Today’s Outline

• Computational Geometry
  – Introduction
  – Line Segments and their properties
  – Sweeping (intersecting line segments)
  – Computing the Convex Hull
    • Graham’s Scan
    • Jarvis’s March
Introduction

• Computational Geometry
  – Algorithms to solve geometric problems
  – Wide range of application areas
    • Computer graphics, VLSI, CAD, molecular modeling, metallurgy, textile layout, GIS,…
  – Inputs
    • E.g., points, line segments, vertices of a polygon,..
  – Outputs
    • Whether lines intersect, convex hull,…
Introduction

- We consider problems in 2D plane
- Objects represented by points \{p_1, p_2, p_3, \ldots\}
- Each \( p_i = (x_i, y_i) \) where \( x_i, y_i \) are real nums
• Say $p_1 = (x_1, y_1)$, $p_2 = (x_2, y_2)$ are 2 points
• $p_3 = (x_3, y_3)$ is a convex combination of $p_1$, $p_2$ such that for some $0 \leq \alpha \leq 1$

$$x_3 = \alpha x_1 + (1-\alpha) x_2$$
$$y_3 = \alpha y_1 + (1-\alpha) y_2$$

• Intuitively
  – $p_3$ is any point on the line passing through $p_1$ and $p_2$; $p_3$ lies on or between $p_1$ and $p_2$
Terminology, Notations

• Line segment $p_1p_2$ is the set of convex combinations of $p_1$, $p_2$
• $p_1$, $p_2$ are the endpoints of segment $p_1p_2$
• If ordering of $p_1$, $p_2$ matters
  – We can have directed segment $p_1p_2$
  – If $p_1=(0,0)$, origin, then $p_1p_2$ is the vector $p_2$
Line Segment Properties

• Questions to explore

Q1 – Given 2 directed segments \( p_0p_1 \) and \( p_0p_2 \), is \( p_0p_1 \) clockwise from \( p_0p_2 \) w.r.t \( p_0 \)?

Q2 – Given 2 line segments \( p_0p_1 \) and \( p_1p_2 \), if we traverse \( p_0p_1 \) and then \( p_1p_2 \), do we make a left turn at point \( p_1 \)?

Q3 – Do line segments \( p_1p_2 \) and \( p_3p_4 \) intersect?
Line Segment Properties

• Can answer each question in $O(1)$ time
  – Using only add, subtract, multiply, compare
  – No need of division, trigonometric functions
    • Which are computationally expensive, have problems due to round-off errors
    • [think of the straight-forward method to determine whether two line segments intersect; this require division; very sensitive to precision of division on computers when 2 segments are nearly parallel]
Cross Product

- Cross product is a core operation we need.
- Consider vectors \( p_1 \) and \( p_2 \).
- The cross product as a determinant:

\[
p_1 \times p_2 = \det \begin{pmatrix} x_1 & x_2 \\ y_1 & y_2 \end{pmatrix} = x_1 y_2 - x_2 y_1 = -p_2 \times p_1
\]

- [Strictly speaking cross product is a 3D concept; but we will treat it simply as the value \( x_1 y_2 - x_2 y_1 \).]
If \( p_1 = (x_1, y_1) \) and \( p_2 = (x_2, y_2) \) then

\[
p_1 \times p_2 = \begin{vmatrix} x_1 & x_2 \\ y_1 & y_2 \end{vmatrix} = x_1y_2 - x_2y_1
\]

If \( p_1 \times p_2 > 0 \), then \( p_1 \) is clockwise from \( p_2 \) (w.r.t. the origin);
if \( p_1 \times p_2 < 0 \), then \( p_1 \) is counterclockwise from \( p_2 \);
If \( p_1 \times p_2 = 0 \), then \( p_1 \) and \( p_2 \) are \textbf{colinear} (pointing in either same or opposite directions)
Q1: Segment Clockwise?

