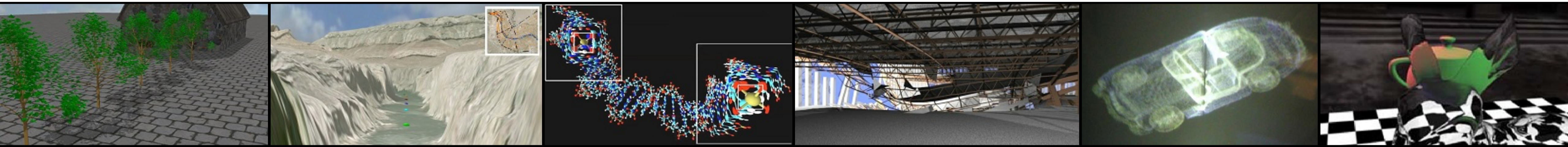
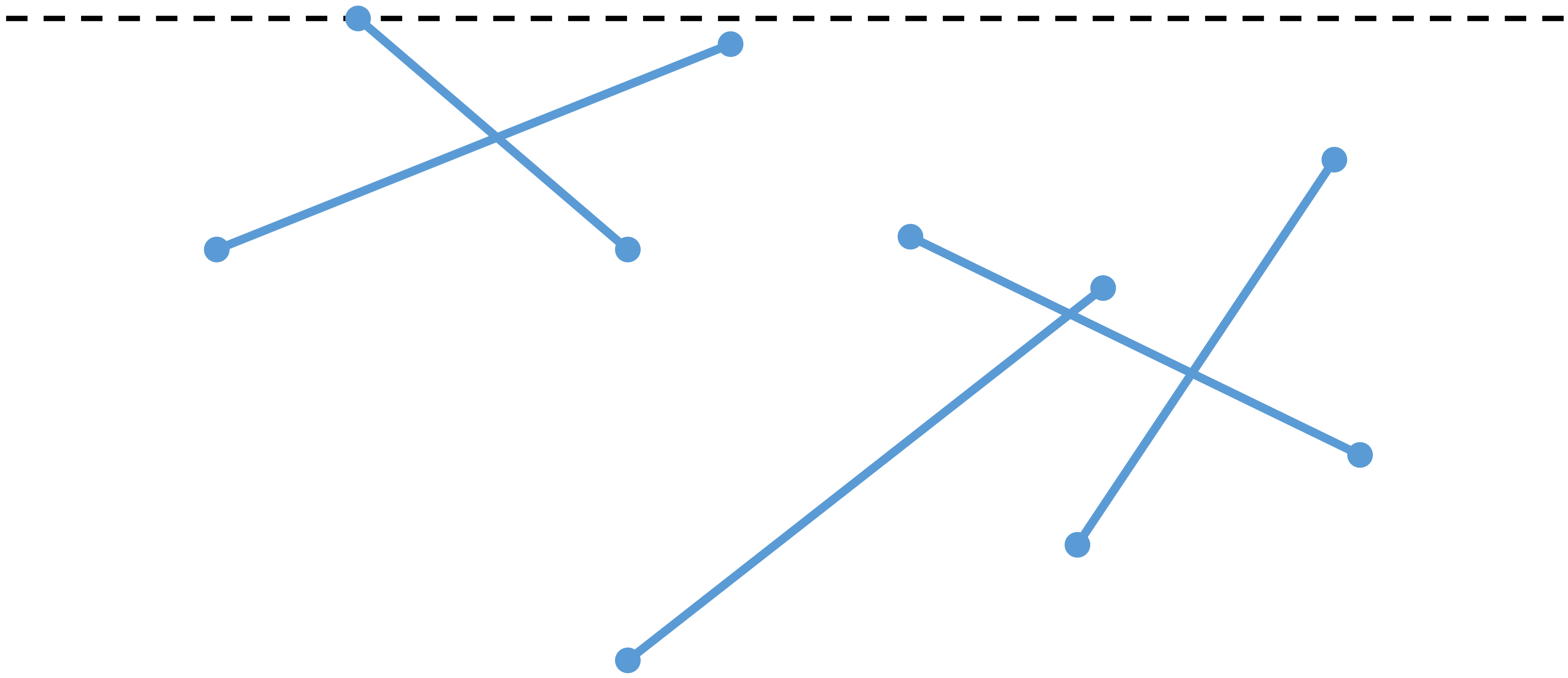
COT 4521: Introduction to Computational Geometry

