

Problem 1787: Make the XOR of All Segments Equal to Zero

Problem Information

Difficulty: **Hard**

Acceptance Rate: 0.00%

Paid Only: No

Problem Description

You are given an array

`nums`

and an integer

`k`

. The

XOR

of a segment

`[left, right]`

where

`left <= right`

is the

XOR

of all the elements with indices between

left

and

right

, inclusive:

$\text{nums}[\text{left}] \text{ XOR } \text{nums}[\text{left}+1] \text{ XOR } \dots \text{ XOR } \text{nums}[\text{right}]$

.

Return

the minimum number of elements to change in the array

such that the

XOR

of all segments of size

k

is equal to zero.

Example 1:

Input:

$\text{nums} = [1, 2, 0, 3, 0]$, $k = 1$

Output:

3

Explanation:

Modify the array from [

1

,

2

,0,

3

,0] to from [

0

,

0

,0,

0

,0].

Example 2:

Input:

nums = [3,4,5,2,1,7,3,4,7], k = 3

Output:

3

Explanation:

Modify the array from [3,4,

5

,

2

,

1

,7,3,4,7] to [3,4,

7

,

3

,

4

,7,3,4,7].

Example 3:

Input:

nums = [1,2,4,1,2,5,1,2,6], k = 3

Output:

3

Explanation:

Modify the array from [1,2,

4,

1,2,

5

,1,2,

6

] to [1,2,

3

,1,2,

3

,1,2,

3

].

Constraints:

$1 \leq k \leq \text{nums.length} \leq 2000$

$0 \leq \text{nums}[i] < 2$

10

Code Snippets

C++:

```
class Solution {  
public:
```

```

int minChanges(vector<int>& nums, int k) {

}

};

```

Java:

```

class Solution {
public int minChanges(int[] nums, int k) {

}

}

```

Python3:

```

class Solution:
def minChanges(self, nums: List[int], k: int) -> int:

```

Python:

```

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

```

JavaScript:

```

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

};

```

TypeScript:

```
function minChanges(nums: number[], k: number): number {  
  
};
```

C#:

```
public class Solution {  
    public int MinChanges(int[] nums, int k) {  
  
    }  
}
```

C:

```
int minChanges(int* nums, int numsSize, int k) {  
  
}
```

Go:

```
func minChanges(nums []int, k int) int {  
  
}
```

Kotlin:

```
class Solution {  
    fun minChanges(nums: IntArray, k: Int): Int {  
  
    }  
}
```

Swift:

```
class Solution {  
    func minChanges(_ nums: [Int], _ k: Int) -> Int {  
  
    }  
}
```

Rust:

```

impl Solution {
  pub fn min_changes(nums: Vec<i32>, k: i32) -> i32 {

  }
}

```

Ruby:

```

# @param {Integer[]} nums
# @param {Integer} k
# @return {Integer}
def min_changes(nums, k)

end

```

PHP:

```

class Solution {

  /**
   * @param Integer[] $nums
   * @param Integer $k
   * @return Integer
   */
  function minChanges($nums, $k) {

  }
}

```

Dart:

```

class Solution {
  int minChanges(List<int> nums, int k) {

  }
}

```

Scala:

```

object Solution {
  def minChanges(nums: Array[Int], k: Int): Int = {

  }
}

```

```
}
```

Elixir:

```
defmodule Solution do
  @spec min_changes(nums :: [integer], k :: integer) :: integer
  def min_changes(nums, k) do

  end
end
```

Erlang:

```
-spec min_changes(Nums :: [integer()], K :: integer()) -> integer().
min_changes(Nums, K) ->
.
```

Racket:

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

Solutions

C++ Solution:

```
/*
 * Problem: Make the XOR of All Segments Equal to Zero
 * Difficulty: Hard
 * 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 minChanges(vector<int>& nums, int k) {
```

```
}  
};
```

Java Solution:

```
/**  
 * Problem: Make the XOR of All Segments Equal to Zero  
 * Difficulty: Hard  
 * 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 minChanges(int[] nums, int k) {  
  
    }  
}
```

Python3 Solution:

```
"""  
Problem: Make the XOR of All Segments Equal to Zero  
Difficulty: Hard  
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 minChanges(self, nums: List[int], k: int) -> int:  
        # TODO: Implement optimized solution  
        pass
```

Python Solution:

```

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

```

JavaScript Solution:

```

/**
 * Problem: Make the XOR of All Segments Equal to Zero
 * Difficulty: Hard
 * 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
 * @param {number} k
 * @return {number}
 */
var minChanges = function(nums, k) {

};

```

TypeScript Solution:

```

/**
 * Problem: Make the XOR of All Segments Equal to Zero
 * Difficulty: Hard
 * 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 minChanges(nums: number[], k: number): number {

```

```
};
```

C# Solution:

```
/*
 * Problem: Make the XOR of All Segments Equal to Zero
 * Difficulty: Hard
 * 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 MinChanges(int[] nums, int k) {

    }
}
```

C Solution:

```
/*
 * Problem: Make the XOR of All Segments Equal to Zero
 * Difficulty: Hard
 * 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 minChanges(int* nums, int numsSize, int k) {

}
```

Go Solution:

```
// Problem: Make the XOR of All Segments Equal to Zero
// Difficulty: Hard
```

```
// 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 minChanges(nums []int, k int) int {

}
```

Kotlin Solution:

```
class Solution {
    fun minChanges(nums: IntArray, k: Int): Int {

    }
}
```

Swift Solution:

```
class Solution {
    func minChanges(_ nums: [Int], _ k: Int) -> Int {

    }
}
```

Rust Solution:

```
// Problem: Make the XOR of All Segments Equal to Zero
// Difficulty: Hard
// 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 min_changes(nums: Vec<i32>, k: i32) -> i32 {

    }
}
```

Ruby Solution:

```
# @param {Integer[]} nums
# @param {Integer} k
# @return {Integer}
def min_changes(nums, k)

end
```

PHP Solution:

```
class Solution {

    /**
     * @param Integer[] $nums
     * @param Integer $k
     * @return Integer
     */
    function minChanges($nums, $k) {

    }

}
```

Dart Solution:

```
class Solution {
  int minChanges(List<int> nums, int k) {

  }
}
```

Scala Solution:

```
object Solution {
  def minChanges(nums: Array[Int], k: Int): Int = {

  }
}
```

Elixir Solution:

```
defmodule Solution do
  @spec min_changes(nums :: [integer], k :: integer) :: integer
  def min_changes(nums, k) do

  end
end
```

Erlang Solution:

```
-spec min_changes(Nums :: [integer()], K :: integer()) -> integer().
min_changes(Nums, K) ->
.
```

Racket Solution:

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