Computational Geometry

Introduction

- Computational geometry is the branch of computer science that studies algorithms for solving geometric problems.
- **Applications:**
  - Computer Graphics
  - Robotics
  - VLSI Design
  - Computer Aided Design
  - Molecular Modeling
  - Textile Layouts
  - Statistics

Introduction

- We shall look at few computational geometry algorithms in two dimensions, i.e. in a plane.
- We represent each input object by a set of points $\{p_1, p_2, p_3, \ldots\}$, where each $p_i = (x_i, y_i)$ and $x_i, y_i \in R$.
- For example, we represent an *n*-vertex polygon P by a sequence $\{p_0, p_1, p_2, \ldots, p_{n-1}\}$ of its vertices in order of their appearance on the boundary of P.
- Computational geometry can also apply to 3-dimensions and even higher dimensional spaces, but these problems and solutions are difficult to visualize.

Line Segment Properties

- **Convex combinations:** A convex combination of two distinct points $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$ is any point $p_3 = (x_3, y_3)$ such that for some α in the range $0 \leq \alpha \leq 1$, we have $x_3 = \alpha x_1 + (1 - \alpha)x_2$ and $y_3 = \alpha y_1 + (1 - \alpha)y_2$. We can also write this as $p_3 = \alpha p_1 + (1 - \alpha)p_2$. Thus p_3 is any point that is on the line passing through p_1 and p_2 and is on or between p_1 and p_2 on the line.
- **Line segment:** Given two distinct points p_1 and p_2, the line segment $\overline{p_1p_2}$ is the set of convex combinations of p_1 and p_2. We call p_1 as left endpoint and p_2 as right endpoints of segment $\overline{p_1p_2}$.
- **Directed segment:** Directed segment $\overrightarrow{p_1p_2}$ is referred as the vector p_2 with p_1 as the origin (0, 0).

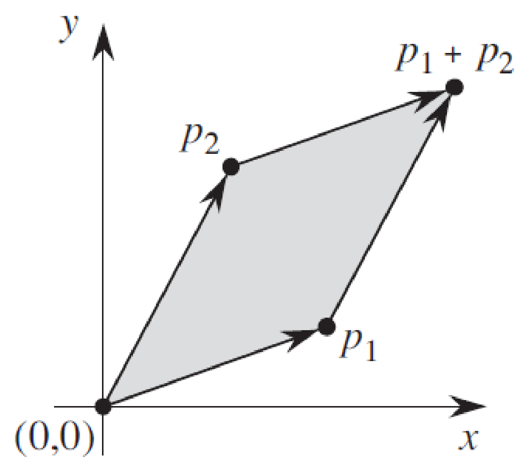
Line Segment Properties

- Computational geometry algorithms requires answers to questions about the properties of line segments. These questions are:
  - Given two directed segments $\overrightarrow{p_0p_1}$ and $\overrightarrow{p_0p_2}$, is $\overrightarrow{p_0p_1}$ clockwise from $\overrightarrow{p_0p_2}$, with respect to their common endpoint p_0?
  - Given two line segments $\overline{p_0p_1}$ and $\overline{p_1p_2}$, if we traverse $\overline{p_0p_1}$, and then $\overline{p_1p_2}$, do we make a left turn at point p_1?
  - Do line segments $\overline{p_1p_2}$ and $\overline{p_3p_4}$ intersect?
- To answer these questions the method which avoids division or trigonometric function is more accurate and that is use of **cross products**.

Line Segment Properties

- **Cross Product:**

  Consider vectors p1 and p2 as shown below



- We can interpret the ***cross product*** $p_1 x p_2$ as the signed area of the parallelogram formed by the points (0, 0), p_1, p_2, and (p_1+p_2).

Line Segment Properties

- **Cross Product:**

  An equivalent, but more useful, definition gives the cross product as the determinant of a matrix:

  $$p_1 \text{ x } p_2 = \det \begin{pmatrix} x_1 & x_2 \\ y_1 & y_2 \end{pmatrix} = x_1y_2 - x_2y_1 = -p_2 \text{ x } p_1$$

- If p_1 x p_2 is positive then p_1 is clockwise from p_2 with respect to origin (0,0) and if p_1 x p_2 is negative then p_1 is counterclockwise from p_2 with respect to origin (0,0).
- If cross product is zero then vectors p_1 and p_2 are collinear pointing in either the same or opposite direction.

Line Segment Properties

- **Determining whether directed segment p_0p_1 is clockwise from p_0p_2:**

  To determine whether a directed segment p_0p_1 is closer to a directed segment p_0p_2 in a clockwise or counterclockwise direction w.r.t. common endpoint p_0, we simply translate to use p_0 as the origin and compute cross product of p_1' and p_2'.

