Previous Next

476. Number Complement

March 24, 2020 | 17.1K views

Average Rating: 5 (6 votes)

Given a **positive** integer `num`, output its complement number. The complement strategy is to flip the bits of its binary representation.

Example 1:

```
Input: num = 5
Output: 2
Explanation: The binary representation of 5 is 101 (no leading zero bits), and its com
```

Example 2:

```
Input: num = 1
Output: 0
Explanation: The binary representation of 1 is 1 (no leading zero bits), and its compl
```

Constraints:

- The given integer `num` is guaranteed to fit within the range of a 32-bit signed integer.
- `num >= 1`
- You could assume no leading zero bit in the integer's binary representation.
- This question is the same as 1009: https://leetcode.com/problems/complement-of-base-10-integer/

Solution

Prerequisites

XOR

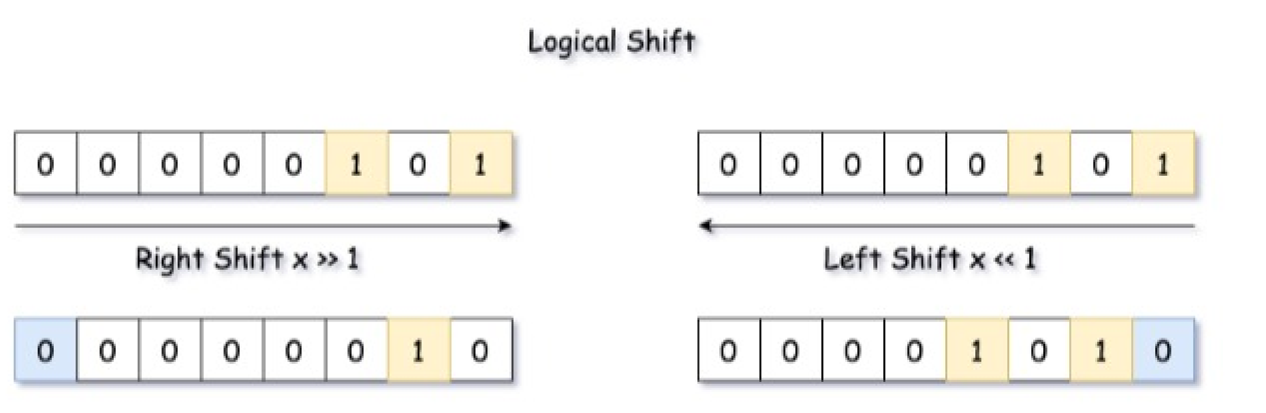
XOR of zero and a bit results in that bit

