Problem 3025: Find the Number of Ways to Place People I

Problem Information

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

Problem Description

You are given a 2D array

points

of size

n x 2

representing integer coordinates of some points on a 2D plane, where

points[i] = [x

i

, y

i

]

.

Count the number of pairs of points

(A, B)

, where

A

is on the

upper left

side of

B

, and

there are no other points in the rectangle (or line) they make (

including the border

), except for the points

A

and

B

.

Return the count.

Example 1:

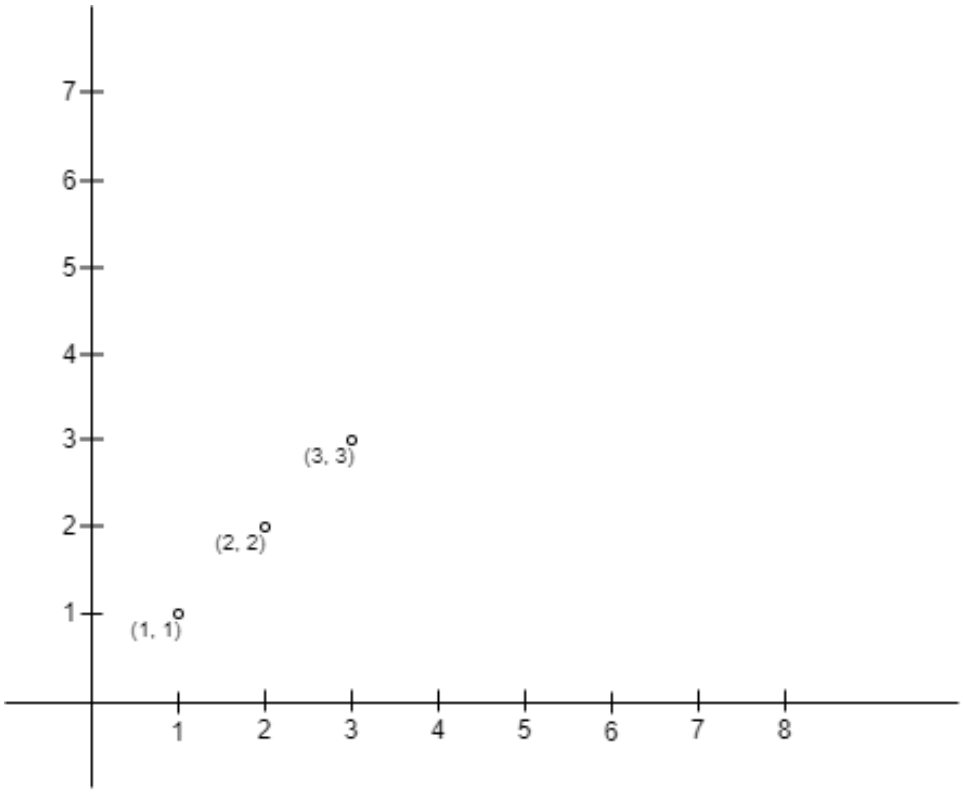
Input:

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

Output:

0

Explanation:



There is no way to choose

A

and

B

such that

A

is on the upper left side of

B

.

Example 2:

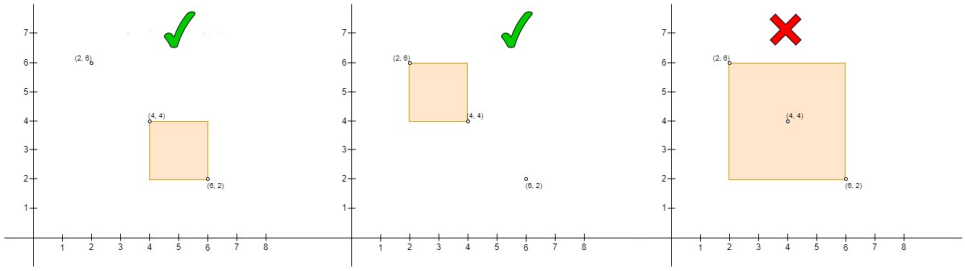
Input:

points = [[6,2],[4,4],[2,6]]

Output:

2

Explanation:



The left one is the pair

(points[1], points[0])

, where

points[1]

is on the upper left side of

points[0]

and the rectangle is empty.

The middle one is the pair

(points[2], points[1])

, same as the left one it is a valid pair.

The right one is the pair

(points[2], points[0])

, where

points[2]

is on the upper left side of

points[0]

, but

points[1]

is inside the rectangle so it's not a valid pair.

Example 3:

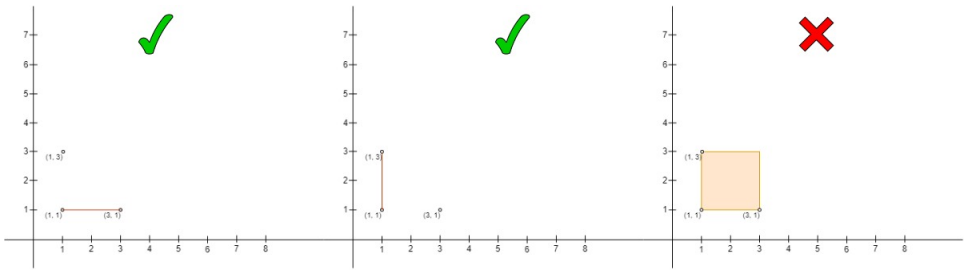
Input:

points = [[3,1],[1,3],[1,1]]

Output:

2

Explanation:



The left one is the pair

(points[2], points[0])

, where

points[2]

is on the upper left side of

points[0]

and there are no other points on the line they form. Note that it is a valid state when the two points form a line.

