October 4

Problem: Missing Number

Problem Statement: Given an array nums containing n distinct numbers in the range [0, n], return the only number in the range that is missing from the array.

Link to problem:

https://leetcode.com/problems/missing-number/

Example 1:
Input: nums = [3,0,1]
Output: 2
Explanation: n = 3 since there are 3 numbers, so all numbers are in the range [0,3]. 2 is the missing number in the range since it does not appear in nums.

Example 2:
Input: nums = [0,1]
Output: 2
Explanation: n = 2 since there are 2 numbers, so all numbers are in the range [0,2]. 2 is the missing number in the range since it does not appear in nums.

Example 3:
Input: nums = [9,6,4,2,3,5,7,0,1]
Output: 8
Explanation: n = 9 since there are 9 numbers, so all numbers are in the range [0,9]. 8 is the missing number in the range since it does not appear in nums.

Solution:

```
class Solution {
    public int missingNumber(int[] nums) {
        int ans = 0;

        // XOR all numbers from 0 to n
        for (int i = 0; i <= nums.length; i++) {
            ans = ans ^ i;
        }

        // XOR all numbers in the array
        for (int i : nums) {
            ans = ans ^ i;
        }

        // The result is the missing number
        return ans;
    }
```

}

Explanation:

- **Concept**: The XOR operation has a useful property that a ^ a = 0 and a ^ 0 = a. This can be used to cancel out all the numbers that appear in both the array and the range [0, n], leaving only the missing number.
- **Step 1**: First, we XOR all numbers from 0 to n (since the numbers are in the range [0, n], there are n+1 numbers).
- **Step 2**: Then, we XOR all the numbers in the given array nums.
- **Step 3**: By XORing the result of the first and second steps, all numbers that appear in both the range [0, n] and the array will cancel each other out, leaving only the missing number.

Time Complexity:

• O(n), where n is the number of elements in the array. We iterate through the array twice (once for XOR from 0 to n and once for XOR over the array).

Space Complexity:

• O(1), since we use only a single extra variable ans for computation.