• Can answer Q1 using the cross product
  – Given 2 directed segments \( \overrightarrow{p_0p_1} \) and \( \overrightarrow{p_0p_2} \), is \( \overrightarrow{p_0p_1} \) clockwise from \( \overrightarrow{p_0p_2} \) w.r.t \( p_0 \)?
  – Translate cross product to use \( p_0 \) as the origin
  – Compute the cross product
    \[
    (p_1 - p_0) \times (p_2 - p_0) = (x_1 - x_0)(y_2 - y_0) - (x_2 - x_0)(y_1 - y_0)
    \]
  – If it is positive, then \( \overrightarrow{p_0p_1} \) is clockwise from \( \overrightarrow{p_0p_2} \)
    else it is counter-clockwise
Q2: Turning Left?

• Given 2 line segments \( p_0p_1 \) and \( p_1p_2 \), if we traverse \( p_0p_1 \) and then \( p_1p_2 \), do we make a left turn at point \( p_1 \)?
  
  • Equivalent: which way the angle \( p_0p_1p_2 \) turns?
  
  – With cross products, can answer without computing the angle
  
  – See next slide
Q2: Turning Left?

![Diagram](image)

**Figure 33.2** Using the cross product to determine how consecutive line segments $\overline{p_0p_1}$ and $\overline{p_1p_2}$ turn at point $p_1$. We check whether the directed segment $\overline{p_0p_2}$ is clockwise or counterclockwise relative to the directed segment $\overline{p_0p_1}$. (a) If counterclockwise, the points make a left turn. (b) If clockwise, they make a right turn.
Q3: Segment Intersection

• Do line segments \( p_1p_2 \) and \( p_3p_4 \) intersect?
• Definition
  – A segment \( p_1p_2 \) straddles a line \( L \) if point \( p_1 \) lies on one side of \( L \) and point \( p_2 \) lies on the other side
  – Boundary case: \( p_1 \) or \( p_2 \) lies directly on \( L \)
Q3: Segment Intersection

- Two line segments intersect iff either or both conditions below hold
  - Each segment straddles the line containing the other
  - An endpoint of one segment lies on the other segment (boundary case)
**Segments-Intersect** $(p_1, p_2, p_3, p_4)$

1. $d_1 \leftarrow \text{Direction}(p_3, p_4, p_1)$
2. $d_2 \leftarrow \text{Direction}(p_3, p_4, p_2)$
3. $d_3 \leftarrow \text{Direction}(p_1, p_2, p_3)$
4. $d_4 \leftarrow \text{Direction}(p_1, p_2, p_4)$
5. \textbf{if} $((d_1 > 0 \text{ and } d_2 < 0) \text{ or } (d_1 < 0 \text{ and } d_2 > 0))$ and $((d_3 > 0 \text{ and } d_4 < 0) \text{ or } (d_3 < 0 \text{ and } d_4 > 0))$
   \hspace{1cm} \textbf{then return } \text{TRUE}$
7. \textbf{elseif} $d_1 = 0$ and $\text{ON-Segment}(p_3, p_4, p_1)$
   \hspace{1cm} \textbf{then return } \text{TRUE}$
8. \textbf{elseif} $d_2 = 0$ and $\text{ON-Segment}(p_3, p_4, p_2)$
   \hspace{1cm} \textbf{then return } \text{TRUE}$
9. \textbf{elseif} $d_3 = 0$ and $\text{ON-Segment}(p_1, p_2, p_3)$
   \hspace{1cm} \textbf{then return } \text{TRUE}$
10. \textbf{elseif} $d_4 = 0$ and $\text{ON-Segment}(p_1, p_2, p_4)$
    \hspace{1cm} \textbf{then return } \text{TRUE}$
11. \textbf{else return } \text{FALSE}
In lines 1-4 of the main algorithm (previous slide), relative orientation $d_i$ of each endpoint $p_i$ with respect to the other line segment is computed.
Figure 33.3  Cases in the procedure SEGMENTS-INTERSECT.  (a) The segments $\overline{p_1p_2}$ and $\overline{p_3p_4}$ straddle each other's lines. Because $\overline{p_3p_4}$ straddles the line containing $\overline{p_1p_2}$, the signs of the cross products $(p_3 - p_1) \times (p_2 - p_1)$ and $(p_4 - p_1) \times (p_2 - p_1)$ differ. Because $\overline{p_1p_2}$ straddles the line containing $\overline{p_3p_4}$, the signs of the cross products $(p_1 - p_3) \times (p_4 - p_3)$ and $(p_2 - p_3) \times (p_4 - p_3)$ differ. (b) Segment $\overline{p_3p_4}$ straddles the line containing $\overline{p_1p_2}$, but $\overline{p_1p_2}$ does not straddle the line containing $\overline{p_3p_4}$. The signs of the cross products $(p_1 - p_3) \times (p_4 - p_3)$ and $(p_2 - p_3) \times (p_4 - p_3)$ are the same. (c) Point $p_3$ is collinear with $\overline{p_1p_2}$ and is between $p_1$ and $p_2$. (d) Point $p_3$ is collinear with $\overline{p_1p_2}$, but it is not between $p_1$ and $p_2$. The segments do not intersect.
Intersecting Segment Pairs