Segment Intersection Line Sweep

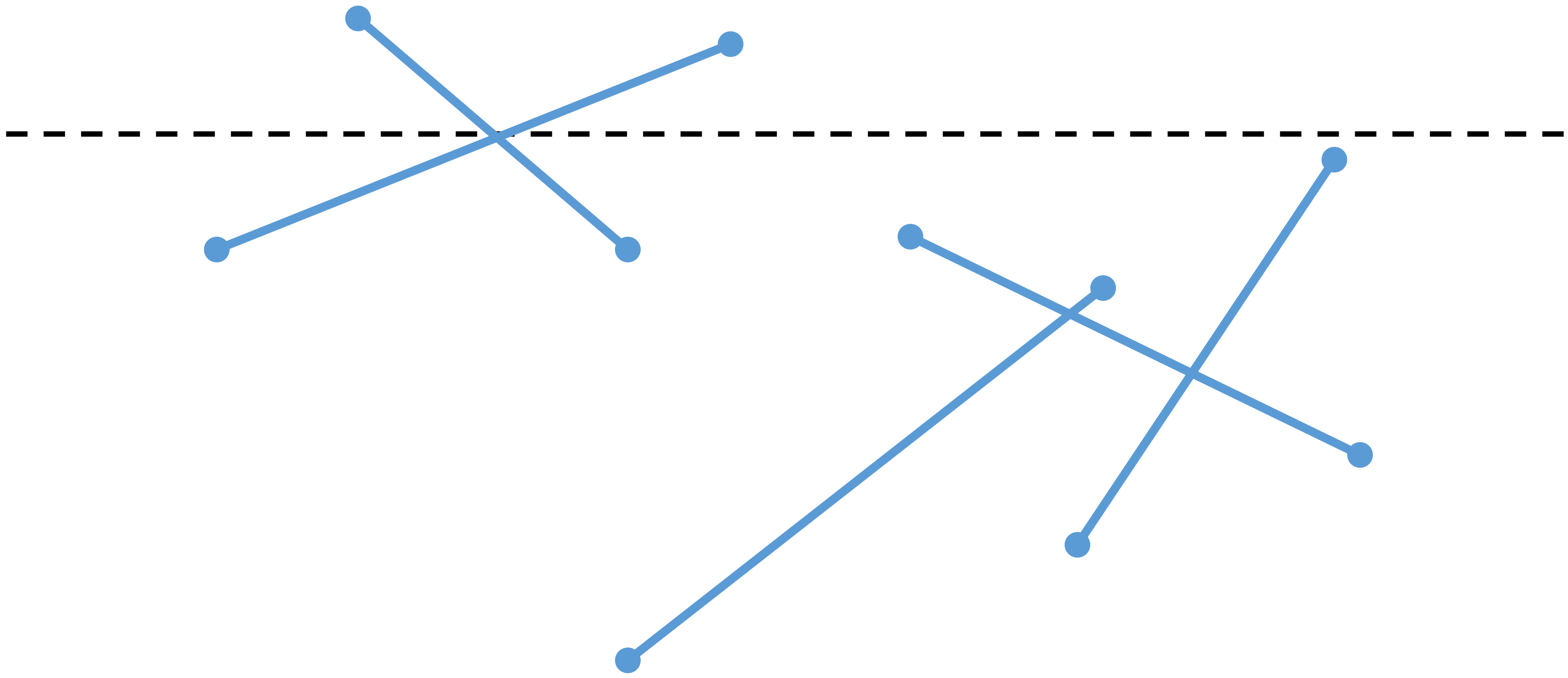
Paul Rosen
Assistant Professor
University of South Florida



