Problem 2275: Largest Combination With Bitwise AND Greater Than Zero

Problem Information

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

Problem Description

The

bitwise AND

of an array

nums

is the bitwise AND of all integers in

nums

.

For example, for

nums = [1, 5, 3]

, the bitwise AND is equal to

1 & 5 & 3 = 1

.

Also, for

nums = [7]

, the bitwise AND is

7

.

You are given an array of positive integers

candidates

. Compute the

bitwise AND

for all possible

combinations

of elements in the

candidates

array.

Return

the size of the

largest

combination of

candidates

with a bitwise AND

greater

than

0

.

Example 1:

Input:

candidates = [16,17,71,62,12,24,14]

Output:

4

Explanation:

The combination [16,17,62,24] has a bitwise AND of 16 & 17 & 62 & 24 = 16 > 0. The size of the combination is 4. It can be shown that no combination with a size greater than 4 has a bitwise AND greater than 0. Note that more than one combination may have the largest size. For example, the combination [62,12,24,14] has a bitwise AND of 62 & 12 & 24 & 14 = 8 > 0.

Example 2:

Input:

candidates = [8,8]

Output:

2

Explanation:

The largest combination [8,8] has a bitwise AND of 8 & 8 = 8 > 0. The size of the combination is 2, so we return 2.

Constraints:

1 <= candidates.length <= 10

5

1 <= candidates[i] <= 10

7

Code Snippets

C++:

```
class Solution {
public:
int largestCombination(vector<int>& candidates) {

}
};
```

Java:

```
class Solution {
public int largestCombination(int[] candidates) {

}
}
```

Python3:

```
class Solution:
def largestCombination(self, candidates: List[int]) -> int:
```

Python:

```
class Solution(object):
def largestCombination(self, candidates):
"""
:type candidates: List[int]
```

```
:rtype: int
"""
```

JavaScript:

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

};
```

TypeScript:

```
function largestCombination(candidates: number[]): number {

};
```

C#:

```
public class Solution {
public int LargestCombination(int[] candidates) {

}
}
```

C:

```
int largestCombination(int* candidates, int candidatesSize) {

}
```

Go:

```
func largestCombination(candidates []int) int {

}
```

Kotlin:

```
class Solution {
fun largestCombination(candidates: IntArray): Int {

}
}
```

Swift:

```
class Solution {
func largestCombination(_ candidates: [Int]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn largest_combination(candidates: Vec<i32>) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[]} candidates
# @return {Integer}
def largest_combination(candidates)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $candidates
* @return Integer
*/
function largestCombination($candidates) {

}
}
```

Dart:

```
class Solution {
int largestCombination(List<int> candidates) {

}
}
```

Scala:

```
object Solution {
def largestCombination(candidates: Array[Int]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec largest_combination(candidates :: [integer]) :: integer
def largest_combination(candidates) do

end
end
```

Erlang:

```
-spec largest_combination(Candidates :: [integer()]) -> integer().
largest_combination(Candidates) ->
.
```

Racket:

```
(define/contract (largest-combination candidates)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Largest Combination With Bitwise AND Greater Than Zero
* Difficulty: Medium
* Tags: array, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public:
int largestCombination(vector<int>& candidates) {

}
};
```

Java Solution:

```
/**
* Problem: Largest Combination With Bitwise AND Greater Than Zero
* Difficulty: Medium
* Tags: array, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public int largestCombination(int[] candidates) {

}
}
```

Python3 Solution:

```
"""
Problem: Largest Combination With Bitwise AND Greater Than Zero
Difficulty: Medium
Tags: array, hash
```

```
Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

class Solution:
def largestCombination(self, candidates: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Largest Combination With Bitwise AND Greater Than Zero
* Difficulty: Medium
* Tags: array, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

/**
* @param {number[]} candidates
* @return {number}
*/
var largestCombination = function(candidates) {

};
```

TypeScript Solution:

```
/**
* Problem: Largest Combination With Bitwise AND Greater Than Zero
* Difficulty: Medium
* Tags: array, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

function largestCombination(candidates: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Largest Combination With Bitwise AND Greater Than Zero
* Difficulty: Medium
* Tags: array, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {
public int LargestCombination(int[] candidates) {

}
}
```

C Solution:

```
/*
* Problem: Largest Combination With Bitwise AND Greater Than Zero
* Difficulty: Medium
* Tags: array, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
```

```
*/

int largestCombination(int* candidates, int candidatesSize) {

}
```

Go Solution:

```
// Problem: Largest Combination With Bitwise AND Greater Than Zero
// Difficulty: Medium
// Tags: array, hash
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func largestCombination(candidates []int) int {

}
```

Kotlin Solution:

```
class Solution {
fun largestCombination(candidates: IntArray): Int {

}
}
```

Swift Solution:

```
class Solution {
func largestCombination(_ candidates: [Int]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Largest Combination With Bitwise AND Greater Than Zero
// Difficulty: Medium
// Tags: array, hash
```

```
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

impl Solution {
pub fn largest_combination(candidates: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} candidates
# @return {Integer}
def largest_combination(candidates)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $candidates
* @return Integer
*/
function largestCombination($candidates) {

}
}
```

Dart Solution:

```
class Solution {
int largestCombination(List<int> candidates) {

}
}
```

Scala Solution:

```
object Solution {
def largestCombination(candidates: Array[Int]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec largest_combination(candidates :: [integer]) :: integer
def largest_combination(candidates) do

end
end
```

Erlang Solution:

```
-spec largest_combination(Candidates :: [integer()]) -> integer().
largest_combination(Candidates) ->
.
```

Racket Solution:

```
(define/contract (largest-combination candidates)
(-> (listof exact-integer?) exact-integer?)
)
```