Problem 3382: Maximum Area Rectangle With Point Constraints II

Problem Information

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

Problem Description

There are n points on an infinite plane. You are given two integer arrays

xCoord

and

yCoord

where

(xCoord[i], yCoord[i])

represents the coordinates of the

i

th

point.

Your task is to find the

maximum

area of a rectangle that:

Can be formed using

four

of these points as its corners.

Does

not

contain any other point inside or on its border.

Has its edges

parallel

to the axes.

Return the

maximum area

that you can obtain or -1 if no such rectangle is possible.

Example 1:

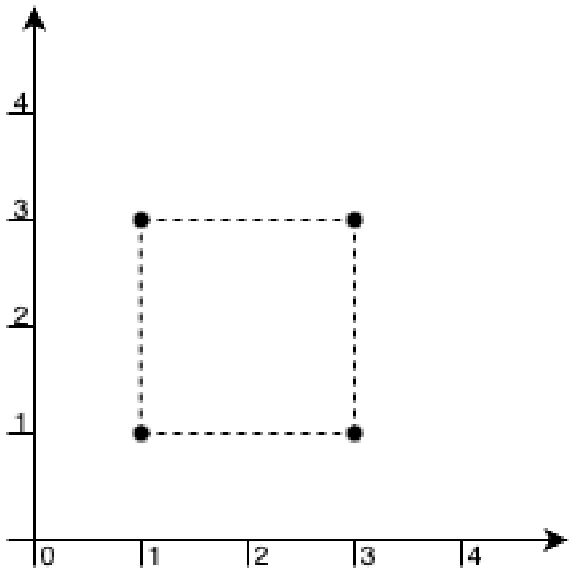
Input:

xCoord = [1,1,3,3], yCoord = [1,3,1,3]

Output:

4

Explanation:



We can make a rectangle with these 4 points as corners and there is no other point that lies inside or on the border. Hence, the maximum possible area would be 4.

Example 2:

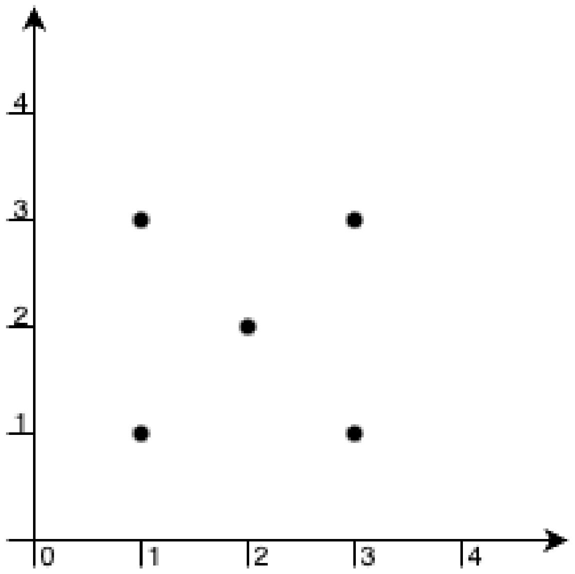
Input:

xCoord = [1,1,3,3,2], yCoord = [1,3,1,3,2]

Output:

-1

Explanation:



There is only one rectangle possible is with points

[1,1], [1,3], [3,1]

and

[3,3]

but

[2,2]

will always lie inside it. Hence, returning -1.

Example 3:

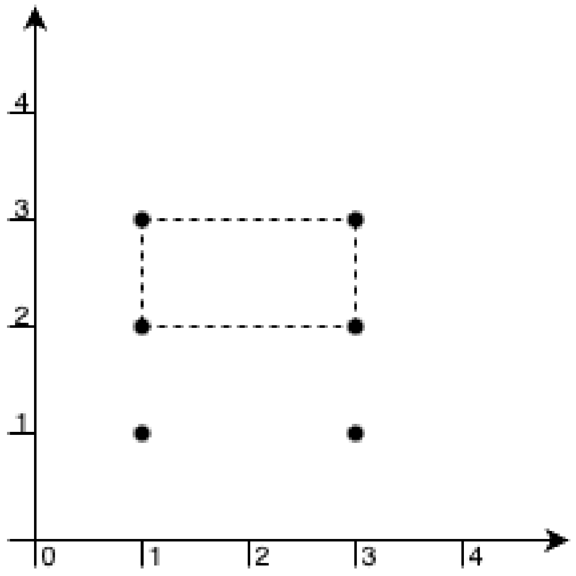
Input:

xCoord = [1,1,3,3,1,3], yCoord = [1,3,1,3,2,2]

Output:

2

Explanation:



The maximum area rectangle is formed by the points

[1,3], [1,2], [3,2], [3,3]

, which has an area of 2. Additionally, the points

[1,1], [1,2], [3,1], [3,2]

also form a valid rectangle with the same area.

Constraints:

1 <= xCoord.length == yCoord.length <= 2 * 10

5

0 <= xCoord[i], yCoord[i] <= 8 * 10

7

All the given points are

unique

.

