Problem 3623: Count Number of Trapezoids I

Problem Information

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

Problem Description

You are given a 2D integer array

points

, where

points[i] = [x

i

, y

i

]

represents the coordinates of the

i

th

point on the Cartesian plane.

A

horizontal

trapezoid

is a convex quadrilateral with

at least one pair

of horizontal sides (i.e. parallel to the x-axis). Two lines are parallel if and only if they have the same slope.

Return the

number of unique

horizontal

trapezoids

that can be formed by choosing any four distinct points from

points

.

Since the answer may be very large, return it

modulo

10

9

+ 7

.

Example 1:

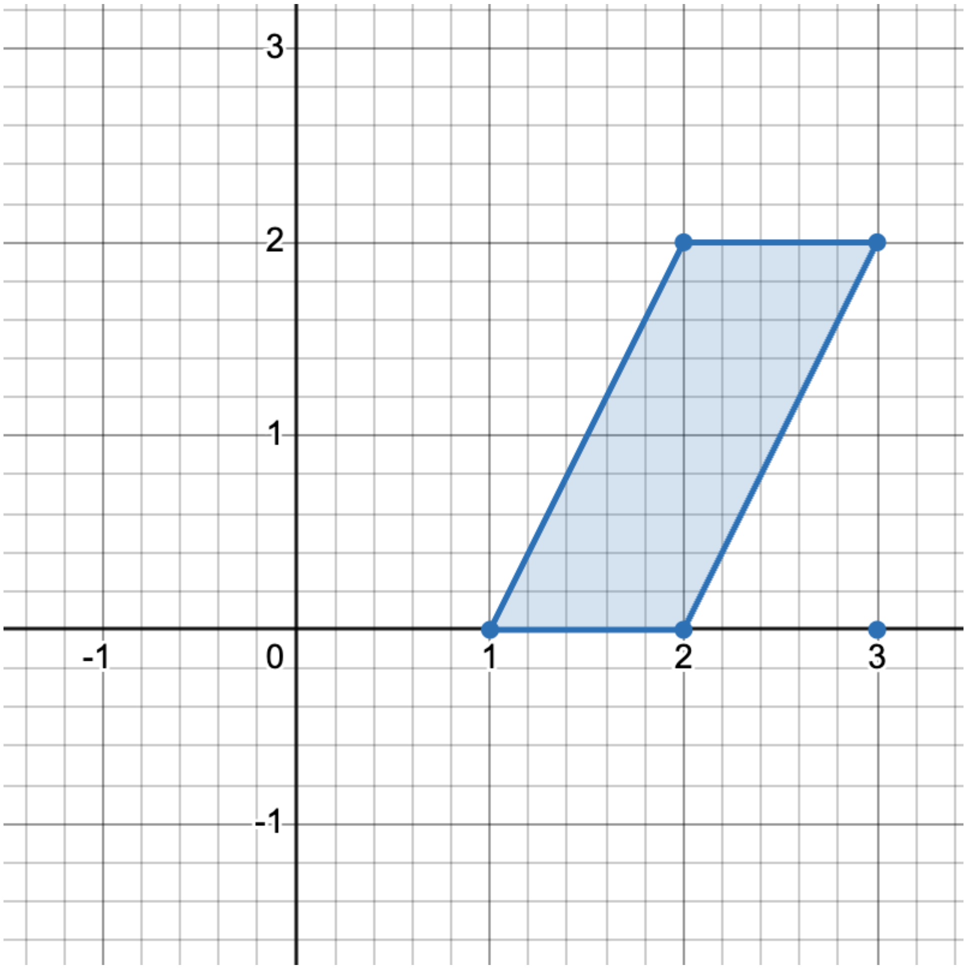
Input:

