Problem 3688: Bitwise OR of Even Numbers in an Array

Problem Information

Difficulty: Easy
Acceptance Rate: 0.00%
Paid Only: No

Problem Description

You are given an integer array

nums

.

Return the bitwise

OR

of all

even

numbers in the array.

If there are no even numbers in

nums

, return 0.

Example 1:

Input:

nums = [1,2,3,4,5,6]

Output:

6

Explanation:

The even numbers are 2, 4, and 6. Their bitwise OR equals 6.

Example 2:

Input:

nums = [7,9,11]

Output:

0

Explanation:

There are no even numbers, so the result is 0.

Example 3:

Input:

nums = [1,8,16]

Output:

24

Explanation:

The even numbers are 8 and 16. Their bitwise OR equals 24.

Constraints:

1 <= nums.length <= 100

1 <= nums[i] <= 100

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function evenNumberBitwiseORs(nums: number[]): number {

};
```

C#:

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

}
}
```

C:

```
int evenNumberBitwiseORs(int* nums, int numsSize) {

}
```

Go:

```
func evenNumberBitwiseORs(nums []int) int {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (even-number-bitwise-o-rs nums)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Bitwise OR of Even Numbers in an Array
* Difficulty: Easy
* Tags: array
*
* 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 evenNumberBitwiseORs(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Bitwise OR of Even Numbers in an Array
* Difficulty: Easy
* Tags: array
*
* 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 evenNumberBitwiseORs(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Bitwise OR of Even Numbers in an Array
Difficulty: Easy
Tags: array

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 evenNumberBitwiseORs(self, nums: List[int]) -> int:
# TODO: Implement optimized solution
```

```
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Bitwise OR of Even Numbers in an Array
* Difficulty: Easy
* Tags: array
*
* 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 evenNumberBitwiseORs = function(nums) {

};
```

TypeScript Solution:

```
/**
* Problem: Bitwise OR of Even Numbers in an Array
* Difficulty: Easy
* Tags: array
*
* 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 evenNumberBitwiseORs(nums: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Bitwise OR of Even Numbers in an Array
* Difficulty: Easy
* Tags: array
*
* 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 EvenNumberBitwiseORs(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Bitwise OR of Even Numbers in an Array
* Difficulty: Easy
* Tags: array
*
* 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 evenNumberBitwiseORs(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Bitwise OR of Even Numbers in an Array
// Difficulty: Easy
// Tags: array
//
// 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 evenNumberBitwiseORs(nums []int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Bitwise OR of Even Numbers in an Array
// Difficulty: Easy
// Tags: array
//
// 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 even_number_bitwise_o_rs(nums: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (even-number-bitwise-o-rs nums)
(-> (listof exact-integer?) exact-integer?)
)
```