CSCI 4560/6560 Computational Geometry
https://www.cs.rpi.edu/~cutler/classes/computationalgeometry/F23/

Lecture 2: Line Segment Intersections
Outline for Today

- Questions about Homework 1?
- Questions about CGAL/Qt installation?
- Motivating Applications for Line Segment Intersection Problem
- Line/Segment Intersection Math
- Naive vs. Output Sensitive Algorithms
- A Plane/Line Sweep Algorithm
- Specific Choices for Data Structures
- Analysis
- Corner Cases / Degeneracies
- Next Time
Homework 1

- Questions?
- Installation Success/Failure?
Outline for Today

- Questions about Homework 1?
- Questions about CGAL/Qt installation?
- Motivating Applications for Line Segment Intersection Problem
- Line/Segment Intersection Math
- Naive vs. Output Sensitive Algorithms
- A Plane/Line Sweep Algorithm
- Specific Choices for Data Structures
- Analysis
- Corner Cases / Degeneracies
- Next Time
Motivating Application: Cartography Map Overlay

- 2 map layers storing the rivers & roads in NYS
- Each road/river stored as a *polyline* - sequence of *line segments*
- Find all intersections between a *road segment* and a *river segment*
- These are the bridges we need to build, inspect, repair, etc.

Application: Machine Learning

- Is my data classifiable? Is my data separable?

Figure 7.2 A two-variable classification problem.

Figure 7.3 Two sets are separable if and only if their convex hulls are disjoint.

Computational Geometry: An Introduction
Preparata & Shamos, Springer 1985
Self-Intersection of Non Convex Polygons

Figure 7.13 Simple and nonsimple polygons.
Hidden Line (Hidden Surface) Removal

- A classic problem from the early days of Computer Graphics
- Identify and remove portions of the object that are not visible from a particular viewing angle

Figure 7.1  Elimination of hidden lines.
Outline for Today

● Questions about Homework 1?
  Questions about CGAL/Qt installation?
● Motivating Applications for Line Segment Intersection Problem
● Line/Segment Intersection Math
● Naive vs. Output Sensitive Algorithms
● A Plane/Line Sweep Algorithm
● Specific Choices for Data Structures
● Analysis
● Corner Cases / Degeneracies
● Next Time
Intersection of 2 Lines in a Plane

https://en.wikipedia.org/wiki/Linear_equation
Intersection of 2 Lines in a Plane

- Using line slope equations:
  \[ y = ax + c \text{ and } y = bx + d \]
- Set them equal to each other:
  \[ ax + c = bx + d \]
- Solve for x and y:
  \[ x = \frac{d - c}{a - b} \]
  \[ y = a \frac{d - c}{a - b} + c \]
- Concerns?

https://en.wikipedia.org/wiki/Linear_equation
Intersection of 2 Lines in a Plane

- Using line slope equations:
  \[ y = ax + c \quad \text{and} \quad y = bx + d \]
- Set them equal to each other:
  \[ ax + c = bx + d \]
- Solve for x and y:
  \[ x = \frac{d - c}{a - b} \quad \text{and} \quad y = a \frac{d - c}{a - b} + c \]
- Concerns?
  - Does it handle vertical lines?
  - How do we detect parallel (non-intersecting) lines?
  - How do we determine if line segments intersect (between endpoints)?
Intersection of 2 Line Segments in a Plane

● Let’s use the *Parametric Equation* for a line segment:

\[
L_1 = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + t \begin{bmatrix} x_2 - x_1 \\ y_2 - y_1 \end{bmatrix}
\]

● For every value of \( t \) from in the interval \([0,1]\), plug \( t \) into this equation, and you’ll get a point on the line segment.

● Linearly interpolating between the endpoints.

