Problem 3153: Sum of Digit Differences of All Pairs

Problem Information

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

Problem Description

You are given an array

nums

consisting of

positive

integers where all integers have the

same

number of digits.

The

digit difference

between two integers is the

count

of different digits that are in the

same

position in the two integers.

Return the

sum

of the

digit differences

between

all

pairs of integers in

nums

.

Example 1:

Input:

nums = [13,23,12]

Output:

4

Explanation:

We have the following:

- The digit difference between

1

3 and

2

3 is 1.

- The digit difference between 1

3

and 1

2

is 1.

- The digit difference between

23

and

12

is 2.

So the total sum of digit differences between all pairs of integers is

1 + 1 + 2 = 4

.

Example 2:

Input:

nums = [10,10,10,10]

Output:

0

Explanation:

All the integers in the array are the same. So the total sum of digit differences between all pairs of integers will be 0.

Constraints:

2 <= nums.length <= 10

5

1 <= nums[i] < 10

9

All integers in

nums

have the same number of digits.

Code Snippets

C++:

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

}
};
```

Java:

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

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

```
public class Solution {
public long SumDigitDifferences(int[] nums) {

}
}
```

C:

```
long long sumDigitDifferences(int* nums, int numsSize) {

}
```

Go:

```
func sumDigitDifferences(nums []int) int64 {

}
```

Kotlin:

```
class Solution {
fun sumDigitDifferences(nums: IntArray): Long {

}
}
```

Swift:

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

}
}
```

Rust:

```
impl Solution {
pub fn sum_digit_differences(nums: Vec<i32>) -> i64 {

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (sum-digit-differences nums)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Sum of Digit Differences of All Pairs
* 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:
long long sumDigitDifferences(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Sum of Digit Differences of All Pairs
* 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 long sumDigitDifferences(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Sum of Digit Differences of All Pairs
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 sumDigitDifferences(self, nums: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Sum of Digit Differences of All Pairs
* 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 sumDigitDifferences = function(nums) {

};
```

TypeScript Solution:

```
/**
* Problem: Sum of Digit Differences of All Pairs
* 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 sumDigitDifferences(nums: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Sum of Digit Differences of All Pairs
* 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 long SumDigitDifferences(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Sum of Digit Differences of All Pairs
* 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
*/

long long sumDigitDifferences(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Sum of Digit Differences of All Pairs
// 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 sumDigitDifferences(nums []int) int64 {

}
```

Kotlin Solution:

```
class Solution {
fun sumDigitDifferences(nums: IntArray): Long {

}
}
```

Swift Solution:

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

Rust Solution:

```
// Problem: Sum of Digit Differences of All Pairs
// 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 sum_digit_differences(nums: Vec<i32>) -> i64 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (sum-digit-differences nums)
(-> (listof exact-integer?) exact-integer?)
)
```