Separating Chains Method
Last Time

- Arrangements of Line Segments
  - Special case: simple polygons
- Computing the trapezoidal decomposition

- Incremental algorithm
  - Decomposition DS
  - Point location DS
Last Time

- **Arrangements of Line Segments**
  - Special case: *simple polygons*
  - Computing the **trapezoidal decomposition**

- **Incremental algorithm**
  - Decomposition DS
  - Point location DS

- **Update**
  - Point location step
  - Tracing step

- **Solution specific to trapezoidal decompositions**
Today

- Point location in polygonal subdivisions in 2D

- Important examples
  - Triangulations
  - Line arrangements
  - Voronoi diagrams

- Based on the notion of separators (separating chains)

- References
  - [Lee, Preparata 1977] [Edelsbrunner, Guibas, Stolfi 1986]

- Several simple algorithmic ideas that come together nicely
**Point Location in 2D Polygonal Subdivisions**

- **Input**: A partition of the Euclidean plane $\mathbb{E}^2$ into polygonal regions
  - For example, given as a **vertex-edge-face** data structure

- **Output**: **Point location data structure** and **algorithm**
  - Preprocess to build an **index** structure
  - Support queries: given a point, **find the element** that contains it
Measures of Performance

- **Input size**
  - Number of edges \( m \)
  - Number of faces (regions) \( n \)
  - Euler: \( n = O(m) \)

- **Space**, \( S(m) \)

- **Preprocessing time**, \( P(m) \)

- **Query time**, \( Q(m) \)

- **Separating chains method**
  \[
  S(m) = O(m) \quad P(m) = O(m) \quad Q(m) = O(\log m)
  \]
Separating Chains Method

1D intuition

- Subdivision of a line
- Partitioned by points
Separating Chains Method

- **1D intuition**
  - Subdivision of a line
  - Partitioned by points

- **Input**: Subdivision structure is a linked list of vertices
- **Output**: A query DS and algorithm that reports an element of the subdivision that contains a given point
Separating Chains Method

- **1D intuition**
  - Subdivision of a line
  - Partitioned by points

- **Input:** Subdivision structure is a linked list of vertices
- **Output:** A query DS and algorithm that reports an element of the subdivision that contains a given point

- **Solution:** Binary search tree

\[ S(m) = O(m) \quad P(m) = O(m) \quad Q(m) = O(\log m) \]
Monotone Polygons

- **Def:** A polygon is **monotone** if it is intersected by any vertical line either along **exactly one line segment** or **not at all**

- **Def:** A polygonal subdivision is **monotone** if all its polygons are monotone
- Assume no vertical edges

- Any polygon boundary is **properly partitioned** into a “**floor**” and a “**ceiling**”
  - Starting and ending points of intersections with vertical lines
Observation: A subdivision is monotone iff no vertex has all adjacent edges on the same side of the vertical line through it.
Regularization

- **Observation**: A subdivision is monotone iff no vertex has all adjacent edges on the same side of the vertical line through it.

- **Triangulation**
- **Trapezoidal decomposition**

  - [Chazelle] $O(m)$ worst case
  - [Seidel] $O(m \log^* m)$ expected
  - [Bentley, Ottman] + diagonals $O(m \log m)$ worst case

- **Note**: effects preprocessing time!
Properties of Monotone Subdivisions

- **Def**: For all $A, B \subseteq \mathbb{E}^2$, we say that
  
  $A$ is above $B$ ($A \succcurlyeq B$)

  or

  $B$ is below $A$ ($B \ll A$)

  if any vertical line intersects $A$ strictly above $B$ (if the intersections are not empty)

- Not every two elements are comparable
Properties of Monotone Subdivisions

- **Lemma**: The relation is **acyclic**
  - The relation graph is a **DAG**
  - Extend by reflexivity
  - It is a **partial order**

- **Proof**: Consider the shortest cycle, prove that it can be shortened
Properties of Monotone Subdivisions

- **Lemma**: The relation is **acyclic**
  - The relation graph is a **DAG**
  - Extend by reflexivity
  - It is a **partial order**

- **Proof**: Consider the shortest cycle, prove that it can be shortened

- **Consequence**: There exists **total ordering**
  - **Consistent** with the partial order
  - **Topological sorting** of the relation DAG
    $$R_0 \ll R_1 \ll R_2 \ll R_4 \ll R_5 \ll R_6 \ll R_7$$
Separators

- **Def**: A separator is a polygonal line which
  - consists of the edges of the subdivision
  - intersects any vertical line in **exactly one point**

- Separator is a “horizontal” line

- **Def**: A complete family of separators is a collection of \( n - 1 \) distinct separators

\[
S_1 \ll S_2 \ll \cdots \ll S_{n-1}
\]
Existence of Separators

- **Theorem**: Every monotone subdivision admits a complete family of separators
Existence of Separators

- **Theorem**: Every monotone subdivision admits a complete family of separators

- **Proof**: Explicit construction

  - Linear ordering of the faces $R_0 \ll R_2 \cdots \ll R_{n-1}$

  ![Diagram](image)

