Intersection

Geometric intersections

- One of the most basic problems in computational geometry
- Applications
  - Solid modeling : CSG(constructive solid geometry) - build up complex shapes by applying various boolean operations (intersection, union, difference) to simple primitive shapes.
  - Robotics and motion planning : collision detection, avoidance
  - GIS : map overlay
  - Computer graphics : ray shooting

Line segment intersection

- Given n line segments in the plane, report all intersection points.
- # of possible intersections?
  - $0 \sim {}_nC_2 = O(n^2)$
  - So, $O(n^2)$ algorithm is worst case optimal.
  - Need output sensitive algorithm!
  - I : # intersections
  - Lower bound : $\Omega(n \log n + I)$

Lower bound

- Reduction from element uniqueness at O(n) time.
- Element uniqueness – given n real numbers, are all of these numbers distinct? – requires Ω(n log n) time in algebraic decision tree model of computation.
- Given n numbers, construct n horizontal line segments with the same y-coord.

Idea

- For output-sensitive algorithm, instead of checking all pairs of segments, just consider segments that are close.
- Use line (plane) sweep algorithm!

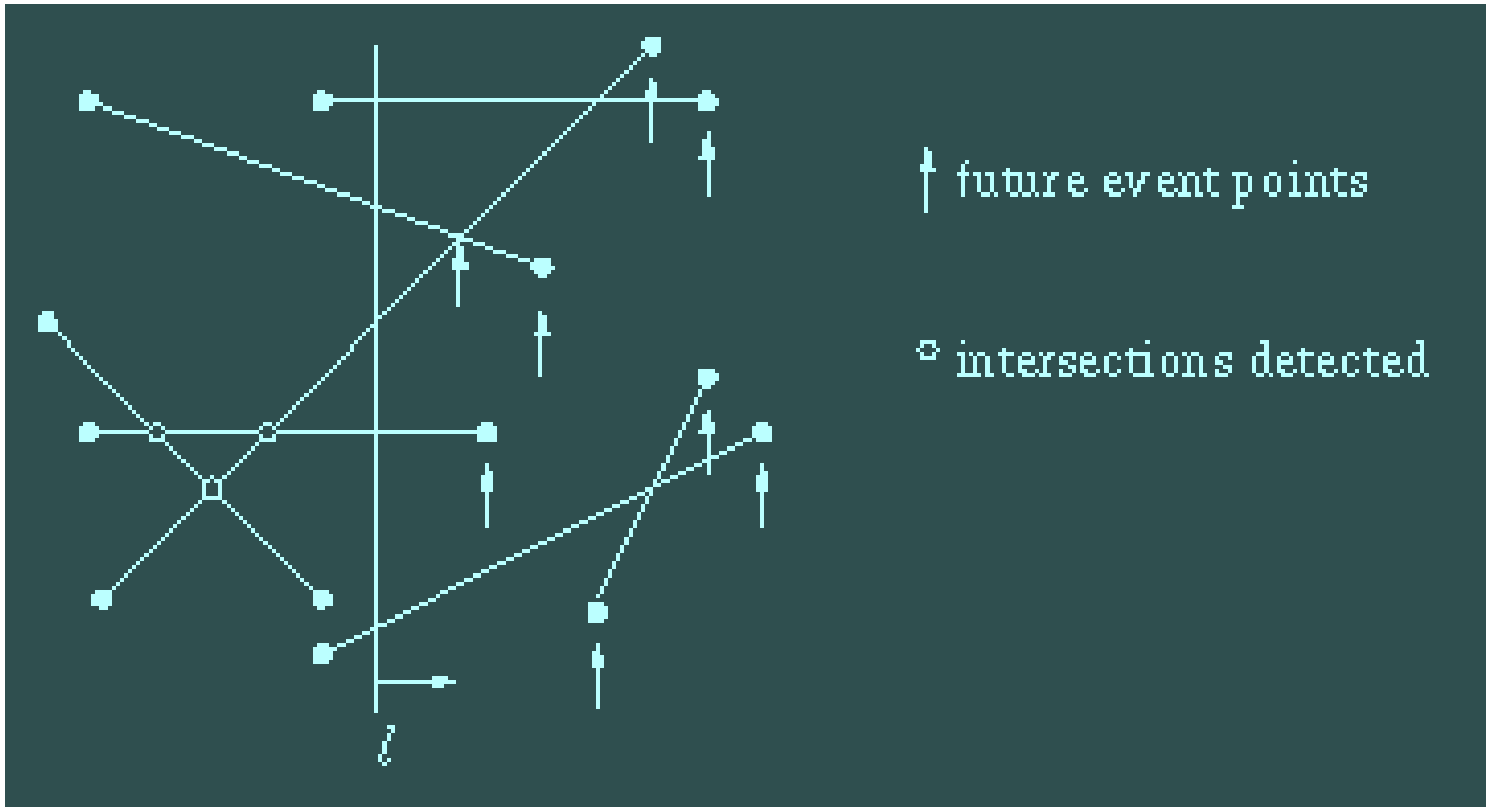
General position assumption

- No line segment is vertical.
- If 2 line segments intersect, they intersect in a single point (they are not collinear).
- No 3 line segments intersect in a common point.

