Documentation on Nth Digit Finder

1. Problem Statement

Given an integer n, return the nth digit of the infinite integer sequence: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, and so on. For example, if n = 3, the output should be 3, and if n = 11, the output should be 0 since the 11th digit in the sequence is from the number 10.

2. Intuition

The sequence is constructed from consecutive integers, making it inefficient to generate and store the entire sequence. Instead, we can efficiently locate the exact digit corresponding to n by understanding the numerical structure of the sequence.0

3. Key Observations

The number of digits increases as we progress. Single-digit numbers (1-9) contribute 9 digits, two-digit numbers (10-99) contribute 180 digits, three-digit numbers (100-999) contribute 2700 digits, and so forth. To find the nth digit, we first determine which range it falls into, then compute the exact number containing that digit, and finally extract the required digit.

4. Approach

We begin by identifying the range in which n falls. Starting with single-digit numbers, we iteratively expand our search to two-digit, three-digit, and higher ranges until we locate n. Once the range is identified, we calculate the exact number that holds the nth digit. Finally, we convert that number into a string and extract the specific digit based on its position.

5. Edge Cases

Several edge cases need to be considered. The smallest input (n = 1) should return 1, while the transition from single-digit to double-digit numbers (n = 10, 11, 12) must be handled correctly. Additionally, for large values of n, the algorithm must be efficient to avoid performance bottlenecks.

6. Complexity Analysis

The approach operates in logarithmic time complexity, **O(log n)** since it reduces n significantly in each iteration while determining its range. The space complexity is **O(1)** as only a few integer variables are used, making the solution optimal for large inputs.

7. Alternative Approaches

A brute-force approach would involve constructing the sequence character by character until reaching the nth digit, which is highly inefficient for large values of n. A more optimized method could involve a binary search to determine the range, reducing the number of iterations even further.

8. Test Cases

Different test cases can be designed to verify the correctness of the approach. Simple cases include n = 3 returning 3 and n = 11 returning 0. Additional tests should cover transition points, such as n = 190 returning 1 from the number 100, and n = 250 returning 0 from the number 107.

9. Final Thoughts

This approach efficiently finds the nth digit in an infinite sequence by leveraging numerical properties rather than explicitly constructing the sequence. With **O(log n)** time complexity and **O(1)** space complexity, it provides an optimal solution for large inputs. Alternative methods such as binary search can further refine efficiency, but the current approach strikes a balance between simplicity and performance.