Problem 391: Perfect Rectangle

Problem Information

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

Problem Description

Given an array

rectangles

where

rectangles[i] = [x

i

, y

i

, a

i

, b

i

]

represents an axis-aligned rectangle. The bottom-left point of the rectangle is

(x

i

, y

i

)

and the top-right point of it is

(a

i

, b

i

)

.

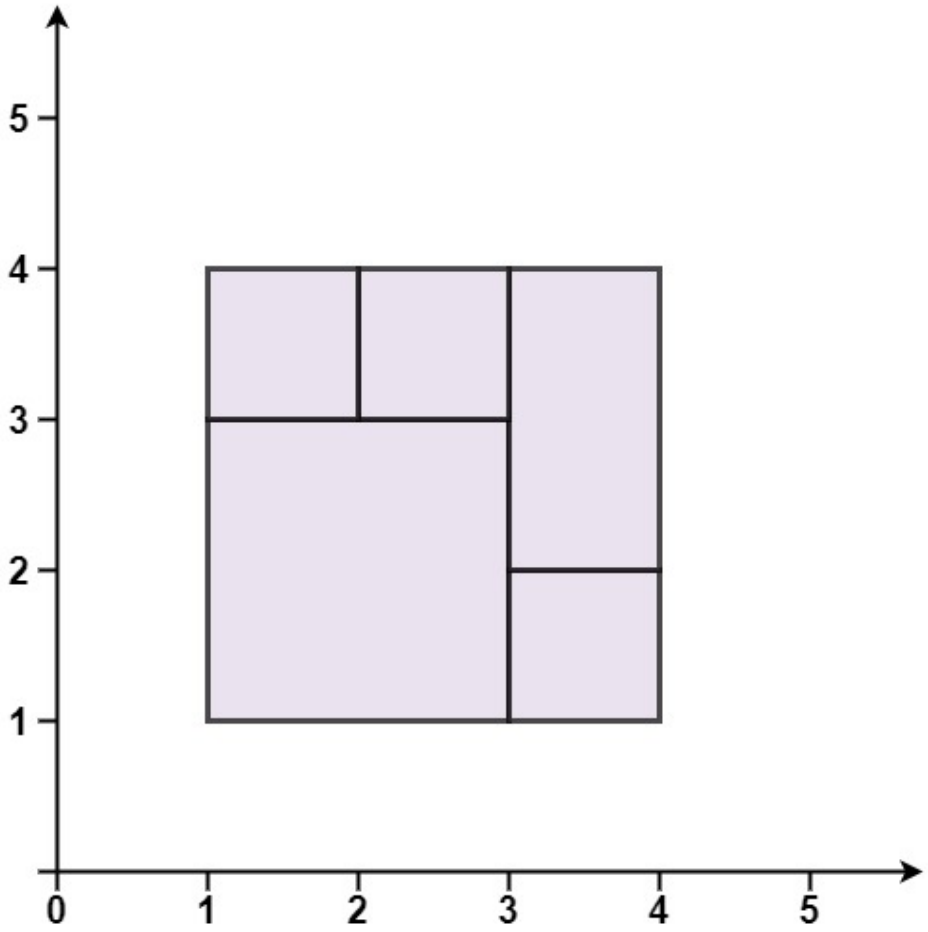
Return

true

if all the rectangles together form an exact cover of a rectangular region

.

Example 1:



Input:

rectangles = [[1,1,3,3],[3,1,4,2],[3,2,4,4],[1,3,2,4],[2,3,3,4]]

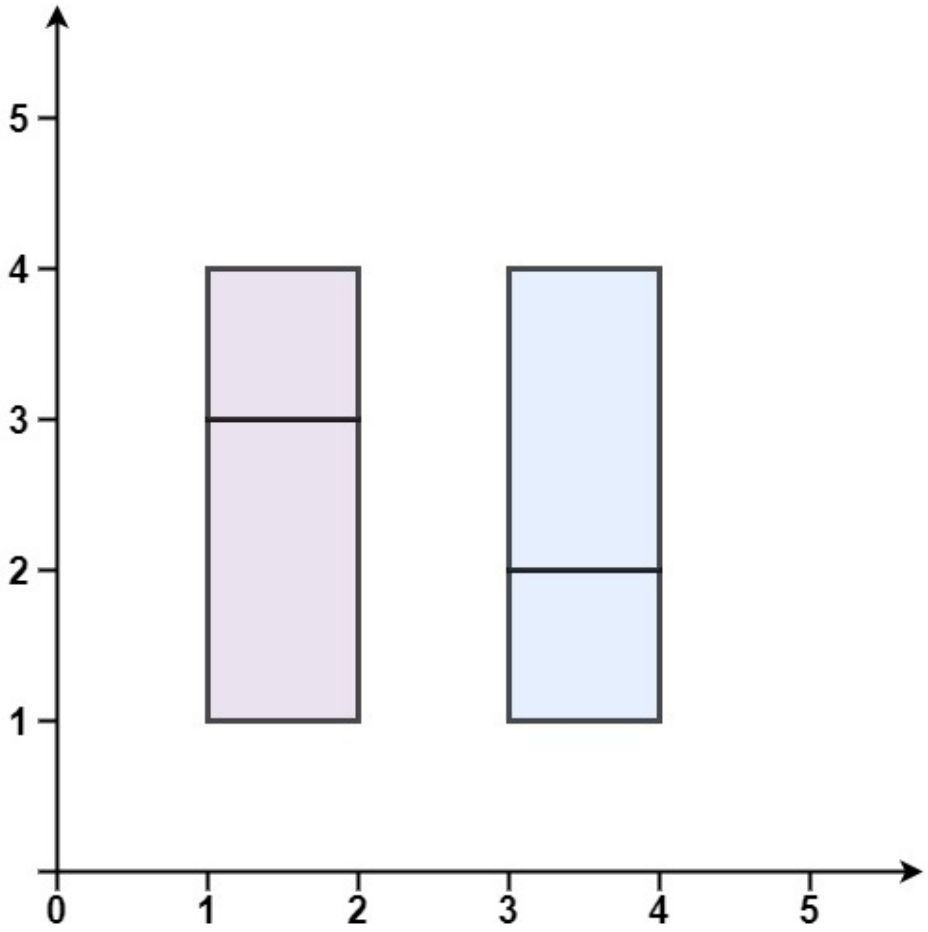
Output:

