Problem 2044: Count Number of Maximum Bitwise-OR Subsets

Problem Information

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

Problem Description

Given an integer array

nums

, find the

maximum

possible

bitwise OR

of a subset of

nums

and return

the

number of different non-empty subsets

with the maximum bitwise OR

.

An array

a

is a

subset

of an array

b

if

a

can be obtained from

b

by deleting some (possibly zero) elements of

b

. Two subsets are considered

different

if the indices of the elements chosen are different.

The bitwise OR of an array

a

is equal to

a[0]

OR

a[1]

OR

...

OR

a[a.length - 1]

(

0-indexed

).

Example 1:

Input:

nums = [3,1]

Output:

2

Explanation:

The maximum possible bitwise OR of a subset is 3. There are 2 subsets with a bitwise OR of 3: - [3] - [3,1]

Example 2:

Input:

nums = [2,2,2]

Output:

7

Explanation:

All non-empty subsets of [2,2,2] have a bitwise OR of 2. There are 2

3

- 1 = 7 total subsets.

Example 3:

Input:

nums = [3,2,1,5]

Output:

6

Explanation:

The maximum possible bitwise OR of a subset is 7. There are 6 subsets with a bitwise OR of 7: - [3,5] - [3,1,5] - [3,2,5] - [3,2,1,5] - [2,5] - [2,1,5]

Constraints:

1 <= nums.length <= 16

1 <= nums[i] <= 10

5

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (count-max-or-subsets nums)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Count Number of Maximum Bitwise-OR Subsets
* Difficulty: Medium
* 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 countMaxOrSubsets(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Count Number of Maximum Bitwise-OR Subsets
* Difficulty: Medium
* 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 countMaxOrSubsets(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Count Number of Maximum Bitwise-OR Subsets
Difficulty: Medium
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 countMaxOrSubsets(self, nums: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Count Number of Maximum Bitwise-OR Subsets
* Difficulty: Medium
* 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 countMaxOrSubsets = function(nums) {

};
```

TypeScript Solution:

```
/**
* Problem: Count Number of Maximum Bitwise-OR Subsets
* Difficulty: Medium
* 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 countMaxOrSubsets(nums: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Count Number of Maximum Bitwise-OR Subsets
* Difficulty: Medium
* 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 CountMaxOrSubsets(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Count Number of Maximum Bitwise-OR Subsets
* Difficulty: Medium
* 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 countMaxOrSubsets(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Count Number of Maximum Bitwise-OR Subsets
// Difficulty: Medium
// 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 countMaxOrSubsets(nums []int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Count Number of Maximum Bitwise-OR Subsets
// Difficulty: Medium
// 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 count_max_or_subsets(nums: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (count-max-or-subsets nums)
(-> (listof exact-integer?) exact-integer?)
)
```