Plane sweep algorithm

- As we sweep a vertical line from left to right, we report the intersections.
- Events
  - Endpoint events : where the sweep line meets an endpoint of line segment
  - Intersection events : where the sweep line meets an intersection point of 2 line segments.
- How to find intersection events?
  - When 2 segments become adjacent along the sweep line, check for an intersection occurring to the right of the sweep line.

Plane sweep algorithm



Data structures

- Event queue : holds future events sorted by x coordinate.
  - Insert a new event
  - Extract the min event
  - Use a balanced binary search tree or skip list : O(log M) where M is the number of entries in queue.
  - Same x-coord? : use lexicographic order
- Sweep line status : set of segments intersecting the sweep line in sorted order (from top to bottom)
  - Use a balanced binary search tree or skip list.
  - Deleting, inserting, swapping, determining 2 immediate predecessor and successor : O(log n)

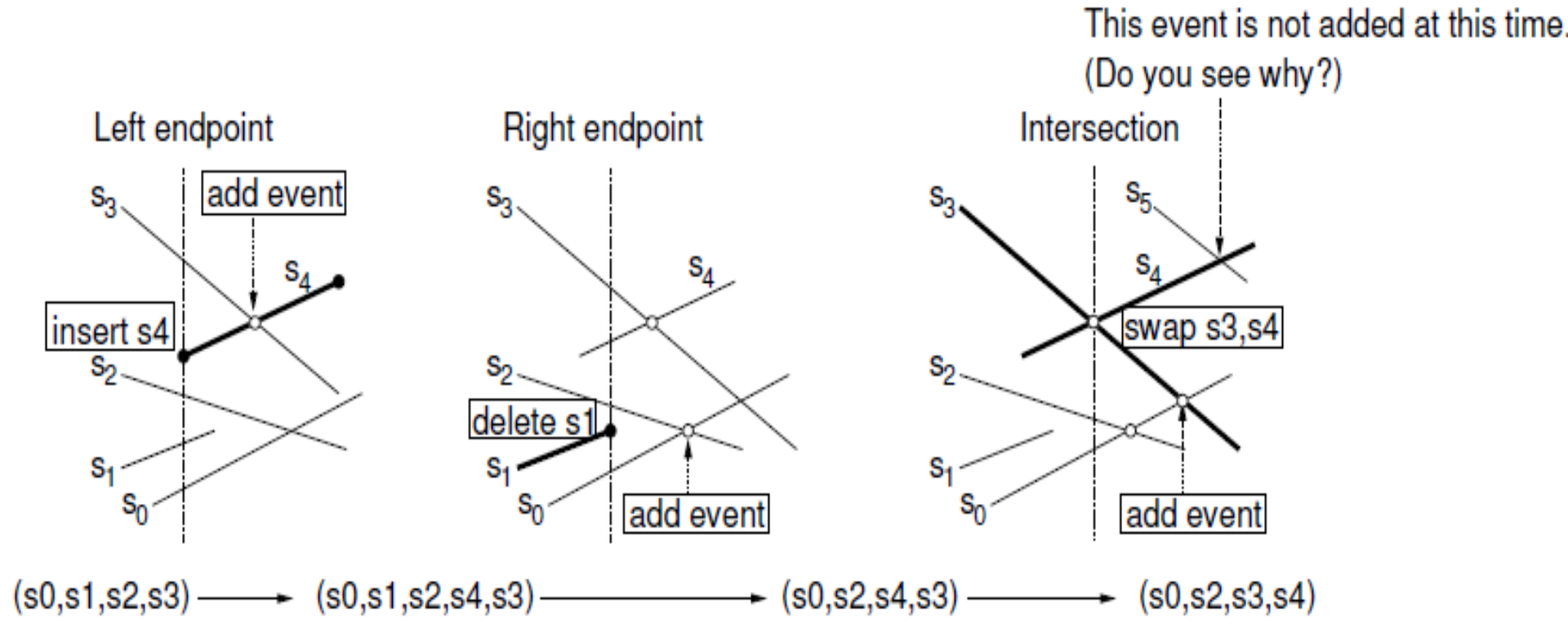
Plane sweep algorithm

- Initially, insert all endpoints of the line segments into the event queue.
- While the event queue is nonempty, extract the next event in the queue.
  - Left endpoint
  - Right endpoint
  - Intersection point

Event

- Left endpoint :
  - Insert the line segment into the sweep line status
  - Test for intersections with the segment immediately above and below
- Right endpoint :
  - Delete the line segment from the sweep line status.
  - Test for intersections for the segments immediately above and below.
- Intersection point :
  - Swap the 2 line segments in order along the sweep line .
  - Test for intersections for the new upper, lower segments.

Plane sweep algorithm



Analysis

- # operations, time for each operation
- Sweep line status : at most n segments intersect the sweep line, O(log n) time per operation.
- Total # events on event queue = $2n + I \Rightarrow O(\log (2n+I)) \Rightarrow$ Since $I \leq n^2$, O(log n) time per operation
- Each event involves constant # of accesses or operations to the sweep line status or the event queue, so
- Total time = $O((2n+I) \log n) = O(n \log n + I \log n)$