● A weighted average of the endpoints.

\[P_0 = (x_1, y_1)\]

\[P_1 = (x_2, y_2)\]
Intersection of 2 Line Segments in a Plane

- Two parametric equations:
  \[ L_1 = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + t \begin{bmatrix} x_2 - x_1 \\ y_2 - y_1 \end{bmatrix}, \quad L_2 = \begin{bmatrix} x_3 \\ y_3 \end{bmatrix} + u \begin{bmatrix} x_4 - x_3 \\ y_4 - y_3 \end{bmatrix} \]

- Solve for \( t \) and \( u \):
  \[
  t = \frac{\begin{vmatrix} x_1 - x_3 & x_3 - x_4 \\ y_1 - y_3 & y_3 - y_4 \\ x_1 - x_2 & x_3 - x_4 \\ y_1 - y_2 & y_3 - y_4 \end{vmatrix}}{\begin{vmatrix} x_1 - x_2 & x_3 - x_4 \\ y_1 - y_2 & y_3 - y_4 \end{vmatrix}} = \frac{(x_1 - x_3)(y_3 - y_4) - (y_1 - y_3)(x_3 - x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)}
  
  u = \frac{\begin{vmatrix} x_1 - x_3 & x_1 - x_2 \\ y_1 - y_3 & y_1 - y_2 \\ x_1 - x_2 & x_3 - x_4 \\ y_1 - y_2 & y_3 - y_4 \end{vmatrix}}{\begin{vmatrix} x_1 - x_2 & x_3 - x_4 \\ y_1 - y_2 & y_3 - y_4 \end{vmatrix}} = \frac{(x_1 - x_3)(y_1 - y_2) - (y_1 - y_3)(x_1 - x_2)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)}
  
- Concerns?
  - Vertical lines?
  - Parallel lines?
  - Line vs. segment intersection?

https://en.wikipedia.org/wiki/Line%E2%80%93line_intersection
Intersection of 2 Line Segments in a Plane

- Two parametric equations:
  
  \[ L_1 = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + t \begin{bmatrix} x_2 - x_1 \\ y_2 - y_1 \end{bmatrix}, \quad L_2 = \begin{bmatrix} x_3 \\ y_3 \end{bmatrix} + u \begin{bmatrix} x_4 - x_3 \\ y_4 - y_3 \end{bmatrix} \]

- Solve for \( t \) and \( u \):
  
  \[
t = \frac{\begin{vmatrix} x_1 - x_3 & x_3 - x_4 \\ y_1 - y_3 & y_3 - y_4 \end{vmatrix}}{\begin{vmatrix} x_1 - x_2 & x_3 - x_4 \\ y_1 - y_2 & y_3 - y_4 \end{vmatrix}} = \frac{(x_1 - x_3)(y_3 - y_4) - (y_1 - y_3)(x_3 - x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)}
  
  \[
u = \frac{\begin{vmatrix} x_1 - x_3 & x_1 - x_2 \\ y_1 - y_3 & y_1 - y_2 \end{vmatrix}}{\begin{vmatrix} x_1 - x_2 & x_3 - x_4 \\ y_1 - y_2 & y_3 - y_4 \end{vmatrix}} = \frac{(x_1 - x_3)(y_1 - y_2) - (y_1 - y_3)(x_1 - x_2)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)}
  
- Concerns?
  - Vertical lines?
  - Parallel lines?
  - Line vs. segment intersection?

\[ (P_x, P_y) = (x_1 + t(x_2 - x_1), \ y_1 + t(y_2 - y_1)) \]

\[ 0.0 \leq t \leq 1.0 \text{ and } 0.0 \leq u \leq 1.0 \]

https://en.wikipedia.org/wiki/Line%E2%80%93line_intersection
Outline for Today

- Questions about Homework 1?
- Questions about CGAL/Qt installation?
- Motivating Applications for Line Segment Intersection Problem
- Line/Segment Intersection Math
- Naive vs. Output Sensitive Algorithms
- A Plane/Line Sweep Algorithm
- Specific Choices for Data Structures
- Analysis
- Corner Cases / Degeneracies
- Next Time
Line Segment Intersection - Brute Force Solution

- Ignore labeling of road vs. river (just compare everything)

- Analysis?
Line Segment Intersection - Brute Force Solution

- Ignore labeling of road vs. river (just compare everything)
  - Nested for loop: Intersect each segment with every other segment

- Analysis? $O(n^2)$

- Can we do better?
Definition: Output Sensitive

- When algorithm running time depends on the size of the output for that specific input.