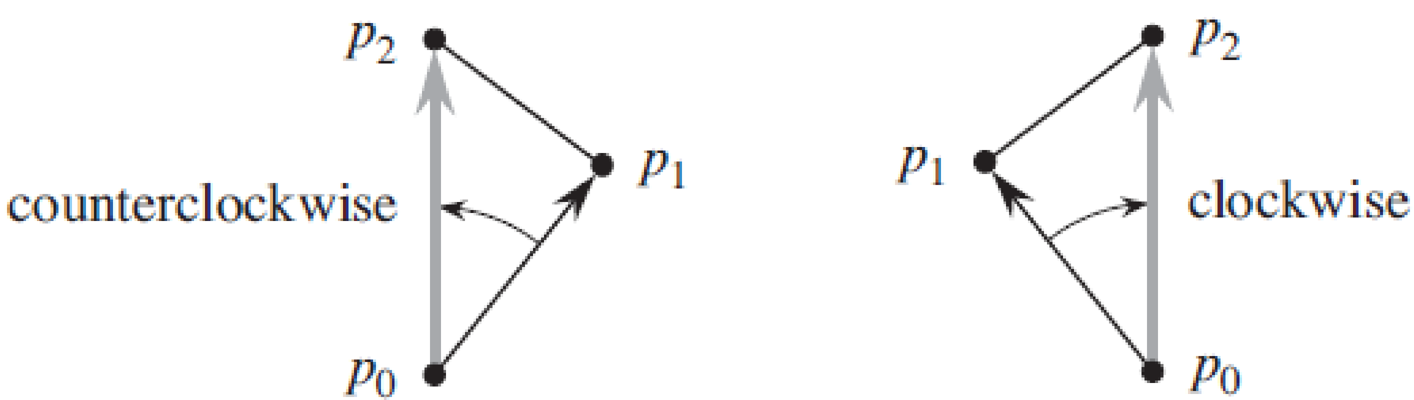
  $p_1' \times p_2' = (p_1-p_0) \times (p_2-p_0)$

  $$= \det\begin{pmatrix} x_1 - x_0 & x_2 - x_0 \\ y_1 - y_0 & y_2 - y_0 \end{pmatrix}$$

- If cross product is positive then p_1' is clockwise from p_2' & if cross product is negative then p_1' is counterclockwise from p_2'.

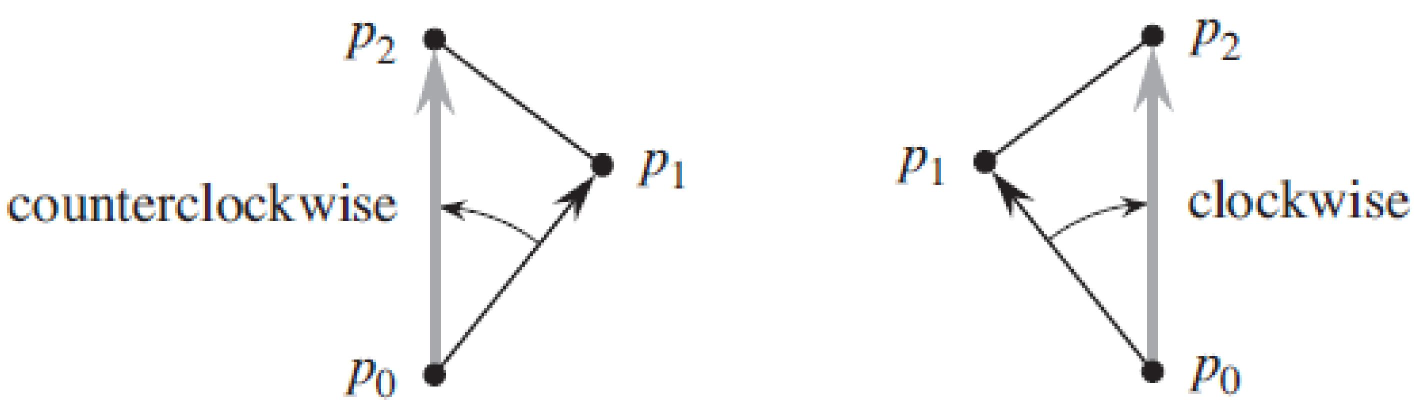
Line Segment Properties

- **Determining whether consecutive segments turn left or right:** To determine whether consecutive line segment p_0p_1 and p_1p_2 turn left or right at point p_1
- In other words we want a method to determine which way a given angle $p_0p_1p_2$ turns.
- Cross product allow us to answer this question too, without computing the angle.



Line Segment Properties

- Check whether directed segment p_0p_2 is clockwise or counterclockwise relative to directed segment p_0p_1.
- A positive cross product $(p_2 – p_0) \times (p_1 – p_0)$ indicates p_0p_2 is clockwise w.r.t. p_0p_1 and it makes a right turn at p .



