

## Grokking the Coding Interview: Patterns for Coding Questions

73% completed

 Search Course

### Introduction

### Pattern: Sliding Window

### Pattern: Two Pointers

### Pattern: Fast & Slow pointers

### Pattern: Merge Intervals

### Pattern: Cyclic Sort

### Pattern: In-place Reversal of a LinkedList

### Pattern: Tree Breadth First Search

### Pattern: Tree Depth First Search

### Pattern: Two Heaps

### Pattern: Subsets

### Pattern: Modified Binary Search

### Pattern: Bitwise XOR

- Introduction
- Single Number (easy)
- Two Single Numbers (medium)
- Complement of Base 10 Number (medium)
- Problem Challenge 1
- Solution Review: Problem Challenge 1

### Pattern: Top 'K' Elements

- Introduction
- Top 'K' Numbers (easy)
- Kth Smallest Number (easy)
- 'K' Closest Points to the Origin (easy)
- Connect Ropes (easy)
- Top 'K' Frequent Numbers (medium)
- Frequency Sort (medium)
- Kth Largest Number in a Stream

## Complement of Base 10 Number (medium)

### We'll cover the following

- Problem Statement
- Try it yourself
- Solution
- Code
- Time Complexity
- Space Complexity

### Problem Statement #

Every non-negative integer N has a binary representation, for example, 8 can be represented as “1000” in binary and 7 as “0111” in binary.

The complement of a binary representation is the number in binary that we get when we change every 1 to a 0 and every 0 to a 1. For example, the binary complement of “1010” is “0101”.

For a given positive number N in base-10, return the complement of its binary representation as a base-10 integer.

### Example 1:

```
Input: 8
Output: 7
Explanation: 8 is 1000 in binary, its complement is 0111 in binary, which is 7 in base-10.
```

### Example 2:

```
Input: 10
Output: 5
Explanation: 10 is 1010 in binary, its complement is 0101 in binary, which is 5 in base-10.
```

### Try it yourself #

Try solving this question here:

Java
Python3
JS
C++

```
1 import java.lang.Math;
2
3 class CalculateComplement {
4     public static int bitwiseComplement(int n) {
5         // TODO: Write your code here
6         return -1;
7     }
8
9     public static void main( String args[] ) {
10        System.out.println("Bitwise complement is: " + CalculateComplement.bitwiseComplement(8));
11        System.out.println("Bitwise complement is: " + CalculateComplement.bitwiseComplement(10));
12    }
13 }
```

RUN
SAVE
RESET

### Solution #

Recall the following properties of XOR:

1. It will return 1 if we take XOR of two different bits i.e.  $1 \oplus 0 = 0 \oplus 1 = 1$ .
2. It will return 0 if we take XOR of two same bits i.e.  $0 \oplus 0 = 1 \oplus 1 = 0$ . In other words, XOR of two same numbers is 0.
3. It returns the same number if we XOR with 0.

From the above-mentioned first property, we can conclude that XOR of a number with its complement will result in a number that has all of its bits set to 1. For example, the binary complement of “101” is “010”, and if we take XOR of these two numbers, we will get a number with all bits set to 1, i.e.,  $101 \oplus 010 = 111$ .

We can write this fact in the following equation:

```
number ^ complement = all_bits_set
```

Let's add 'number' on both sides:

```
number ^ number ^ complement = number ^ all_bits_set
```

- (medium) 'K' Closest Numbers (medium)
- (medium) Maximum Distinct Elements (medium)
- (medium) Sum of Elements (medium)
- (hard) Rearrange String (hard)
- Problem Challenge 1
- Solution Review: Problem Challenge 1
- Problem Challenge 2
- Solution Review: Problem Challenge 2
- Problem Challenge 3
- Solution Review: Problem Challenge 3

## Pattern: K-way merge ^

- Introduction
- Merge K Sorted Lists (medium)
- Kth Smallest Number in M Sorted Lists (Medium)
- Kth Smallest Number in a Sorted Matrix (Hard)
- Smallest Number Range (Hard)
- Problem Challenge 1
- Solution Review: Problem Challenge 1

## Pattern : 0/1 Knapsack ^ (Dynamic Programming)

- Introduction
- 0/1 Knapsack (medium)
- Equal Subset Sum Partition (medium)
- Subset Sum (medium)
- Minimum Subset Sum Difference (hard)
- Problem Challenge 1
- Solution Review: Problem Challenge 1
- Problem Challenge 2
- Solution Review: Problem Challenge 2



Explore



Tracks



My Courses



Edpresso



Refer a Friend

- (medium) 'K' Closest Numbers (medium)
- (medium) Maximum Distinct Elements (medium)
- (medium) Sum of Elements (medium)
- (hard) Rearrange String (hard)
- Problem Challenge 1
- Solution Review: Problem Challenge 1
- Problem Challenge 2
- Solution Review: Problem Challenge 2
- Problem Challenge 3
- Solution Review: Problem Challenge 3

From the above-mentioned second property:

```
0 ^ complement = number ^ all_bits_set
```

From the above-mentioned third property:

```
complement = number ^ all_bits_set
```

We can use the above fact to find the complement of any number.

**How do we calculate 'all\_bits\_set'?** One way to calculate `all_bits_set` will be to first count the bits required to store the given number. We can then use the fact that for a number which is a complete power of '2' i.e., it can be written as  $\text{pow}(2, n)$ , if we subtract '1' from such a number, we get a number which has 'n' least significant bits set to '1'. For example, '4' which is a complete power of '2', and '3' (which is one less than 4) has a binary representation of '11' i.e., it has '2' least significant bits set to '1'.

## Code #

Here is what our algorithm will look like:

Java
Python3
C++
JS

```

1 import java.lang.Math;
2
3 class CalculateComplement {
4     public static int bitwiseComplement(int num) {
5         // count number of total bits in 'num'
6         int bitCount = 0;
7         int n = num;
8         while (n > 0) {
9             bitCount++;
10            n = n >> 1;
11        }
12
13        // for a number which is a complete power of '2' i.e., it can be written as pow(2, n), if we
14        // subtract '1' from such a number, we get a number which has 'n' least significant bits set to '1'.
15        // For example, '4' which is a complete power of '2', and '3' (which is one less than 4) has a binary
16        // representation of '11' i.e., it has '2' least significant bits set to '1'
17        int all_bits_set = (int) Math.pow(2, bitCount) - 1;
18
19        // from the solution description: complement = number ^ all_bits_set
20        return num ^ all_bits_set;
21    }
22
23    public static void main(String[] args) {
24        System.out.println("Bitwise complement is: " + CalculateComplement.bitwiseComplement(8));
25        System.out.println("Bitwise complement is: " + CalculateComplement.bitwiseComplement(10));
26    }
27 }
```

RUN
SAVE
RESET
Close

Output

```
Bitwise complement is: 7
Bitwise complement is: 5
```

1.426s

## Time Complexity #

Time complexity of this solution is  $O(b)$  where 'b' is the number of bits required to store the given number.

## Space Complexity #

Space complexity of this solution is  $O(1)$ .

Interviewing soon? We've partnered with Hired so that companies apply to you instead of you applying to them. [See how](#)

[Back](#)

[Next](#)

Two Single Numbers (medium)

Problem Challenge 1

[Mark as Completed](#)

Report an Issue Ask a Question

## Miscellaneous ^

- Kth Smallest Number (hard)

## Conclusions ^

- Where to Go from Here



[Mark Course as Completed](#)