Activity 4.1

José Carlos Martínez Núñez, Tania Sayuri Guizado Hernández

November 4, 2022

Contents

Problem Description

In teams of two and using the computational geometry programming technique, build a C++ program that implements the algorithm to determine if two line segments intersect.

The output of the program will be a boolean array indicated with True if the pair of line segments (4 points per line) intersect, False otherwise.

Algorithm Implementation

The algorithm to determine if two line segments intersect is based on the following steps:

1. Determine the orientation of the points of the first line segment with respect to the second line segment.
2. Determine the orientation of the points of the second line segment with respect to the first line segment.
3. If the orientations of the points of the first line segment are different and the orientations of the points of the second line segment are different, then the line segments intersect.
4. If the orientations of the points of the first line segment are the same and the orientations of the points of the second line segment are the same, and the line segments are not overlapping, then the line segments do not intersect.

We create the following `LineSegment` class to represent a line segment:

```
class LineSegment
{
public:
    LineSegment(Point p1, Point p2)
    {
        this->p1 = p1;
        this->p2 = p2;
    }

    Point p1;
    Point p2;

    Orientation getOrientation(Point p)
    {
        int val = (p2.y - p1.y) * (p.x - p2.x) - (p2.x - p1.x) * (p.y - p2.y);
```

```
    if (val == 0)
        return COLINEAR;
    return (val > 0) ? CLOCKWISE : COUNTERCLOCKWISE;
}

bool onSegment(Point p)
{
    if (p.x <= max(p1.x, p2.x) && p.x >= min(p1.x, p2.x) && p.y <= max(p1.y, p2.y) && p.y >=
    ↪ min(p1.y, p2.y))
        return true;
    return false;
}

bool doesItIntersect(LineSegment l)
{
    Orientation o1 = getOrientation(l.p1);
    Orientation o2 = getOrientation(l.p2);
    Orientation o3 = l.getOrientation(p1);
    Orientation o4 = l.getOrientation(p2);

    if (o1 != o2 && o3 != o4)
        return true;

    if (o1 == COLINEAR && onSegment(l.p1))
        return true;

    if (o2 == COLINEAR && onSegment(l.p2))
        return true;

    if (o3 == COLINEAR && l.onSegment(p1))
        return true;

    if (o4 == COLINEAR && l.onSegment(p2))
        return true;

    return false;
}
```

```
};
```

The time complexity of the algorithm is $O(1)$.

Test Cases

The following test cases are used to test the program:

Test Case 1

```
4
0 0 10 10 2 2 12 12
```

This test case has two line segments that intersect.

Test Case 2

```
8
1 1 10 1 1 2 10 2
10 0 0 10 0 0 10 10
```

This test case has four line segments. The first two line segments **don't** intersect, and the last two line segments intersect.

Test Case 3

```
4
-5 -5 0 0 1 1 10 10
```

This test case has two line segments that **don't** intersect.

Test Case 4

```
8
0 0 5 5 2 -10 3 10
1 5 5 8 2 1 3 10
```

This test case has four line segments. The first two line segments intersect, and the last two line segments also intersect.

The results of each test case are located in the `results` folder.