Problem 2345: Finding the Number of Visible Mountains

Problem Information

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

Problem Description

You are given a

0-indexed

2D integer array

peaks

where

peaks[i] = [x

i

, y

i

]

states that mountain

i

has a peak at coordinates

(x

i

, y

i

)

. A mountain can be described as a right-angled isosceles triangle, with its base along the

x

-axis and a right angle at its peak. More formally, the

gradients

of ascending and descending the mountain are

1

and

-1

respectively.

A mountain is considered

visible

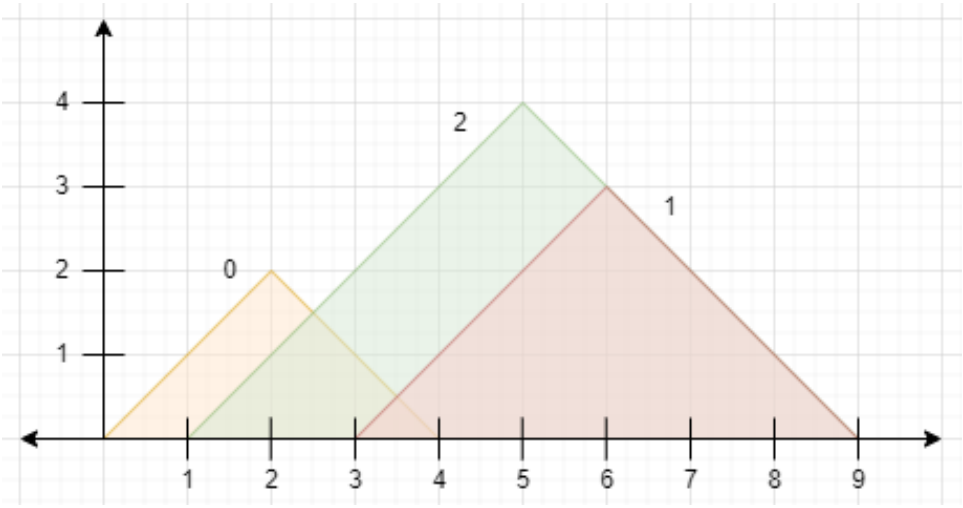
if its peak does not lie within another mountain (including the border of other mountains).

Return

the number of visible mountains

.

Example 1:



Input:

peaks = [[2,2],[6,3],[5,4]]

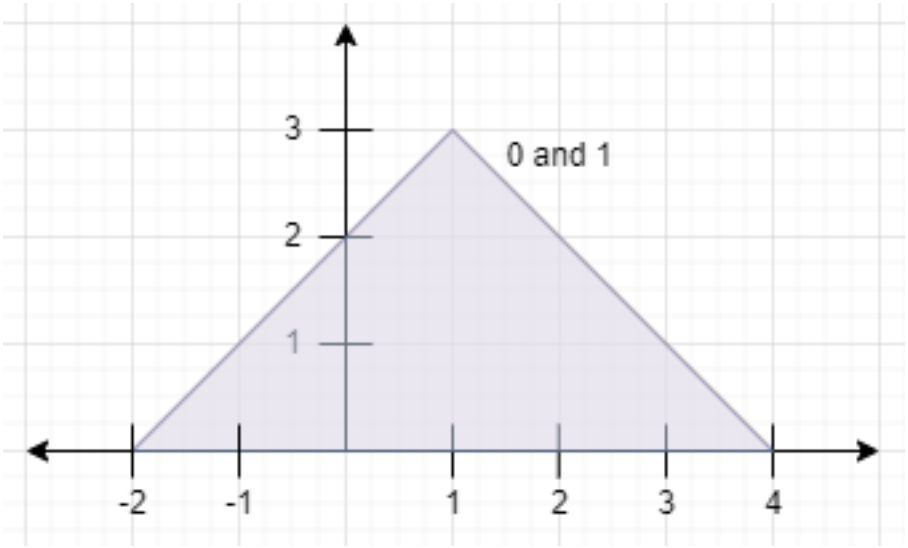
Output:

2

Explanation:

The diagram above shows the mountains. - Mountain 0 is visible since its peak does not lie within another mountain or its sides. - Mountain 1 is not visible since its peak lies within the side of mountain 2. - Mountain 2 is visible since its peak does not lie within another mountain or its sides. There are 2 mountains that are visible.

Example 2:



Input:

peaks = [[1,3],[1,3]]

Output:

0

Explanation:

The diagram above shows the mountains (they completely overlap). Both mountains are not visible since their peaks lie within each other.

