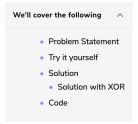


# Grokking the Coding Interview: Patterns for **Coding Questions** 72% completed Q Search Course Pattern: Two Heaps Pattern: Subsets Pattern: Modified Binary Search Pattern: Bitwise XOR Introduction Single Number (easy) Two Single Numbers (medium) Complement of Base 10 Number (medium) Problem Challenge 1 Solution Review: Problem Challenge 1 Pattern: Top 'K' **Elements** 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 'K' Closest Numbers (medium) Maximum Distinct Elements Sum of Elements (medium) 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

# Single Number (easy)



#### **Problem Statement**

In a non-empty array of integers, every number appears twice except for one, find that single number.

## Example 1:

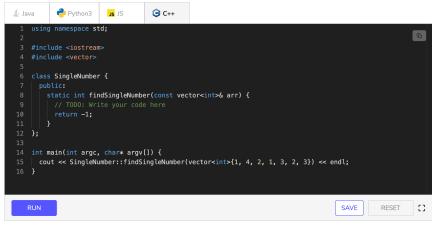
```
Input: 1, 4, 2, 1, 3, 2, 3
Output: 4
```

#### Example 2:

```
Input: 7, 9, 7
Output: 9
```

#### Try it yourself

Try solving this question here:



# Solution

One straight forward solution can be to use a **HashMap** kind of data structure and iterate through the input:

- If number is already present in **HashMap**, remove it.
- If number is not present in  ${\bf HashMap},$  add it.
- In the end, only number left in the  ${\bf HashMap}$  is our required single number.

Time and space complexity Time Complexity of the above solution will be O(n) and space complexity will also be O(n).

Can we do better than this using the XOR Pattern?

Solution with XOR

Recall the following two properties of XOR:

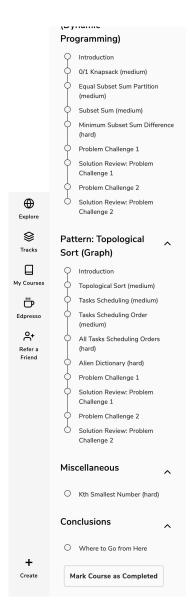
- It returns zero if we take XOR of two same numbers.
- It returns the same number if we XOR with zero.

So we can XOR all the numbers in the input; duplicate numbers will zero out each other and we will be left with the single number.

## Code

Here is what our algorithm will look like:





**Time Complexity**: Time complexity of this solution is O(n) as we iterate through all numbers of the input once.

Space Complexity : The algorithm runs in constant space O(1).

Interviewing soon? We've partnered with Hired so that companies apply to you instead of you applying to them. See how ①

