|| Computational Geometry

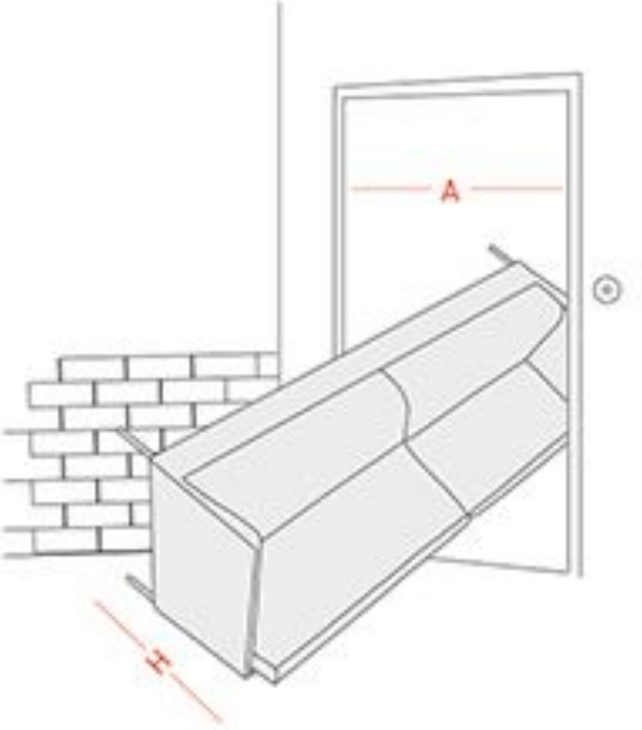
Workshop in Competitive Programming – 234900

Agenda

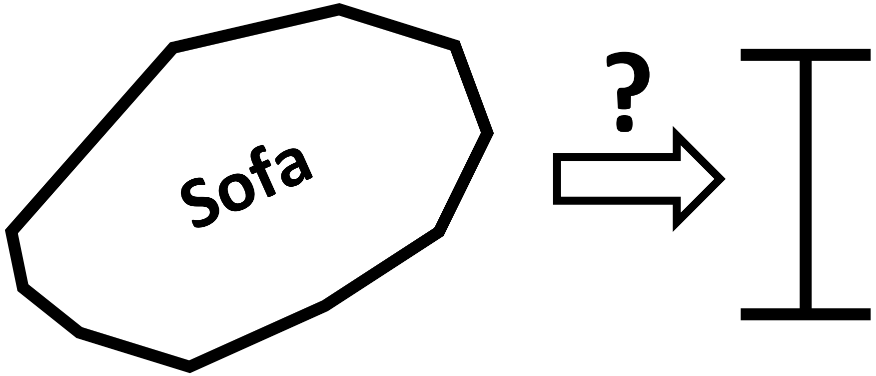
- Rotating Calipers (Shamos’s algorithm)
- The Sweep Line Paradigm

Rotating Calipers

Or “Will the sofa pass the door?” problem



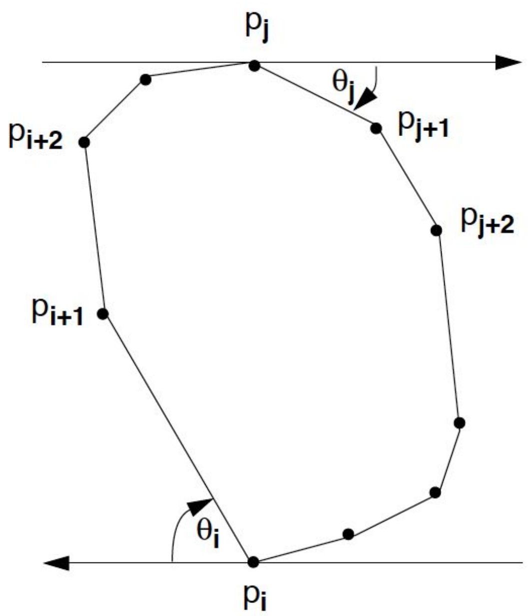
- One simplification - 2D sofa
- One complication - Any convex polygon



