Problem 3551: Minimum Swaps to Sort by Digit Sum

Problem Information

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

Problem Description

You are given an array

nums

of

distinct

positive integers. You need to sort the array in

increasing

order based on the sum of the digits of each number. If two numbers have the same digit sum, the

smaller

number appears first in the sorted order.

Return the

minimum

number of swaps required to rearrange

nums

into this sorted order.

A

swap

is defined as exchanging the values at two distinct positions in the array.

Example 1:

Input:

nums = [37,100]

Output:

1

Explanation:

Compute the digit sum for each integer:

[3 + 7 = 10, 1 + 0 + 0 = 1] $\rightarrow$ [10, 1]

Sort the integers based on digit sum:

[100, 37]

. Swap

37

with

100

to obtain the sorted order.

Thus, the minimum number of swaps required to rearrange

nums

is 1.

Example 2:

Input:

nums = [22,14,33,7]

Output:

0

Explanation:

Compute the digit sum for each integer:

[2 + 2 = 4, 1 + 4 = 5, 3 + 3 = 6, 7 = 7] $\rightarrow$ [4, 5, 6, 7]

Sort the integers based on digit sum:

[22, 14, 33, 7]

. The array is already sorted.

Thus, the minimum number of swaps required to rearrange

nums

is 0.

Example 3:

Input:

nums = [18,43,34,16]

Output:

2

Explanation:

Compute the digit sum for each integer:

[1 + 8 = 9, 4 + 3 = 7, 3 + 4 = 7, 1 + 6 = 7] $\rightarrow$ [9, 7, 7, 7]

Sort the integers based on digit sum:

[16, 34, 43, 18]

. Swap

18

with

16

, and swap

43

with

34

to obtain the sorted order.

Thus, the minimum number of swaps required to rearrange

nums

is 2.

Constraints:

1 <= nums.length <= 10

5

1 <= nums[i] <= 10

9

nums

consists of

distinct

positive integers.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Minimum Swaps to Sort by Digit Sum
* Difficulty: Medium
* Tags: array, hash, sort
*
* 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 minSwaps(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Minimum Swaps to Sort by Digit Sum
* Difficulty: Medium
* Tags: array, hash, sort
*
* 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 minSwaps(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Minimum Swaps to Sort by Digit Sum
Difficulty: Medium
Tags: array, hash, sort

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Minimum Swaps to Sort by Digit Sum
* Difficulty: Medium
* Tags: array, hash, sort
*
* 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 minSwaps = function(nums) {
};
```

TypeScript Solution:

```
/**
* Problem: Minimum Swaps to Sort by Digit Sum
* Difficulty: Medium
* Tags: array, hash, sort
*
* 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 minSwaps(nums: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Minimum Swaps to Sort by Digit Sum
* Difficulty: Medium
* Tags: array, hash, sort
*
* 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 MinSwaps(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Minimum Swaps to Sort by Digit Sum
* Difficulty: Medium
* Tags: array, hash, sort
*
* 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 minSwaps(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Minimum Swaps to Sort by Digit Sum
// Difficulty: Medium
// Tags: array, hash, sort
//
// 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 minSwaps(nums []int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Minimum Swaps to Sort by Digit Sum
// Difficulty: Medium
// Tags: array, hash, sort
//
// 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 min_swaps(nums: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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