Problem 1274: Number of Ships in a Rectangle

Problem Information

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

Problem Description

(This problem is an

interactive problem

.)

Each ship is located at an integer point on the sea represented by a cartesian plane, and each integer point may contain at most 1 ship.

You have a function

Sea.hasShips(topRight, bottomLeft)

which takes two points as arguments and returns

true

If there is at least one ship in the rectangle represented by the two points, including on the boundary.

Given two points: the top right and bottom left corners of a rectangle, return the number of ships present in that rectangle. It is guaranteed that there are

at most 10 ships

in that rectangle.

Submissions making

more than 400 calls

to

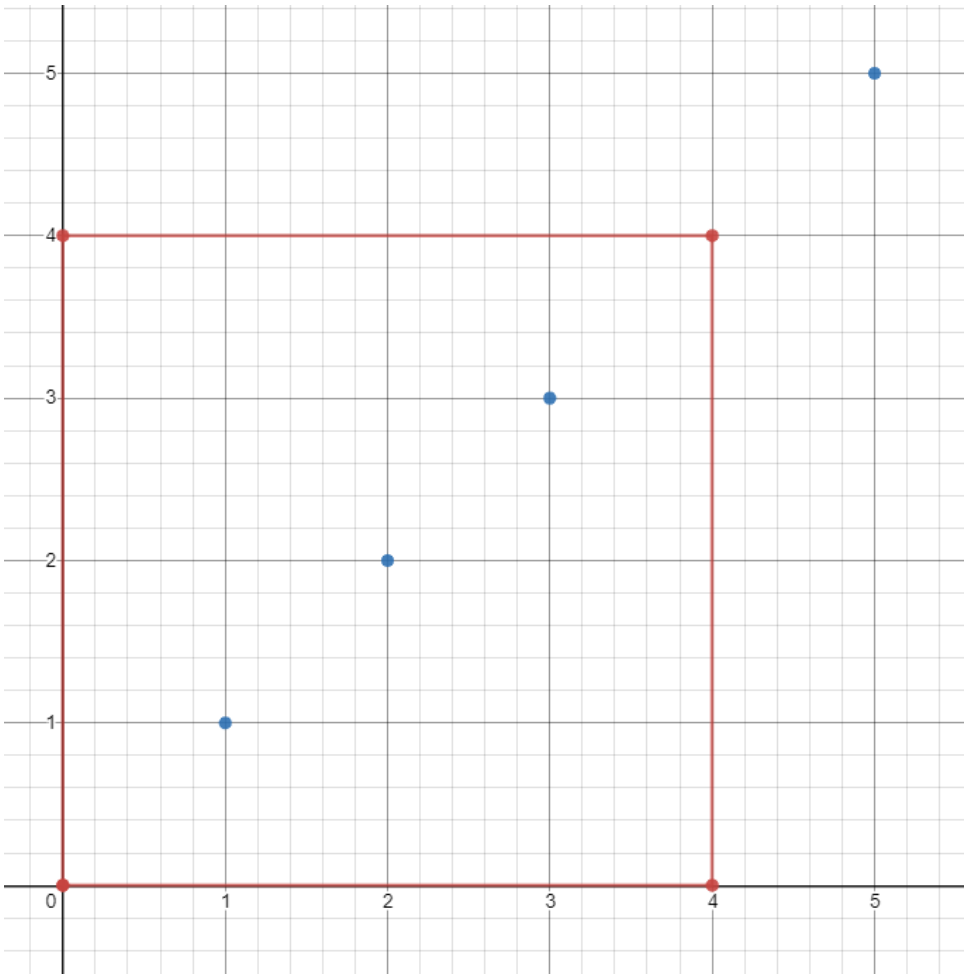
hasShips

will be judged

Wrong Answer

. Also, any solutions that attempt to circumvent the judge will result in disqualification.

Example :



Input:

ships = [[1,1],[2,2],[3,3],[5,5]], topRight = [4,4], bottomLeft = [0,0]

Output:

3

Explanation:

From [0,0] to [4,4] we can count 3 ships within the range.

Example 2:

Input:

ans = [[1,1],[2,2],[3,3]], topRight = [1000,1000], bottomLeft = [0,0]

Output:

3

Constraints:

On the input

ships

is only given to initialize the map internally. You must solve this problem "blindfolded". In other words, you must find the answer using the given

hasShips

API, without knowing the

ships

position.