$$0 \oplus x = x$$

> XOR of one and a bit flips that bit

$$1 \oplus x = 1 - x$$

Right Shift and Left Shift



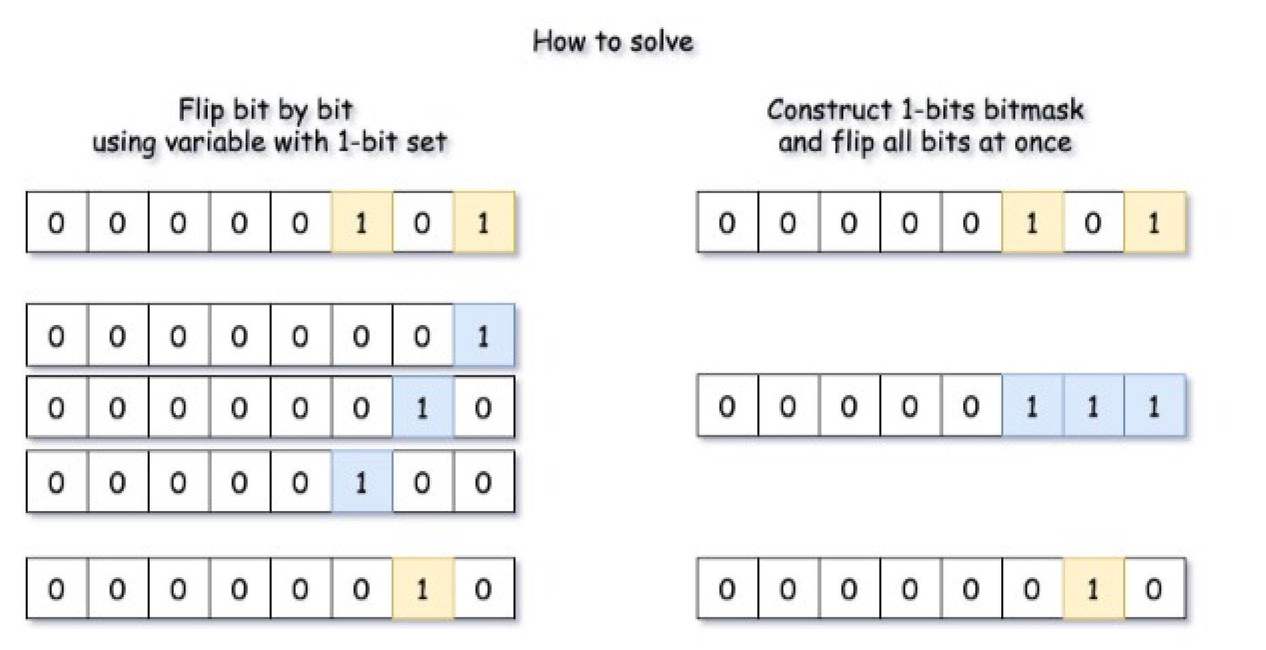
Overview

> The article is long, and the best approach is the one number 4. In the case of limited time, you could jump to it directly.

There are two standard ways to solve the problem:

- To move along the number and flip bit by bit.
- To construct 1-bits bitmask which has the same length as the input number, and to get the answer as `bitmask - num` or `bitmask ^ num`.

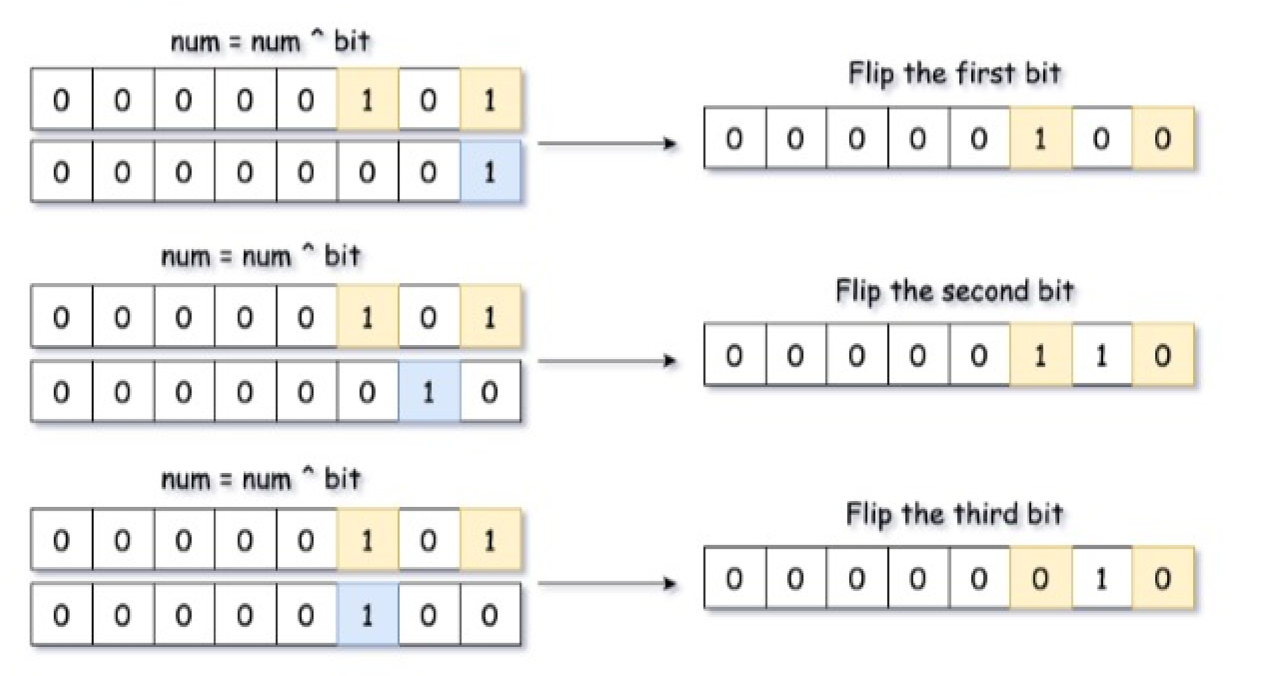
  For example, for $\text{num} = 5 = (101)_2$ the bitmask is $\text{bitmask} = (111)_2$, and the complement number is $\text{bitmask} \oplus \text{num} = (010)_2 = 2$.



