Problem 356: Line Reflection

Problem Information

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

Problem Description

Given

n

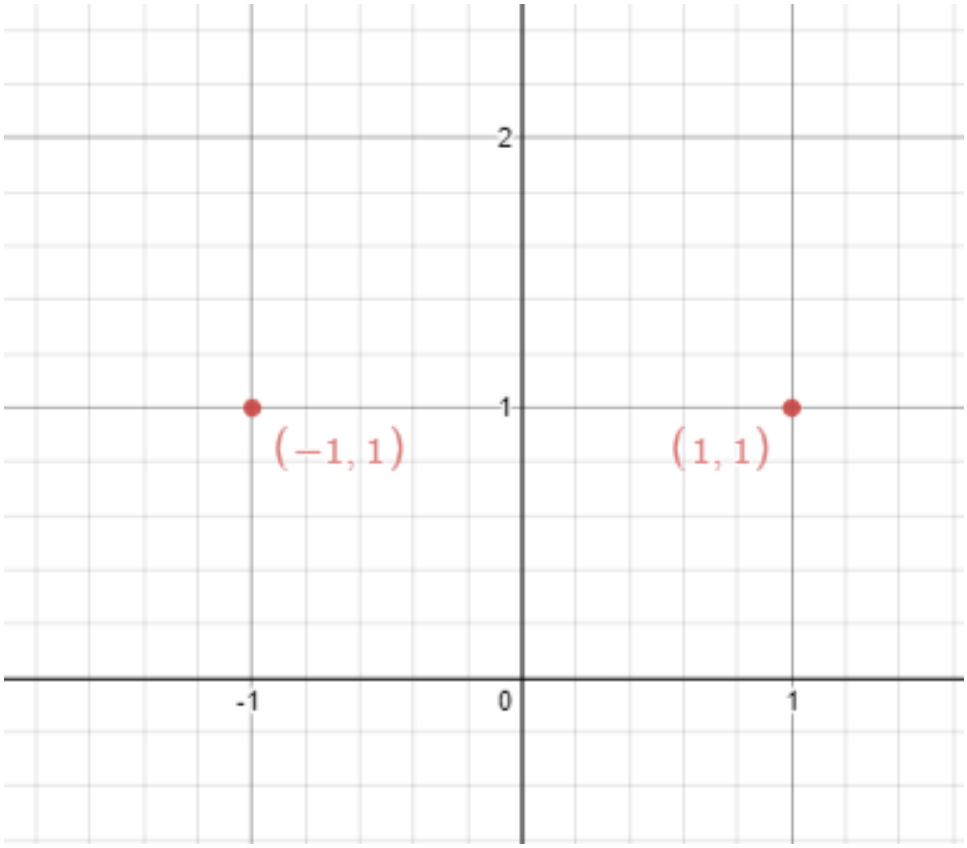
points on a 2D plane, find if there is such a line parallel to the y-axis that reflects the given points symmetrically.

In other words, answer whether or not if there exists a line that after reflecting all points over the given line, the original points' set is the same as the reflected ones.

Note

that there can be repeated points.

Example 1:



Input:

points = [[1,1],[-1,1]]