Some definitions…

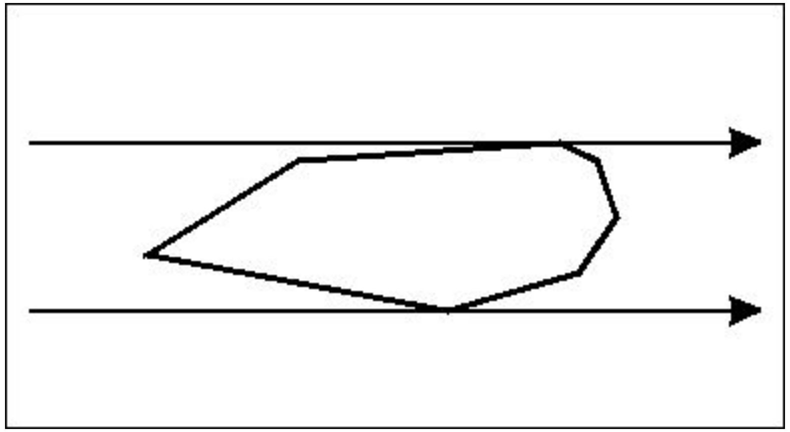
- Notation: $P = \{p_1, p_2, \dots, p_n\}$ denotes a convex polygon with n vertices in clockwise order.
- Definition: A line l is a **line of support** of P if the interior of P lies completely to one side of l. (Assume l is directed such that P lies to the right of l.)

  - Definition: A pair of vertices p_i, p_j is an **antipodal** pair if it admits parallel lines of support
    - Width – minimum distance between parallel lines of support of P
    - Diameter – maximum distance between parallel lines of support of P



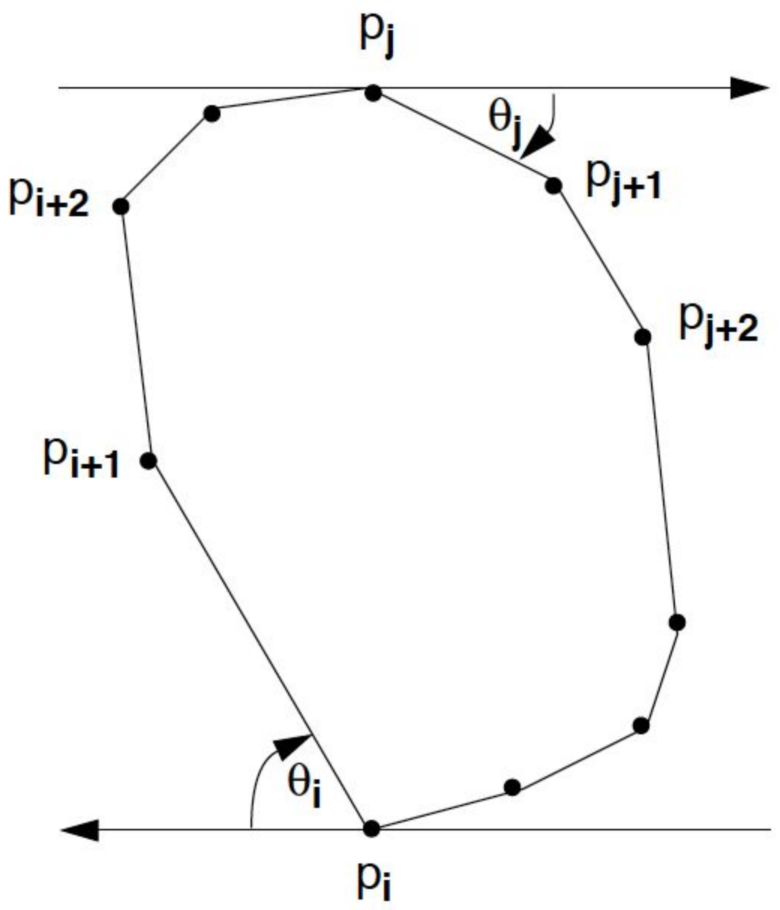
Rotating Calipers – Shamo’s Alg.

- Problem: Compute the width or diameter of a convex polygon $P = \{p_1, p_2, \dots, p_n\}$
- Shamo’s Alg. generates all O(n) antipodal pairs of vertices.
- The procedure resembles rotating calipers around the polygon.