- The Convex Hull Algorithms from last:
  - \( n = \# \text{ of input points} \)
  - \( h = \# \text{ of points on final convex hull} \)
  - Naive: \( O(n^3) \)
  - Compute Upper Hull: \( O(n \log n) \)
  - Gift Wrapping: \( O(n \times h) \) ← output sensitive!

- … there are also \( O(n \log h) \) convex hull algorithms!

https://medium.com/@harshitsikchi/convex-hulls-explained-baab662c4e94
Output Sensitive Line Segment Intersection

- For specific worst case inputs, $O(n^2)$ is the best we can do…

But most problems aren’t worst case!

*Computational Geometry Algorithms and Applications*, de Berg, Cheong, van Kreveld and Overmars, Chapter 2
Outline for Today

- Questions about Homework 1?
  - Questions about CGAL/Qt installation?
- Motivating Applications for Line Segment Intersection Problem
- Line/Segment Intersection Math
- Naive vs. Output Sensitive Algorithms
- A Plane/Line Sweep Algorithm
- Specific Choices for Data Structures
- Analysis
- Corner Cases / Degeneracies
- Next Time
A Classic Computational Geometry Tool: The Line-Sweep (or Plane-Sweep) Algorithm

- Incrementally focus on a subset of the data at a time
- Sweep line will move from top to bottom across our dataset
- Sweep line/plane is used to define the current status
- **Active segments** = those that touch/intersect the sweep line’s current position
A Classic Computational Geometry Tool: The Line-Sweep (or Plane-Sweep) Algorithm

- We will only look for intersections between **green segments**
- We will never check for intersections between a **red line** and a **blue line**
- Why is this ok?
As line sweeps down, handle Events in Event Queue

- Line segment **added** to active set
- Line segment **removed** from active set

We know “when” (vertical position) these events will happen and can pre-schedule them. Simply sort the y coordinates of all of the input line segments.

- Line segment **intersection**

*We don’t know when these will happen! This is what we’re trying to solve for!*

- Add seg
- Add seg
- Add seg
- seg & seg intersect
- Remove seg
- Add seg
- seg & seg intersect
- Remove seg
- Remove seg
Intersections between Active Segments

- Must we intersect every active segment to every other active segment?
Intersections between Active Segments

- Must we intersect every active segment to every other active segment?
  
  - No… We can do better!
    - Maintain the active segments ordered by the x position of intersection with the current sweep line
    - Only compare segments that are adjacent in this ordering
Intersections between Active Segments

- When a segment (f) is removed
Intersections between Active Segments

- When a segment (f) is removed

```
d g f b e a c
d g b e a c
```

The newly adjacent segments (g & b) are checked for intersection
Intersections between Active Segments

- When a segment (g) is added
Intersections between Active Segments

- When a segment (g) is added
  
  `d f b a e c`
  
  `d g f b a e c`

  The newly adjacent segments (d & g, g & f)

  are checked for intersection
Intersections between Active Segments

- When the sweep line reaches an intersection (a&e)
Intersections between Active Segments

- When the sweep line reaches an intersection (a&e)

\[ \text{d g f b a e c} \]

\[ \text{d g f b e a c} \]

Swap the positions in the horizontal ordering

And check for intersections with the new neighbors (b & e, a & c)
Intersections between Active Segments

- Sometimes the intersection is in the past…
  (y coordinate is above current sweep line position)
- We’ve already processed this intersection

Do nothing
Intersections between Active Segments

- Sometimes the intersection is in the future… (y coordinate is below current sweep line position)
- We may or may not have already detected this intersection…