true

Explanation:

All 5 rectangles together form an exact cover of a rectangular region.

Example 2:



Input:

rectangles = [[1,1,2,3],[1,3,2,4],[3,1,4,2],[3,2,4,4]]

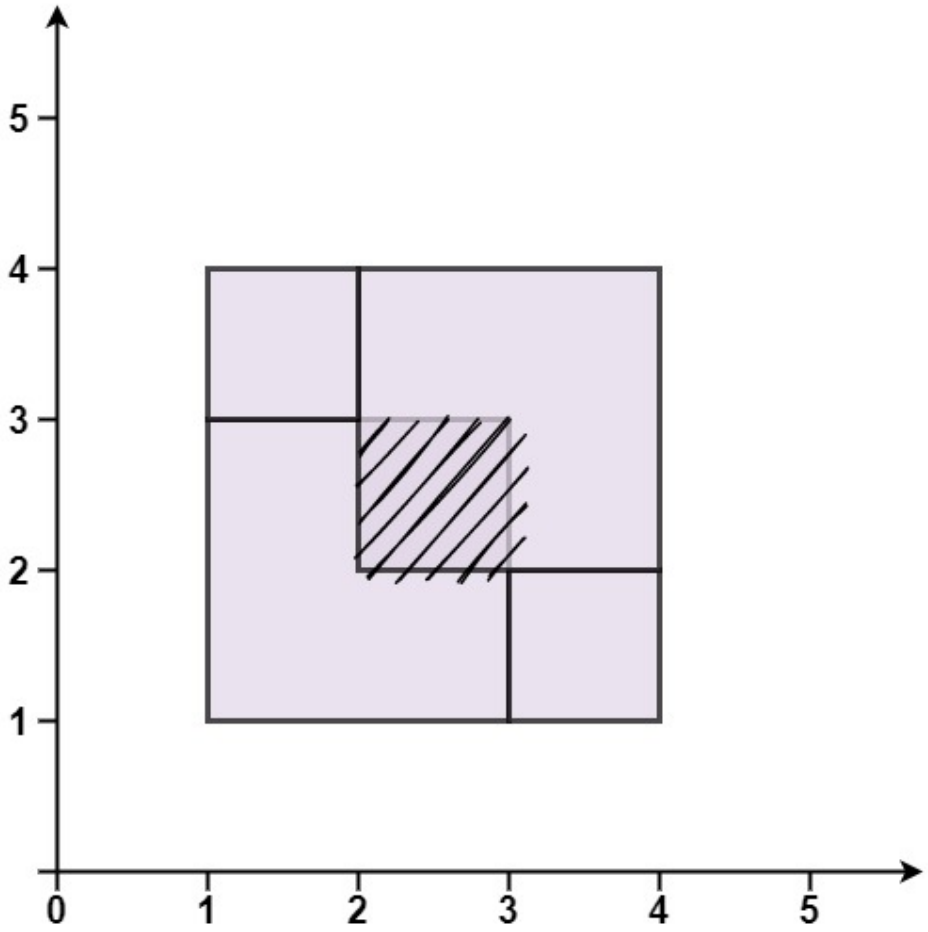
Output:

false

Explanation:

Because there is a gap between the two rectangular regions.

Example 3:



Input:

rectangles = [[1,1,3,3],[3,1,4,2],[1,3,2,4],[2,2,4,4]]

Output:

false

Explanation:

Because two of the rectangles overlap with each other.