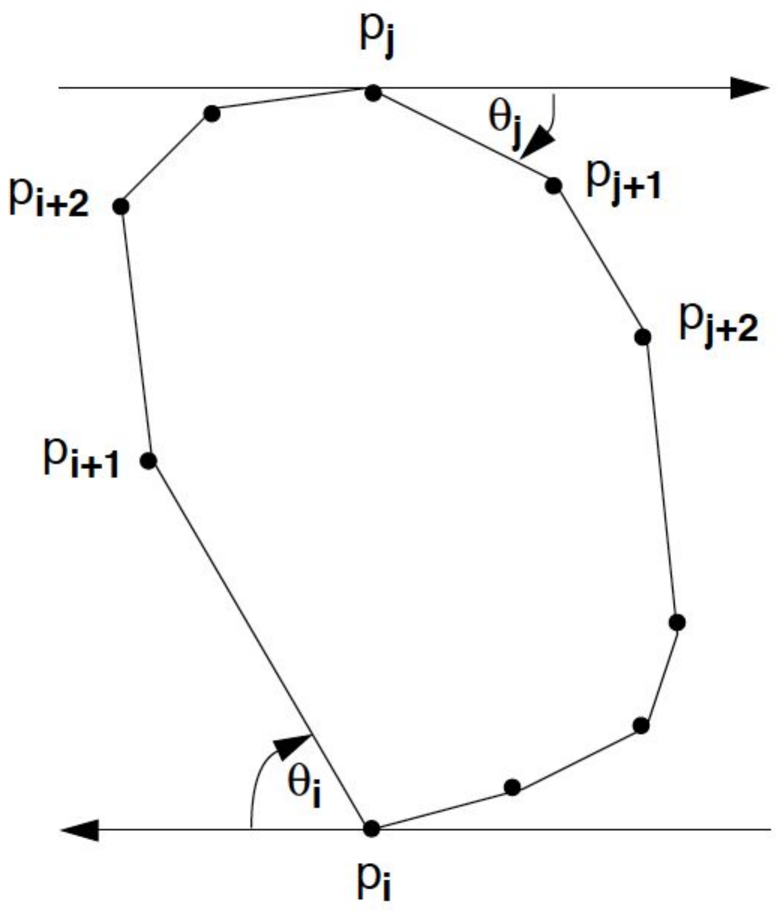
Shamo's Algorithm

- Initialization:
  - Choose a direction (such as the x-axis)
  - Find the two antipodal vertices p_i and p_j (Can be done in $O(n)$)



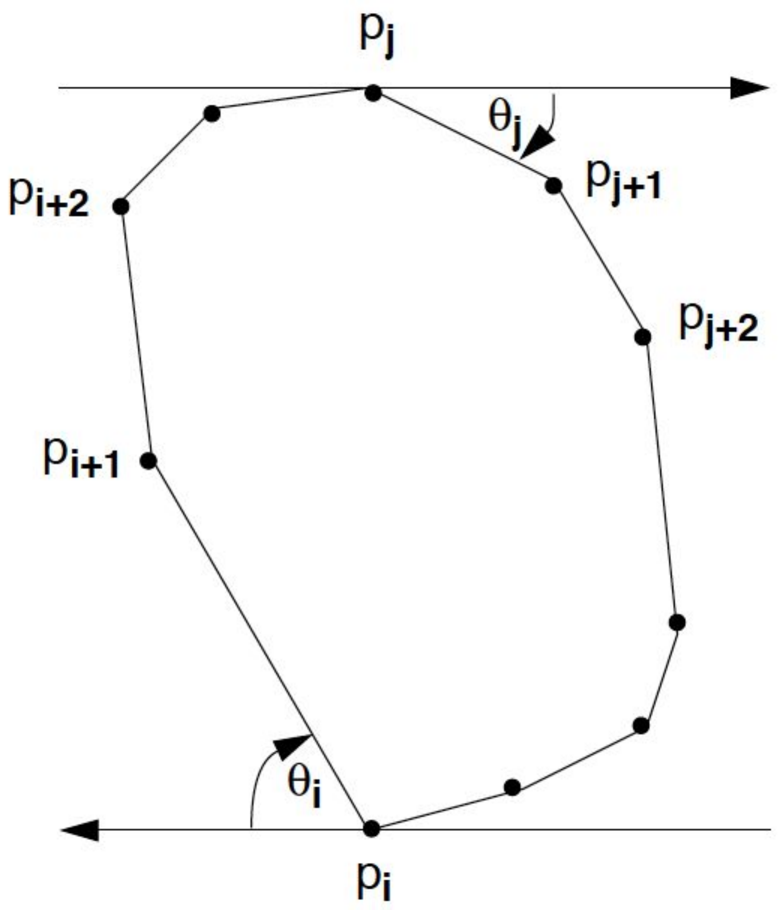
Shamo's Algorithm (cont.)

- Generation of the next antipodal pair:
  - Consider θ_i and θ_j. Let angle $\theta_j < \theta_i$. Then we "rotate" the lines of support by an angle θ_j, and p_{j+1}, p_i becomes the next antipodal pair.
  - This process is continued until we come full circle to the starting position.
  - (In the event that $\theta_j = \theta_i$ three new antipodal pairs are generated.)



