Lecture 16: Windowing, Interval & Segment Trees
Outline for Today

- Review from Last Time: Delaunay Triangulations
- Motivation: Cartography Windowing & Data Selection
- Lecture 8 Review: Points in k-D trees
- 1D Interval Tree
- 1D Interval Tree Analysis
- 2D Interval Tree + Range Tree
- 2D Interval Tree + Range Tree Analysis
- 2D Axis Aligned Segment Query
- Segment Tree for general 2D Segment Query
- Segment Tree Analysis
Construction by Point Insertion

- Start with convex hull
  - Triangulate it
  - \(k-2\) triangles
- For some ordering of the other points
  - Determine which triangle the point lies inside of
  - Replace that triangle with 3 triangles
  - \((n - k) \times 2\) additional triangles
- \(2n - k - 2\) total triangles!
Construction by Line Sweep

- Sort the input points by x
- Form a triangle with the 3 leftmost points
- Add every other point from left to right
  - Determine which points on the current hull are visible from the new point
  - Add a fan of triangles connecting the new point to the visible hull points

“Discrete and Computational Geometry”, Devadoss & O’Rourke, Princeton University Press 2011, Chapter 3
The Flip Graph

- If we did generate every triangulation...
- Let’s organize the triangulations as nodes in a graph
- We’ll put an edge between two nodes if flipping a single edge converts one triangulation into the other triangulation

“Discrete and Computational Geometry”, Devadoss & O’Rourke, Princeton University Press 2011, Chapter 3
Delaunay Construction Analysis Summary

- Brute force (enumerate all triangles, construct circles, reject… )
  \[ \rightarrow O(n^3 \cdot n) = O(n^4) \]
- Construct any triangulation & Flip until all edges are legal
  \[ \rightarrow O(n^2) \]
- Randomized Incremental Construction
  \[ \rightarrow O(n \log n) \]
- By duality, reduce to problem of Constructing the Voronoi Diagram
  \[ \rightarrow O(n \log n) \]
Outline for Today

- Review from Last Time: Delaunay Triangulations
- Motivation: Cartography Windowing & Data Selection
- Lecture 8 Review: Points in k-D trees
- 1D Interval Tree
- 1D Interval Tree Analysis
- 2D Interval Tree + Range Tree
- 2D Interval Tree + Range Tree Analysis
- 2D Axis Aligned Segment Query
- Segment Tree for general 2D Segment Query
- Segment Tree Analysis
Motivation: Cartography (Map-Making)

- Select a small rectangular region to display in a window at larger scale
Motivation: Visibility

Seth Teller, PhD thesis, 1992, Berkeley Soda Hall walkthrough
Graphics: 3D Clipping

- Eliminate portions of objects outside the viewing frustum
- View Frustum
  - boundaries of the image plane projected in 3D
  - a near & far clipping plane
Graphics: 2D Clipping

Why do it?

- Reduce amount of geometry going through graphics pipeline
- Prevent rendering bugs from overflow, wraparound, things behind the camera, etc.

https://www.tutorialandexample.com/clipping-in-computer-graphics
Outline for Today

- Review from Last Time: Delaunay Triangulations
- Motivation: Cartography Windowing & Data Selection
- Lecture 8 Review: Points in k-D trees
- 1D Interval Tree
- 1D Interval Tree Analysis
- 2D Interval Tree + Range Tree
- 2D Interval Tree + Range Tree Analysis
- 2D Axis Aligned Segment Query
- Segment Tree for general 2D Segment Query
- Segment Tree Analysis
Review from Lecture 8: 2D k-d Tree

- Used to store points
- Alternate splitting horizontally & vertically
- If data is available for preprocess, the structure is easy to balance
- Point data is only stored at the leaves
Outline for Today

- Review from Last Time: Delaunay Triangulations
- Motivation: Cartography Windowing & Data Selection
- Lecture 8 Review: Points in k-D trees
- 1D Interval Tree
- 1D Interval Tree Analysis
- 2D Interval Tree + Range Tree
- 2D Interval Tree + Range Tree Analysis
- 2D Axis Aligned Segment Query
- Segment Tree for general 2D Segment Query
- Segment Tree Analysis
What about Segments? Let’s Tackle 1D First…

- Input: A collection of $n$ line segments on the x-axis
- For a query interval, return all line segments that overlap the query interval
Traditional Binary Search Tree

- Select split point near middle of data
- What about segments that overlap the split?
Traditional Binary Search Tree

- Select split point near middle of data
- What about segments that overlap the split?
- Should we store them on both sides?
  - Uses extra memory
  - We may lose our $O(\log n)$ performance!
Interval Tree

- Chose a split point and make 3 groups:
  - $I_{\text{mid}}$ = Segments that overlap the split
  - $I_{\text{left}}$ = Segments completely to the left
  - $I_{\text{right}}$ = Segments completely to the right
Interval Tree

- Recurse down the tree only with items that DO NOT overlap the split point.
Interval Tree

- Items in $I_{\text{mid}}$ group will stay at the current node.