• Given a set of line segments, is there a pair that intersects? (Does any exist?)
• Uses “sweeping” (left-to-right) technique
  – Complexity $O(n \lg n)$, $n =$ # of segments
• Assumptions
  – No segment is vertical
  – No three segments intersect at a single point
• Can order segments (“total pre-order”)
Figure 33.4 The ordering among line segments at various vertical sweep lines. (a) We have \(a >_r c\), \(a >_t b\), \(b >_t c\), \(a >_t c\), and \(b >_u c\). Segment \(d\) is comparable with no other segment shown. (b) When segments \(e\) and \(f\) intersect, their orders are reversed: we have \(e >_v f\) but \(f >_w e\). Any sweep line (such as \(z\)) that passes through the shaded region has \(e\) and \(f\) consecutive in its total order.
Moving the Sweep Line

- Sweeping algorithms manage 2 data sets
  - **Sweep-line status**
    - Gives the relationship among objects that the sweep line intersects; status changes occur only at event points
  - **Event-point schedule**
    - A sequence of points (event points), ordered L-to-R based on x-coordinates. As sweep progresses L→R, when sweep line reaches the x-coordinate of an event point, sweep halts; event point processed; then resumes.

In our algorithm, all event points determined before sweep.
ANY-SEGMENTS-INTERSECT($S$)

1. $T \leftarrow \emptyset$
2. 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
3. for each point $p$ in the sorted list of endpoints 
   do if $p$ is the left endpoint of a segment $s$
     then INSERT($T$, $s$)
6. if (ABOVE($T$, $s$) exists and intersects $s$) or (BELOW($T$, $s$) exists and intersects $s$) 
     then return TRUE
8. if $p$ is the right endpoint of a segment $s$
   then if both ABOVE($T$, $s$) and BELOW($T$, $s$) exist and ABOVE($T$, $s$) intersects BELOW($T$, $s$)
     then return TRUE
10. return FALSE
Figure 33.5  The execution of ANY-SEGMENTS-INTERSECT. Each dashed line is the sweep line at an event point, and the ordering of segment names below each sweep line is the total order $T$ at the end of the for loop in which the corresponding event point is processed. The intersection of segments $d$ and $b$ is found when segment $c$ is deleted.
Convex Hull

- **Convex Hull** \( CH(Q) \) of a set \( Q \) of points
  - The smallest convex polygon \( P \) for which each point in \( Q \) is either on the boundary of \( P \) or in its interior

- **Convex polygon**: given any 2 points on its boundary or in its interior, all points on the line segment between them are contained in the boundary or interior
Convex Hull

• Intuitively
  – A point in Q is a nail sticking out from a board
  – Convex hull is the shape of a tight rubber band that surrounds all the nails

• Assume: all points in Q are unique, Q has at least 3 points that are not colinear
Example

Figure 33.6  A set of points $Q = \{p_0, p_1, \ldots, p_{12}\}$ with its convex hull $\text{CH}(Q)$ in gray.
Finding the Convex Hull

• Two algorithms
  – Both outputs the vertices of the convex hull in counter-clockwise order, use rotational sweep (other methods exist)

• Graham’s Scan
  – Takes $O(n \lg n)$ time given $n$ points as input

• Jarvis’s March
  – Runs in $O(nh)$ time, $h$ is the # vertices in CH
Graham’s Scan

• Maintains a stack $S$ of candidate points
  – Each point is pushed to the stack once
  – Eventually pops points not in the $CH(Q)$
  – Finally, $S$ from bottom to top contains vertices in $CH(Q)$ in the counter-clockwise order