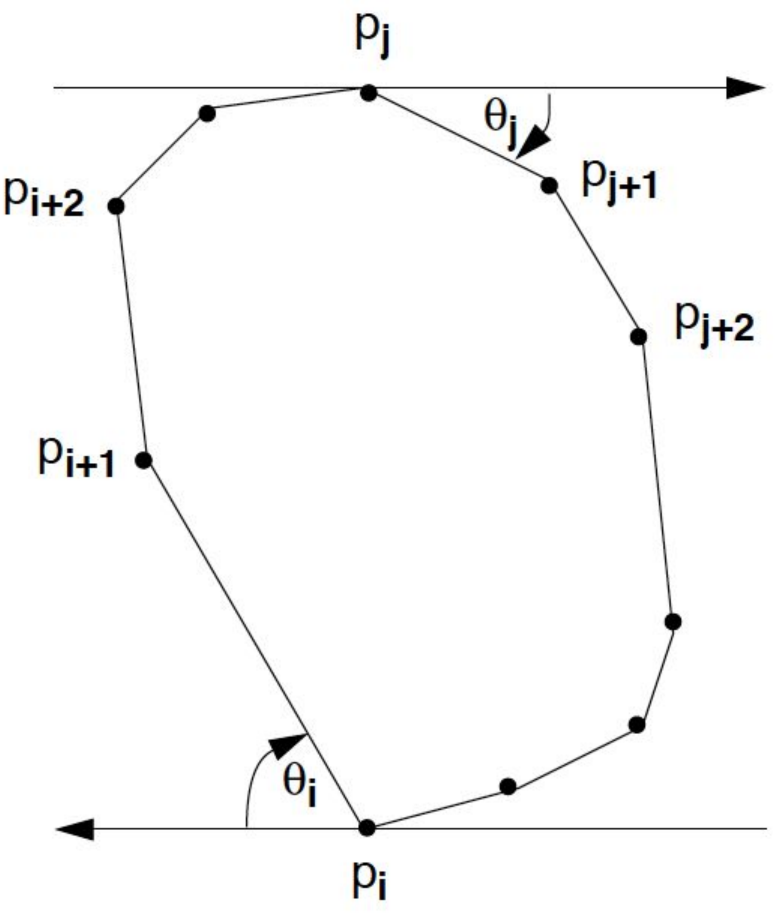
Shamo’s Algorithm (cont.)

- How to find the width?
- Width = minimum distance between parallel support lines
- There are infinitly many support lines
- Check only pairs of support lines where one support line touchs the polygon.
  - Exactly what the algorithm generates!



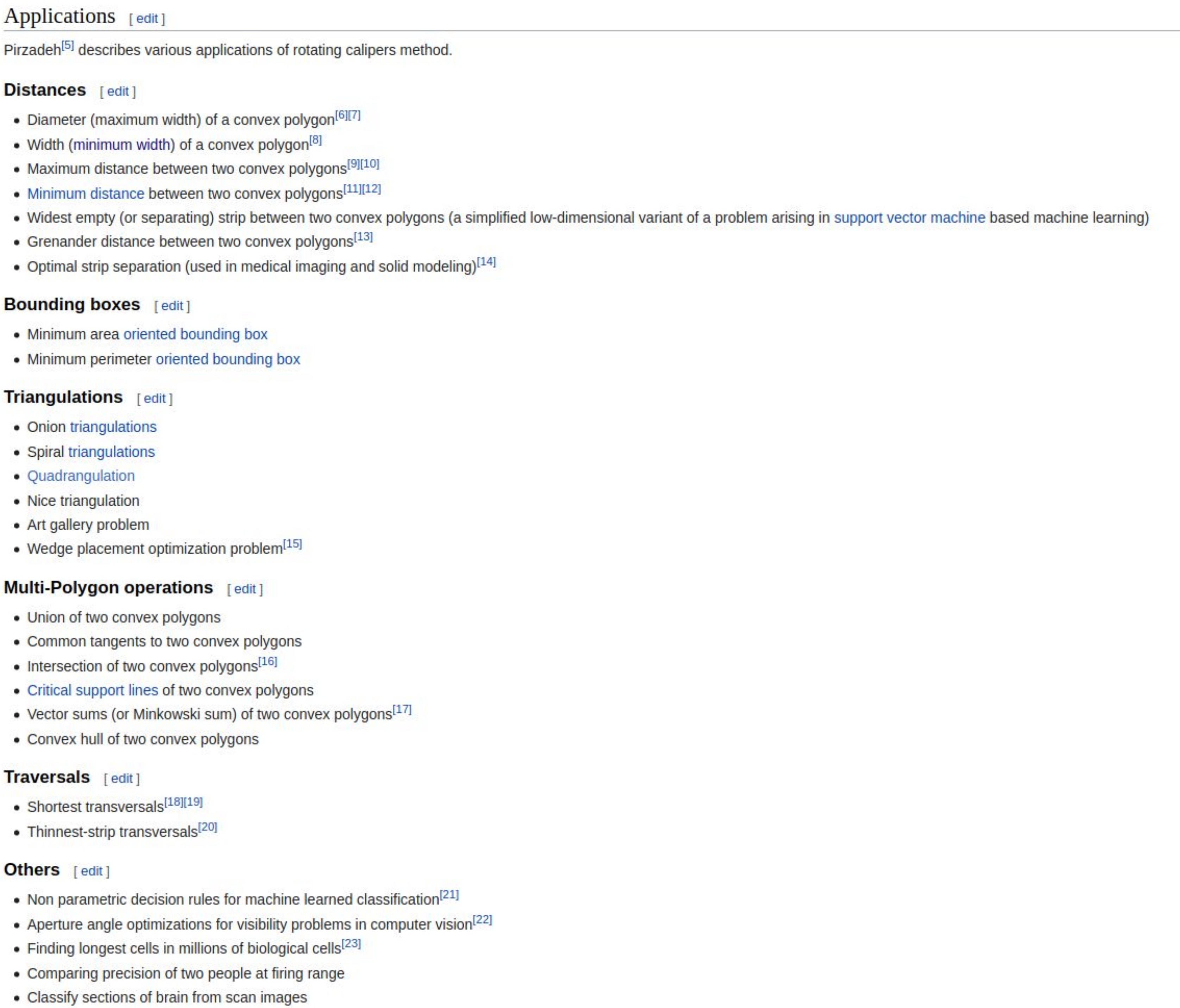
Shamo's Algorithm (cont.)

