

# Problem 3251: Find the Count of Monotonic Pairs II

## Problem Information

Difficulty: **Hard**

Acceptance Rate: 0.00%

Paid Only: No

## Problem Description

You are given an array of

positive

integers

nums

of length

n

.

We call a pair of

non-negative

integer arrays

(arr1, arr2)

monotonic

if:

The lengths of both arrays are

$n$

.

$arr1$

is monotonically

non-decreasing

, in other words,

$arr1[0] \leq arr1[1] \leq \dots \leq arr1[n - 1]$

.

$arr2$

is monotonically

non-increasing

, in other words,

$arr2[0] \geq arr2[1] \geq \dots \geq arr2[n - 1]$

.

$arr1[i] + arr2[i] == nums[i]$

for all

$0 \leq i \leq n - 1$

.

Return the count of

monotonic

pairs.

Since the answer may be very large, return it

modulo

10

9

+ 7

.

Example 1:

Input:

nums = [2,3,2]

Output:

4

Explanation:

The good pairs are:

([0, 1, 1], [2, 2, 1])

([0, 1, 2], [2, 2, 0])

([0, 2, 2], [2, 1, 0])

([1, 2, 2], [1, 1, 0])

Example 2:

Input:

nums = [5,5,5,5]

Output:

126

Constraints:

$1 \leq n == \text{nums.length} \leq 2000$

$1 \leq \text{nums}[i] \leq 1000$

## Code Snippets

**C++:**

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

    }
};
```

**Java:**

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

    }
}
```

**Python3:**

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

## Python:

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

## JavaScript:

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

## TypeScript:

```
function countOfPairs(nums: number[]): number {  
  
};
```

## C#:

```
public class Solution {  
    public int CountOfPairs(int[] nums) {  
  
    }  
}
```

## C:

```
int countOfPairs(int* nums, int numsSize) {  
  
}
```

## Go:

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

```
}
```

### Kotlin:

```
class Solution {  
    fun countOfPairs(nums: IntArray): Int {  
  
    }  
}
```

### Swift:

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

### Rust:

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

### Ruby:

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

### PHP:

```
class Solution {  
  
    /**  
     * @param Integer[] $nums  
     * @return Integer  
     */  
}
```

```
function countOfPairs($nums) {

}

}
```

### Dart:

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

  }
}
```

### Scala:

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

  }
}
```

### Elixir:

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

  end
end
```

### Erlang:

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

### Racket:

```
(define/contract (count-of-pairs nums)
  (-> (listof exact-integer?) exact-integer?)
)
```

## Solutions

### C++ Solution:

```
/*
 * Problem: Find the Count of Monotonic Pairs II
 * Difficulty: Hard
 * Tags: array, dp, math
 *
 * 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 countOfPairs(vector<int>& nums) {

    }
};
```

### Java Solution:

```
/**
 * Problem: Find the Count of Monotonic Pairs II
 * Difficulty: Hard
 * Tags: array, dp, math
 *
 * 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 countOfPairs(int[] nums) {

    }
}
```

### Python3 Solution:



```

"""
Problem: Find the Count of Monotonic Pairs II
Difficulty: Hard
Tags: array, dp, math

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

```

### Python Solution:

```

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

```

### JavaScript Solution:

```

/**
 * Problem: Find the Count of Monotonic Pairs II
 * Difficulty: Hard
 * Tags: array, dp, math
 *
 * 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 countOfPairs = function(nums) {

```

```
};
```

### TypeScript Solution:

```
/**
 * Problem: Find the Count of Monotonic Pairs II
 * Difficulty: Hard
 * Tags: array, dp, math
 *
 * 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 countOfPairs(nums: number[]): number {

};
```

### C# Solution:

```
/*
 * Problem: Find the Count of Monotonic Pairs II
 * Difficulty: Hard
 * Tags: array, dp, math
 *
 * 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 CountOfPairs(int[] nums) {

    }
}
```

### C Solution:

```
/*
 * Problem: Find the Count of Monotonic Pairs II
 * Difficulty: Hard
```

```

* Tags: array, dp, math
*
* 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 countOfPairs(int* nums, int numsSize) {

}

```

### Go Solution:

```

// Problem: Find the Count of Monotonic Pairs II
// Difficulty: Hard
// Tags: array, dp, math
//
// 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 countOfPairs(nums []int) int {

}

```

### Kotlin Solution:

```

class Solution {
    fun countOfPairs(nums: IntArray): Int {

    }
}

```

### Swift Solution:

```

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

    }
}

```

### Rust Solution:

```
// Problem: Find the Count of Monotonic Pairs II
// Difficulty: Hard
// Tags: array, dp, math
//
// 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 count_of_pairs(nums: Vec<i32>) -> i32 {

    }
}
```

### Ruby Solution:

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

end
```

### PHP Solution:

```
class Solution {

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

    }

}
```

### Dart Solution:

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

```
}  
}
```

### Scala Solution:

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

### Elixir Solution:

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

### Erlang Solution:

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

### Racket Solution:

```
(define/contract (count-of-pairs nums)  
  (-> (listof exact-integer?) exact-integer?)  
)
```