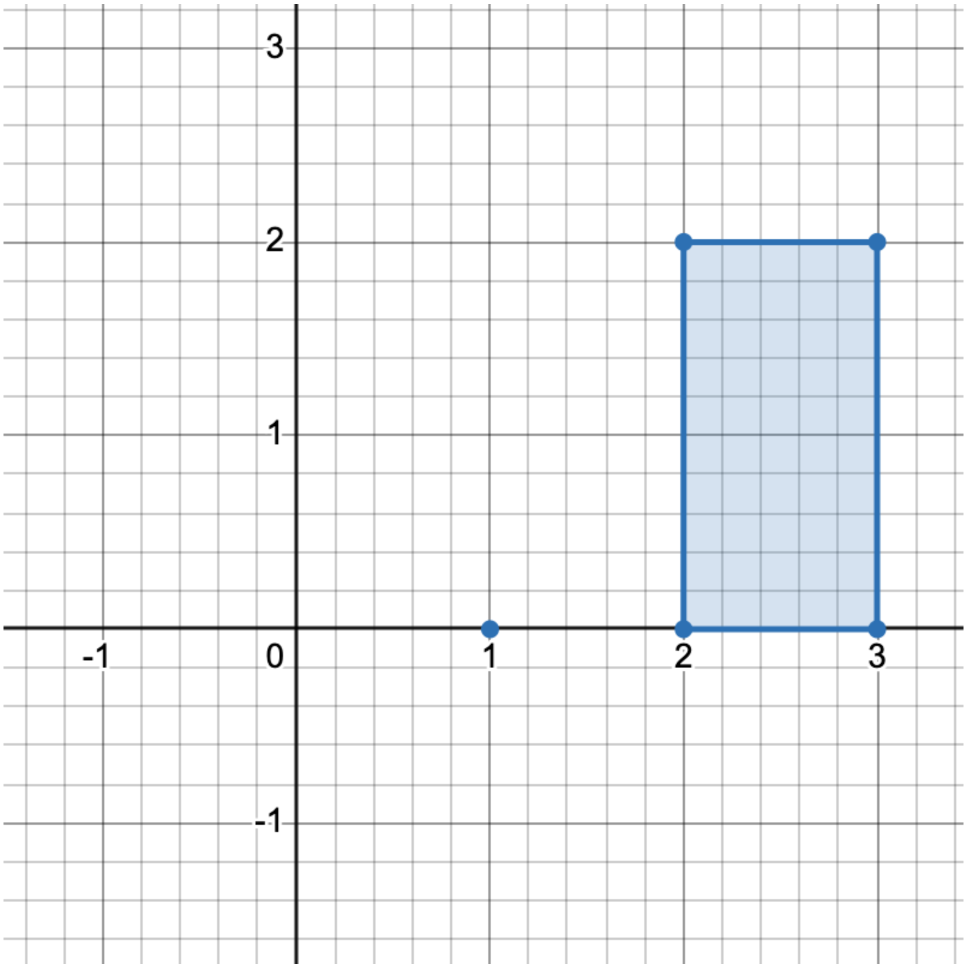
points = [[1,0],[2,0],[3,0],[2,2],[3,2]]

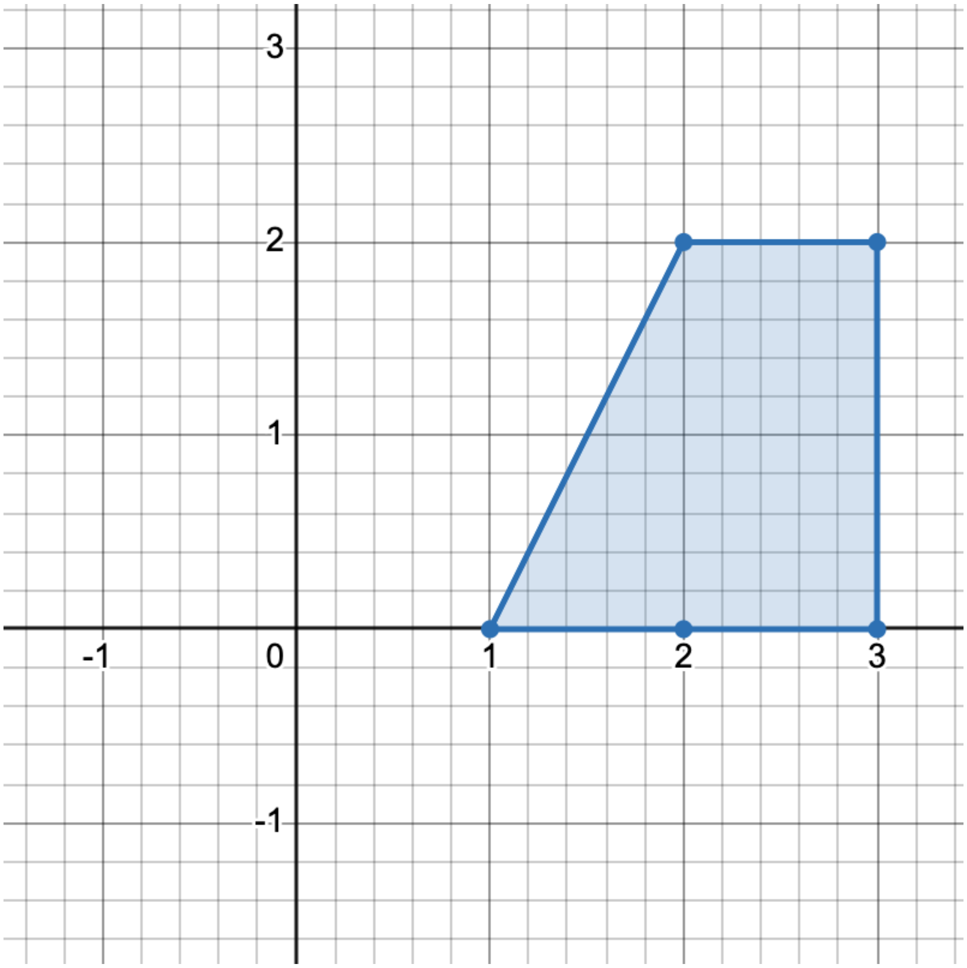
Output:

3

Explanation:




3
2
1
-1
0
1
2
3
-1



There are three distinct ways to pick four points that form a horizontal trapezoid:

Using points

[1,0]

,

[2,0]

,

[3,2]

, and

[2,2]

.

Using points

[2,0]

,

[3,0]

,

[3,2]

, and

[2,2]

.