- How to find the diameter?
- Width = maximum distance between antipodal points
- There are infinitly many support lines
- Check all the antipodal points and find the maximum distance



Myriad of other applications!

[wikipedia]

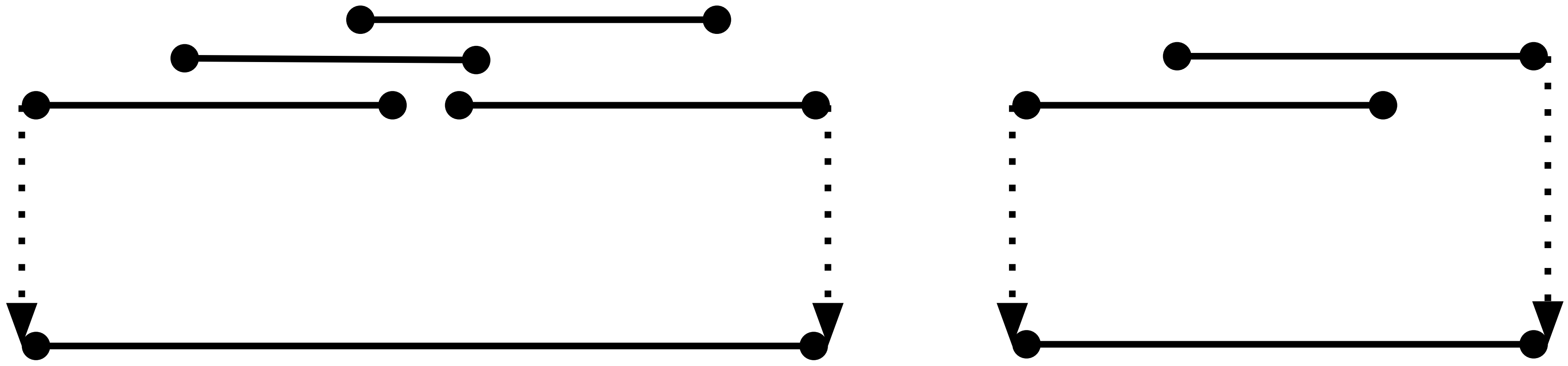


The Sweep Line Paradigm

Sweeping: Example #1

Sweeping: Example #1

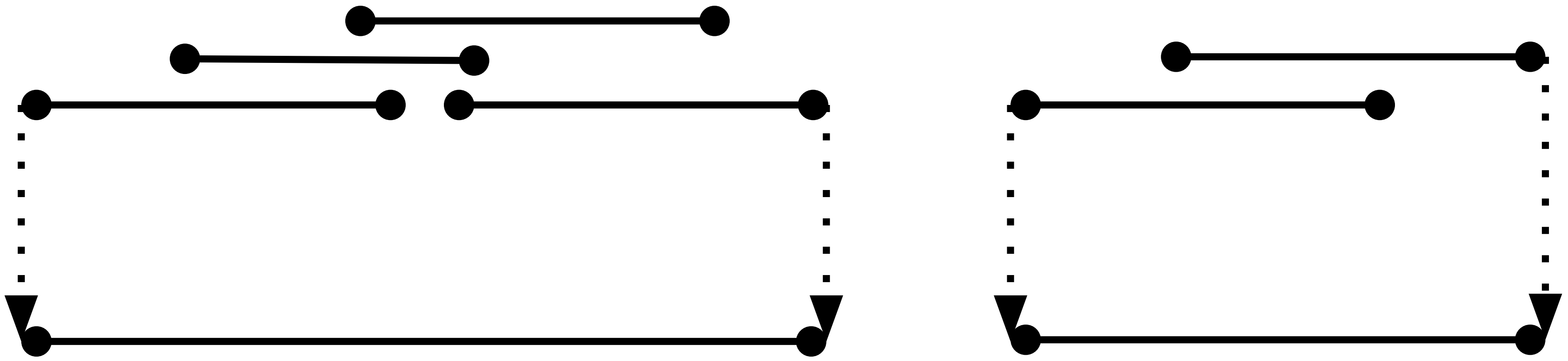
- Given a set of 1D segments, what is the union of them all?



