

Problem 1885: Count Pairs in Two Arrays

Problem Information

Difficulty: Medium

Acceptance Rate: 0.00%

Paid Only: No

Problem Description

Given two integer arrays

`nums1`

and

`nums2`

of length

`n`

, count the pairs of indices

(i, j)

such that

$i < j$

and

$\text{nums1}[i] + \text{nums1}[j] > \text{nums2}[i] + \text{nums2}[j]$

Return

the

number of pairs

satisfying the condition.

Example 1:

Input:

nums1 = [2,1,2,1], nums2 = [1,2,1,2]

Output:

1

Explanation

: The pairs satisfying the condition are: - (0, 2) where $2 + 2 > 1 + 1$.

Example 2:

Input:

nums1 = [1,10,6,2], nums2 = [1,4,1,5]

Output:

5

Explanation

: The pairs satisfying the condition are: - (0, 1) where $1 + 10 > 1 + 4$. - (0, 2) where $1 + 6 > 1 + 1$. - (1, 2) where $10 + 6 > 4 + 1$. - (1, 3) where $10 + 2 > 4 + 5$. - (2, 3) where $6 + 2 > 1 + 5$.

Constraints:

```
n == nums1.length == nums2.length
```

```
1 <= n <= 10
```

```
5
```

```
1 <= nums1[i], nums2[i] <= 10
```

```
5
```

Code Snippets

C++:

```
class Solution {  
public:  
    long long countPairs(vector<int>& nums1, vector<int>& nums2) {  
  
    }  
};
```

Java:

```
class Solution {  
public long countPairs(int[] nums1, int[] nums2) {  
  
}  
}
```

Python3:

```
class Solution:  
    def countPairs(self, nums1: List[int], nums2: List[int]) -> int:
```

Python:

```
class Solution(object):  
    def countPairs(self, nums1, nums2):  
        """  
        :type nums1: List[int]
```

```
:type nums2: List[int]
:rtype: int
"""

```

JavaScript:

```
/**
 * @param {number[]} nums1
 * @param {number[]} nums2
 * @return {number}
 */
var countPairs = function(nums1, nums2) {
}
```

TypeScript:

```
function countPairs(nums1: number[], nums2: number[]): number {
}
```

C#:

```
public class Solution {
public long CountPairs(int[] nums1, int[] nums2) {

}
}
```

C:

```
long long countPairs(int* nums1, int nums1Size, int* nums2, int nums2Size) {
}
```

Go:

```
func countPairs(nums1 []int, nums2 []int) int64 {
}
```

Kotlin:

```
class Solution {  
    fun countPairs(nums1: IntArray, nums2: IntArray): Long {  
        }  
        }  
}
```

Swift:

```
class Solution {  
    func countPairs(_ nums1: [Int], _ nums2: [Int]) -> Int {  
        }  
        }  
}
```

Rust:

```
impl Solution {  
    pub fn count_pairs(nums1: Vec<i32>, nums2: Vec<i32>) -> i64 {  
        }  
        }  
}
```

Ruby:

```
# @param {Integer[]} nums1  
# @param {Integer[]} nums2  
# @return {Integer}  
def count_pairs(nums1, nums2)  
  
end
```

PHP:

```
class Solution {  
  
    /**  
     * @param Integer[] $nums1  
     * @param Integer[] $nums2  
     * @return Integer  
     */  
    function countPairs($nums1, $nums2) {  
  
    }
```

```
}
```

Dart:

```
class Solution {  
    int countPairs(List<int> nums1, List<int> nums2) {  
          
    }  
}
```

Scala:

```
object Solution {  
    def countPairs(nums1: Array[Int], nums2: Array[Int]): Long = {  
          
    }  
}
```

Elixir:

```
defmodule Solution do  
    @spec count_pairs([integer], [integer]) :: integer  
    def count_pairs(nums1, nums2) do  
  
    end  
end
```

Erlang:

```
-spec count_pairs([integer()], [integer()]) -> integer().  
count_pairs(Nums1, Nums2) ->  
    .
```