Approach 1: Flip Bit by Bit

Algorithm

- Initiate 1-bit variable which will be used to flip bits one by one. Set it to the smallest register `bit = 1`.
- Initiate the marker variable which will be used to stop the loop over the bits `todo = num`.
- Loop over the bits. While `todo != 0`:
  - Flip the current bit: `num = num ^ bit`.
  - Prepare for the next run. Shift flip variable to the left and `todo` variable to the right.
- Return `num`.



Implementation

Java | Python3 | Copy

```
class Solution:
    def findComplement(self, num):
        todo, bit = num, 1
        while todo:
            # flip current bit
            num = num ^ bit
            # prepare for the next run
            bit = bit << 1
            todo = todo >> 1
        return num
```

Complexity

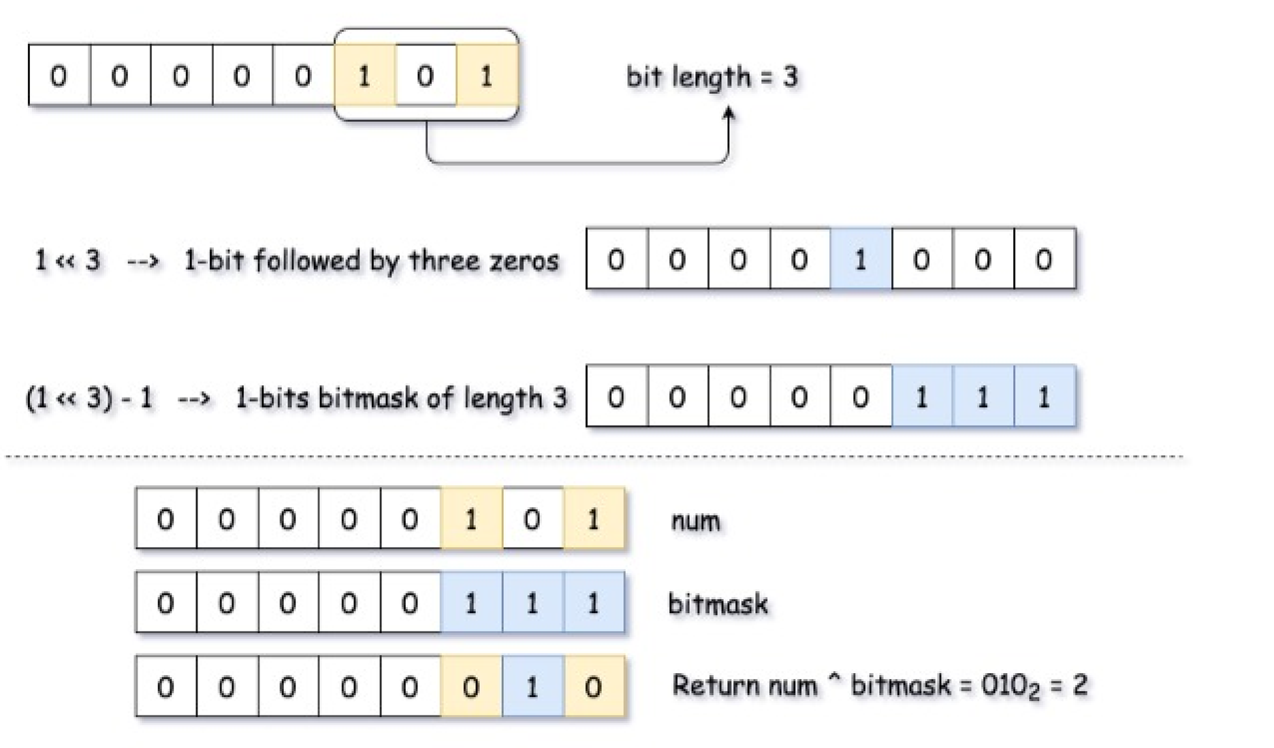
- Time Complexity: $\mathcal{O}(1)$, since we're doing not more than 32 iterations here.
- Space Complexity: $\mathcal{O}(1)$.