- Solution: Sort all the points, and count the number of ‘active’ segments.

Sweeping: Example #1

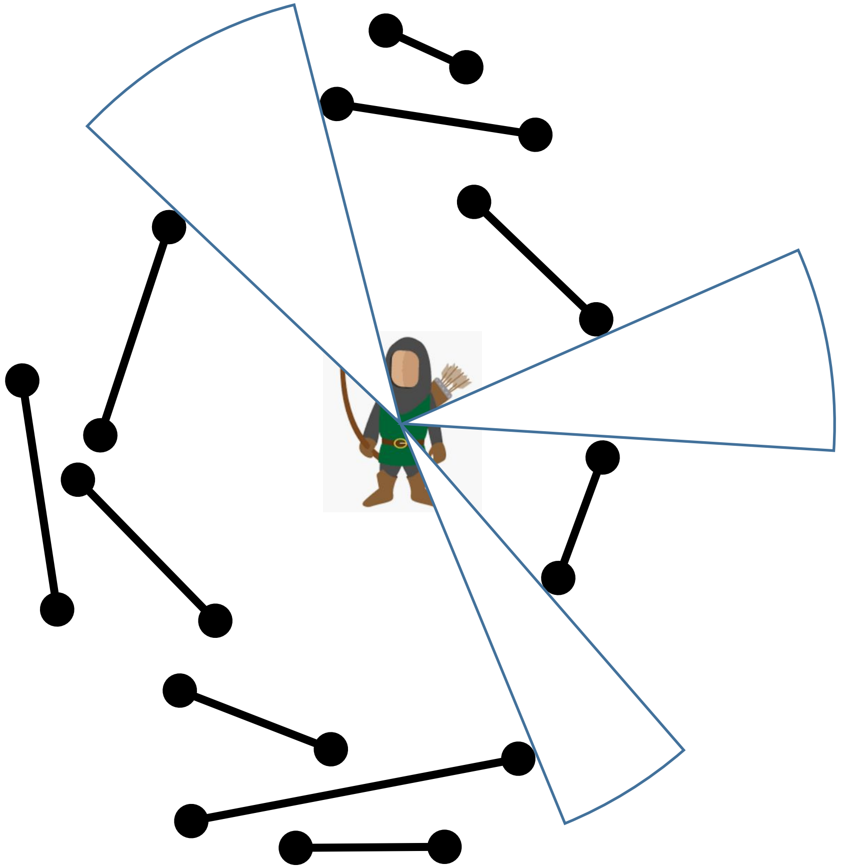
- We have traversed a discrete set of **Events**, in a certain **Order**, while maintaining some **Status** of the algorithm.
- **Events [What data was processed]:** start of segment, end of segment.
- **Order [In what order we traverse the events]:** From left to right
- **Status [Additional information maintained]:** number of active segments.
- Complexity: $O(n \log n)$



