ALGORITHMS FOR SPATIAL DBMS

Basic Geometry (in 2D)

• cross-product $u \times v$:
  - $(x_1, y_1) \times (x_2, y_2) := x_1 y_2 - y_1 x_2$
  - $u \times v = |u| |v| \sin \Theta$
  where $\Theta$ is angle between $u$ and $v$

• can use to get angle between two vectors
  - how to test whether $p$ is to the left/right of segment $uv$?
  - how to test whether two line segments intersect?

• can use to calculate area of parallelogram/triangle

Basic Polygon Tasks

• calculate area of polygon

Formula:

$$\frac{1}{2} \sum_{i=1}^{n} (x_i y_{i+1} - x_{i+1} y_i)$$
Point in Polygon

- horizontal stabbing: $O(n)$
- if pre-processing is possible:
  - point location query: $O(\log n)$

Overlays, etc.

Points of interests in overlays:
- intersection points

Also useful for:
- intersection
- union
- difference

Line Segment Intersection

Given: $\{s_1, ..., s_n\}$ a set of $n$ line segments
Output (detect): do any two of them intersect
or
Output (compute): list all intersections between the line segments
Line Segment Intersection

- trivial (detect/compute): $O(n^2)$
- plane sweep approach
  - natural: before two segments intersect, they are next to each other (assuming no three lines intersect in a point)
  - events: endpoints
  - active list: segments intersecting the current sweep line in order

detect Line Segment Intersection

\[
\begin{align*}
\text{detect intersects} & : (x, y) \\
\text{begin} & : \text{set the 2x endpoints of segments in } S \text{ and place them on } E \\
\text{for } i = 1 \text{ to } |E| & : \text{do} \\
\text{if } i = 1 & : \text{end for} \\
\text{end }
\end{align*}
\]

- $L$ needs to be dynamic binary search tree
- time: $O(n \log n)$, space: $O(n)$
- problem: does not report all intersections; as it is, it can't, why?

compute Line Segment Intersection

\[
\begin{align*}
\text{compute intersects} & : (x, y) \\
\text{begin} & : \text{set the 2x endpoints of segments in } S \text{ and place them on } E \\
\text{let } E' & : \text{be a set of points} \\
\text{for } i = 1 \text{ to } |E| & : \text{do} \\
\text{if } i \text{ is on the right of a segment } a & : \text{end for} \\
\text{end }
\end{align*}
\]

- what do we need to implement $L$?
- time/space analysis?
Red/Blue Intersection

For our applications:
- can assume two intersection-free sets (e.g. two polygons), think of them as red/blue
- simplifies general problem
- can be solved in time $O(n \log n + k)$
- algorithm is a bit tricky though; main problem:

![Diagram of red and blue intersections]

Polyline Intersection

The algorithms can be adapted to work for intersections of polylines (i.e. allowing common endpoints).

Polygon Intersection (detect)

Given: two polygons $P, Q$
Output: do $P$ and $Q$ intersect?

Algorithm:
- run line segment intersection test on edges of $P$ (red) and $Q$ (blue)
  - if intersect, then yes,
  - else
    - $p :=$ point of $P$;
    - if $p$ in $Q$, then yes
      - else
        - $q :=$ point of $Q$;
        - if $q$ in $P$, then yes
        - else no.
Polygon Operations

General approach:
- calculate common faces
- orient edges along boundary (face is to the left of an edge), leads to doubly connected edge lists, each edge has two sides
- run line segment intersection and update edge lists
- recalculate faces
- get union, difference, intersection, etc.

Doubly edge connected lists

Record of
- vertices
- half-edges (sides)
- faces (boundary traversal in clockwise order)
- faces can have holes (traversed counterclockwise: face is always to left of half-edge)

Updating edge lists
Convex Polygon Intersection

Given: convex polygons $P$, $Q$
Output: intersection

Naïve algorithm: $|P| \cdot |Q|$
Can be done in time $O(|P| + |Q|)$:
- Shamos-Hoey
  - split into slabs, do line sweep
- O’Rourke
  - follow boundary

Clipping