<https://leetcode.com/problems/rectangle-overlap/description/>

An axis-aligned rectangle is represented as a list [x1, y1, x2, y2], where (x1, y1) is the coordinate of its bottom-left corner, and (x2, y2) is the coordinate of its top-right corner. Its top and bottom edges are parallel to the X-axis, and its left and right edges are parallel to the Y-axis.

Two rectangles overlap if the area of their intersection is **positive**. To be clear, two rectangles that only touch at the corner or edges do not overlap.

Given two axis-aligned rectangles rec1 and rec2, return true *if they overlap, otherwise return* false.

**Example 1:**

Input: rec1 = [0,0,2,2], rec2 = [1,1,3,3]

Output: true

**Example 2:**

Input: rec1 = [0,0,1,1], rec2 = [1,0,2,1]

Output: false

**Example 3:**

Input: rec1 = [0,0,1,1], rec2 = [2,2,3,3]

Output: false

**Constraints:**

* rec1.length == 4
* rec2.length == 4
* -109 <= rec1[i], rec2[i] <= 109
* rec1 and rec2 represent a valid rectangle with a non-zero area.

**Attempt 1: 2023-12-03**

**Wrong Solution:**

class Solution {

public boolean isRectangleOverlap(int[] rec1, int[] rec2) {

if((rec1[2] <= rec2[0] || rec2[2] <= rec1[0]) && (rec1[3] <= rec2[1] || rec2[3] <= rec1[1])) {

return false;

}

return true;

}

}

**Test out by**

rec1 = [0,0,1,1]

rec2 = [1,0,2,1]

**Step by step debug**

**The issue is coming from 'AND' operator between condition 1 and condition 2 as '&&', if check above error out example, when condition 1 satisfied, which means on x-axis no overlap, then no matter where how y-axis looks like, even two rectangles y-axis have intersection doesn't matter, there is no chance these two rectangles will intersect when their x-axis have no intersections, and vice versa, if condition 2 satisfied, we don't need condition 1 satisfied, so two conditions relation should be 'OR' as '||'**

[x4,y4]

Rec2

[x3,y3]

[x2,y2]

Rec1

[x1,y1]

The '=' comes from "To be clear, two rectangles that only touch at the corner or edges do not overlap."

No overlap condition 1:

if [x1,x2] and [x3,x4] no overlap

x2 <= x3 || x4 <= x1

No overlap condition 2:

if [y1,y2] and [y3,y4] no overlap

y2 <= y3 || y4 <= y1

Rec1 & Rec2 No overlap = condition 1 && condition 2

(x2 <= x3 || x4 <= x1) && (y2 <= y3 || y4 <= y1)

Mapping to rec1 and rec2 representation:

x1 = rec1[0], y1 = rec1[1]

x2 = rec1[2], y2 = rec1[3]

x3 = rec2[0], y3 = rec2[1]

x4 = rec2[2], y4 = rec2[3]

The final "No overlap" condtion for two rectangles is below:

=> (rec1[2] <= rec2[0] || rec2[2] <= rec1[0]) && (rec1[3] <= rec2[1] || rec2[3] <= rec1[1])

But for given example rec1 = [0,0,1,1], rec2 = [1,0,2,1] it shows:

(1 <= 1 || 2 <= 0) && (1 <= 0 || 1 <= 0) => for condition 2 it not match and decision is overlapped

But from the actual drawing graph, the two rectangles no overlap

\* \* [1,1] \* [2,1]

\* \* \*

[0,0] \* [1,0] \* \*

The issue is coming from 'AND' operator between condition 1 and condition 2 as '&&', if check above error out example, when condition 1 satisfied, which means on x-axis no overlap, then no matter where how y-axis looks like, even two rectangles y-axis have intersection doesn't matter, there is no chance these two rectangles will intersect when their x-axis have no intersections, and vice versa, if condition 2 satisfied, we don't need condition 1 satisfied, so two conditions relation should be 'OR' as '||'

**Solution 1:  Intersection detect (10 min)**

**Style 1:**

class Solution {

public boolean isRectangleOverlap(int[] rec1, int[] rec2) {

if(rec1[2] <= rec2[0] || rec2[2] <= rec1[0] || rec1[3] <= rec2[1] || rec2[3] <= rec1[1]) {

return false;

}

return true;

}

}

Time Complexity: O(1)

Space Complexity: O(1)

**Style 2:**

class Solution {

public boolean isRectangleOverlap(int[] rec1, int[] rec2) {

// Interval overlapping: Max(left1, left2) < Min(right1, right2)

// x has overlapping && y has overlapping

int x1 = rec1[0], y1 = rec1[1], x2 = rec1[2], y2 = rec1[3];

int x3 = rec2[0], y3 = rec2[1], x4 = rec2[2], y4 = rec2[3];

boolean overlapX = false;

boolean overlapY = false;

if(Math.max(x1, x3) < Math.min(x2, x4))

overlapX = true;

if(Math.max(y1, y3) < Math.min(y2, y4))

overlapY = true;

return overlapX && overlapY;

}

}

Time Complexity: O(1)

Space Complexity: O(1)

**Refer to**

<https://leetcode.com/problems/rectangle-overlap/solutions/185809/template-interval-overlapping/>

For overlapping questions:

Interval A = [leftA, rightA]

Interval B = [leftB, rightB]

Overlapping region: [max(leftA, leftB) , min(rightA, rightB)]

which means if(max(leftA, leftB) < min(rightA, rightB)), there is an overlap.

So the code of this problem is to check whether x is overlapped && y is overlapped.

class Solution {

public boolean isRectangleOverlap(int[] rec1, int[] rec2) {

// Interval overlapping: Max(left1, left2) < Min(right1, right2)

// x has overlapping && y has overlapping

int x1 = rec1[0], y1 = rec1[1], x2 = rec1[2], y2 = rec1[3];

int x3 = rec2[0], y3 = rec2[1], x4 = rec2[2], y4 = rec2[3];

boolean overlapX = false;

boolean overlapY = false;

if(Math.max(x1, x3) < Math.min(x2, x4))

overlapX = true;

if(Math.max(y1, y3) < Math.min(y2, y4))

overlapY = true;

return overlapX && overlapY;

}

}