Problem 2250: Count Number of Rectangles Containing Each Point

Problem Information

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

Problem Description

You are given a 2D integer array

rectangles

where

rectangles[i] = [l

i

, h

i

]

indicates that

i

th

rectangle has a length of

l

i

and a height of

h

i

. You are also given a 2D integer array

points

where

points[j] = [x

j

, y

j

]

is a point with coordinates

(x

j

, y

j

)

.

The

i

th

rectangle has its

bottom-left corner

point at the coordinates

(0, 0)

and its

top-right corner

point at

(l

i

, h

i

)

.

Return

an integer array

count

of length

points.length

where

count[j]

is the number of rectangles that

contain

the

j

th

point.

The

i

th

rectangle

contains

the

j

th

point if

0 <= x

j

<= l

i

and

0 <= y

j

<= h

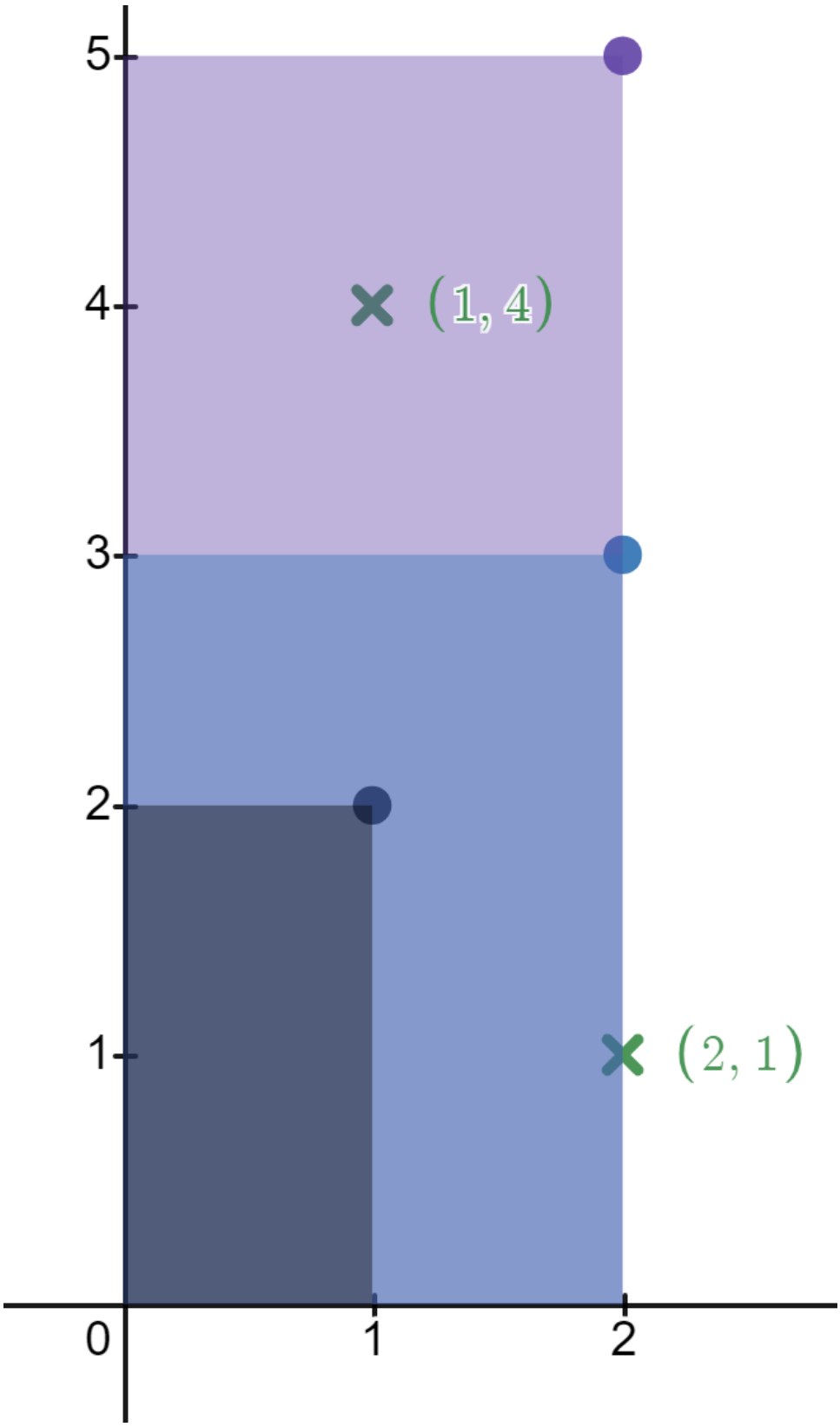
i

. Note that points that lie on the

edges

of a rectangle are also considered to be contained by that rectangle.

Example 1:



Input:

rectangles = [[1,2],[2,3],[2,5]], points = [[2,1],[1,4]]

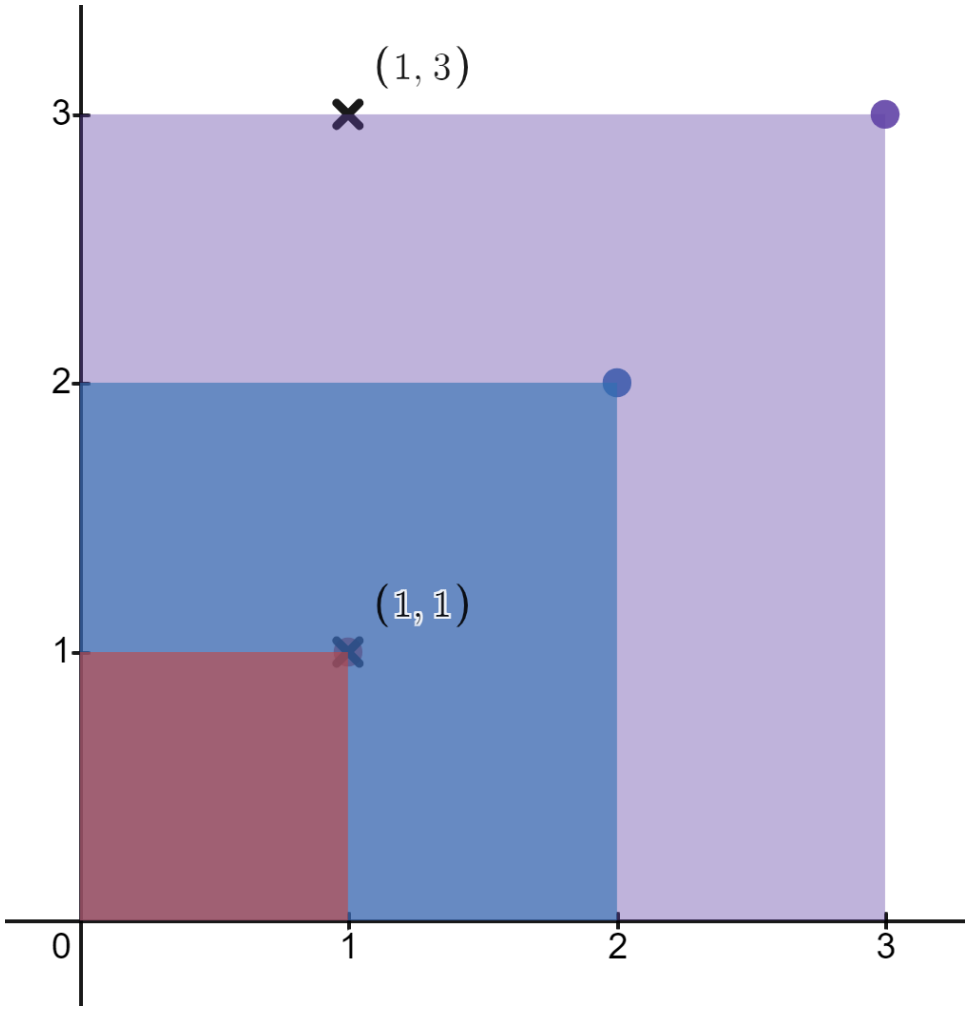
Output:

[2,1]

Explanation:

The first rectangle contains no points. The second rectangle contains only the point (2, 1). The third rectangle contains the points (2, 1) and (1, 4). The number of rectangles that contain the point (2, 1) is 2. The number of rectangles that contain the point (1, 4) is 1. Therefore, we return [2, 1].