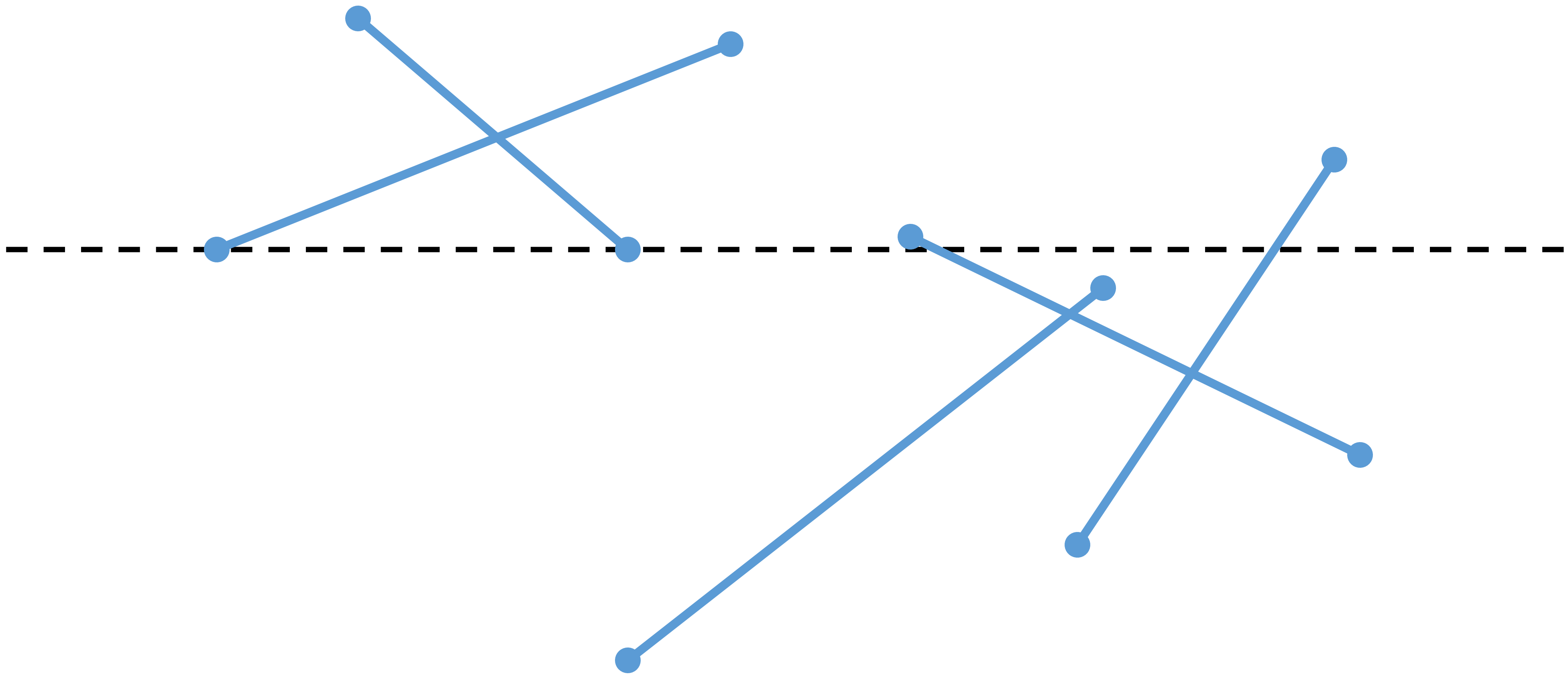
SWEEP ALGORITHM



Sweep Algorithm



Sweep Algorithm – Where do Events Occur?



EVENTS

- WHEN DO THE EVENTS HAPPEN?
  - When the sweep is:
    - At an upper endpoint of the line segment;
    - At a lower endpoint of the line segment; or
    - At an intersection point of line segments
  - At each type, the status changes

Intersection of line segments

- Algorithm idea (Shamos-Hoey algorithm)
- Crucial observation:
  - For two segments s_1 and s_2 to intersect, there must be some Y for which s_1 and s_2 are consecutive in the X ordering.
  - This suggests that the sequence of intersections of the segments with the horizontal line contains the information needed to find the intersections between the segments.

