: array, greedy, math, sort
//
// 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 minimumReplacement(nums []int) int64 {

}
```

Kotlin Solution:

```
class Solution {
fun minimumReplacement(nums: IntArray): Long {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Minimum Replacements to Sort the Array
// Difficulty: Hard
// Tags: array, greedy, math, sort
//
// 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 minimum_replacement(nums: Vec<i32>) -> i64 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {
```

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (minimum-replacement nums)
(-> (listof exact-integer?) exact-integer?)
)
```