Approach 2: Compute Bit Length and Construct 1-bits Bitmask

Instead of flipping bits one by one, let's construct 1-bits bitmask and flip all the bits at once.

There are many ways to do it, let's start from the simplest one:

- Compute bit length of the input number $l = [\log_2 \text{num}] + 1$.
- Compute 1-bits bitmask of length l: $\text{bitmask} = (1 << l) - 1$.
- Return `num ^ bitmask`.



Implementation

Java | Python3 | Copy

```
from math import log2
class Solution:
    def findComplement(self, num):
        # n is a length of num in binary representation
        n = floor(log2(num)) + 1
        # bitmask has the same length as num and contains only ones 1...1
        bitmask = (1 << n) - 1
        # flip all bits
        return bitmask ^ num
```

Complexity

- Time Complexity: $\mathcal{O}(1)$.
- Space Complexity: $\mathcal{O}(1)$.

Approach 3: Built-in Functions to Construct 1-bits Bitmask

Approach 2 could be rewritten with the help of built-in functions: `bit_length` in Python and `highestOneBit` in Java. The first one is trivial, and `Integer.highestOneBit(int x)` method in Java returns int with leftmost bit set in x, i.e. `Integer.highestOneBit(3) = 2`.

Implementation

Java | Python3 | Copy

```
class Solution:
    def findComplement(self, num):
        return (1 << num.bit_length()) - 1 - num
```

