Problem 1725: Number Of Rectangles That Can Form The Largest Square

Problem Information

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

Problem Description

You are given an array

rectangles

where

rectangles[i] = [l

i

, w

i

]

represents the

i

th

rectangle of length

l

i

and width

w

i

.

You can cut the

i

th

rectangle to form a square with a side length of

k

if both

k <= l

i

and

k <= w

i

. For example, if you have a rectangle

[4,6]

, you can cut it to get a square with a side length of at most

4

.

Let

maxLen

be the side length of the

largest

square you can obtain from any of the given rectangles.

Return

the

number

of rectangles that can make a square with a side length of

maxLen

.

Example 1:

Input:

rectangles = [[5,8],[3,9],[5,12],[16,5]]

Output:

3

Explanation:

The largest squares you can get from each rectangle are of lengths [5,3,5,5]. The largest possible square is of length 5, and you can get it out of 3 rectangles.

Example 2:

Input:

rectangles = [[2,3],[3,7],[4,3],[3,7]]

Output:

3

Constraints:

1 <= rectangles.length <= 1000

rectangles[i].length == 2

1 <= l

i

, w

i

<= 10

9

l

i

!= w

i

Code Snippets

C++:

```
class Solution {
public:
int countGoodRectangles(vector<vector<int>>& rectangles) {

}
};
```

Java:

```
class Solution {
public int countGoodRectangles(int[][] rectangles) {

}
}
```

Python3:

```
class Solution:
def countGoodRectangles(self, rectangles: List[List[int]]) -> int:
```

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function countGoodRectangles(rectangles: number[][]): number {

};
```

C#:

```
public class Solution {
public int CountGoodRectangles(int[][] rectangles) {

}
}
```

C:

```
int countGoodRectangles(int** rectangles, int rectanglesSize, int*
rectanglesColSize){

}
```

Go:

```
func countGoodRectangles(rectangles [][]int) int {

}
```

Kotlin:

```
class Solution {
fun countGoodRectangles(rectangles: Array<IntArray>): Int {

}
}
```

Swift:

```
class Solution {
func countGoodRectangles(_ rectangles: [[Int]]) -> Int {

}
```

```
}
```

Rust:

```
impl Solution {
pub fn count_good_rectangles(rectangles: Vec<Vec<i32>>) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[][]} rectangles
# @return {Integer}
def count_good_rectangles(rectangles)

end
```

PHP:

```
class Solution {

/**
* @param Integer[][] $rectangles
* @return Integer
*/
function countGoodRectangles($rectangles) {

}
}
```

Scala:

```
object Solution {
def countGoodRectangles(rectangles: Array[Array[Int]]): Int = {

}
}
```

Solutions

C++ Solution:

```
/*
* Problem: Number Of Rectangles That Can Form The Largest Square
* 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 countGoodRectangles(vector<vector<int>>& rectangles) {

}
};
```

Java Solution:

```
/**
* Problem: Number Of Rectangles That Can Form The Largest Square
* 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 countGoodRectangles(int[][] rectangles) {

}
}
```

Python3 Solution:

```
"""
Problem: Number Of Rectangles That Can Form The Largest Square
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 countGoodRectangles(self, rectangles: List[List[int]]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Number Of Rectangles That Can Form The Largest Square
* 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[][]} rectangles
* @return {number}
*/
var countGoodRectangles = function(rectangles) {

};
```

TypeScript Solution:

```
/**
* Problem: Number Of Rectangles That Can Form The Largest Square
* 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 countGoodRectangles(rectangles: number[][]): number {

};
```

C# Solution:

```
/*
* Problem: Number Of Rectangles That Can Form The Largest Square
* 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 CountGoodRectangles(int[][] rectangles) {

}
}
```

C Solution:

```
/*
* Problem: Number Of Rectangles That Can Form The Largest Square
* 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 countGoodRectangles(int** rectangles, int rectanglesSize, int*
rectanglesColSize){

}
```

Go Solution:

```
// Problem: Number Of Rectangles That Can Form The Largest Square
// 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 countGoodRectangles(rectangles [][]int) int {

}
```

Kotlin Solution:

```
class Solution {
fun countGoodRectangles(rectangles: Array<IntArray>): Int {

}
}
```

Swift Solution:

```
class Solution {
func countGoodRectangles(_ rectangles: [[Int]]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Number Of Rectangles That Can Form The Largest Square
// 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 count_good_rectangles(rectangles: Vec<Vec<i32>>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[][]} rectangles
# @return {Integer}
def count_good_rectangles(rectangles)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[][] $rectangles
* @return Integer
*/
function countGoodRectangles($rectangles) {

}
}
```

Scala Solution:

```
object Solution {
def countGoodRectangles(rectangles: Array[Array[Int]]): Int = {

}
}
```