1317. Convert Integer to the Sum of Two No-Zero Integers

Easy Math

Problem Description

The "No-Zero integer" problem requires us to find two positive integers (`a` and `b`) such that when they are added together, they equal a given integer `n`, and neither `a` nor `b` contains the digit `0` in their decimal representation. In other words, we need to find a pair of integers where each integer does not have any zeroes, and their sum is exactly `n`.

Intuition

The intuition behind the solution is to sequentially check pairs of integers that add up to `n`. We start by setting the first number `a` to `1` and compute the second number `b` as `n - a`. Each time, we convert the numbers to their string representation and check if the character `0` is present in either. If `0` is not found in both `a` and `b`, we have found our pair of No-Zero integers and return them as a list `[a, b]`.

The reason we start with `a` at `1` is because we are looking for positive integers (integers greater than `0`), and we increase `a` by `1` each iteration up to `n-1` since the smallest valid `b` is `1`. This approach is guaranteed to find a valid solution because there must be at least one pair without the digit `0` in the range from `1` to `n-1` based on the constraints given in the problem. By checking each possible pair, we will surely encounter a valid pair that meets the conditions.

Solution Approach

The solution utilizes a simple brute force approach to implement the algorithm because it systematically checks each pair of positive integers `(a, b)` such that `a + b = n` and ensures neither `a` nor `b` contains the digit `0`.

To iterate through the possible pairs, we use a `for` loop, starting `a` at `1` and incrementing `a` by `1` each time until `n-1`. This is because `a` cannot be `0` and `b` cannot be less than `1`.

In Python, within the loop, for each value of `a`, we calculate `b` by subtracting `a` from `n` (`b = n - a`). We need to check if either of these numbers includes the digit `0`. To do this, we convert both numbers to strings using the `str()` function, concatenate them, and check for the presence of the character `'0'`.

We use the `not in` operator to check if `'0'` is not present in the concatenated string of `a` and `b`. If `'0'` is not present, this means both `a` and `b` are No-Zero integers, and we can then return them as a list `[a, b]`.

Here's the snippet of code encapsulated in the implementation strategy:

```
class Solution:
    def getNoZeroIntegers(self, n: int) -> List[int]:
        for a in range(1, n):
            b = n - a
            if "0" not in str(a) + str(b):
                return [a, b]
```

No additional data structures are needed for this implementation. The pattern used here is straightforward enumeration, which refers to trying all possible solutions until the correct one is found. This pattern is often used when the search space is small enough to be practicable, which is the case here since it is guaranteed that there will be at least one valid pair `(a, b)` within the range from `1` to `n-1`.

Example Walkthrough

Let's walk through a small example to illustrate the solution approach. Suppose we are given an integer `n = 11`. Our task is to find two No-Zero integers `a` and `b` such that their sum equals `n`.

We start by initializing `a` to `1` since it needs to be a positive integer. We then calculate `b` as `n - a`, which in this case would be `11 - 1 = 10`. Since `b` contains the digit `0`, this pair does not satisfy the condition.

Next, we increment `a` to `2` and calculate the new `b` as `11 - 2 = 9`. Neither `a=2` nor `b=9` contain the digit `0`, so this pair satisfies the condition and can be returned as a valid answer.

The function would therefore return `[2, 9]` as output since both are positive integers, their sum is `11`, and neither contains the digit `0` in their decimal representation.

Here's how the code would execute the above logic:

```
class Solution:
    def getNoZeroIntegers(self, n: int) -> List[int]:
        for a in range(1, n):
            b = n - a
            if "0" not in str(a) + str(b):
                return [a, b]

# Example usage:
solution = Solution()
result = solution.getNoZeroIntegers(11)
print(result)  # Output will be [2, 9]
```

In this example, the brute force solution quickly finds the right pair by simply checking all possibilities in a sequential manner, ensuring the logic is both understandable and efficient for small values of `n`.

