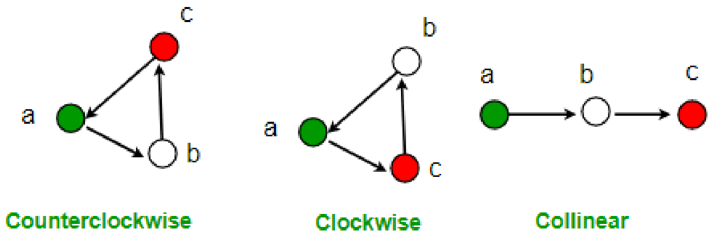
Orientation of 3 ordered points

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

- counterclockwise
- clockwise
- colinear

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



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

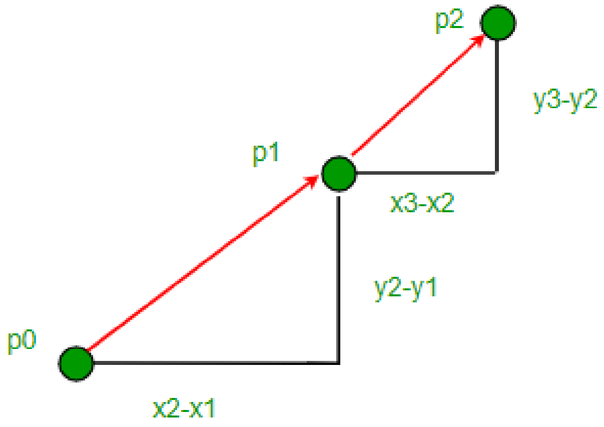
Given three points p1, p2 and p3, find orientation of (p1, p2, p3).
Example:

```
Input:   p1 = {0, 0}, p2 = {4, 4}, p3 = {1, 2}
Output:  CounterClockWise

Input:   p1 = {0, 0}, p2 = {4, 4}, p3 = {1, 1}
Output:  Colinear
```

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 counterclockwise (left turn)
If  σ = τ, the orientation is collinear
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
Above expression is 0 when σ = τ, i.e., collinear
Above expression is positive when σ > τ, i.e., clockwise
```

Below is the implementation of above idea.

C++

```
// A C++ program to find orientation of three points
#include <iostream>
using namespace std;

struct Point
{
    int x, y;
};

// To find orientation of ordered triplet (p1, p2, p3).
// The function returns following values
// 0 --> p, q and r are colinear
// 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;  // colinear

    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;
        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 colinear
    // 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;  // colinear

        // 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.
```

Output:

```
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

Recommended Posts:

Closest Pair of Points using Divide and Conquer algorithm

Closest Pair of Points | O(nlogn) Implementation

Find Simple Closed Path for a given set of points

Count Integral points inside a Triangle

Number of Integral Points between Two Points

Area of a polygon with given n ordered vertices

Non-crossing lines to connect points in a circle

Count maximum points on same line

Circle and Lattice Points

Queries on count of points lie inside a circle

Minimum lines to cover all points

How to check if given four points form a square

Dynamic Convex hull | Adding Points to an Existing Convex Hull

Angular Sweep (Maximum points that can be enclosed in a circle of given radius)

Deleting points from Convex Hull

Article Tags : Geometric

Practice Tags : Geometric

👍

1

□ To-do □ Done

2.1

Based on **41** vote(s)

Feedback Notes Improve Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

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

Load Comments

Share this post!

A computer science portal for geeks

710-B, Advant Navis Business Park,
Sector-142, Noida, Uttar Pradesh - 201305
feedback@geeksforgeeks.org