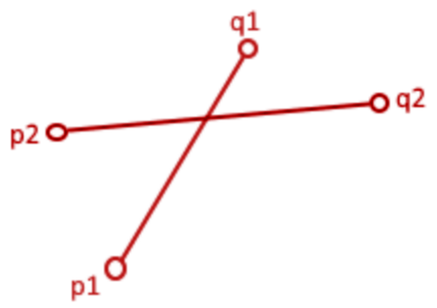
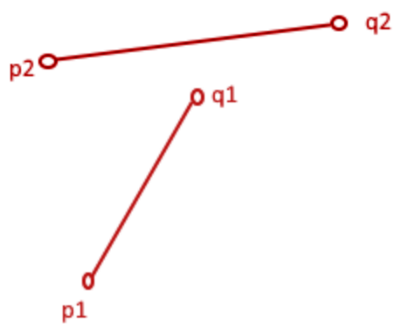
Check if two line segments intersect

Problem Statement : You are given two line segments, l1 and l2, each of which is characterized by its two endpoints p1 and p2. You are required to find out whether the line segments intersect.

Solution:



i)



ii)

Above here is shown two possible scenarios. i) For intersection and ii) For non-intersection
In case i)

- orientation of (p1, q1, p2) is anticlockwise and (p1, q1, q2) is clockwise
- orientation of (p2, q2, p1) is clockwise and (p2, q2, q1) is anticlockwise

Hence the both are different.

While in case ii)

- orientation of (p1, q1, p2) is anticlockwise and (p1, q1, q2) is clockwise
- but, the orientation of (p2, q2, p1) and (p2, q2, q1) are both clockwise hence they do not intersect.

So we can say that for lines to intersect, the orientation of an endpoint with respect to another line segment should be opposite to the other endpoint. We also should handle the special cases when one of the points lies on the other line segment.

Here is the implementation which checks the orientation of 3 points by calculating the cross - product and checks if a point lies on the line segment.

Pseudocode:

```
/*
        function to check if the point p lies on the segment defined by line l1
*/
function onSegment(line l1, Point p)
```

```
    if (p.x <= max(l1.p1.x, l1.p2.x) && p.x <= min(l1.p1.x, l1.p2.x) &&
        (p.y <= max(l1.p1.y, l1.p2.y) && p.y <= min(l1.p1.y, l1.p2.y)))
        return true
    return false


/*
    function to check if the point c lies on the line a->b
    or is oriented clockwise or anticlockwise
*/
function direction(Point a, Point b, Point c)

    //   calculate the cross product (b - a) x (c - b)
    val = (b.y-a.y)*(c.x-b.x) - (b.x-a.x)*(c.y-b.y)
    if (val == 0)
        //   collinear
        return 0
    else if val < 0
        //   anti-clockwise direction
        return 2
    else
        //   clockwise direction
        return 1


/*
    function to check if the line l1 and l2 intersect
*/
function Intersects(line l1, line l2)

    //   four direction for two lines and points of other line
    dir1 = direction(l1.p1, l1.p2, l2.p1)
    dir2 = direction(l1.p1, l1.p2, l2.p2)
    dir3 = direction(l2.p1, l2.p2, l1.p1)
    dir4 = direction(l2.p1, l2.p2, l1.p2)

    if dir1 != dir2 && dir3 != dir4
        //   they are intersecting
        return true

    //   when p2 of line2 are on the line1
    if dir1 == 0 && onSegment(l1, l2.p1)
        return true
```

```
//   when p1 of line2 are on the line1
if dir2 == 0 && onSegment(l1, l2.p2)
        return true

//   when p2 of line1 are on the line2
if dir3 == 0 && onSegment(l2, l1.p1)
        return true

//   when p1 of line1 are on the line2
if dir4 == 0 && onSegment(l2, l1.p2)
        return true

return false
```

TIme complexity : O(1), since we are doing only a constant number of operations to calculate the 4 directions and checking if the point is on segment.
Space complexity : O(1), since constant additional space is used.