Solution Implementation

Python

```
class Solution:
    # Define a function to find two integers that add up to `n`,
    # none of which contains the digit 0.
    def getNoZeroIntegers(self, n: int) -> List[int]:
        # Iterate through all numbers from 1 to n-1
        for num1 in range(1, n):
            # Calculate the second number as the difference between n and the first number
            num2 = n - num1

            # Convert both numbers to strings and check if '0' is not present in either
            if "0" not in str(num1) + str(num2):
                # If neither contains '0', return the pair of numbers as a list
                return [num1, num2]

# Note: You must import List from typing before using it
from typing import List  # This should be at the top of the file
```

Java

```
class Solution {

    // This method returns an array of two integers that add up to 'n' and contain no zeroes.
    public int[] getNoZeroIntegers(int n) {
        // Start trying with 'a' starting at 1 and incrementing.
        for (int a = 1;; a++) {
            // Calculate 'b' so that the sum of 'a' and 'b' equals 'n'.
            int b = n - a;

            // Convert both 'a' and 'b' to a string and check if the concatenated result contains '0'.
            if (!containsZero(a) && !containsZero(b)) {
                // If neither 'a' nor 'b' contains '0', we return the pair as the result.
                return new int[] {a, b};
            }
        }
    }

    // Helper method to check if an integer contains the digit '0'.
    private boolean containsZero(int number) {
        // Convert the number to a string.
        String numberStr = Integer.toString(number);
        // Return true if the string representation contains '0', otherwise false.
        return numberStr.contains("0");
    }
}
```

C++

```
#include <vector>
#include <string>
using namespace std;

class Solution {
public:
    // Function that returns two positive integers that add up to n, and neither of which contain any zero digits.
    vector<int> getNoZeroIntegers(int n) {
        // Start a loop which will iterate until we find a valid pair (a, b)
        for (int num1 = 1;; ++num1) { // Only increment num1, num2 is derived from n and num1
            int num2 = n - num1; // Compute the second number of the pair

            // Convert both numbers to strings and concatenate
            string combinedStr = to_string(num1) + to_string(num2);

            // Find if the combined string contains a zero
            if (combinedStr.find('0') == string::npos) { // npos is returned if character not found
                // If no zero is found, return the pair as a vector of two integers
                return {num1, num2};
            }
            // If a zero is found, the loop continues, num1 is incremented and we try the next pair
        }
    }
};
```

TypeScript

```
// Import the necessary functionality for string manipulation in TypeScript
function getNoZeroIntegers(n: number): number[] {
    // This function finds two positive integers that add up to 'n',
    // and neither of them contain any zero digits.

    // Start a loop which will iterate until we find a valid pair of integers
    for (let num1 = 1;; ++num1) { // Only increment num1, num2 is derived from n - num1
        let num2 = n - num1; // Compute the second number of the pair

        // Convert both numbers to strings to check for the presence of zero
        let combinedStr = num1.toString() + num2.toString();

        // Check if the combined string does not contain a '0'
        if (!combinedStr.includes('0')) {
            // If no zero is found, return the pair as an array of two numbers
            return [num1, num2];
        }
        // If a zero is found, continue the loop, increment num1, and try again with the next pair.
    }
    // The loop is theoretically infinite, but it will always find a solution before reaching n
    // because there's guaranteed to be at least one such pair according to the problem statement.
}


class Solution:
    # Define a function to find two integers that add up to `n`,
    # none of which contains the digit 0.
    def getNoZeroIntegers(self, n: int) -> List[int]:
        # Iterate through all numbers from 1 to n-1
        for num1 in range(1, n):
            # Calculate the second number as the difference between n and the first number
            num2 = n - num1

            # Convert both numbers to strings and check if '0' is not present in either
            if "0" not in str(num1) + str(num2):
                # If neither contains '0', return the pair of numbers as a list
                return [num1, num2]

# Note: You must import List from typing before using it
from typing import List  # This should be at the top of the file
```

Time and Space Complexity

Time Complexity

The time complexity of the provided code is `O(N)`, where `N` is the input value to the function. It's because the code uses a loop that runs from `1` up to `n - 1` in the worst case, testing each value of `a` and calculating `b` as `n - a`. Since it checks each pair `(a, b)` once without any nested loops, the time complexity is linear with respect to the input number `n`.

Space Complexity

The space complexity of the code is `O(1)`. Aside from the input and the two variables `a` and `b` that are used within the loop, no additional space that scales with the input size is used. The output list `[a, b]` is of constant size (2 elements) and does not affect the space complexity.