Problem 469: Convex Polygon

Problem Information

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

Problem Description

You are given an array of points on the

X-Y

plane

points

where

points[i] = [x

i

, y

i

]

. The points form a polygon when joined sequentially.

Return

true

if this polygon is

convex

and

false

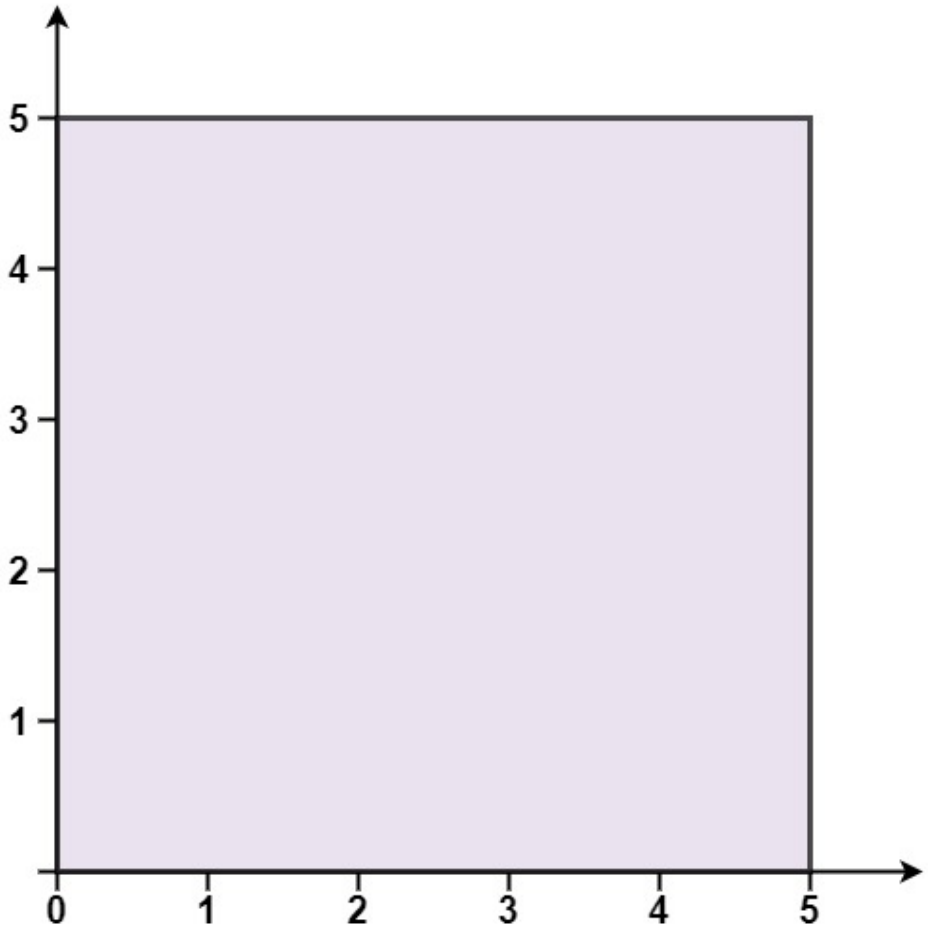
otherwise.

You may assume the polygon formed by given points is always a

simple polygon

. In other words, we ensure that exactly two edges intersect at each vertex and that edges otherwise don't intersect each other.

Example 1:



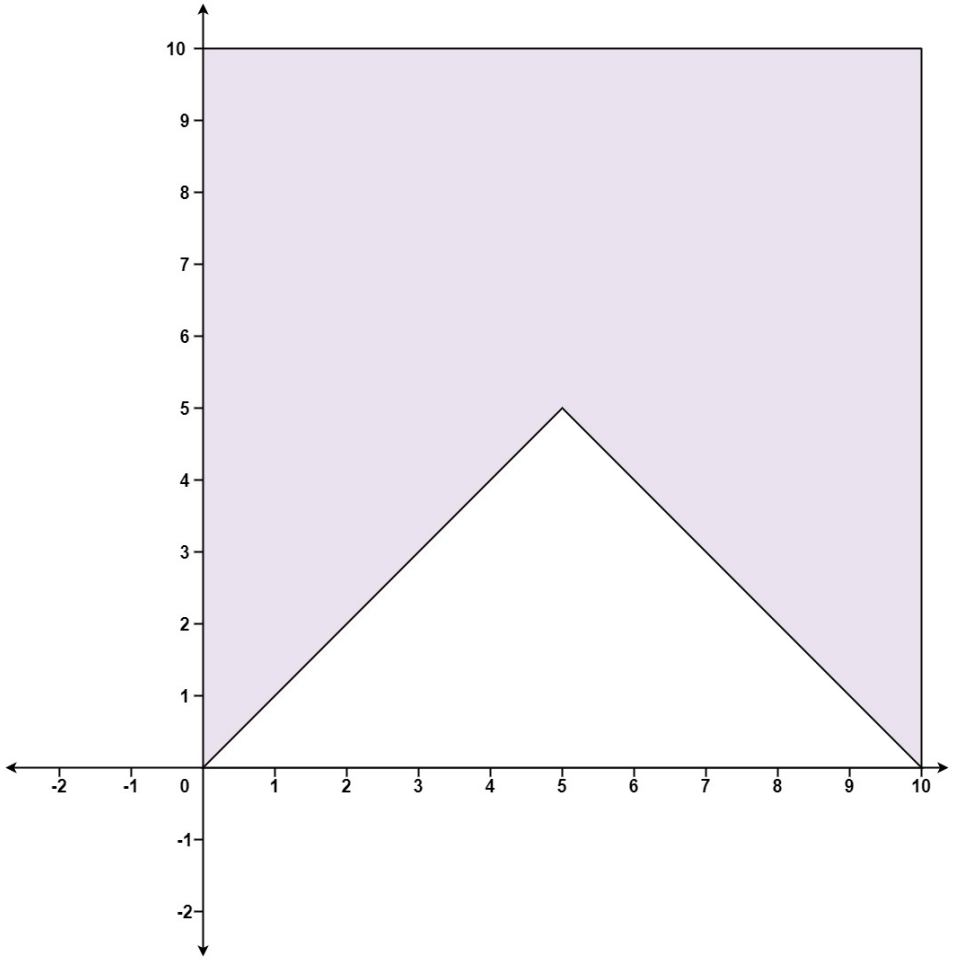
Input:

points = [[0,0],[0,5],[5,5],[5,0]]

Output:

true

Example 2:



Input:

points = [[0,0],[0,10],[10,10],[10,0],[5,5]]

Output:

false

Constraints:

3 <= points.length <= 10

4

points[i].length == 2

-10

4

<= x

i

, y

i

<= 10

4

All the given points are

unique

.

Code Snippets

C++:

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

```
}
};
```

Java:

```
class Solution {
public boolean isConvex(List<List<Integer>> points) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

```
public class Solution {
public bool IsConvex(IList<IList<int>> points) {

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

```
class Solution {
fun isConvex(points: List<List<Int>>): Boolean {

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

```
object Solution {
def isConvex(points: List[List[Int]]): Boolean = {

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Convex Polygon
* Difficulty: Medium
* Tags: array, math
*
* 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:
bool isConvex(vector<vector<int>>& points) {

}
};
```

Java Solution:

```
/**
* Problem: Convex Polygon
* Difficulty: Medium
* Tags: array, math
*
* 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 boolean isConvex(List<List<Integer>> points) {

}
}
```

Python3 Solution:

```
"""
Problem: Convex Polygon
Difficulty: Medium
Tags: array, math

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Convex Polygon
* Difficulty: Medium
```

```
* Tags: array, math
*
* 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 {boolean}
*/
var isConvex = function(points) {

};
```

TypeScript Solution:

```
/**
* Problem: Convex Polygon
* Difficulty: Medium
* Tags: array, math
*
* 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 isConvex(points: number[][]): boolean {

};
```

C# Solution:

```
/*
* Problem: Convex Polygon
* Difficulty: Medium
* Tags: array, math
*
* 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 bool IsConvex(IList<IList<int>> points) {

}
}
```

C Solution:

```
/*
* Problem: Convex Polygon
* Difficulty: Medium
* Tags: array, math
*
* 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
*/

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

}
```

Go Solution:

```
// Problem: Convex Polygon
// Difficulty: Medium
// Tags: array, math
//
// 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 isConvex(points [][]int) bool {

}
```

Kotlin Solution:

```
class Solution {
fun isConvex(points: List<List<Int>>): Boolean {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Convex Polygon
// Difficulty: Medium
// Tags: array, math
//
// 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 is_convex(points: Vec<Vec<i32>>) -> bool {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {
```

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

```
object Solution {
def isConvex(points: List[List[Int]]): Boolean = {

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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