Sweeping: Example #2

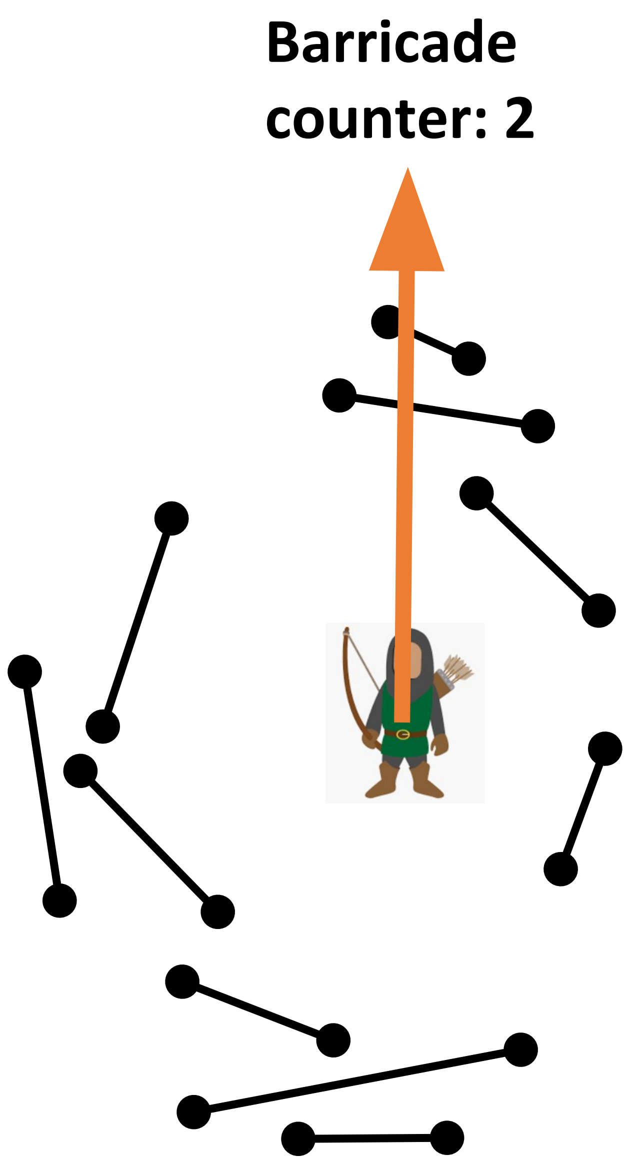
Sweeping: Example #2

- An archer is surrounded by a set of barricades. What are his lines of sight?
- **Order:** Scan the segments by angle.
- **Status:** Number of 'active' barricades.
  - Init in $O(n)$.
- **Events:**
  - Start of a segment: increase number of barricades.
  - End of a segment: decrease number of barricades.
- Report angles with 0 barricades.



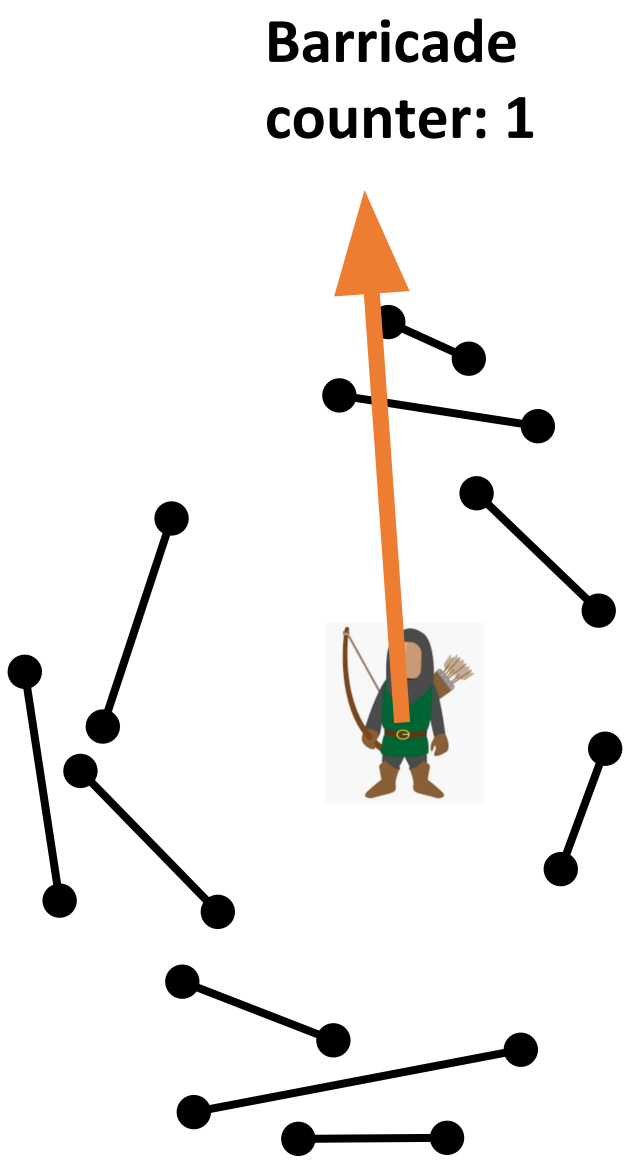
Sweeping: Example #2

- An archer is surrounded by a set of barricades. What are his lines of sight?
- **Order:** Scan the segments by angle.
- **Status:** Number of 'active' barricades.
  - Init in $O(n)$.
- **Events:**
  - Start of a segment: increase number of barricades.
  - End of a segment: decrease number of barricades.
- Report angles with 0 barricades.



