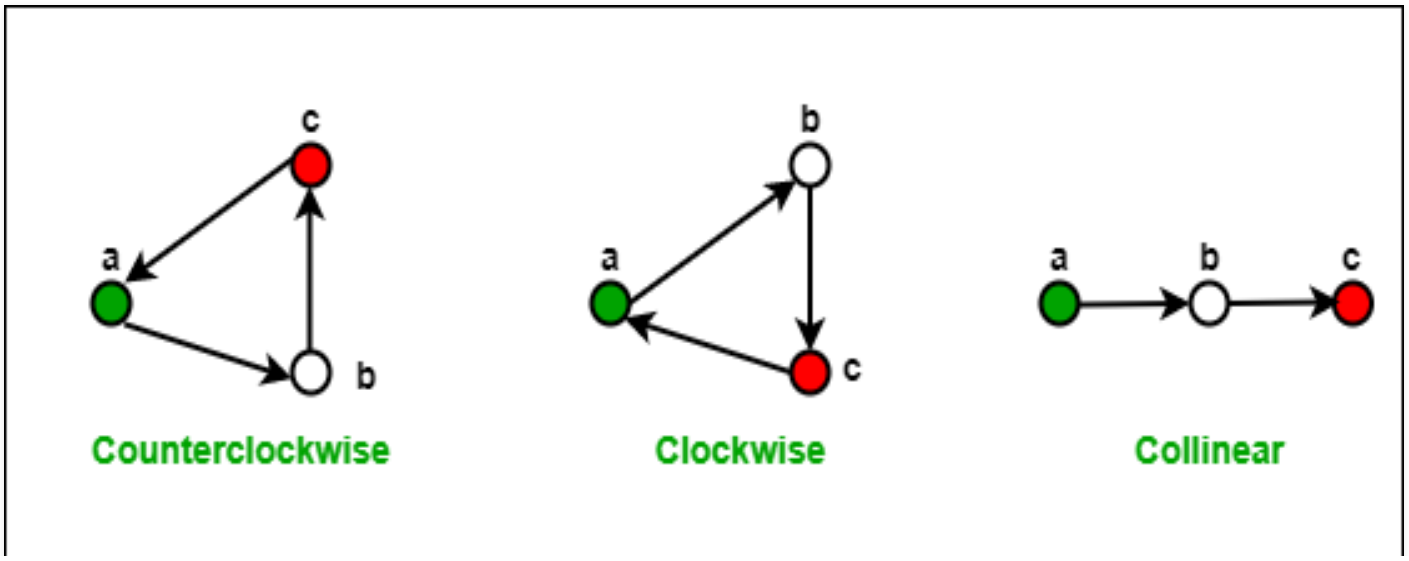
Orientation of 3 ordered points

Difficulty Level : Easy • Last Updated : 01 Sep, 2021

Orientation of an ordered triplet of points in the plane can be

- counterclockwise
- clockwise
- collinear

The following diagram shows different possible orientations of (a,b,c)



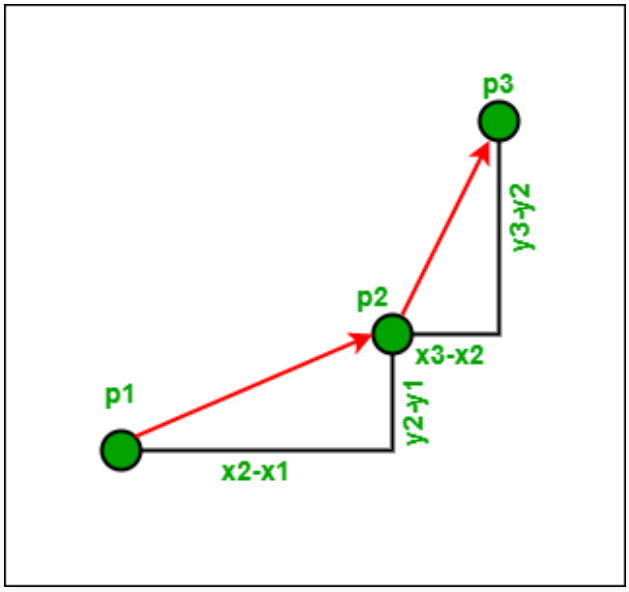
If orientation of (p1, p2, p3) is clockwise, then orientation of (p3, p2, p1) is counterclockwise and vice versa is also true.

Example: Given three points p1, p2 and p3, find orientation of (p1, p2, p3).

Output: Collinear

How to compute Orientation?

The idea is to use slope.



Slope of line segment (p1, p2): σ = (y2 - y1)/(x2 - x1)
Slope of line segment (p2, p3): τ = (y3 - y2)/(x3 - x2)

If σ > τ, the orientation is clockwise (right turn)

Using above values of σ and τ, we can conclude that,
the orientation depends on sign of below expression:

(y2 - y1)*(x3 - x2) - (y3 - y2)*(x2 - x1)

