Problem 1814: Count Nice Pairs in an Array

Problem Information

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

Problem Description

You are given an array

nums

that consists of non-negative integers. Let us define

rev(x)

as the reverse of the non-negative integer

x

. For example,

rev(123) = 321

, and

rev(120) = 21

. A pair of indices

(i, j)

is

nice

if it satisfies all of the following conditions:

0 <= i < j < nums.length

nums[i] + rev(nums[j]) == nums[j] + rev(nums[i])

Return

the number of nice pairs of indices

. Since that number can be too large, return it

modulo

10

9

+ 7

.

Example 1:

Input:

nums = [42,11,1,97]

Output:

2

Explanation:

The two pairs are: - (0,3) : 42 + rev(97) = 42 + 79 = 121, 97 + rev(42) = 97 + 24 = 121. - (1,2) : 11 + rev(1) = 11 + 1 = 12, 1 + rev(11) = 1 + 11 = 12.

Example 2:

Input:

nums = [13,10,35,24,76]

Output:

4

Constraints:

1 <= nums.length <= 10

5

0 <= nums[i] <= 10

9

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Count Nice Pairs in an Array
* Difficulty: Medium
* Tags: array, math, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

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

}
};
```

Java Solution:

```
/**
* Problem: Count Nice Pairs in an Array
* Difficulty: Medium
* Tags: array, math, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

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

}
}
```

Python3 Solution:

```
"""
Problem: Count Nice Pairs in an Array
Difficulty: Medium
Tags: array, math, hash

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Count Nice Pairs in an Array
* Difficulty: Medium
* Tags: array, math, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

/**
```

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

};
```

TypeScript Solution:

```
/**
* Problem: Count Nice Pairs in an Array
* Difficulty: Medium
* Tags: array, math, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

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

};
```

C# Solution:

```
/*
* Problem: Count Nice Pairs in an Array
* Difficulty: Medium
* Tags: array, math, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

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

}
}
```

C Solution:

```
/*
* Problem: Count Nice Pairs in an Array
* Difficulty: Medium
* Tags: array, math, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

int countNicePairs(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Count Nice Pairs in an Array
// Difficulty: Medium
// Tags: array, math, hash
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func countNicePairs(nums []int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

Rust Solution:

```
// Problem: Count Nice Pairs in an Array
// Difficulty: Medium
// Tags: array, math, hash
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

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

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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