Constraints:

1 <= peaks.length <= 10

5

peaks[i].length == 2

1 <= x

i

, y

i

<= 10

5

Code Snippets

C++:

```
class Solution {
public:
int visibleMountains(vector<vector<int>>& peaks) {

}
};
```

Java:

```
class Solution {
public int visibleMountains(int[][] peaks) {

}
}
```

Python3:

```
class Solution:
def visibleMountains(self, peaks: List[List[int]]) -> int:
```

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function visibleMountains(peaks: number[][]): number {

};
```

C#:

```
public class Solution {
public int VisibleMountains(int[][] peaks) {

}
}
```

C:

```
int visibleMountains(int** peaks, int peaksSize, int* peaksColSize) {

}
```

Go:

```
func visibleMountains(peaks [][]int) int {

}
```

Kotlin:

```
class Solution {
fun visibleMountains(peaks: Array<IntArray>): Int {

}
}
```

Swift:

```
class Solution {
func visibleMountains(_ peaks: [[Int]]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn visible_mountains(peaks: Vec<Vec<i32>>) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[][]} peaks
# @return {Integer}
def visible_mountains(peaks)

end
```

PHP:

```
class Solution {

/**
* @param Integer[][] $peaks
* @return Integer
*/
function visibleMountains($peaks) {

}
}
```

Dart:

```
class Solution {
int visibleMountains(List<List<int>> peaks) {

}
}
```

Scala:

```
object Solution {
def visibleMountains(peaks: Array[Array[Int]]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec visible_mountains(peaks :: [[integer]]) :: integer
def visible_mountains(peaks) do

end
end
```

Erlang:

```
-spec visible_mountains(Peaks :: [[integer()]]) -> integer().
visible_mountains(Peaks) ->
.
```

Racket:

```
(define/contract (visible-mountains peaks)
(-> (listof (listof exact-integer?)) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Finding the Number of Visible Mountains
* Difficulty: Medium
* Tags: array, sort, stack
*
* 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 visibleMountains(vector<vector<int>>& peaks) {

}
};
```

Java Solution:

```
/**
* Problem: Finding the Number of Visible Mountains
* Difficulty: Medium
* Tags: array, sort, stack
*
* 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 visibleMountains(int[][] peaks) {

}
}
```

Python3 Solution:

```
"""
Problem: Finding the Number of Visible Mountains
Difficulty: Medium
Tags: array, sort, stack

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 visibleMountains(self, peaks: List[List[int]]) -> int:
# TODO: Implement optimized solution
```

```
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Finding the Number of Visible Mountains
* Difficulty: Medium
* Tags: array, sort, stack
*
* 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[][]} peaks
* @return {number}
*/
var visibleMountains = function(peaks) {

};
```

TypeScript Solution:

```
/**
* Problem: Finding the Number of Visible Mountains
* Difficulty: Medium
* Tags: array, sort, stack
*
* 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 visibleMountains(peaks: number[][]): number {

};
```

C# Solution:

```
/*
* Problem: Finding the Number of Visible Mountains
* Difficulty: Medium
* Tags: array, sort, stack
*
* 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 VisibleMountains(int[][] peaks) {

}
}
```

C Solution:

```
/*
* Problem: Finding the Number of Visible Mountains
* Difficulty: Medium
* Tags: array, sort, stack
*
* 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 visibleMountains(int** peaks, int peaksSize, int* peaksColSize) {

}
```

Go Solution:

```
// Problem: Finding the Number of Visible Mountains
// Difficulty: Medium
// Tags: array, sort, stack
//
// 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 visibleMountains(peaks [][]int) int {

}
```

Kotlin Solution:

```
class Solution {
fun visibleMountains(peaks: Array<IntArray>): Int {

}
}
```

Swift Solution:

```
class Solution {
func visibleMountains(_ peaks: [[Int]]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Finding the Number of Visible Mountains
// Difficulty: Medium
// Tags: array, sort, stack
//
// 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 visible_mountains(peaks: Vec<Vec<i32>>) -> i32 {

}
```

```
}
```

Ruby Solution:

```
# @param {Integer[][]} peaks
# @return {Integer}
def visible_mountains(peaks)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[][] $peaks
* @return Integer
*/
function visibleMountains($peaks) {

}
}
```

Dart Solution:

```
class Solution {
int visibleMountains(List<List<int>> peaks) {

}
}
```

Scala Solution:

```
object Solution {
def visibleMountains(peaks: Array[Array[Int]]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec visible_mountains(peaks :: [[integer]]) :: integer
def visible_mountains(peaks) do

end
end
```

Erlang Solution:

```
-spec visible_mountains(Peaks :: [[integer()]]) -> integer().
visible_mountains(Peaks) ->
.
```

Racket Solution:

```
(define/contract (visible-mountains peaks)
(-> (listof (listof exact-integer?)) exact-integer?)
)
```