Racket:

```
(define/contract (count-pairs nums1 nums2)  
  (-> (listof exact-integer?) (listof exact-integer?) exact-integer?)  
)
```

Solutions

C++ Solution:

```
/*
 * Problem: Count Pairs in Two Arrays
 * Difficulty: Medium
 * Tags: array, sort, search
 *
 * 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 countPairs(vector<int>& nums1, vector<int>& nums2) {
}
```

Java Solution:

```
/**
 * Problem: Count Pairs in Two Arrays
 * Difficulty: Medium
 * Tags: array, sort, search
 *
 * 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 countPairs(int[] nums1, int[] nums2) {
}
```

Python3 Solution:

```
"""
Problem: Count Pairs in Two Arrays
Difficulty: Medium
Tags: array, sort, search
```

```

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 countPairs(self, nums1: List[int], nums2: List[int]) -> int:
# TODO: Implement optimized solution
pass

```

Python Solution:

```

class Solution(object):
def countPairs(self, nums1, nums2):
"""

:type nums1: List[int]
:type nums2: List[int]
:rtype: int
"""

```

JavaScript Solution:

```

/**
 * Problem: Count Pairs in Two Arrays
 * Difficulty: Medium
 * Tags: array, sort, search
 *
 * 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
 */

var countPairs = function(nums1, nums2) {

};


```

TypeScript Solution:

```
/**  
 * Problem: Count Pairs in Two Arrays  
 * Difficulty: Medium  
 * Tags: array, sort, search  
 *  
 * 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 countPairs(nums1: number[], nums2: number[]): number {  
  
};
```

C# Solution:

```
/*  
 * Problem: Count Pairs in Two Arrays  
 * Difficulty: Medium  
 * Tags: array, sort, search  
 *  
 * 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 CountPairs(int[] nums1, int[] nums2) {  
  
    }  
}
```

C Solution:

```
/*  
 * Problem: Count Pairs in Two Arrays  
 * Difficulty: Medium  
 * Tags: array, sort, search  
 *  
 * 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 countPairs(int* nums1, int nums1Size, int* nums2, int nums2Size) {
}

```

Go Solution:

```

// Problem: Count Pairs in Two Arrays
// Difficulty: Medium
// Tags: array, sort, search
//
// 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 countPairs(nums1 []int, nums2 []int) int64 {
}

```

Kotlin Solution:

```

class Solution {
    fun countPairs(nums1: IntArray, nums2: IntArray): Long {
    }
}

```

Swift Solution:

```

class Solution {
    func countPairs(_ nums1: [Int], _ nums2: [Int]) -> Int {
    }
}

```

Rust Solution:

```

// Problem: Count Pairs in Two Arrays
// Difficulty: Medium
// Tags: array, sort, search
//
// 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 count_pairs(nums1: Vec<i32>, nums2: Vec<i32>) -> i64 {
        }

    }
}

```

Ruby Solution:

```

# @param {Integer[]} nums1
# @param {Integer[]} nums2
# @return {Integer}
def count_pairs(nums1, nums2)

end

```

PHP Solution:

```

class Solution {

    /**
     * @param Integer[] $nums1
     * @param Integer[] $nums2
     * @return Integer
     */
    function countPairs($nums1, $nums2) {

    }
}

```

Dart Solution:

```

class Solution {
    int countPairs(List<int> nums1, List<int> nums2) {

```

```
}
```

```
}
```

Scala Solution:

```
object Solution {  
    def countPairs(nums1: Array[Int], nums2: Array[Int]): Long = {  
  
    }  
    }  
}
```

Elixir Solution:

```
defmodule Solution do  
  @spec count_pairs(list(integer()), list(integer())) :: integer  
  def count_pairs(nums1, nums2) do  
  
  end  
end
```

Erlang Solution:

```
-spec count_pairs(list(integer()), list(integer())) -> integer().  
count_pairs(Nums1, Nums2) ->  
.
```

Racket Solution:

```
(define/contract (count-pairs numsl nums2)  
  (-> (listof exact-integer?) (listof exact-integer?) exact-integer?)  
  )
```