The middle one is the pair

(points[1], points[2])

, it is a valid pair same as the left one.

The right one is the pair

(points[1], points[0])

, it is not a valid pair as

points[2]

is on the border of the rectangle.

Constraints:

2 <= n <= 50

points[i].length == 2

0 <= points[i][0], points[i][1] <= 50

All

points[i]

are distinct.

Code Snippets

C++:

```
class Solution {
public:
int numberOfPairs(vector<vector<int>>& points) {

}
};
```

Java:

```
class Solution {
public int numberOfPairs(int[][] points) {

}
}
```

Python3:

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

Python:

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

JavaScript:

```
/**
* @param {number[][]} points
```

```
 * @return {number}
 */
var numberOfPairs = function(points) {

};
```

TypeScript:

```
function numberOfPairs(points: number[][]): number {

};
```

C#:

```
public class Solution {
public int NumberOfPairs(int[][] points) {

}
}
```

C:

```
int numberOfPairs(int** points, int pointsSize, int* pointsColSize) {

}
```

Go:

```
func numberOfPairs(points [][]int) int {

}
```

Kotlin:

```
class Solution {
fun numberOfPairs(points: Array<IntArray>): Int {

}
}
```

Swift:

```
class Solution {
func numberOfPairs(_ points: [[Int]]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn number_of_pairs(points: Vec<Vec<i32>>) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[][]} points
# @return {Integer}
def number_of_pairs(points)

end
```

PHP:

```
class Solution {

/**
* @param Integer[][] $points
* @return Integer
*/
function numberOfPairs($points) {

}
}
```

Dart:

```
class Solution {
int numberOfPairs(List<List<int>> points) {

}
}
```

Scala:

```
object Solution {
def numberOfPairs(points: Array[Array[Int]]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec number_of_pairs(points :: [[integer]]) :: integer
def number_of_pairs(points) do

end
end
```

Erlang:

```
-spec number_of_pairs(Points :: [[integer()]]) -> integer().
number_of_pairs(Points) ->
.
```

Racket:

```
(define/contract (number-of-pairs points)
(-> (listof (listof exact-integer?)) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find the Number of Ways to Place People I
* Difficulty: Medium
* Tags: array, math, sort
*
* 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 numberOfPairs(vector<vector<int>>& points) {

}
};
```

Java Solution:

```
/**
* Problem: Find the Number of Ways to Place People I
* Difficulty: Medium
* Tags: array, math, sort
*
* 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 numberOfPairs(int[][] points) {

}
}
```

Python3 Solution:

```
"""
Problem: Find the Number of Ways to Place People I
Difficulty: Medium
Tags: array, math, sort

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

```
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Find the Number of Ways to Place People I
* Difficulty: Medium
* Tags: array, math, sort
*
* 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[][]} points
* @return {number}
*/
var numberOfPairs = function(points) {

};
```

TypeScript Solution:

```
/**
* Problem: Find the Number of Ways to Place People I
* Difficulty: Medium
* Tags: array, math, sort
*
* 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 numberOfPairs(points: number[][]): number {

};
```

C# Solution:

```
/*
* Problem: Find the Number of Ways to Place People I
* Difficulty: Medium
* Tags: array, math, sort
*
* 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 NumberOfPairs(int[][] points) {

}
}
```

C Solution:

```
/*
* Problem: Find the Number of Ways to Place People I
* Difficulty: Medium
* Tags: array, math, sort
*
* 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 numberOfPairs(int** points, int pointsSize, int* pointsColSize) {

}
```

Go Solution:

```
// Problem: Find the Number of Ways to Place People I
// Difficulty: Medium
// Tags: array, math, sort
//
// 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 numberOfPairs(points [][]int) int {

}
```

Kotlin Solution:

```
class Solution {
fun numberOfPairs(points: Array<IntArray>): Int {

}
}
```

Swift Solution:

```
class Solution {
func numberOfPairs(_ points: [[Int]]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Find the Number of Ways to Place People I
// Difficulty: Medium
// Tags: array, math, sort
//
// 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 number_of_pairs(points: Vec<Vec<i32>>) -> i32 {

}
```

```
}
```

Ruby Solution:

```
# @param {Integer[][]} points
# @return {Integer}
def number_of_pairs(points)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[][] $points
* @return Integer
*/
function numberOfPairs($points) {

}
}
```

Dart Solution:

```
class Solution {
int numberOfPairs(List<List<int>> points) {

}
}
```

Scala Solution:

```
object Solution {
def numberOfPairs(points: Array[Array[Int]]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec number_of_pairs(points :: [[integer]]) :: integer
def number_of_pairs(points) do

end
end
```

Erlang Solution:

```
-spec number_of_pairs(Points :: [[integer()]]) -> integer().
number_of_pairs(Points) ->
.
```

Racket Solution:

```
(define/contract (number-of-pairs points)
(-> (listof (listof exact-integer?)) exact-integer?)
)
```