- Plane sweep algorithms often use two data structures:
  1. Sweep-line status
  2. Event-point schedule

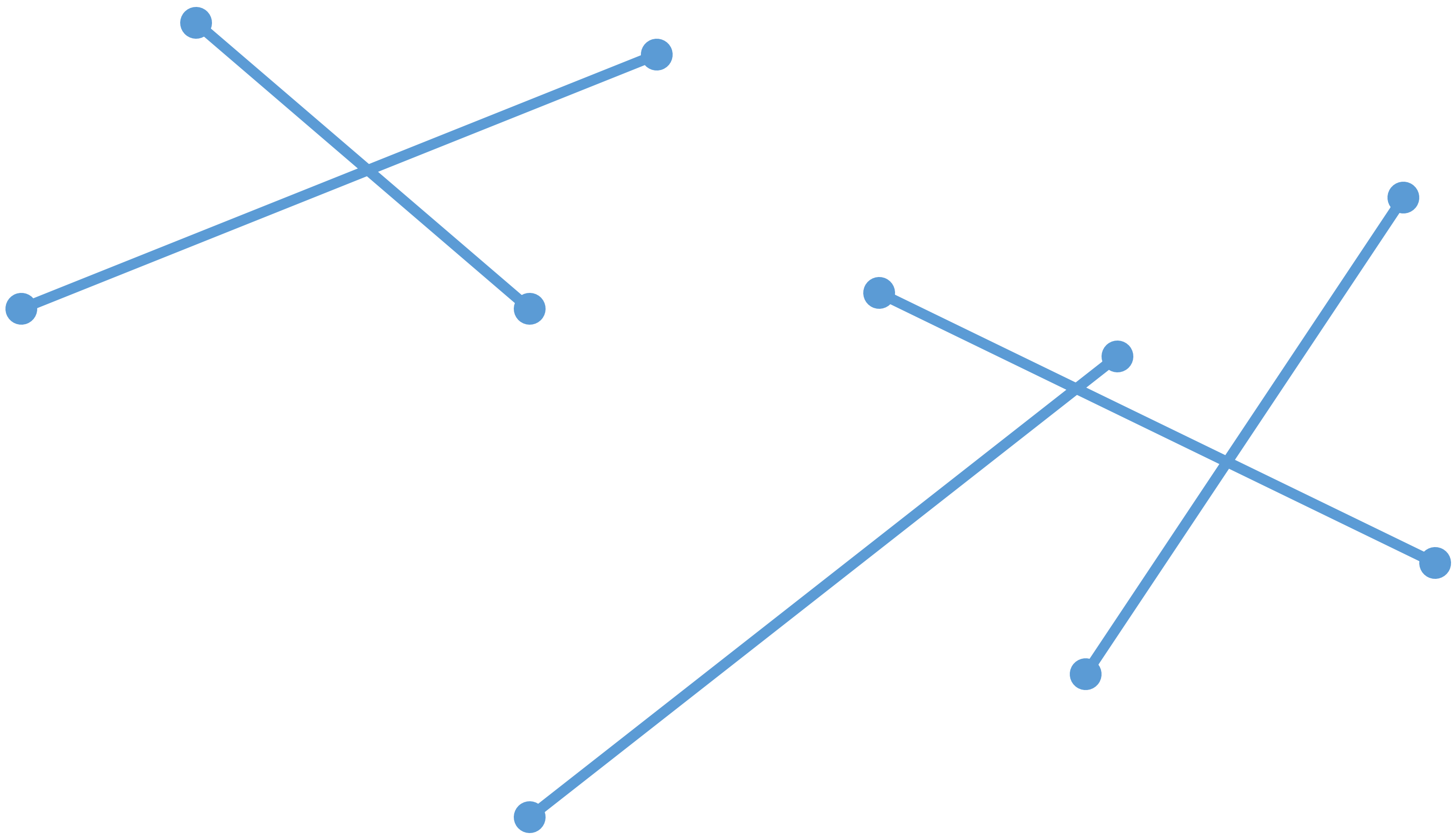
INTERSECTION OF LINE SEGMENTS

- SWEEP-LINE STATUS
  - The sweep-line status is a list of the currently comparable segments, ordered by the relation in X.
- THE SWEEP-LINE STATUS DATA STRUCTURE L IS USED TO STORE THE ORDERING OF THE CURRENTLY COMPARABLE SEGMENTS. BECAUSE THE SET OF CURRENTLY COMPARABLE SEGMENTS CHANGES, THE DATA STRUCTURE FOR L MUST SUPPORT THESE OPERATIONS:
  1. INSERT(s, L). Insert segment s into the total order in L.
  2. DELETE(s, L). Delete segment s from L.
  3. LEFT(s, L). Return the name of the segment immediately left of s in the ordering in L.
  4. RIGHT(s, L). Return the name of the segment immediately right of s in the ordering in L.
- THESE OPERATIONS CAN BE PERFORMED IN $O(\log N)$ TIME (OR BETTER).

Intersection of line segments

- Event-point schedule
  - As the sweep-line is swept from top to bottom, the set of the currently comparable segments and/or their ordering by the relation of X changes at a finite set of y values; those are known as events.
- The events for this problem are segment endpoints and segment intersections. The event-point schedule data structure E is used to store events prior to their processing. For E the following operations are needed:
  1. MIN(E). Determine the smallest element in E (based on y), return it, and delete it.
  2. INSERT(p, E). Insert abscissa p, representing an event, into E.
  3. MEMBER(p, E). Determine if abscissa p is a member of E.
- The priority queue data structure can perform all of these operations in O(log N) time.

Sweep Algorithm



What Data Structures Should We Use?

- Self-balancing trees (AVL, Red-black)