  - Separator $S_i$ ($1 \leq i \leq n-1$) consists of all segments on the common boundary of any two faces $A, B$ such that $A \ll R_{i-1} \ll R_i \ll B$
Existence of Separators

- Verifying the definition for arbitrary $S_i$: need to prove
  - Consists of *polygon edges*
  - Intersected *exactly once* by any vertical line
Existence of Separators

- Verifying the definition for arbitrary $S_i$: need to prove
  - Consists of **polygon edges**
  - Intersected **exactly once** by any vertical line
- **At least one** intersection
  - Consider the sequence of intersected faces
Existence of Separators

- Verifying the definition for arbitrary $S_i$: need to prove
  - Consists of **polygon edges**
  - Intersected **exactly once** by any vertical line

- **At least one** intersection
  - Consider the sequence of intersected faces

- **At most one** intersection
  - **Contradiction**: $R_i \ll C \ll B \ll R_{i-1}$
  - Also implies that $S_i$ is a **polygonal line**
Existence of a Separator System

- Verifying the definition for $S_1, S_2, \ldots, S_{n-1}$. Need to prove
  - Ordering $S_1 \ll S_2 \ll \cdots \ll S_{n-1}$
  - No two equal
Existence of a Separator System

- Verifying the definition for $S_1, S_2, \ldots, S_{n-1}$. Need to prove
  - Ordering $S_1 \ll S_2 \ll \cdots \ll S_{n-1}$
  - No two equal

- **Fact**: $S_i \ll R_i \ll S_{i+1}$ $(1 \leq i \leq n-2)$
  - By construction, all segments in $S_i$ are $\ll R_i$ and $\gg R_{i-1}$
Existence of a Separator System

- Verifying the definition for $S_1, S_2, \ldots, S_{n-1}$. Need to prove
  - Ordering $S_1 \ll S_2 \ll \cdots \ll S_{n-1}$
  - No two equal

- **Fact**: $S_i \ll R_i \ll S_{i+1} \ (1 \leq i \leq n - 2)$
  - By construction, all segments in $S_i$ are $\ll R_i$ and $\gg R_{i-1}$

- Sufficient: no two **consecutive** $S_i$ and $S_{i+1}$ are equal
Properties of Separators

- **Consequence:**
  \[ R_0 \ll S_1 \ll R_1 \ll S_2 \ll \cdots \ll R_{n-2} \ll S_{n-1} \ll R_{n-1} \]

- **Consequence:** If \( e \) is in the common boundary of \( R_i \) and \( R_j \), the separators that contain it are exactly \( S_i, S_i, \ldots, S_{j-1} \)

- Computing separators?
**Point Location Data Structure**

- **Binary search tree on the ordered set of separators**
  - Binary search on \( y \)-coordinate
- **Within each node, binary search tree on the segments of the associated separator**
  - Segments ordered by \( x \)-coordinate
- **Two geometric primitives**
  - Query point **above/below** a segment (signed distance computation)
  - Query point **left/right** of a subdivision vertex (segment endpoint) (\( x \)-coordinate comparison)
Input: Point $p$

$i \leftarrow 0; j \leftarrow n - 1; k \leftarrow \text{root}$;

while $(i < j)$ {

  if $(i < k \leq j)$ {

    find $e \in S_k$ such that $e \ll p$ or $e \gg p$

    if $p \in e$ return $e$

    if $(p \gg e)$ $i \leftarrow$ index of the region above $e$

    if $(p \ll e)$ $j \leftarrow$ index of the region below $e$

  }

  else if $(k > j)$ $k \leftarrow$ left child of $k$

  else if $(k \leq i)$ $k \leftarrow$ right child of $k$

}
Analysis of the Basic Algorithm

- **Space:** $S(m) = O(m^2)$
  - “Outer” search: number of faces $O(n) = O(m)$
  - “Inner” search: number of segments in a separator $O(m)$

- **Query time:** $Q(m) = O(\log^2 m)$
  - Two nested binary searches

- Still don’t care about the preprocessing time
Improving the Space Requirement

- Avoid storing entire separators

**Space:** $S(m) = O(m)$

**Preprocessing time:** $P(m) = O(m)$

**$O(1)$-time computation of lca**

\[
lca(i, j) = j \& \sim (\text{msb}(i \^ j) - 1)
\]

$\text{msb}(x)$: obtained by zeroing all but the most significant bit of $x$
(precompute in a table)

$\^$: bitwise XOR \hspace{1cm} \sim$: bitwise NOT \hspace{1cm} \&$: bitwise AND
Improving the Time Requirement

- The idea of **fractional cascading**
- Replace **independent** binary searches by **correlated** ones

- Start from the bottom level of the “outer” tree
- Propagate a **fixed fraction** of the breakpoints to the parent
- Nodes in the next level repeat

- Only perform the “inner” binary search **once**, pay $O(\log m)$
- For each level of the “outer” search, do only $O(1)$ additional work

- **Query time** $Q(m) = O(\log m + \log m)$