0 <= bottomLeft[0] <= topRight[0] <= 1000

0 <= bottomLeft[1] <= topRight[1] <= 1000

topRight != bottomLeft

Code Snippets

C++:

```
/**
* // This is Sea's API interface.
* // You should not implement it, or speculate about its implementation
* class Sea {
* public:
* bool hasShips(vector<int> topRight, vector<int> bottomLeft);
* };
*/

class Solution {
public:
int countShips(Sea sea, vector<int> topRight, vector<int> bottomLeft) {

}
};
```

Java:

```
/**
* // This is Sea's API interface.
* // You should not implement it, or speculate about its implementation
* class Sea {
* public boolean hasShips(int[] topRight, int[] bottomLeft);
* }
*/

class Solution {
public int countShips(Sea sea, int[] topRight, int[] bottomLeft) {

}
}
```

Python3:

```
# """
# This is Sea's API interface.
# You should not implement it, or speculate about its implementation
# """
#class Sea:
# def hasShips(self, topRight: 'Point', bottomLeft: 'Point') -> bool:
#
#class Point:
# def __init__(self, x: int, y: int):
# self.x = x
# self.y = y

class Solution:
def countShips(self, sea: 'Sea', topRight: 'Point', bottomLeft: 'Point') ->
int:
```

Python:

```
# """
# This is Sea's API interface.
# You should not implement it, or speculate about its implementation
# """
#class Sea(object):
# def hasShips(self, topRight, bottomLeft):
# """
# :type topRight: Point
# :type bottomLeft: Point
# :rtype bool
# """
#
#class Point(object):
# def __init__(self, x, y):
# self.x = x
# self.y = y

class Solution(object):
def countShips(self, sea, topRight, bottomLeft):
"""
:type sea: Sea
:type topRight: Point
:type bottomLeft: Point
```

```
:rtype: integer
"""
```

JavaScript:

```
/**
* // This is Sea's API interface.
* // You should not implement it, or speculate about its implementation
* function Sea() {
* @param {integer[]} topRight
* @param {integer[]} bottomLeft
* @return {boolean}
* this.hasShips = function(topRight, bottomLeft) {
* ...
* };
* };
*/

/**
* @param {Sea} sea
* @param {integer[]} topRight
* @param {integer[]} bottomLeft
* @return {integer}
*/
var countShips = function(sea, topRight, bottomLeft) {

};
```

TypeScript:

```
/**
* // This is the Sea's API interface.
* // You should not implement it, or speculate about its implementation
* class Sea {
* hasShips(topRight: number[], bottomLeft: number[]): boolean {}
* }
*/

function countShips(sea: Sea, topRight: number[], bottomLeft: number[]):
number {

};
```

C#:

```
/**
* // This is Sea's API interface.
* // You should not implement it, or speculate about its implementation
* class Sea {
* public bool HasShips(int[] topRight, int[] bottomLeft);
* }
*/

class Solution {
public int CountShips(Sea sea, int[] topRight, int[] bottomLeft) {

}
}
```

C:

```
/**
* // The hasShips API is already defined for you.
* // You should not implement it, or speculate about its implementation
* bool hasShips(int topRightX, int topRightY, int bottomLeftX, int
bottomLeftY);
*/

int countShips(int topRightX, int topRightY, int bottomLeftX, int
bottomLeftY) {

}
```

Go:

```
/**
* // This is Sea's API interface.
* // You should not implement it, or speculate about its implementation
* type Sea struct {
* func hasShips(topRight, bottomLeft []int) bool {}
* }
*/

func countShips(sea Sea, topRight, bottomLeft []int) int {

}
```

Kotlin:

```
/**
* // This is Sea's API interface.
* // You should not implement it, or speculate about its implementation
* class Sea {
* fun hasShips(topRight: IntArray, bottomLeft: IntArray): Boolean{}
* }
*/

class Solution {
fun countShips(sea: Sea, topRight: IntArray, bottomLeft: IntArray): Int {

}
}
```