Line Segment Properties

- **Determining whether two line segments intersect:**

  We check whether each segment straddles the line containing the other. A line segment p_1p_2 straddles a line if point p_1 lies on one side of the line and point p_2 lies on the other side. A boundary case arises if p_1 and p_2 lies directly on the line.

- Thus two line segments intersect if and only if either or both of the following conditions hold.

  1. Each segment straddles the line containing other.
  2. An endpoint of one lies on the other segment (Boundary case).

Line Segment Properties

- Following procedures implement this idea:

1. **DIRECTION :** computes relative orientations
2. **ON-SEGMENT :** determines whether a point known to be collinear with a segment lies on that segment.
3. **SEGMENT-INTERSECT :** returns TRUE if segments p_1p_2 and p_3p_4 intersect, otherwise returns FALSE

DIRECTION(p_i, p_j, p_k)

return $(p_k - p_i) \times (p_j - p_i)$

ON-SEGMENT(p_i, p_j, p_k)

if $\min(x_i, x_j) \le x_k \le \max(x_i, x_j)$ AND $\min(y_i, y_j) \le y_k \le \max(y_i, y_j)$

return TRUE

else return FALSE

Line Segment Properties

SEGMENT-INTERSECT (p_1, p_2, p_3 , p_4){

// algorithm returns TRUE if p_1p_2 and p_3p_4 intersect,

d_1 = DIRECTION (p_3, p_4, p_1) // if -ve, p_1 is left to p_3p_4

d_2 = DIRECTION (p_3, p_4, p_2) // if +ve, p_2 is right to p_3p_4

d_3 = DIRECTION (p_1, p_2, p_3) // if -ve, p_3 is left to p_1p_2

d_4 = DIRECTION (p_1, p_2, p_4) // if +ve, p_4 is right to p_1p_2

if (($d_1 > 0$ AND $d_2 < 0$) OR ($d_1 < 0$ AND $d_2 > 0$)) AND (($d_3 > 0$ AND $d_4 < 0$) OR ($d_3 < 0$ AND $d_4 > 0$))

return TRUE

elseif $d_1 = 0$ AND ON-SEGMENT(p_3, p_4, p_1) return TRUE;

elseif $d_2 = 0$ AND ON-SEGMENT(p_3, p_4, p_2) return TRUE;

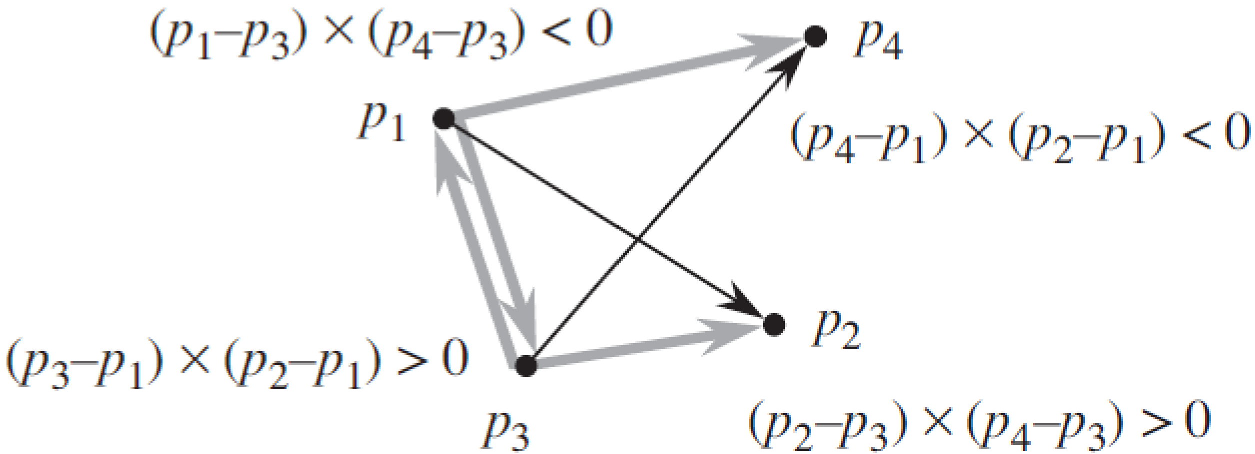
elseif $d_3 = 0$ AND ON-SEGMENT(p_1, p_2, p_3) return TRUE;

elseif $d_4 = 0$ AND ON-SEGMENT(p_1, p_2, p_4) return TRUE;

else return FALSE;

}

Line Segment Properties



- $d_1 = (p_1 - p_3) \times (p_4 - p_3)$, DIRECTION (p_3, p_4, p_1)
- $d_2 = (p_2 - p_3) \times (p_4 - p_3)$, DIRECTION (p_3, p_4, p_2)
- $d_3 = (p_3 - p_1) \times (p_2 - p_1)$, DIRECTION (p_1, p_2, p_3)
- $d_4 = (p_4 - p_1) \times (p_2 - p_1)$, DIRECTION (p_1, p_2, p_4)

Determining Whether Any Pair of Segments Intersect

- Algorithm uses a technique known as "sweeping", which is common to many computational geometry algorithms.
- It determines only whether or not any intersection exists, it does not print all the intersections.