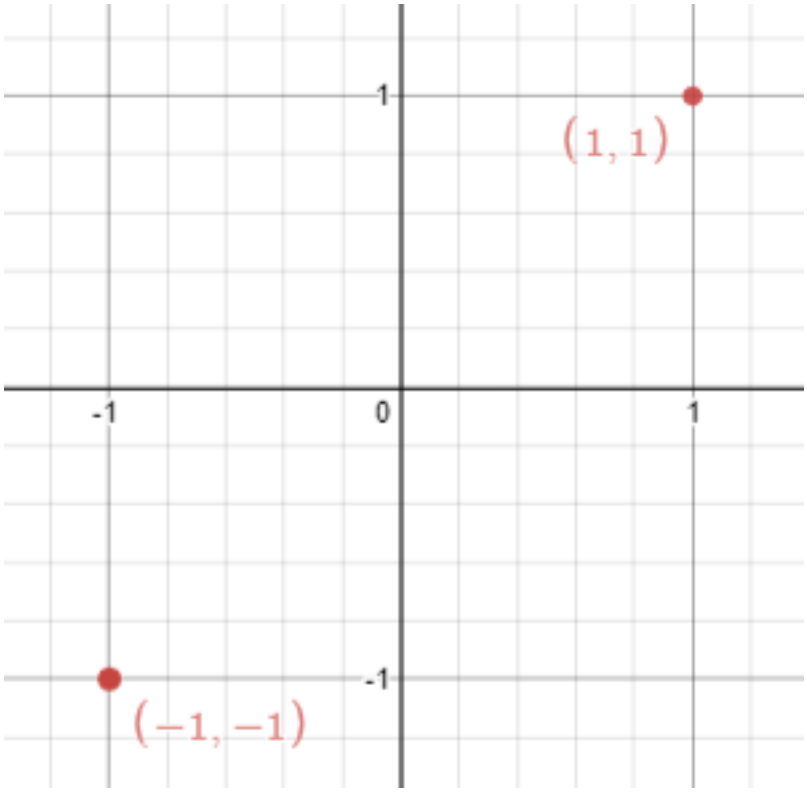
Output:

true

Explanation:

We can choose the line x = 0.

Example 2:



Input:

points = [[1,1],[-1,-1]]

Output:

false

Explanation:

We can't choose a line.

Constraints:

n == points.length

1 <= n <= 10

4

-10

8

<= points[i][j] <= 10

8

Follow up:

Could you do better than

O(n

2

)

?

Code Snippets

C++:

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

}
};
```

Java:

```
class Solution {
public boolean isReflected(int[][] points) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function isReflected(points: number[][]): boolean {

};
```

C#:

```
public class Solution {
public bool IsReflected(int[][] points) {

}
}
```

C:

```
bool isReflected(int** points, int pointsSize, int* pointsColSize) {

}
```

Go:

```
func isReflected(points [][]int) bool {

}
```

Kotlin:

```
class Solution {
fun isReflected(points: Array<IntArray>): Boolean {

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

/**
* @param Integer[][] $points
* @return Boolean
```

```
*/
function isReflected($points) {

}
}
```

Dart:

```
class Solution {
bool isReflected(List<List<int>> points) {

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (is-reflected points)
(-> (listof (listof exact-integer?)) boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Line Reflection
* Difficulty: Medium
* Tags: array, 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:
bool isReflected(vector<vector<int>>& points) {

}
};
```

Java Solution:

```
/**
* Problem: Line Reflection
* Difficulty: Medium
* Tags: array, 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 boolean isReflected(int[][] points) {

}
}
```

Python3 Solution:

```
"""
Problem: Line Reflection
Difficulty: Medium
Tags: array, 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 isReflected(self, points: List[List[int]]) -> bool:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Line Reflection
* Difficulty: Medium
* Tags: array, 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 {boolean}
*/
var isReflected = function(points) {
```

```
};
```

TypeScript Solution:

```
/**
* Problem: Line Reflection
* Difficulty: Medium
* Tags: array, 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 isReflected(points: number[][]): boolean {

};
```

C# Solution:

```
/*
* Problem: Line Reflection
* Difficulty: Medium
* Tags: array, 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 bool IsReflected(int[][] points) {

}
}
```

C Solution:

```
/*
* Problem: Line Reflection
* Difficulty: Medium
```

```
* Tags: array, 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
*/

bool isReflected(int** points, int pointsSize, int* pointsColSize) {

}
```

Go Solution:

```
// Problem: Line Reflection
// Difficulty: Medium
// Tags: array, 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 isReflected(points [][]int) bool {

}
```

Kotlin Solution:

```
class Solution {
fun isReflected(points: Array<IntArray>): Boolean {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Line Reflection
// Difficulty: Medium
// Tags: array, 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 is_reflected(points: Vec<Vec<i32>>) -> bool {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

```
class Solution {
bool isReflected(List<List<int>> points) {
```

```
}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (is-reflected points)
(-> (listof (listof exact-integer?)) boolean?)
)
```