Complement of Base 10 Integer LeetCode

The "Bitwise Complement" question on LeetCode is asking for the bitwise complement of a given non-negative integer num. The bitwise complement of a number is calculated by flipping all the bits in the binary representation of the number. For example, the bitwise complement of the binary number 101 is 010.

In this question, you are given a non-negative integer num that is guaranteed to fit within the range of a 32-bit signed integer. You need to compute the bitwise complement of num and return the result as an integer.

For instance, if the input num is 5, the binary representation of 5 is 101. To calculate the bitwise complement of 5, you flip all the bits in 101, resulting in 010, which is 2 in decimal form. Therefore, the output for the input num = 5 should be 2.

Approach 1: Using Bitwise manipulation

This code computes the bitwise complement of a decimal (base 10) integer using a bitwise manipulation approach.

The bitwiseComplement() function takes an integer n as input, and returns its bitwise complement. Here's how it works:

If n is 0, the function returns 1 as the complement of 0 is 1.

The function initializes mask and temp to 0 and n, respectively.

The function then enters a while loop that runs until temp becomes 0.

In each iteration of the while loop, the function shifts temp one bit to the right (temp >>= 1), and sets the rightmost bit of mask to 1 by shifting mask one bit to the left and OR-ing it with 1 (mask = (mask << 1) | 1).

After the loop completes, the mask variable will contain a sequence of 1's with the same length as the binary representation of n.

The function computes the bitwise complement of n by performing a bitwise NOT (~n) and AND-ing it with the mask variable. The result is stored in the ans variable, which is then returned.

Space And Time Complexity:

The time complexity of the bitwiseComplement function is O(log n), where n is the input integer. This is because the while loop iterates log n times, where the base of the logarithm is 2, since we are shifting the bits to the right by 1 in each iteration. The bitwise operations inside the loop are constant time operations.

The space complexity of the function is O(1), since we are using a constant amount of space for the integer variables mask and temp, regardless of the input size.