It may or may not already be in the **Event Queue**

We’ll process it when the sweep line gets to that position

(just make sure we don’t add a duplicate!)
Outline for Today

● Questions about Homework 1?
  Questions about CGAL/Qt installation?
● Motivating Applications for Line Segment Intersection Problem
● Line/Segment Intersection Math
● Naive vs. Output Sensitive Algorithms
● A Plane/Line Sweep Algorithm
● Specific Choices for Data Structures
● Analysis
● Corner Cases / Degeneracies
● Next Time
Data / Data Structures maintained during Sweep

- What data structure do we use for the vertically-ordered **Event Queue**?
  - Is it an array?
  - Is it a linked list?
  - Is it a priority queue?
  - Is it a binary search tree?
  - Is it a hash table?
Data / Data Structures maintained during Sweep

- What data structure do we use for the vertically-ordered **Event Queue**?
  - Is it an array?
  - Is it a linked list?
  - Is it a priority queue?
  - Is it a binary search tree?
  - Is it a hash table?

- We start with a vertically-sorted collection of all of the end points
- We remove events one at a time in order
- We insert intersection points as they are detected, one at a time, not necessarily in a particular order
- We need to check for existence before adding a duplicate
Data / Data Structures maintained during Sweep

- What data structure do we use for the horizontally-ordered Active Segment Status Structure?
  - Is it an array?
  - Is it a linked list?
  - Is it a priority queue?
  - Is it a binary search tree?
  - Is it a hash table?
Data / Data Structures maintained during Sweep

- What data structure do we use for the horizontally-ordered **Active Segment Status Structure**?
  - Is it an array?
  - Is it a linked list?
  - Is it a priority queue?
  - Is it a binary search tree?
  - Is it a hash table?

- Initially empty
- Segments are added, removed, and swapped
- Adjacent neighbors are queried often

*Computational Geometry Algorithms and Applications*, de Berg, Cheong, van Kreveld and Overmars, Chapter 2
Outline for Today

- Questions about Homework 1?
  - Questions about CGAL/Qt installation?
- Motivating Applications for Line Segment Intersection Problem
- Line/Segment Intersection Math
- Naive vs. Output Sensitive Algorithms
- A Plane/Line Sweep Algorithm
- Specific Choices for Data Structures
- Analysis
- Corner Cases / Degeneracies
- Next Time
Analysis - Running Time

- For $n = \# \text{ of input segments}$,
  $k = \# \text{ of output intersections}$
  $s = \max \# \text{ of items on sweep line / in status structure at one time}$

- Step 1: Create add segment and remove segment events, sort and initialize the Event Queue

- Step 2: For each entry in the Event Queue
  - Update the Active Segment Status Structure
  - Compute intersections between newly adjacent segments
  - Add new intersections to the Event Queue

- Overall:
Analysis - Running Time

- For \( n = \# \text{ of input segments} \),
  \( k = \# \text{ of output intersections} \rightarrow k \leq n(n-1)/2 \)
  \( s = \max \# \text{ of items on sweep line / in status structure at one time} \rightarrow s \leq n \)

- Step 1: Create add segment and remove segment events, sort and initialize the Event Queue \( \rightarrow O(n \log n) \)

- Step 2: For each entry in the Event Queue \( \rightarrow O(n + k) \)
  - Update the Active Segment Status Structure \( \rightarrow O(\log s) \)
  - Compute intersections between newly adjacent segments \( \rightarrow O(1) \)
  - Add new intersections to the Event Queue
    \( \rightarrow O(\log (n+k)) \rightarrow O(\log (n+n^2)) \rightarrow O(2 \times \log n) \rightarrow O(\log n) \)

- Overall: \( O(n \times \log n + (n+k) \times (\log n)) \rightarrow O((k+n) \times \log n) \)

Algorithm & result has been improved… lower bound is: \( \Omega(n \log n + k) \)
Analysis - Storage / Memory