Example 2:



Input:

rectangles = [[1,1],[2,2],[3,3]], points = [[1,3],[1,1]]

Output:

[1,3]

Explanation:

The first rectangle contains only the point (1, 1). The second rectangle contains only the point (1, 1). The third rectangle contains the points (1, 3) and (1, 1). The number of rectangles that contain the point (1, 3) is 1. The number of rectangles that contain the point (1, 1) is 3. Therefore, we return [1, 3].

Constraints:

1 <= rectangles.length, points.length <= 5 * 10

4

rectangles[i].length == points[j].length == 2

1 <= l

i

, x

j

<= 10

9

1 <= h

i

, y

j

<= 100

All the

rectangles

are

unique

.

All the

points

are

unique

.

Code Snippets

C++:

```
class Solution {
public:
vector<int> countRectangles(vector<vector<int>>& rectangles,
vector<vector<int>>& points) {

}
};
```

Java:

```
class Solution {
public int[] countRectangles(int[][] rectangles, int[][] points) {
```

```
}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function countRectangles(rectangles: number[][], points: number[][]):
number[] {

};
```

C#:

```
public class Solution {
public int[] CountRectangles(int[][] rectangles, int[][] points) {
```

```
}
}
```

C:

```
/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* countRectangles(int** rectangles, int rectanglesSize, int*
rectanglesColSize, int** points, int pointsSize, int* pointsColSize, int*
returnSize) {

}
```

Go:

```
func countRectangles(rectangles [][]int, points [][]int) []int {

}
```

Kotlin:

```
class Solution {
fun countRectangles(rectangles: Array<IntArray>, points: Array<IntArray>):
IntArray {

}
}
```

Swift:

```
class Solution {
func countRectangles(_ rectangles: [[Int]], _ points: [[Int]]) -> [Int] {

}
}
```

Rust:

```
impl Solution {
pub fn count_rectangles(rectangles: Vec<Vec<i32>>, points: Vec<Vec<i32>>) ->
Vec<i32> {
```

```
}
}
```

Ruby:

```
# @param {Integer[][]} rectangles
# @param {Integer[][]} points
# @return {Integer[]}
def count_rectangles(rectangles, points)

end
```

PHP:

```
class Solution {

/**
* @param Integer[][] $rectangles
* @param Integer[][] $points
* @return Integer[]
*/
function countRectangles($rectangles, $points) {

}
}
```

Dart:

```
class Solution {
List<int> countRectangles(List<List<int>> rectangles, List<List<int>> points)
{

}
}
```

Scala:

```
object Solution {
def countRectangles(rectangles: Array[Array[Int]], points:
Array[Array[Int]]): Array[Int] = {
```

```
}
}
```

Elixir:

```
defmodule Solution do
@spec count_rectangles(rectangles :: [[integer]], points :: [[integer]]) ::
[integer]
def count_rectangles(rectangles, points) do

end
end
```

Erlang:

```
-spec count_rectangles(Rectangles :: [[integer()]], Points :: [[integer()]])
-> [integer()].
count_rectangles(Rectangles, Points) ->
.
```

Racket:

```
(define/contract (count-rectangles rectangles points)
(-> (listof (listof exact-integer?)) (listof (listof exact-integer?)) (listof
exact-integer?))
)
```

Solutions

C++ Solution:

```
/*
* Problem: Count Number of Rectangles Containing Each Point
* Difficulty: Medium
* Tags: array, tree, hash, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/
```

```
class Solution {
public:
vector<int> countRectangles(vector<vector<int>>& rectangles,
vector<vector<int>>& points) {

}
};
```