- Each node stores two sorted lists:
  - $L_{\text{left}} = I_{\text{mid}}$ sorted by left endpoint (increasing)
  - $L_{\text{right}} = I_{\text{mid}}$ sorted by right endpoint (decreasing)
Interval Tree

- For a specific query
Interval Tree

- For a specific query
- Determine if the query is to the right (or left) of the current node
- **Binary search** within the $L_{right}$ list (or $L_{left}$ list) by right (or left) endpoint
  - Return all segments with endpoint further away from the query
- And recurse down the right (or left)
Outline for Today

● Review from Last Time: Delaunay Triangulations
● Motivation: Cartography Windowing & Data Selection
● Lecture 8 Review: Points in k-D trees
● 1D Interval Tree
● 1D Interval Tree Analysis
● 2D Interval Tree + Range Tree
● 2D Interval Tree + Range Tree Analysis
● 2D Axis Aligned Segment Query
● Segment Tree for general 2D Segment Query
● Segment Tree Analysis
1D Interval Tree Analysis

- For $n$ input segments and a query that will return $k$ items

- Memory Usage:

- Construction Time:

- Query Time:
1D Interval Tree Analysis

- For $n$ input segments and a query that will return $k$ items

- Memory Usage: $O(n)$

- Construction Time: $O(n \log n)$

- Query Time: $O(\log n + k)$

*Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 10*
Outline for Today

● Review from Last Time: Delaunay Triangulations
● Motivation: Cartography Windowing & Data Selection
● Lecture 8 Review: Points in k-D trees
● 1D Interval Tree
● 1D Interval Tree Analysis
● 2D Interval Tree + Range Tree
● 2D Interval Tree + Range Tree Analysis
● 2D Axis Aligned Segment Query
● Segment Tree for general 2D Segment Query
● Segment Tree Analysis
How do we Extend to 2D?

- First consider only horizontal input line segments
- And instead of a query line, we’ll have a *query line segment*
How do we Extend to 2D?

- We’ll replace the sorted lists of the interval tree with a 2D range query (Lecture 8)
- This will require $O(\log n)$ additional memory…

---

*Computational Geometry Algorithms and Applications,* de Berg, Cheong, van Kreveld and Overmars, Chapters 8 & 10
Outline for Today

- Review from Last Time: Delaunay Triangulations
- Motivation: Cartography Windowing & Data Selection
- Lecture 8 Review: Points in k-D trees
- 1D Interval Tree
- 1D Interval Tree Analysis
- 2D Interval Tree + Range Tree
- 2D Interval Tree + Range Tree Analysis
- 2D Axis Aligned Segment Query
- Segment Tree for general 2D Segment Query
- Segment Tree Analysis
2D Interval Tree + Range Tree Analysis

- For $n$ horizontal input segments and a query segment that will return $k$ items

- Memory Usage:

- Construction Time:

- Query Time:
2D Interval Tree + Range Tree Analysis

- For $n$ horizontal input segments and a query segment that will return $k$ items

- Memory Usage:
  \[ \rightarrow O(n \log n) \]

- Construction Time:
  \[ \rightarrow O(n \log n) \]

- Query Time:
  \[ \rightarrow O(\log n + k) \]
Outline for Today

● Review from Last Time: Delaunay Triangulations
● Motivation: Cartography Windowing & Data Selection
● Lecture 8 Review: Points in k-D trees
● 1D Interval Tree
● 1D Interval Tree Analysis
● 2D Interval Tree + Range Tree
● 2D Interval Tree + Range Tree Analysis
● 2D Axis Aligned Segment Query
● Segment Tree for general 2D Segment Query
● Segment Tree Analysis
How to handle a 2D Axis-Aligned Query Box?

- Initially, let’s restrict to horizontal & vertical segments
How to handle a 2D Axis-Aligned Query Box?

- Initially, let’s restrict to horizontal & vertical segments

- Case Analysis:
  Segments that touch the query box will:
  - Have one endpoint inside the box, OR
  - Will have both endpoints outside the box AND
    - Will be a horizontal segment that overlaps the left edge of the box OR
    - Will be a vertical segment that overlaps the bottom edge of the box

Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 10
How to handle a 2D Axis-Aligned Query Box?

