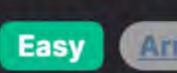
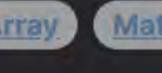
2535. Difference Between Element Sum and Digit Sum of an Array





Problem Description

Math Array Leetcode Link

In this problem, we are provided with an array of positive integers named nums. We need to perform two types of summations:

- 1. Element Sum: This is the sum of all the elements of the array itself. For example, if our array is [1, 2, 3], the element sum would be 1 + 2 + 3 = 6.
- 2. Digit Sum: This is the sum of all digits present in the elements of the array. Continuing the above example, the digit sum would be 1 + 2 + 3 = 6 as well (since each element is comprised of only one digit).

Our goal is to calculate the absolute difference between these two sums. The absolute difference between two numbers x and y is the non-negative value of x - y (denoted as |x - y|).

Intuition

sum all the numbers in the given array. The digit sum requires a bit more work; for each number in the array, we need to extract each digit and sum them together.

To solve this problem, we need to calculate both the element sum and the digit sum. The element sum is straightforward—we simply

Element Sum Calculation:

 Iterate through each number in the array and add it to elementSum. After the loop, elementSum holds the total sum of the array elements.

Initialize a variable (let's call it elementSum) to 0.

- Digit Sum Calculation:

Loop through each number in the array.

For each number, use a nested loop or division and modulo operations to extract each digit.

Initialize a variable (let's call it digitSum) to 0.

Add each extracted digit to digitSum.

happens in the provided reference solution:

- Finally, we calculate the absolute difference between elementSum and digitSum using the abs function, which returns the absolute
- value of the passed argument. The solution code achieves this with the use of simple, efficient loops without any need for complex data structures or algorithms.

After all numbers are processed, digitSum holds the total sum of all digits present in the array elements.

relative to the number of digits across all elements in the array.

By iterating through the array only once and handling each element's digits within that iteration, we keep the time complexity linear

Solution Approach The implementation of the solution is pretty straightforward without involving complex algorithms or data structures. Here's what

The Solution class defines the method differenceOfSum which accepts an integer array nums.

 Two variables, a and b, are initialized at the start of this method. The variable a stores the result of calling the Python sum function on the nums list. This computes the Element Sum

The variable b is initialized to 0. It will be used to store the Digit Sum as we process each integer in nums.

efficiently, as the sum function is a Python built-in that iterates through the list and calculates the total sum of its elements.

- A for loop iterates over each integer x in nums. For each integer:
- A while loop runs as long as x is non-zero (that is, until all digits of x have been processed). ■ Inside the while loop, b (the Digit Sum) is incremented by x % 10, which gives the last digit of x.
- Sum (a) and the Digit Sum (b) using the abs function: abs(a b). • The abs (a - b) expression calculates the non-negative difference between a and b and the resulting integer is returned as the

• x is then floor-divided by 10 using x //= 10, which effectively removes the last digit of x already added to b.

Once all digits of all numbers in nums are processed and added to b, we calculate the absolute difference between the Element

- No additional data structures are used in this approach; we only use variables to store the sums and iterate over the list of integers. The algorithm is efficient because it only involves one pass over the input array and a straightforward digit extraction process for
- each integer element, resulting in an overall time complexity of O(N * K), where N is the number of elements in the nums array, and K is the average number of digits per element.

Example Walkthrough Let's take a small example to illustrate the solution approach using the array nums = [23, 59, 2]. Step 1: Calculate Element Sum We initialize elementSum to 0 and iterate over nums to compute the element sum.

Thus, the element sum is 84.

Step 2: Calculate Digit Sum

For the first integer 23:

elementSum = 0 + 23 + 59 + 2 = 84.

final answer.

We initialize digitSum to 0 and start processing each integer to extract digits and compute the digit sum.

```
o 2 % 10 gives 2, add 2 to digitSum.
```

 Now 2 // 10 is 0, we stop processing this number. For the second integer 59:

We then do integer division 23 // 10 which gives 2.

o 59 % 10 gives 9, add 9 to digitSum. ∘ 59 // 10 gives 5.

o 5 % 10 gives 5, add 5 to digitSum.

23 % 10 gives 3, so we add 3 to digitSum.

 For the third integer 2 (single-digit): o 2 % 10 gives 2, and we add 2 to digitSum.

Adding all digits together: digitSum = 0 + 3 + 2 + 9 + 5 + 2 = 21.

Now we calculate the absolute difference between elementSum and digitSum.

o 5 // 10 is 0, we stop processing this number.

2 // 10 is 0, so we stop.