Java Solution:

```
/**
* Problem: Count Number of Rectangles Containing Each Point
* Difficulty: Medium
* Tags: array, tree, hash, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

class Solution {
public int[] countRectangles(int[][] rectangles, int[][] points) {

}
}
```

Python3 Solution:

```
"""
Problem: Count Number of Rectangles Containing Each Point
Difficulty: Medium
Tags: array, tree, hash, sort, search

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(h) for recursion stack where h is height
"""

class Solution:
def countRectangles(self, rectangles: List[List[int]], points:
```

```
List[List[int]]) -> List[int]:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Count Number of Rectangles Containing Each Point
* Difficulty: Medium
* Tags: array, tree, hash, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

/**
* @param {number[][]} rectangles
* @param {number[][]} points
* @return {number[]}
*/
var countRectangles = function(rectangles, points) {

};
```

TypeScript Solution:

```
/**
* Problem: Count Number of Rectangles Containing Each Point
* Difficulty: Medium
* Tags: array, tree, hash, sort, search
*
```

```
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

function countRectangles(rectangles: number[][], points: number[][]):
number[] {

};
```

C# Solution:

```
/*
* Problem: Count Number of Rectangles Containing Each Point
* Difficulty: Medium
* Tags: array, tree, hash, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

public class Solution {
public int[] CountRectangles(int[][] rectangles, int[][] points) {

}
}
```

C Solution:

```
/*
* Problem: Count Number of Rectangles Containing Each Point
* Difficulty: Medium
* Tags: array, tree, hash, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

/**
```

```
* Note: The returned array must be malloced, assume caller calls free().
*/
int* countRectangles(int** rectangles, int rectanglesSize, int*
rectanglesColSize, int** points, int pointsSize, int* pointsColSize, int*
returnSize) {

}
```

Go Solution:

```
// Problem: Count Number of Rectangles Containing Each Point
// Difficulty: Medium
// Tags: array, tree, hash, sort, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

func countRectangles(rectangles [][]int, points [][]int) []int {

}
```

Kotlin Solution:

```
class Solution {
fun countRectangles(rectangles: Array<IntArray>, points: Array<IntArray>):
IntArray {

}
}
```

Swift Solution:

```
class Solution {
func countRectangles(_ rectangles: [[Int]], _ points: [[Int]]) -> [Int] {

}
}
```

Rust Solution:

```
// Problem: Count Number of Rectangles Containing Each Point
// Difficulty: Medium
// Tags: array, tree, hash, sort, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

impl Solution {
pub fn count_rectangles(rectangles: Vec<Vec<i32>>, points: Vec<Vec<i32>>) ->
Vec<i32> {

}
}
```

Ruby Solution:

```
# @param {Integer[][]} rectangles
# @param {Integer[][]} points
# @return {Integer[]}
def count_rectangles(rectangles, points)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[][] $rectangles
* @param Integer[][] $points
* @return Integer[]
*/
function countRectangles($rectangles, $points) {

}
}
```

Dart Solution:

```
class Solution {
List<int> countRectangles(List<List<int>> rectangles, List<List<int>> points)
```

```
{

}
}
```

Scala Solution:

```
object Solution {
def countRectangles(rectangles: Array[Array[Int]], points:
Array[Array[Int]]): Array[Int] = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec count_rectangles(rectangles :: [[integer]], points :: [[integer]]) ::
[integer]
def count_rectangles(rectangles, points) do

end
end
```

Erlang Solution:

```
-spec count_rectangles(Rectangles :: [[integer()]], Points :: [[integer()]])
-> [integer()].
count_rectangles(Rectangles, Points) ->
.
```

Racket Solution:

```
(define/contract (count-rectangles rectangles points)
(-> (listof (listof exact-integer?)) (listof (listof exact-integer?)) (listof
exact-integer?))
)
```