Swift:

```
/**
* // This is Sea's API interface.
* // You should not implement it, or speculate about its implementation
* class Sea {
* public func hasShips(_ topRight: [Int], _ bottomLeft: [Int]) -> Bool {}
* }
*/

class Solution {
func countShips(_ sea: Sea, _ topRight: [Int], _ bottomLeft: [Int]) -> Int {

}
}
```

Rust:

```
/**
* // This is Sea's API interface.
* // You should not implement it, or speculate about its implementation
* struct Sea;
* impl Sea {
* pub fn hasShips(topRight: Vec<i32>,bottomLeft: Vec<i32>)->bool{}
* }
*/
```

```
impl Solution {
pub fn count_ships(sea: &Sea, topRight: Vec<i32>, bottomLeft: Vec<i32>) ->
i32 {

}
}
```

Ruby:

```
# This is Sea's API interface.
# You should not implement it, or speculate about its implementation
# class Sea
# def hasShips(topRight, bottomLeft)
#
# end
# end

# @param {Sea} sea
# @param {List[int]} topRight
# @param {List[int]} bottomLeft
# @return {int}
def countShips(sea, topRight, bottomLeft)

end
```

PHP:

```
/**
* // This is Sea's API interface.
* // You should not implement it, or speculate about its implementation
* class Sea {
* function hasShips ($topRight, $bottomLeft) {}
* }
*/

class Solution {
/**
* @param Sea $sea
* @param Integer[] $topRight
* @param Integer[] $bottomLeft
* @return Integer[]
*/
```

```
function countShips ($sea, $topRight, $bottomLeft) {

}
}
```

Scala:

```
/**
* // This is Sea's API interface.
* // You should not implement it, or speculate about its implementation
* class Sea {
* def hasShips(topRight: Array[Int], bottomLeft: Array[Int]): Boolean = {}
* }
*/

object Solution {
def countShips(sea: Sea, topRight: Array[Int], bottomLeft: Array[Int]): Int =
{

}
}
```

Solutions

C++ Solution:

```
/*
* Problem: Number of Ships in a Rectangle
* Difficulty: Hard
* 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
*/

/**
* // This is Sea's API interface.
* // You should not implement it, or speculate about its implementation
* class Sea {
```

```
* public:
* bool hasShips(vector<int> topRight, vector<int> bottomLeft);
* };
*/

class Solution {
public:
int countShips(Sea sea, vector<int> topRight, vector<int> bottomLeft) {

}
};
```

Java Solution:

```
/**
* Problem: Number of Ships in a Rectangle
* Difficulty: Hard
* 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
*/

/**
* // This is Sea's API interface.
* // You should not implement it, or speculate about its implementation
* class Sea {
* public boolean hasShips(int[] topRight, int[] bottomLeft);
* }
*/

class Solution {
public int countShips(Sea sea, int[] topRight, int[] bottomLeft) {

}
}
```

Python3 Solution:

```
"""
Problem: Number of Ships in a Rectangle
Difficulty: Hard
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
"""

# """
# This is Sea's API interface.
# You should not implement it, or speculate about its implementation
# """
#class Sea:
# def hasShips(self, topRight: 'Point', bottomLeft: 'Point') -> bool:
#
#class Point:
# def __init__(self, x: int, y: int):
# self.x = x
# self.y = y

class Solution:
def countShips(self, sea: 'Sea', topRight: 'Point', bottomLeft: 'Point') ->
int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
# """
# This is Sea's API interface.
# You should not implement it, or speculate about its implementation
# """
#class Sea(object):
# def hasShips(self, topRight, bottomLeft):
# """
# :type topRight: Point
# :type bottomLeft: Point
# :rtype bool
# """
#
```

```
#class Point(object):
# def __init__(self, x, y):
# self.x = x
# self.y = y

class Solution(object):
def countShips(self, sea, topRight, bottomLeft):
"""
:type sea: Sea
:type topRight: Point
:type bottomLeft: Point
:rtype: integer
"""
```

JavaScript Solution:

```
/**
* Problem: Number of Ships in a Rectangle
* Difficulty: Hard
* 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
*/

/**
* // This is Sea's API interface.
* // You should not implement it, or speculate about its implementation
* function Sea() {
* @param {integer[]} topRight
* @param {integer[]} bottomLeft
* @return {boolean}
* this.hasShips = function(topRight, bottomLeft) {
* ...
* };
* };
*/

/**
* @param {Sea} sea
```

```
* @param {integer[]} topRight
* @param {integer[]} bottomLeft
* @return {integer}
*/
var countShips = function(sea, topRight, bottomLeft) {

};
```