Above expression is negative when σ < τ, i.e., counterclockwise

Below is the implementation of above idea.

```
{
    int x, y;
};

// To find orientation of ordered triplet (p1, p2, p3).
// The function returns following values
// 0 --> p, q and r are collinear
// 1 --> Clockwise
// 2 --> Counterclockwise
int orientation(Point p1, Point p2, Point p3)
{
    // See 10th slides from following link for derivation
    // of the formula
    int val = (p2.y - p1.y) * (p3.x - p2.x) -
              (p2.x - p1.x) * (p3.y - p2.y);

    if (val == 0) return 0;  // collinear

    return (val > 0)? 1: 2; // clock or counterclock wise
}

// Driver program to test above functions
int main()
{
    Point p1 = {0, 0}, p2 = {4, 4}, p3 = {1, 2};
    int o = orientation(p1, p2, p3);
    if (o==0)         cout << "Linear";
    else if (o == 1)  cout << "Clockwise";
    else                 cout << "CounterClockwise";
    return 0;
}
```

Java

```
// JAVA Code to find Orientation of 3
// ordered points
class Point
{
    int x, y;
    Point(int x,int y){
        this.x=x;
```

```
    // 0 --> p, q and r are collinear
    // 1 --> Clockwise
    // 2 --> Counterclockwise
    public static int orientation(Point p1, Point p2,
                                  Point p3)
    {
        // See 10th slides from following link
        // for derivation of the formula
        int val = (p2.y - p1.y) * (p3.x - p2.x) -
                  (p2.x - p1.x) * (p3.y - p2.y);

        if (val == 0) return 0;  // collinear

        // clock or counterclock wise
        return (val > 0)? 1: 2;
    }

    /* Driver program to test above function */
    public static void main(String[] args)
    {
            Point p1 = new Point(0, 0);
            Point p2 = new Point(4, 4);
            Point p3 = new Point(1, 2);

            int o = orientation(p1, p2, p3);

            if (o==0)
            System.out.print("Linear");
            else if (o == 1)
            System.out.print("Clockwise");
            else
            System.out.print("CounterClockwise");

    }
}

//This code is contributed by Arnav Kr. Mandal.
```

Python3

```
# A Python3 program to find orientation of 3 points
```

```
    # to find the orientation of
    # an ordered triplet (p1,p2,p3)
    # function returns the following values:
    # 0 : Collinear points
    # 1 : Clockwise points
    # 2 : Counterclockwise
    val = (float(p2.y - p1.y) * (p3.x - p2.x)) - \
           (float(p2.x - p1.x) * (p3.y - p2.y))
    if (val > 0):

        # Clockwise orientation
        return 1
    elif (val < 0):

        # Counterclockwise orientation
        return 2
    else:

        # Collinear orientation
        return 0

# Driver code
p1 = Point(0, 0)
p2 = Point(4, 4)
p3 = Point(1, 2)

o = orientation(p1, p2, p3)

if (o == 0):
    print("Linear")
elif (o == 1):
    print("Clockwise")
else:
    print("CounterClockwise")

# This code is contributed by Ansh Riyal
```

C#

```
// C# Code to find Orientation of 3
```

```
        this.y = y;
    }
}

class GFG
{

    // To find orientation of ordered triplet
    // (p1, p2, p3). The function returns
    // following values
    // 0 --> p, q and r are collinear
    // 1 --> Clockwise
    // 2 --> Counterclockwise
    public static int orientation(Point p1, Point p2,
                                  Point p3)
    {
        // See 10th slides from following link
        // for derivation of the formula
        int val = (p2.y - p1.y) * (p3.x - p2.x) -
                (p2.x - p1.x) * (p3.y - p2.y);

        if (val == 0) return 0; // collinear

        // clock or counterclock wise
        return (val > 0)? 1: 2;
    }

    /* Driver program to test above function */<strong>
    public static void Main(String[] args)
    {
            Point p1 = new Point(0, 0);
            Point p2 = new Point(4, 4);
            Point p3 = new Point(1, 2);

            int o = orientation(p1, p2, p3);

            if (o == 0)
                Console.WriteLine("Linear");
            else if (o == 1)
                Console.WriteLine("Clockwise");
            else
                Console.WriteLine("CounterClockwise");
```

```
CounterClockwise
```

The concept of orientation is used in below articles:

- Find Simple Closed Path for a given set of points
- How to check if two given line segments intersect?
- Convex Hull | Set 1 (Jarvis's Algorithm or Wrapping)
- Convex Hull | Set 2 (Graham Scan)

Source: http://www.dcs.gla.ac.uk/~pat/52233/slides/Geometry1x1.pdf
This article is contributed by **Rajeev Agrawal**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

Previous Next

RECOMMENDED ARTICLES

01 **Number of ordered points pair satisfying line equation**
12, Apr 18

02 **Number of Integral Points between Two Points**
24, Feb 16

03 **Count of obtuse angles in a circle with 'k' equidistant points between 2 given points**
31, Aug 17

04 **Minimum number of points to be removed to get remaining points on one side of axis**
15, Dec 17

05 **Ways to choose three points with distance between the most distant points <= L**
31, May 18

06 **Find the point on X-axis from given N points having least Sum of Distances from all other points**
12, Aug 20

07 **Area of a polygon with given n ordered vertices**
16, Mar 16

08 **Closest Pair of Points using Divide and Conquer algorithm**
28, Nov 12

Article Contributed By :

GeeksforGeeks

Vote for difficulty

Current difficulty : Easy

Easy | Normal | Medium | Hard | Expert

Improve Article | Report Issue

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

GeeksforGeeks
5th Floor, A-118,
Sector-136, Noida, Uttar Pradesh - 201305

feedback@geeksforgeeks.org