Code Snippets

C++:

```
class Solution {
public:
long long maxRectangleArea(vector<int>& xCoord, vector<int>& yCoord) {

}
};
```

Java:

```
class Solution {
public long maxRectangleArea(int[] xCoord, int[] yCoord) {

}
}
```

Python3:

```
class Solution:
def maxRectangleArea(self, xCoord: List[int], yCoord: List[int]) -> int:
```

Python:

```
class Solution(object):
def maxRectangleArea(self, xCoord, yCoord):
"""
:type xCoord: List[int]
:type yCoord: List[int]
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number[]} xCoord
* @param {number[]} yCoord
* @return {number}
*/
```

```
var maxRectangleArea = function(xCoord, yCoord) {

};
```

TypeScript:

```
function maxRectangleArea(xCoord: number[], yCoord: number[]): number {

};
```

C#:

```
public class Solution {
public long MaxRectangleArea(int[] xCoord, int[] yCoord) {

}
}
```

C:

```
long long maxRectangleArea(int* xCoord, int xCoordSize, int* yCoord, int
yCoordSize) {

}
```

Go:

```
func maxRectangleArea(xCoord []int, yCoord []int) int64 {

}
```

Kotlin:

```
class Solution {
fun maxRectangleArea(xCoord: IntArray, yCoord: IntArray): Long {

}
}
```

Swift:

```
class Solution {
func maxRectangleArea(_ xCoord: [Int], _ yCoord: [Int]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn max_rectangle_area(x_coord: Vec<i32>, y_coord: Vec<i32>) -> i64 {

}
}
```

Ruby:

```
# @param {Integer[]} x_coord
# @param {Integer[]} y_coord
# @return {Integer}
def max_rectangle_area(x_coord, y_coord)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $xCoord
* @param Integer[] $yCoord
* @return Integer
*/
function maxRectangleArea($xCoord, $yCoord) {

}
}
```

Dart:

```
class Solution {
int maxRectangleArea(List<int> xCoord, List<int> yCoord) {

}
}
```

Scala:

```
object Solution {
def maxRectangleArea(xCoord: Array[Int], yCoord: Array[Int]): Long = {

}
}
```

Elixir:

```
defmodule Solution do
@spec max_rectangle_area(x_coord :: [integer], y_coord :: [integer]) ::
integer
def max_rectangle_area(x_coord, y_coord) do

end
end
```

Erlang:

```
-spec max_rectangle_area(XCoord :: [integer()], YCoord :: [integer()]) ->
integer().
max_rectangle_area(XCoord, YCoord) ->
.
```

Racket:

```
(define/contract (max-rectangle-area xCoord yCoord)
(-> (listof exact-integer?) (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Maximum Area Rectangle With Point Constraints II
* Difficulty: Hard
* Tags: array, tree, math, sort
*
* 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:
long long maxRectangleArea(vector<int>& xCoord, vector<int>& yCoord) {

}
};
```

Java Solution:

```
/**
* Problem: Maximum Area Rectangle With Point Constraints II
* Difficulty: Hard
* Tags: array, tree, math, sort
*
* 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 long maxRectangleArea(int[] xCoord, int[] yCoord) {

}
}
```

Python3 Solution:

```
"""
Problem: Maximum Area Rectangle With Point Constraints II
Difficulty: Hard
Tags: array, tree, math, sort

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 maxRectangleArea(self, xCoord: List[int], yCoord: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def maxRectangleArea(self, xCoord, yCoord):
"""
:type xCoord: List[int]
:type yCoord: List[int]
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Maximum Area Rectangle With Point Constraints II
* Difficulty: Hard
* Tags: array, tree, math, sort
*
* 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[]} xCoord
* @param {number[]} yCoord
* @return {number}
*/
var maxRectangleArea = function(xCoord, yCoord) {

};
```

TypeScript Solution:

```
/**
* Problem: Maximum Area Rectangle With Point Constraints II
* Difficulty: Hard
* Tags: array, tree, math, sort
*
* 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 maxRectangleArea(xCoord: number[], yCoord: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Maximum Area Rectangle With Point Constraints II
* Difficulty: Hard
* Tags: array, tree, math, sort
*
* 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 long MaxRectangleArea(int[] xCoord, int[] yCoord) {

}
}
```

C Solution:

```
/*
* Problem: Maximum Area Rectangle With Point Constraints II
* Difficulty: Hard
* Tags: array, tree, math, sort
*
* 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
```

```
*/

long long maxRectangleArea(int* xCoord, int xCoordSize, int* yCoord, int
yCoordSize) {

}
```

Go Solution:

```
// Problem: Maximum Area Rectangle With Point Constraints II
// Difficulty: Hard
// Tags: array, tree, math, sort
//
// 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 maxRectangleArea(xCoord []int, yCoord []int) int64 {

}
```

Kotlin Solution:

```
class Solution {
fun maxRectangleArea(xCoord: IntArray, yCoord: IntArray): Long {

}
}
```

Swift Solution:

```
class Solution {
func maxRectangleArea(_ xCoord: [Int], _ yCoord: [Int]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Maximum Area Rectangle With Point Constraints II
// Difficulty: Hard
// Tags: array, tree, math, sort
//
// 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 max_rectangle_area(x_coord: Vec<i32>, y_coord: Vec<i32>) -> i64 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} x_coord
# @param {Integer[]} y_coord
# @return {Integer}
def max_rectangle_area(x_coord, y_coord)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $xCoord
* @param Integer[] $yCoord
* @return Integer
*/
function maxRectangleArea($xCoord, $yCoord) {

}
}
```

Dart Solution:

```
class Solution {
int maxRectangleArea(List<int> xCoord, List<int> yCoord) {

}
}
```

Scala Solution:

```
object Solution {
def maxRectangleArea(xCoord: Array[Int], yCoord: Array[Int]): Long = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec max_rectangle_area(x_coord :: [integer], y_coord :: [integer]) ::
integer
def max_rectangle_area(x_coord, y_coord) do

end
end
```

Erlang Solution:

```
-spec max_rectangle_area(XCoord :: [integer()], YCoord :: [integer()]) ->
integer().
max_rectangle_area(XCoord, YCoord) ->
.
```

Racket Solution:

```
(define/contract (max-rectangle-area xCoord yCoord)
(-> (listof exact-integer?) (listof exact-integer?) exact-integer?)
)
```