Constraints:

1 <= rectangles.length <= 2 * 10

4

rectangles[i].length == 4

-10

5

<= x

i

< a

i

<= 10

5

-10

5

<= y

i

< b

i

<= 10

5

Code Snippets

C++:

```
class Solution {
public:
bool isRectangleCover(vector<vector<int>>& rectangles) {

}
};
```

Java:

```
class Solution {
public boolean isRectangleCover(int[][] rectangles) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function isRectangleCover(rectangles: number[][]): boolean {
```

```
};
```

C#:

```
public class Solution {
public bool IsRectangleCover(int[][] rectangles) {

}
}
```

C:

```
bool isRectangleCover(int** rectangles, int rectanglesSize, int*
rectanglesColSize) {

}
```

Go:

```
func isRectangleCover(rectangles [][]int) bool {

}
```

Kotlin:

```
class Solution {
fun isRectangleCover(rectangles: Array<IntArray>): Boolean {

}
}
```

Swift:

```
class Solution {
func isRectangleCover(_ rectangles: [[Int]]) -> Bool {

}
}
```

Rust:

```
impl Solution {
pub fn is_rectangle_cover(rectangles: Vec<Vec<i32>>) -> bool {

}
}
```

Ruby:

```
# @param {Integer[][]} rectangles
# @return {Boolean}
def is_rectangle_cover(rectangles)

end
```

PHP:

```
class Solution {

/**
* @param Integer[][] $rectangles
* @return Boolean
*/
function isRectangleCover($rectangles) {

}
}
```

Dart:

```
class Solution {
bool isRectangleCover(List<List<int>> rectangles) {

}
}
```

Scala:

```
object Solution {
def isRectangleCover(rectangles: Array[Array[Int]]): Boolean = {

}
}
```

Elixir:

```
defmodule Solution do
@spec is_rectangle_cover(rectangles :: [[integer]]) :: boolean
def is_rectangle_cover(rectangles) do

end
end
```

Erlang:

```
-spec is_rectangle_cover(Rectangles :: [[integer()]]) -> boolean().
is_rectangle_cover(Rectangles) ->
.
```

Racket:

```
(define/contract (is-rectangle-cover rectangles)
(-> (listof (listof exact-integer?)) boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Perfect Rectangle
* Difficulty: Hard
* 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 isRectangleCover(vector<vector<int>>& rectangles) {

}
};
```

Java Solution:

```
/**
* Problem: Perfect Rectangle
* Difficulty: Hard
* 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 isRectangleCover(int[][] rectangles) {

}
}
```

Python3 Solution:

```
"""
Problem: Perfect Rectangle
Difficulty: Hard
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 isRectangleCover(self, rectangles: List[List[int]]) -> bool:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

```
"""
```

JavaScript Solution:

```
/**
* Problem: Perfect Rectangle
* Difficulty: Hard
* 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[][]} rectangles
* @return {boolean}
*/
var isRectangleCover = function(rectangles) {

};
```

TypeScript Solution:

```
/**
* Problem: Perfect Rectangle
* Difficulty: Hard
* 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 isRectangleCover(rectangles: number[][]): boolean {

};
```

C# Solution:

```
/*
* Problem: Perfect Rectangle
* Difficulty: Hard
* 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 IsRectangleCover(int[][] rectangles) {

}
}
```

C Solution:

```
/*
* Problem: Perfect Rectangle
* Difficulty: Hard
* 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 isRectangleCover(int** rectangles, int rectanglesSize, int*
rectanglesColSize) {

}
```

Go Solution:

```
// Problem: Perfect Rectangle
// Difficulty: Hard
// 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 isRectangleCover(rectangles [][]int) bool {

}
```

Kotlin Solution:

```
class Solution {
fun isRectangleCover(rectangles: Array<IntArray>): Boolean {

}
}
```

Swift Solution:

```
class Solution {
func isRectangleCover(_ rectangles: [[Int]]) -> Bool {

}
}
```

Rust Solution:

```
// Problem: Perfect Rectangle
// Difficulty: Hard
// 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_rectangle_cover(rectangles: Vec<Vec<i32>>) -> bool {

}
}
```

Ruby Solution:

```
# @param {Integer[][]} rectangles
# @return {Boolean}
```

```
def is_rectangle_cover(rectangles)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[][] $rectangles
* @return Boolean
*/
function isRectangleCover($rectangles) {

}
}
```

Dart Solution:

```
class Solution {
bool isRectangleCover(List<List<int>> rectangles) {

}
}
```

Scala Solution:

```
object Solution {
def isRectangleCover(rectangles: Array[Array[Int]]): Boolean = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec is_rectangle_cover(rectangles :: [[integer]]) :: boolean
def is_rectangle_cover(rectangles) do

end
end
```

Erlang Solution:

```
-spec is_rectangle_cover(Rectangles :: [[integer()]]) -> boolean().
is_rectangle_cover(Rectangles) ->
.
```

Racket Solution:

```
(define/contract (is-rectangle-cover rectangles)
(-> (listof (listof exact-integer?)) boolean?)
)
```