TypeScript Solution:

```
/**
* Problem: Number of Ships in a Rectangle
* Difficulty: Hard
* 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
*/

/**
* // This is the Sea's API interface.
* // You should not implement it, or speculate about its implementation
* class Sea {
* hasShips(topRight: number[], bottomLeft: number[]): boolean {}
* }
*/

function countShips(sea: Sea, topRight: number[], bottomLeft: number[]):
number {

};
```

C# Solution:

```
/*
* Problem: Number of Ships in a Rectangle
* Difficulty: Hard
* 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
*/

/**
* // This is Sea's API interface.
* // You should not implement it, or speculate about its implementation
* class Sea {
* public bool HasShips(int[] topRight, int[] bottomLeft);
* }
*/

class Solution {
public int CountShips(Sea sea, int[] topRight, int[] bottomLeft) {

}
}
```

C Solution:

```
/*
* Problem: Number of Ships in a Rectangle
* Difficulty: Hard
* 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
*/

/**
* // The hasShips API is already defined for you.
* // You should not implement it, or speculate about its implementation
* bool hasShips(int topRightX, int topRightY, int bottomLeftX, int
bottomLeftY);
*/

int countShips(int topRightX, int topRightY, int bottomLeftX, int
bottomLeftY) {

}
```

Go Solution:

```
// Problem: Number of Ships in a Rectangle
// Difficulty: Hard
// 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

/**
* // This is Sea's API interface.
* // You should not implement it, or speculate about its implementation
* type Sea struct {
* func hasShips(topRight, bottomLeft []int) bool {}
* }
*/

func countShips(sea Sea, topRight, bottomLeft []int) int {

}
```

Kotlin Solution:

```
/**
* // This is Sea's API interface.
* // You should not implement it, or speculate about its implementation
* class Sea {
* fun hasShips(topRight: IntArray, bottomLeft: IntArray): Boolean{}
* }
*/

class Solution {
fun countShips(sea: Sea, topRight: IntArray, bottomLeft: IntArray): Int {

}
}
```

Swift Solution:

```
/**
* // This is Sea's API interface.
```

```
* // You should not implement it, or speculate about its implementation
* class Sea {
* public func hasShips(_ topRight: [Int], _ bottomLeft: [Int]) -> Bool {}
* }
*/

class Solution {
func countShips(_ sea: Sea, _ topRight: [Int], _ bottomLeft: [Int]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Number of Ships in a Rectangle
// Difficulty: Hard
// 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

/**
* // This is Sea's API interface.
* // You should not implement it, or speculate about its implementation
* struct Sea;
* impl Sea {
* pub fn hasShips(topRight: Vec<i32>,bottomLeft: Vec<i32>)->bool{}
* }
*/

impl Solution {
pub fn count_ships(sea: &Sea, topRight: Vec<i32>, bottomLeft: Vec<i32>) ->
i32 {

}
}
```

Ruby Solution:

```
# This is Sea's API interface.
# You should not implement it, or speculate about its implementation
# class Sea
# def hasShips(topRight, bottomLeft)
#
# end
# end

# @param {Sea} sea
# @param {List[int]} topRight
# @param {List[int]} bottomLeft
# @return {int}
def countShips(sea, topRight, bottomLeft)

end
```

PHP Solution:

```
/**
* // This is Sea's API interface.
* // You should not implement it, or speculate about its implementation
* class Sea {
* function hasShips ($topRight, $bottomLeft) {}
* }
*/

class Solution {
/**
* @param Sea $sea
* @param Integer[] $topRight
* @param Integer[] $bottomLeft
* @return Integer[]
*/
function countShips ($sea, $topRight, $bottomLeft) {

}
}
```

Scala Solution:

```
/**
* // This is Sea's API interface.
```

```
* // You should not implement it, or speculate about its implementation
* class Sea {
* def hasShips(topRight: Array[Int], bottomLeft: Array[Int]): Boolean = {}
* }
*/

object Solution {
def countShips(sea: Sea, topRight: Array[Int], bottomLeft: Array[Int]): Int =
{

}
}
```