Using points

[1,0]

,

[3,0]

,

[3,2]

, and

[2,2]

.

Example 2:

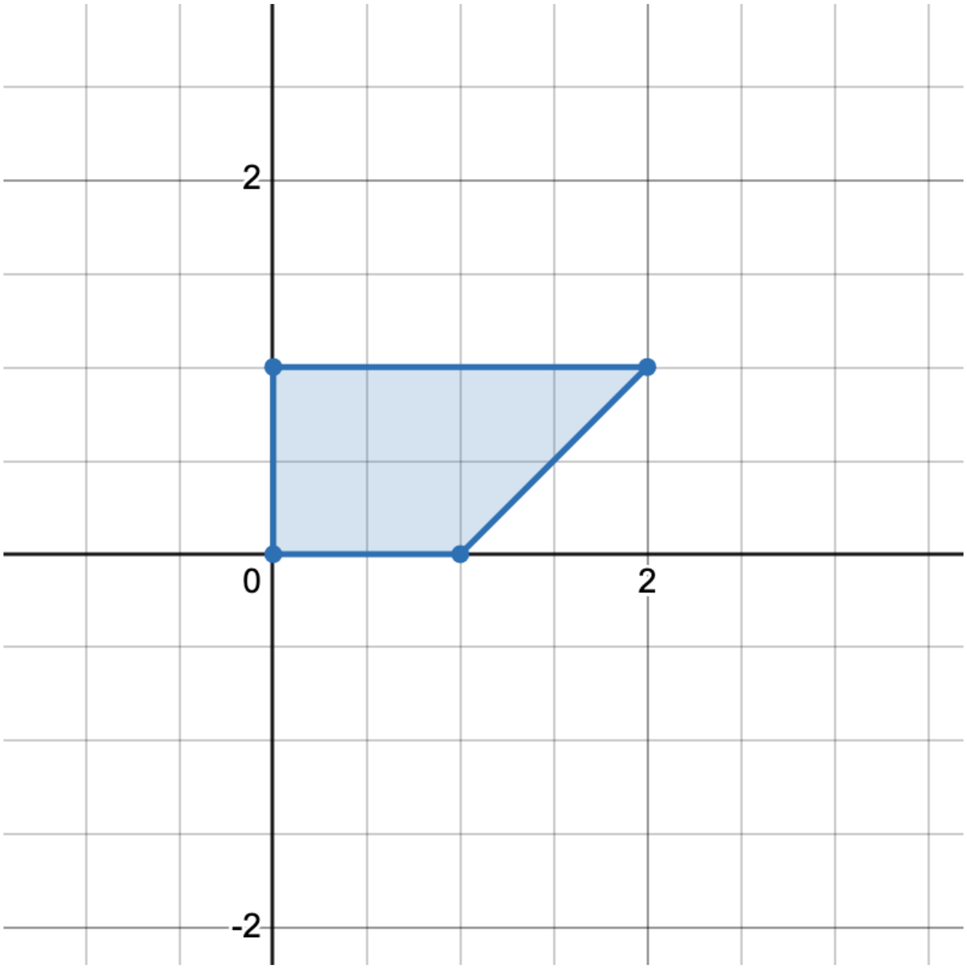
Input:

points = [[0,0],[1,0],[0,1],[2,1]]

Output:

1

Explanation:



There is only one horizontal trapezoid that can be formed.

Constraints:

4 <= points.length <= 10

5

–10

8

<= x

i

, y

i

<= 10

8

All points are pairwise distinct.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Count Number of Trapezoids I
* Difficulty: Medium
* Tags: array, graph, math, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

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

}
};
```

Java Solution:

```
/**
* Problem: Count Number of Trapezoids I
* Difficulty: Medium
* Tags: array, graph, math, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public int countTrapezoids(int[][] points) {

}
}
```

Python3 Solution:

```
"""
Problem: Count Number of Trapezoids I
Difficulty: Medium
Tags: array, graph, math, hash

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

class Solution:
def countTrapezoids(self, points: List[List[int]]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Count Number of Trapezoids I
* Difficulty: Medium
* Tags: array, graph, math, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

/**
```

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

};
```

TypeScript Solution:

```
/**
* Problem: Count Number of Trapezoids I
* Difficulty: Medium
* Tags: array, graph, math, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

function countTrapezoids(points: number[][]): number {

};
```

C# Solution:

```
/*
* Problem: Count Number of Trapezoids I
* Difficulty: Medium
* Tags: array, graph, math, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {
public int CountTrapezoids(int[][] points) {

}
}
```

C Solution:

```
/*
* Problem: Count Number of Trapezoids I
* Difficulty: Medium
* Tags: array, graph, math, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

int countTrapezoids(int** points, int pointsSize, int* pointsColSize) {

}
```

Go Solution:

```
// Problem: Count Number of Trapezoids I
// Difficulty: Medium
// Tags: array, graph, math, hash
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func countTrapezoids(points [][]int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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


}
}
```

Rust Solution:

```
// Problem: Count Number of Trapezoids I
// Difficulty: Medium
// Tags: array, graph, math, hash
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

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

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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