Red–black tree	
Type	tree
Invented	1972
Invented by	Rudolf Bayer

Time complexity in big O notation

Algorithm	Average	Worst case
Space	$O(n)$	$O(n)$
Search	$O(\log n)$[1]	$O(\log n)$[1]
Insert	$O(\log n)$[1]	$O(\log n)$[1]
Delete	$O(\log n)$[1]	$O(\log n)$[1]

What Data Structures Should We Use?

- Priority Queue

Operation	Binary[6]	Leftist	Binomial[6]	Fibonacci[6][2]	Pairing[7]	Brodal[8][a]	Rank-pairing[10]	Strict Fibonacci[11]	2-3 heap
find-min	$\Theta(1)$	$\Theta(1)$	$\Theta(\log n)$	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$	?
delete-min	$\Theta(\log n)$	$\Theta(\log n)$	$\Theta(\log n)$	$O(\log n)$[b]	$O(\log n)$[b]	$O(\log n)$	$O(\log n)$[b]	$O(\log n)$	$O(\log n)$[b]
insert	$O(\log n)$	$\Theta(\log n)$	$\Theta(1)$[b]	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$	$O(\log n)$[b]
decrease-key	$\Theta(\log n)$	$\Theta(n)$	$\Theta(\log n)$	$\Theta(1)$[b]	$o(\log n)$[b][c]	$\Theta(1)$	$\Theta(1)$[b]	$\Theta(1)$	$\Theta(1)$
merge	$\Theta(n)$	$\Theta(\log n)$	$O(\log n)$[d]	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$	?

Intersection of line segment

- Analysis of Shamos-Hoey algorithm
  - Preprocessing: $O(N \log N)$; sort of endpoints.
  - Query: $O((N + K) \log N)$;
    - each of 2N endpoints and K intersections are inserted into E, an O(log N) operation.
  - Storage: $O(N + K)$; at most $2N$ endpoints and K intersections are stored in E.
- Comments
  - As given, assumption that no segments of S are horizontal.
  - As given, assumption that no three (or more) segments meet at a point.
  - Care must be taken with intersections at segment end points.
  - Query time of $O((N + K) \log N)$ is suboptimum; an optimum $O(N \log N + K)$ algorithm exists but is quite difficult.