- Initially, let’s restrict to horizontal & vertical segments

- Case Analysis:

Segments that touch the query box will:

- Have one endpoint inside the box, OR
- Will have both endpoints outside the box AND
  - Will be a horizontal segment that overlaps the left edge of the box OR
  - Will be a vertical segment that overlaps the bottom edge of the box

Handled with a Lecture 8 2D Range Query

Handled with an Interval Tree + 2D Range Query

*Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 10*
How to handle a 2D Axis-Aligned Query Box?

- Initially, let’s restrict to horizontal & vertical segments

Case Analysis:
Segments that touch the query box will:
- Have one endpoint inside the box, OR
- Will have both endpoints outside the box AND
  - Will be a horizontal segment that overlaps the left edge of the box OR
  - Will be a vertical segment that overlaps the bottom edge of the box

Caution: Need to prevent duplicates in output

Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 10
Outline for Today

- Review from Last Time: Delaunay Triangulations
- Motivation: Cartography Windowing & Data Selection
- Lecture 8 Review: Points in k-D trees
- 1D Interval Tree
- 1D Interval Tree Analysis
- 2D Interval Tree + Range Tree
- 2D Interval Tree + Range Tree Analysis
- 2D Axis Aligned Segment Query
- Segment Tree for General 2D Segment Query
- Segment Tree Analysis
How do we handle General 2D Segments?

- Not restricted to horizontal & vertical segments!
- (Note: We will later insist that the segments do not cross… )
How do we handle General 2D Segments?

- *Do the (sloppy?) Computer Graphics thing…*
  
  Output the segment if its bounding overlaps the axis-aligned query box
How do we handle General 2D Segments?

- *Do the (sloppy?) Computer Graphics thing…*
  Output the segment if its bounding overlaps the axis-aligned query box

- *We might have LOTS of false positives!*

*Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 10*
How do we handle General 2D Segments?

- Do the (sloppy?) Computer Graphics thing…
  Output the segment if its bounding overlaps the axis-aligned query box

- We might have LOTS of false positives!

- Can we do better?
  - Ensure good (output sensitive) Performance
  - Avoid false positives?
Segment Tree - First Dimension (x)

- First, sort the x coordinates of the start and end points of every segment.
- Construct a balanced binary search tree with these x values.
- Insert every segment into the structure.
- If a segment overlaps both the left and right subranges of the node, store it at the node (do not recurse).

Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 10
Segment Tree - First Dimension \((x)\)

- For a vertical query slab \((x_{\text{min}}, x_{\text{max}})\)
- Walk down the tree
- If the node is in range, return all items at that node
- Recurse left and/or right as appropriate
- & filter duplicates…

Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 10
Segment Tree - Second Dimension (y)

- To efficiently query a vertical range in addition to the horizontal range:
- Sort the segments stored at each node by \( y \)
- Remember: this is only the segments that completely overlap the node’s range
- Note: this is why we require no crossings in the input segments

Now we can perform binary search at each node to only return the segments in the vertical query range
Segment Tree - Second Dimension (y)

- To efficiently query a vertical range in addition to the horizontal range:
  - Sort the segments stored at each node by $y$
  - **Remember: this is only the segments that completely overlaps the node’s range**
  - Note: this is why we require no crossings in the input segments

![Diagram showing segment tree](image)

Now we can perform binary search at each node to only return the segments in the vertical query range.
Outline for Today

● Review from Last Time: Delaunay Triangulations
● Motivation: Cartography Windowing & Data Selection
● Lecture 8 Review: Points in k-D trees
● 1D Interval Tree
● 1D Interval Tree Analysis
● 2D Interval Tree + Range Tree
● 2D Interval Tree + Range Tree Analysis
● 2D Axis Aligned Segment Query
● Segment Tree for General 2D Segment Query
● Segment Tree Analysis
Segment Tree - Analysis

- For $n$ input segments, for a query that will return $k$ segments
- Memory:
  Each segment is stored in at most 2 nodes per level
- Construction Time:
  Presort all endpoints by $x$ & $y$ $O(n \log n)$
- Query Time:

*Computational Geometry Algorithms and Applications*, de Berg, Cheong, van Kreveld and Overmars, Chapter 10
Segment Tree - Analysis

- For $n$ input segments, for a query that will return $k$ segments

- Memory:
  Each segment is stored in at most 2 nodes per level
  $\rightarrow O(n \log n)$

- Construction Time:
  Presort all endpoints by $x$ & $y$ $O(n \log n)$
  $\rightarrow O(n \log n)$

- Query Time:
  $\rightarrow O(\log n \times \log n + k)$
  $\rightarrow O(\log^2 n + k)$

*Computational Geometry Algorithms and Applications*, de Berg, Cheong, van Kreveld and Overmars, Chapter 10