

# Problem 2527: Find Xor-Beauty of Array

## Problem Information

Difficulty: **Medium**

Acceptance Rate: 0.00%

Paid Only: No

## Problem Description

You are given a

0-indexed

integer array

nums

.

The

effective value

of three indices

i

,

j

, and

k

is defined as

$((\text{nums}[i] \mid \text{nums}[j]) \& \text{nums}[k])$

.

The

xor-beauty

of the array is the XORing of

the effective values of all the possible triplets

of indices

$(i, j, k)$

where

$0 \leq i, j, k < n$

.

Return

the xor-beauty of

nums

.

Note

that:

$\text{val1} \mid \text{val2}$

is bitwise OR of

val1

and

val2

.

val1 & val2

is bitwise AND of

val1

and

val2

.

Example 1:

Input:

nums = [1,4]

Output:

5

Explanation:

The triplets and their corresponding effective values are listed below: - (0,0,0) with effective value  $((1 \mid 1) \& 1) = 1$  - (0,0,1) with effective value  $((1 \mid 1) \& 4) = 0$  - (0,1,0) with effective value  $((1 \mid 4) \& 1) = 1$  - (0,1,1) with effective value  $((1 \mid 4) \& 4) = 4$  - (1,0,0) with effective value  $((4 \mid 1) \& 1) = 1$  - (1,0,1) with effective value  $((4 \mid 1) \& 4) = 4$  - (1,1,0) with effective value  $((4 \mid 4) \& 1) = 0$  - (1,1,1) with effective value  $((4 \mid 4) \& 4) = 4$  Xor-beauty of array will be bitwise XOR of all beauties =  $1 \wedge 0 \wedge 1 \wedge 4 \wedge 1 \wedge 4 \wedge 0 \wedge 4 = 5$ .

Example 2:

Input:

nums = [15,45,20,2,34,35,5,44,32,30]

Output:

34

Explanation:

The xor-beauty of the given array is 34.

Constraints:

$1 \leq \text{nums.length} \leq 10$

5

$1 \leq \text{nums}[i] \leq 10$

9

## Code Snippets

**C++:**

```
class Solution {  
public:  
    int xorBeauty(vector<int>& nums) {  
  
    }  
};
```

**Java:**

```
class Solution {  
    public int xorBeauty(int[] nums) {
```

```
}  
}
```

### Python3:

```
class Solution:  
    def xorBeauty(self, nums: List[int]) -> int:
```

### Python:

```
class Solution(object):  
    def xorBeauty(self, nums):  
        """  
        :type nums: List[int]  
        :rtype: int  
        """
```

### JavaScript:

```
/**  
 * @param {number[]} nums  
 * @return {number}  
 */  
var xorBeauty = function(nums) {  
  
};
```

### TypeScript:

```
function xorBeauty(nums: number[]): number {  
  
};
```

### C#:

```
public class Solution {  
    public int XorBeauty(int[] nums) {  
  
    }  
}
```

**C:**

```
int xorBeauty(int* nums, int numsSize) {  
  
}
```

**Go:**

```
func xorBeauty(nums []int) int {  
  
}
```

**Kotlin:**

```
class Solution {  
    fun xorBeauty(nums: IntArray): Int {  
  
    }  
}
```

**Swift:**

```
class Solution {  
    func xorBeauty(_ nums: [Int]) -> Int {  
  
    }  
}
```

**Rust:**

```
impl Solution {  
    pub fn xor_beauty(nums: Vec<i32>) -> i32 {  
  
    }  
}
```

**Ruby:**

```
# @param {Integer[]} nums  
# @return {Integer}  
def xor_beauty(nums)  
  
end
```

## PHP:

```
class Solution {

    /**
     * @param Integer[] $nums
     * @return Integer
     */
    function xorBeauty($nums) {

    }

}
```

## Dart:

```
class Solution {
  int xorBeauty(List<int> nums) {

  }
}
```

## Scala:

```
object Solution {
  def xorBeauty(nums: Array[Int]): Int = {

  }
}
```

## Elixir:

```
defmodule Solution do
  @spec xor_beauty(nums :: [integer]) :: integer
  def xor_beauty(nums) do

  end

end
```