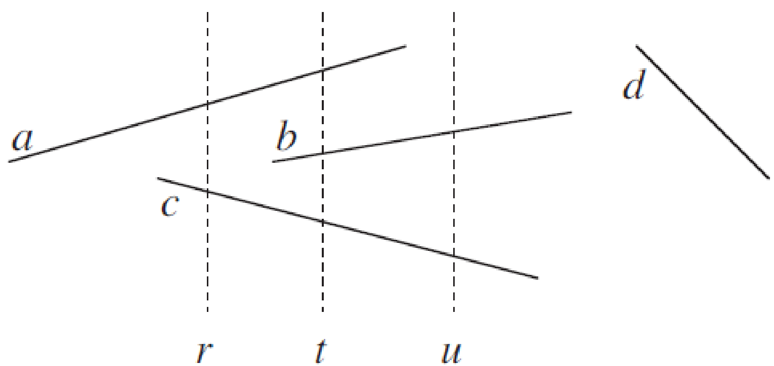
Sweeping : In sweeping an imaginary vertical sweep line that passes through the given set of geometric objects, usually from left to right.

- The line-segment-intersection algorithm considers all the line segment endpoints in left to right order and checks for an intersection each time it encounters an endpoint.

Determining Whether Any Pair of Segments Intersect

- The algorithm makes to simplify assumptions:
  - No input segment is vertical and
  - No three or more input segments intersect at a single point.
- Consider two segments s_1 and s_2. We say that these segments are comparable at x if the vertical sweep line with x-coordinate x intersects both of them.
- We say that s_1 is above s_2 at x, written as $s_1 \geq_x s_2$, if s_1 and s_2 are comparable at x, and the intersection of s_1 with the sweep line at x is higher than the intersection of s_2 with the same sweep line or if s_1 and s_2 intersect at the same point.

Determining Whether Any Pair of Segments Intersect



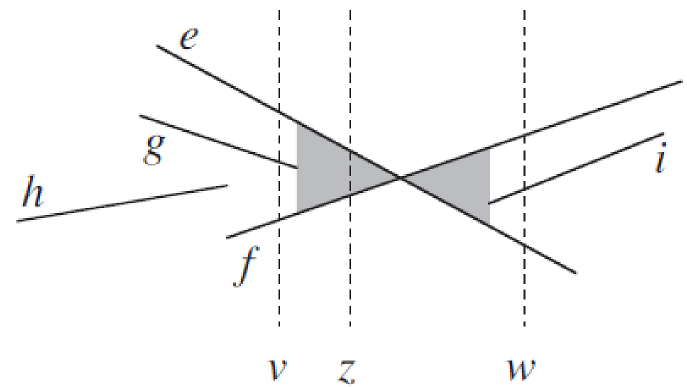


Figure 1: $a \geq_r c$,

$a \geq_t b$, $b \geq_t c$, $a \geq_t c$

Figure 2: $e \geq_v f$

$f \geq_w e$

Determining Whether Any Pair of Segments Intersect

Moving the sweep line :

- Thus sweep line status is a total preorder T (BST) for which we require the following operations:
  1. **INSERT (T, s) :** Insert segment s into T
  2. **DELETE (T, s) :** Deletes segment s from T
  3. **ABOVE (T, s) :** returns the segment immediately above segment s in T.
  4. **BELOW (T, s) :** returns the segment immediately below segment s in T.
- It is possible for segments s_1 and s_2 to be mutually above each other in the total preorder T; this situation can occur if s_1 and s_2 intersect at the sweep line (boundary case). In this case the two segments may occur in either order in T.

Determining Whether Any Pair of Segments Intersect

```
ANY-SEGMENTS-INTERSECT(S)
T = Ø
sort the endpoints of the segments in S from left to right,
        breaking ties by putting left endpoints before right endpoints
        and breaking further ties by putting points with lower
        y-coordinates first
for each point p in the sorted list of endpoints
        if p is the left endpoint of a segment s
                INSERT(T, s)
                if (ABOVE(T, s) exists and intersects s)
                        or (BELOW(T, s) exists and intersects s)
                        return TRUE
        if p is the right endpoint of a segment s
                if both ABOVE(T, s) and BELOW(T, s) exist
                        and ABOVE(T, s) intersects BELOW(T, s)
                        return TRUE
                DELETE(T, s)
return FALSE
```

Determining Whether Any Pair of Segments Intersect