- For \( n = \# \text{ of input segments}, \) 
  \( k = \# \text{ of output intersections} \rightarrow k \leq \frac{n(n-1)}{2} \) 
  \( s = \max \# \text{ of items items on sweep line / in status structure at one time} \rightarrow s \leq n \)

- Step 1: Create add segment and remove segment events, sort and initialize the Event Queue

- Step 2: For each entry in the Event Queue
  Update the Active Segment Status Structure

- Overall:
Analysis - Storage / Memory

- For $n =$ # of input segments,
  $k =$ # of output intersections $\rightarrow k \leq n(n-1)/2$
  $s =$ max # of items items on sweep line / in status structure at one time $\rightarrow s \leq n$

- Step 1: Create add segment and remove segment events, sort and initialize the Event Queue
  $\rightarrow$ “in place” sorting algorithm, $O(1)$ add’l memory

- Step 2: For each entry in the Event Queue
  $\rightarrow$ maximum size $O(n + k)$
  - Update the Active Segment Status Structure
    $\rightarrow$ maximum size, $O(s)$

- Overall: $O(n + k)$ extra memory!
Analysis - Storage / Memory

- For \( n = \# \text{ of input segments} \),
  \( k = \# \text{ of output intersections} \rightarrow k \leq n(n-1)/2 \)
  \( s = \max \# \text{ of items items on sweep line / in status structure at one time} \rightarrow s \leq n \)
- Step 1: Create add segment and remove segment events, sort and initialize the Event Queue
  → “in place” sorting algorithm, \( O(1) \) add’l memory
- Step 2: For each entry in the Event Queue
  → maximum size \( O(n + k) \)
  - Update the Active Segment Status Structure
    → maximum size, \( O(s) \)
  - Overall: \( O(n + k) \) extra memory!
- Better: Don’t store “future” intersection of non-adjacent segments
  → \( O(n) \) extra memory!
Outline for Today

- Questions about Homework 1?
  Questions about CGAL/Qt installation?
- Motivating Applications for Line Segment Intersection Problem
- Line/Segment Intersection Math
- Naive vs. Output Sensitive Algorithms
- A Plane/Line Sweep Algorithm
- Specific Choices for Data Structures
- Analysis
- Corner Cases / Degeneracies
- Next Time
Corner Cases / Degeneracies

- We assumed these situations don’t occur:
Corner Cases / Degeneracies

- We assumed these situations don’t occur:

*Computational Geometry Algorithms and Applications*, de Berg, Cheong, van Kreveld and Overmars, Chapter 2
Corner Cases / Degeneracies

- We assumed these situations don’t occur:
  - 3 or more segments intersect at a point
  - Intersection may be at the segment endpoint (rather than in the middle)
  - Segments may be perfectly horizontal (parallel to sweep line)
  - 2 or more simultaneous events (add segment, remove segment, intersection)
  - And general floating point rounding headaches…

- However, these situations can be handled properly in the algorithm without too much more fuss… see the textbook for details

Algorithm & analysis are still good & valid

Note: segments touching at endpoints is not a rare occurrence for this application. Our river & road polylines are connected at the endpoints!
Outline for Today

- Questions about Homework 1?
  Questions about CGAL/Qt installation?
- Motivating Applications for Line Segment Intersection Problem
- Line/Segment Intersection Math
- Naive vs. Output Sensitive Algorithms
- A Plane/Line Sweep Algorithm
- Specific Choices for Data Structures
- Analysis
- Corner Cases / Degeneracies
- Next Time
Next Time

- Cartography (map making) is not just river and road polylines, it is also the areas or regions.
- How do we describe and store a region?
- How do we overlay, intersect, & union map areas or regions?

https://www.natureconservancy.ca/assets/images/graphics/nat/maps/Forest-regions-map-NCC-1000px-custom.jpg
Next Time

- Complexity of the intersection of non-convex polygons…

Figure 7.11  The intersection of two star-shaped polygons.