## Erlang:

```
-spec xor_beauty(Nums :: [integer()]) -> integer().
xor_beauty(Nums) ->
.
```

## Racket:

```
(define/contract (xor-beauty nums)
  (-> (listof exact-integer?) exact-integer?)
)
```

## Solutions

### C++ Solution:

```
/*
 * Problem: Find Xor-Beauty of Array
 * Difficulty: Medium
 * Tags: array, math
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

class Solution {
public:
    int xorBeauty(vector<int>& nums) {

    }
};
```

### Java Solution:

```
/**
 * Problem: Find Xor-Beauty of Array
 * Difficulty: Medium
 * Tags: array, math
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

class Solution {
    public int xorBeauty(int[] nums) {
```

```
}  
}
```

### Python3 Solution:

```
"""  
Problem: Find Xor-Beauty of Array  
Difficulty: Medium  
Tags: array, math  
  
Approach: Use two pointers or sliding window technique  
Time Complexity: O(n) or O(n log n)  
Space Complexity: O(1) to O(n) depending on approach  
"""  
  
class Solution:  
    def xorBeauty(self, nums: List[int]) -> int:  
        # TODO: Implement optimized solution  
        pass
```

### Python Solution:

```
class Solution(object):  
    def xorBeauty(self, nums):  
        """  
        :type nums: List[int]  
        :rtype: int  
        """
```

### JavaScript Solution:

```
/**  
 * Problem: Find Xor-Beauty of Array  
 * Difficulty: Medium  
 * Tags: array, math  
 *  
 * Approach: Use two pointers or sliding window technique  
 * Time Complexity: O(n) or O(n log n)  
 * Space Complexity: O(1) to O(n) depending on approach  
 */
```

```

/**
 * @param {number[]} nums
 * @return {number}
 */
var xorBeauty = function(nums) {

};

```

### TypeScript Solution:

```

/**
 * Problem: Find Xor-Beauty of Array
 * Difficulty: Medium
 * Tags: array, math
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

function xorBeauty(nums: number[]): number {

};

```

### C# Solution:

```

/*
 * Problem: Find Xor-Beauty of Array
 * Difficulty: Medium
 * Tags: array, math
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

public class Solution {
    public int XorBeauty(int[] nums) {

    }
}

```

```
}
```

### C Solution:

```
/*
 * Problem: Find Xor-Beauty of Array
 * Difficulty: Medium
 * Tags: array, math
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

int xorBeauty(int* nums, int numsSize) {

}
```

### Go Solution:

```
// Problem: Find Xor-Beauty of Array
// Difficulty: Medium
// Tags: array, math
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func xorBeauty(nums []int) int {

}
```

### Kotlin Solution:

```
class Solution {
    fun xorBeauty(nums: IntArray): Int {

    }
}
```

### Swift Solution:

```

class Solution {
func xorBeauty(_ nums: [Int]) -> Int {

}

}

```

### Rust Solution:

```

// Problem: Find Xor-Beauty of Array
// Difficulty: Medium
// Tags: array, math
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn xor_beauty(nums: Vec<i32>) -> i32 {

}

}

```

### Ruby Solution:

```

# @param {Integer[]} nums
# @return {Integer}
def xor_beauty(nums)

end

```

### PHP Solution:

```

class Solution {

/**
 * @param Integer[] $nums
 * @return Integer
 */
function xorBeauty($nums) {

}

}

```

### Dart Solution:

```
class Solution {  
  int xorBeauty(List<int> nums) {  
  
  }  
}
```

### Scala Solution:

```
object Solution {  
  def xorBeauty(nums: Array[Int]): Int = {  
  
  }  
}
```

### Elixir Solution:

```
defmodule Solution do  
  @spec xor_beauty(nums :: [integer]) :: integer  
  def xor_beauty(nums) do  
  
  end  
end
```

### Erlang Solution:

```
-spec xor_beauty(Nums :: [integer()]) -> integer().  
xor_beauty(Nums) ->  
.
```

### Racket Solution:

```
(define/contract (xor-beauty nums)  
  (-> (listof exact-integer?) exact-integer?)  
)
```