Python Solution

class Solution:

16

17

19

21

22

23

24

26

27

28

23

24

25

26

27

28

30

29 }

from typing import List

- Thus, the digit sum is 21. Step 3: Compute Absolute Difference
- So, the absolute difference for the array nums = [23, 59, 2] is 63.

|elementSum - digitSum| = |84 - 21| = 63.

total_sum = sum(nums)

digits_sum = 0

for num in nums:

while num:

num //= 10

def difference_of_sum(self, nums: List[int]) -> int: Calculate the difference between the sum of the numbers in the array

```
and the sum of the individual digits of each number in the array.
           Args:
           nums (List[int]): List of integers.
10
11
           Returns:
            int: The absolute difference between the two calculated sums.
13
14
15
```

Calculate the sum of all numbers in the given list.

Iterate over each number in the list to calculate the sum of digits.

Remove the last digit from the current number.

Add the last digit of the current number to the digit sum.

Initialize the sum of individual digits to 0.

digits_sum += num % 10

```
29
30
           # Return the absolute difference between total sum and digits sum.
           return abs(total_sum - digits_sum)
31
32
Java Solution
   class Solution {
       public int differenceOfSum(int[] nums) {
           // Initialize variables to hold the sum of array elements and the sum of digits.
           int sumOfElements = 0;
           int sumOfDigits = 0;
           // Iterate through each element in the input array.
           for (int num : nums) {
               // Add the current element to the sum of elements.
               sumOfElements += num;
10
11
12
               // Make a copy of the current element to manipulate and get the sum of its digits.
13
               int temp = num;
14
               // Extract digits of the current element and accumulate the sum.
15
               while (temp > 0) {
16
17
                   // Extract the rightmost digit and add to the sum of digits.
18
                   int digit = temp % 10;
19
                   sumOfDigits += digit;
20
21
                   // Remove the rightmost digit to proceed to the next digit.
                   temp /= 10;
```

return Math.abs(sumOfElements - sumOfDigits);

C++ Solution

```
1 #include <vector>
 2 #include <cstdlib> // Include required header for std::abs
   class Solution {
   public:
       // Function that computes the absolute difference between the sum of elements
       // in the vector and the sum of the digits of each element in the vector.
       int differenceOfSum(vector<int>& nums) {
           int sumOfElements = 0; // Variable to store the sum of the elements
           int sumOfDigits = 0;  // Variable to store the sum of the digits of each element
10
           // Loop through each number in the vector
           for (int num : nums) {
13
               sumOfElements += num; // Add current number to the sum of elements
14
               // Inner loop to calculate the sum of digits of the current number
16
               for (int n = num; n > 0; n /= 10) {
17
                   sumOfDigits += n % 10; // Add the rightmost digit of n to the sum of digits
18
19
20
21
22
           // Return the absolute value of the difference between the two sums
23
           return std::abs(sumOfElements - sumOfDigits);
24
25 };
Typescript Solution
   function differenceOfSum(nums: number[]): number {
       // Initialize the result accumulator.
```

// Return the absolute difference between the sum of elements and the sum of their digits.

Time Complexity

12

13

14

15

16

17

18

19

});

let resultAccumulator = 0;

nums.forEach((value) => {

while (value !== 0) {

Time and Space Complexity

resultAccumulator += value;

// Return the final result, which is the difference between 20 // the sum of the numbers and the sum of their digits. 21 return resultAccumulator; 24

The time complexity of the given function differenceOfSum involves two parts:

// Iterate over each value in the input array 'nums'.

resultAccumulator -= value % 10;

value = Math.floor(value / 10);

// Remove the last digit of 'value'.

// Add the current value to the result accumulator.

// As long as 'value' is not zero, subtract the digits from 'value'.

// Subtract the last digit of 'value' from the result accumulator.

- is executed for each number in the list. The while loop for calculating the sum of digits runs as many times as there are digits in
- the number—which, in the worst case, can be assumed to be proportional to the logarithm of the number to base 10, 0(log10(x)). Assuming that the numbers are bounded by some value k, this inner loop operation is 0(log10(k)), which can be considered constant for the purposes of this analysis, since k does not depend on n. Thus, this part is also 0(n) for the entire list. Combining both parts, the total time complexity remains O(n) since both operations are linear with respect to the length of the list.

1. The sum of all elements in the list: This is a linear operation with respect to the number of elements, n, hence it has a time complexity of O(n). 2. The loop to sum the individual digits of each number: Here, each number in the list is processed to sum its digits. This operation

Space Complexity

The space complexity of the function is 0(1). This is because the function uses a fixed amount of additional space regardless of the input size n. Variables a, b, and x are the only extra variables, and their space requirement does not scale with n.