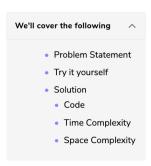
## Two Single Numbers (medium)



# **Problem Statement**

In a non-empty array of numbers, every number appears exactly twice except two numbers that appear only once. Find the two numbers that appear only once.

#### Example 1:

```
Input: [1, 4, 2, 1, 3, 5, 6, 2, 3, 5]
Output: [4, 6]
```

#### Example 2:

```
Input: [2, 1, 3, 2]
Output: [1, 3]
```

## Try it yourself

Try solving this question here:

```
class TwoSingleNumbers {

class TwoSingleNumbers {

public static int[] findSingleNumbers(int[] nums) {

// TODO: Write your code here
return new int[] { -1, -1 };

}

public static void main(String[] args) {

int[] arr = new int[] { 1, 4, 2, 1, 3, 5, 6, 2, 3, 5 };

int[] result = TwoSingleNumbers.findSingleNumbers(arr);

System.out.println("Single numbers are: " + result[0] + ", " + result[1]);

arr = new int[] { 2, 1, 3, 2 };

result = TwoSingleNumbers.findSingleNumbers(arr);

System.out.println("Single numbers are: " + result[0] + ", " + result[1]);

Run

Run

Run

Save Reset (3)
```

## Solution

This problem is quite similar to Single Number, the only difference is that, in this problem, we have two single numbers instead of one. Can we still use XOR to solve this problem?

Let's assume num1 and num2 are the two single numbers. If we do XOR of all elements of the given array, we will be left with XOR of num1 and num2 as all other numbers will cancel each other because all of them appeared twice. Let's call this XOR n1xn2. Now that we have XOR of num1 and num2, how can we find these two single numbers?

As we know that <code>num1</code> and <code>num2</code> are two different numbers, therefore, they should have at least one bit different between them. If a bit in <code>n1xn2</code> is '1', this means that <code>num1</code> and <code>num2</code> have different bits in that place, as we know that we can get '1' only when we do XOR of two different bits, i.e.,

```
1 XOR 0 = 0 XOR 1 = 1
```

We can take any bit which is '1' in n1xn2 and partition all numbers in the given array into two groups based on that bit. One group will have all those numbers with that bit set to '0' and the other with the bit set to '1'. This will ensure that num1 will be in one group and num2 will be in the other. We can take XOR of all numbers in each group separately to get num1 and num2, as all other numbers in each group will cancel each other. Here are the steps of our algorithm:

- 1. Taking XOR of all numbers in the given array will give us XOR of num1 and num2, calling this XOR as
- Find any bit which is set in n1xn2. We can take the rightmost bit which is '1'. Let's call this rightmostSetBit.
- 3. Iterate through all numbers of the input array to partition them into two groups based on rightmostSetBit. Take XOR of all numbers in both the groups separately. Both these XORs are our required numbers.

### Code

Here is what our algorithm will look like:

```
Python3
                          G C++
👙 Java
           TwoSingleNumbers {
       public static int[] findSingleNumbers(int[] nums) {
         int n1xn2 = 0;
         for (int num : nums) {
          n1xn2 ^= num;
         int rightmostSetBit = 1;
         while ((rightmostSetBit & n1xn2) == 0) {
          rightmostSetBit = rightmostSetBit << 1;
         int num1 = 0, num2 = 0;
         for (int num : nums) {
           if ((num & rightmostSetBit) != 0) // the bit is set
             num1 ^= num;
             num2 ^= num;
         return new int[] { num1, num2 };
       public static void main(String[] args) {
         int[] arr = new int[] { 1, 4, 2, 1, 3, 5, 6, 2, 3, 5 };
         int[] result = TwoSingleNumbers.findSingleNumbers(arr);
System.out.println("Single numbers are: " + result[0] +
                                                                           + result[1])
 Run
                                                                                               Save Reset []
```

## Time Complexity

The time complexity of this solution is O(n) where 'n' is the number of elements in the input array.

### Space Complexity

The algorithm runs in constant space O(1).

