<https://leetcode.com/problems/complement-of-base-10-integer/description/>

The complement of an integer is the integer you get when you flip all the 0's to 1's and all the 1's to 0's in its binary representation.

For example, The integer 5 is "101" in binary and its complement is "010" which is the integer 2.

Given an integer n, return its complement.

**Example 1:**

**Input:** n = 5

**Output:** 2

**Explanation:** 5 is "101" in binary, with complement "010" in binary, which is 2 in base-10.

**Example 2:**

**Input:** n = 7

**Output:** 0

**Explanation:** 7 is "111" in binary, with complement "000" in binary, which is 0 in base-10.

**Example 3:**

**Input:** n = 10

**Output:** 5

**Explanation:** 10 is "1010" in binary, with complement "0101" in binary, which is 5 in base-10.

**Constraints:**

1. 0 <= n < 10^9

**Note:** This question is the same as 476: <https://leetcode.com/problems/number-complement/>

**Attempt 1: 2023-03-20**

**Solution 1: Math (10 min)**

class Solution {

    public int bitwiseComplement(int n) {

        int x = 1;

        while(n > x) {

            x = x \* 2 + 1;

        }

        return x - n;

    }

}

**Solution 2: Bit Manipulation (10 min)**

class Solution {

    public int bitwiseComplement(int n) {

        int x = 1;

        while(n > x) {

            x = x << 1;

x = x + 1;

        }

        return x ^ n;

    }

}

**Refer to**

<https://leetcode.com/problems/complement-of-base-10-integer/solutions/256740/java-c-python-find-111-1111-n/>

**Hints**

1. what is the relationship between input and output
2. input + output = 111....11 in binary format
3. Is there any corner case?
4. 0 is a corner case expecting 1, output > input

**Intuition**

Let's find the first number X that X = 1111....1 > N

And also, it has to be noticed that,

N = 0 is a corner case expecting1 as result.

**Solution 1:**

N + bitwiseComplement(N) = 11....11 = X

Then bitwiseComplement(N) = X - N

**Java:**

public int bitwiseComplement(int N) {

int X = 1;

while (N > X) X = X \* 2 + 1;

return X - N;

}

**C++:**

int bitwiseComplement(int N) {

int X = 1;

while (N > X) X = X \* 2 + 1;

return X - N;

}

**Python:**

def bitwiseComplement(self, N):

X = 1

while N > X: X = X \* 2 + 1

return X - N

**Solution 2:**

N ^ bitwiseComplement(N) = 11....11 = X

bitwiseComplement(N) = N ^ X

**Java:**

public int bitwiseComplement(int N) {

int X = 1;

while (N > X) X = X \* 2 + 1;

return N ^ X;

}

**C++:**

int bitwiseComplement(int N) {

int X = 1;

while (N > X) X = X \* 2 + 1;

return N ^ X;

}

**Python:**

def bitwiseComplement(self, N):

X = 1

while N > X: X = X \* 2 + 1;

return N ^ X

**Complexity**

O(logN) Time

O(1) Space

**Python 1-lines**

Use bin

def bitwiseComplement(self, N):

return (1 << len(bin(N)) >> 2) - N - 1

Use translate

def bitwiseComplement(self, N):

return int(bin(N)[2:].translate(string.maketrans('01', '10')), 2)

for those who wonder why the +1 is being added try looking at the pattern below, maybe it helps someone:

(number) => (how-number-is-derived)=>binary-string

x = 1 => 1 => 1

x = 3 => (2\*1 + 1) => 11

x = 7 => (3\*2 + 1) => 111

x = 15 => (7\*2 + 1) => 1111

and so on...