• $|Q| \geq 3$

• Requires functions
  – Top($S$), Next-to-Top($S$)
GRAHAM-SCAN(Q)

1 let $p_0$ be the point in $Q$ with the minimum $y$-coordinate, 
or the leftmost such point in case of a tie
2 let $\langle p_1, p_2, \ldots, p_m \rangle$ be the remaining points in $Q$, 
sorted by polar angle in counterclockwise order around $p_0$ 
(if more than one point has the same angle, remove all but 
the one that is farthest from $p_0$)
3 \textbf{PUSH}(p_0, S)
4 \textbf{PUSH}(p_1, S)
5 \textbf{PUSH}(p_2, S)
6 \textbf{for} i \leftarrow 3 \textbf{to} m
7 \quad \textbf{do while} the angle formed by points NEXT-TO-TOP(S), TOP(S), 
and $p_i$ makes a nonleft turn
8 \quad \quad \textbf{do} \quad \textbf{POP}(S)
9 \quad \quad \textbf{PUSH}(p_i, S)
10 \quad \textbf{return} S
Example (Fig 33.7 in CLRS)

Figure 33.7 The execution of GRAHAM-SCAN on the set $Q$ of Figure 33.6. The current convex hull contained in stack $S$ is shown in gray at each step. (a) The sequence $(p_1, p_2, \ldots, p_{12})$ of points numbered in order of increasing polar angle relative to $p_0$, and the initial stack $S$ containing $p_0$, $p_1$, and $p_2$. (b)–(k) Stack $S$ after each iteration of the for loop of lines 6–9. Dashed lines show nonleft turns, which cause points to be popped from the stack. In part (h), for example, the right turn at angle $\angle p_7p_8p_9$ causes $p_9$ to be popped, and then the right turn at angle $\angle p_6p_7p_9$ causes $p_7$ to be popped. (l) The convex hull returned by the procedure, which matches that of Figure 33.6.
Example

...contd
Figure 33.8 The proof of correctness of GRAHAM-SCAN. (a) Because $p_i$'s polar angle relative to $p_0$ is greater than $p_j$'s polar angle, and because the angle $\angle p_k p_j p_i$ makes a left turn, adding $p_i$ to $\text{CH}(Q_j)$ gives exactly the vertices of $\text{CH}(Q_j \cup \{p_i\})$. (b) If the angle $\angle p_r p_t p_i$ makes a nonleft turn, then $p_r$ is either in the interior of the triangle formed by $p_0$, $p_r$, and $p_i$ or on a side of the triangle, and it cannot be a vertex of $\text{CH}(Q_i)$. 
Javis’s March

- Uses package (gift) wrapping technique
  - Intuitively
    - Simulates a wrapping a taut piece of paper around the set Q, starting with the lowest point p0, continue around keeping it taut

- Runs in $O(nh)$ time
  - $h$ is the # vertices in $CH(Q)$
  - When $h$ is $O(lg n)$, Javis’s March is faster than Graham’s scan
Figure 33.9  The operation of Jarvis’s march. The first vertex chosen is the lowest point $p_0$. The next vertex, $p_1$, has the smallest polar angle of any point with respect to $p_0$. Then, $p_2$ has the smallest polar angle with respect to $p_1$. The right chain goes as high as the highest point $p_3$. Then, the left chain is constructed by finding smallest polar angles with respect to the negative $x$-axis.
References

• The lecture slides are based on the slides prepared by Prof. Sanath Jayasena for this class in previous years.

• Mainly: CLRS book, 3e
  – Part VII: Selected Topics
  – Chapter 33: Computational Geometry

• Slides from Alon Efrat

• Applet by Rashid Bin Muhammad
  • http://www.personal.kent.edu/~rmuhamma/Compgeometry/MyCG/ORourke/ORourkeCompGeom.htm
Conclusion

• We discussed a few problems and algorithms in computational geometry
  – Line segments, cross products
  – Line segment intersections
  – Convex hull
    • Graham’s Scan
    • Jarvis’s March

• Next time
  – Last lecture: Parallel/Multi-threaded algorithms