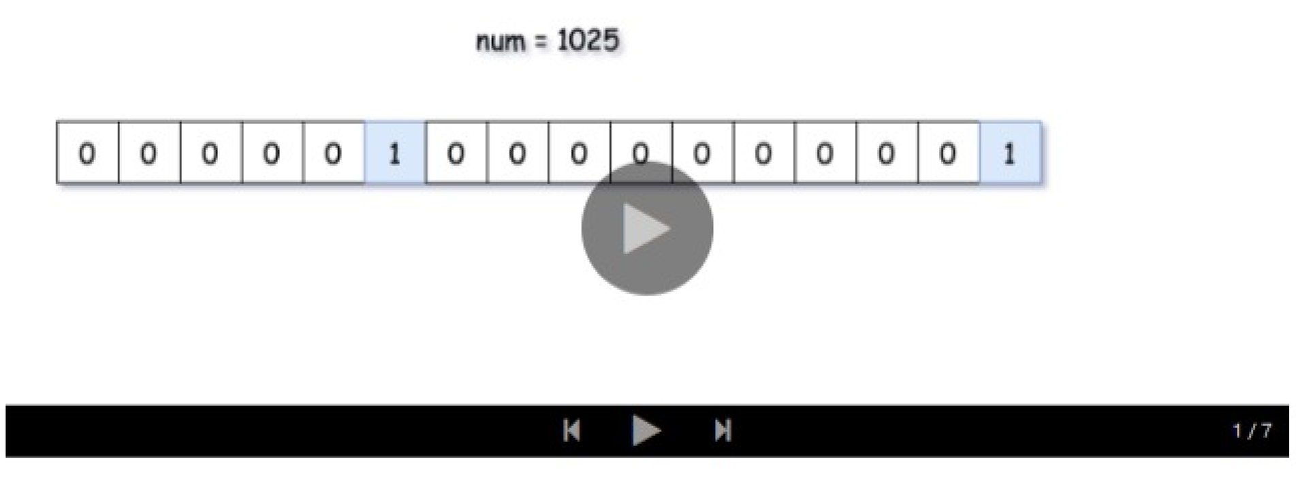
Complexity

- Time Complexity: $\mathcal{O}(1)$ because one deals here with integers of not more than 32 bits.
- Space Complexity: $\mathcal{O}(1)$.

Approach 4: highestOneBit OpenJDK algorithm from Hacker's Delight

The best algorithm for this task is an implementation of `highestOneBit` in OpenJDK. This implementation is taken from "Hacker's Delight" book.

The idea is to create the same 1-bits bitmask by propagating the highest 1-bit into the lower ones.



Implementation

Java | Python3 | Copy

```
class Solution:
    def findComplement(self, num):
        # bitmask has the same length as num and contains only ones 1...1
        bitmask = num
        bitmask |= (bitmask >> 1)
        bitmask |= (bitmask >> 2)
        bitmask |= (bitmask >> 4)
        bitmask |= (bitmask >> 8)
        bitmask |= (bitmask >> 16)
        # flip all bits
        return bitmask ^ num
```

Complexity

- Time Complexity: $\mathcal{O}(1)$.
- Space Complexity: $\mathcal{O}(1)$.

Rate this article: ★★★★★

Previous Next

Comments: 10 Sort By

Type comment here... (Markdown is supported)

Preview Post

abhyasa ★ 14 June 1, 2020 4:46 PM
The last solution is really elegant. Both in terms of understanding it as well the logic.
2 Share Reply

kevin2702 ★ 141 May 5, 2020 8:14 AM
Very elegant
2 Share Reply

stasdude ★ 3 May 4, 2020 8:15 PM
What makes the last solution the best, when all solutions have the same time and space complexity? With more lines of code, it seems less elegant than some of the others.
1 Share Reply
SHOW 2 REPLIES

DenisSchmidt ★ 15 May 4, 2020 3:37 PM
very helpful thanks
1 Share Reply

yfcheng ★ 4447 April 7, 2020 10:10 AM
Solution is so good, especially the last one.
1 Share Reply

sriharik ★ 170 June 10, 2020 3:05 AM
Looks like I'm the only one that utilized an AND mask:

```
public int findComplement(int num) {
    int copy = num, place = 1, complement = 0;
    while (copy > 0) {
```

Read More
0 Share Reply

john_snow ★ 0 May 21, 2020 12:50 PM Report
int findComplement(int num) {
int mask = 0, pos = 0, n = num;
while(n){
n>>=1;
mask += (1<<pos++);
Read More
0 Share Reply

kmvu ★ 0 May 4, 2020 2:49 PM
The last solution seems like it doesn't work for test case "n = 0".

If n = 0, then the last solution returns 0, while the actual result should be 1.

Hence, we have to remember to explicitly check for this case if we want to use this approach.
0 Share Reply
SHOW 4 REPLIES

andvary STAFF ★ 791 March 25, 2020 5:13 PM
@ntkw we do not treat negative numbers here because the fact that num is positive is clearly stated in the description "Given a positive integer".
0 Share Reply

ntkw ★ 58 March 25, 2020 3:45 PM
How should we treat negative Number in this question?

different solutions returns negative numbers differently.

There is no mention about how to deal with negative numbers in the query.
Read More
0 Share Reply