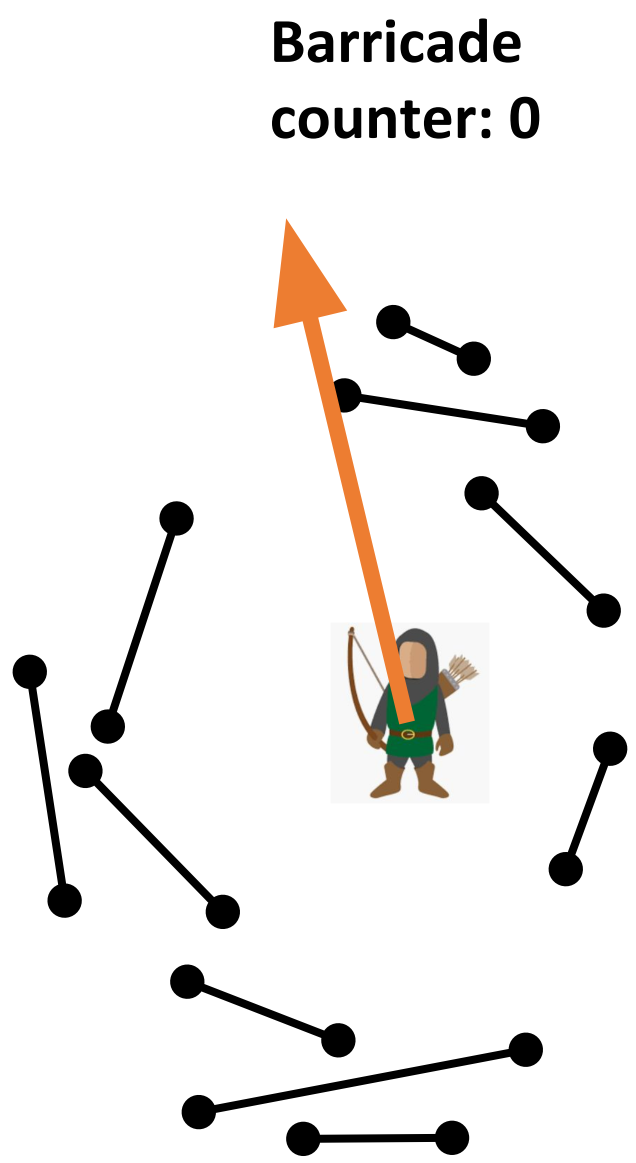
Sweeping: Example #2

- An archer is surrounded by a set of barricades. What are his lines of sight?
- **Order:** Scan the segments by angle.
- **Status:** Number of 'active' barricades.
  - Init in $O(n)$.
- **Events:**
  - Start of a segment: increase number of barricades.
  - End of a segment: decrease number of barricades.
- Report angles with 0 barricades.



Sweeping: Example #2

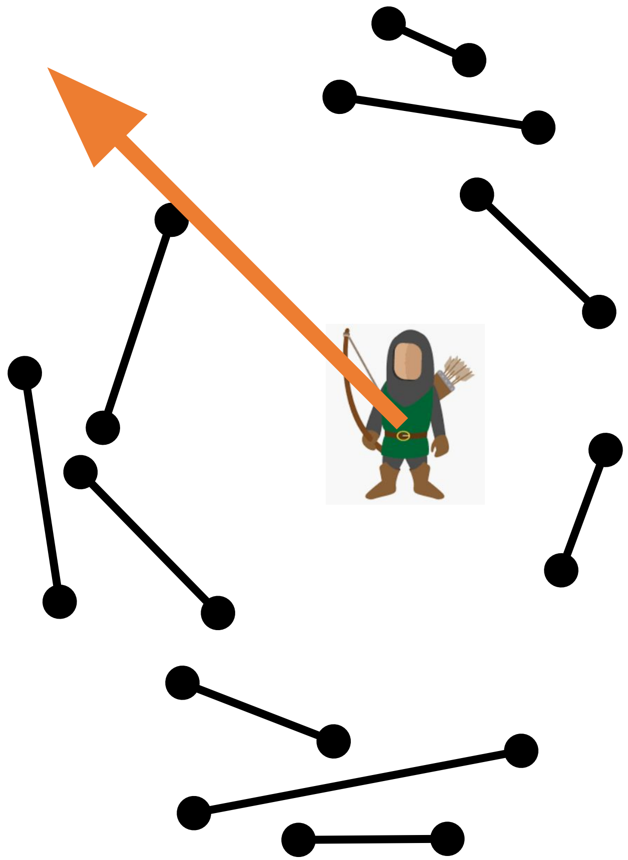
- An archer is surrounded by a set of barricades. What are his lines of sight?
- **Order:** Scan the segments by angle.
- **Status:** Number of ‘active’ barricades.
  - Init in $O(n)$.
- **Events:**
  - Start of a segment: increase number of barricades.
  - End of a segment: decrease number of barricades.
- Report angles with 0 barricades.



Sweeping: Example #2

Barricade counter: 1

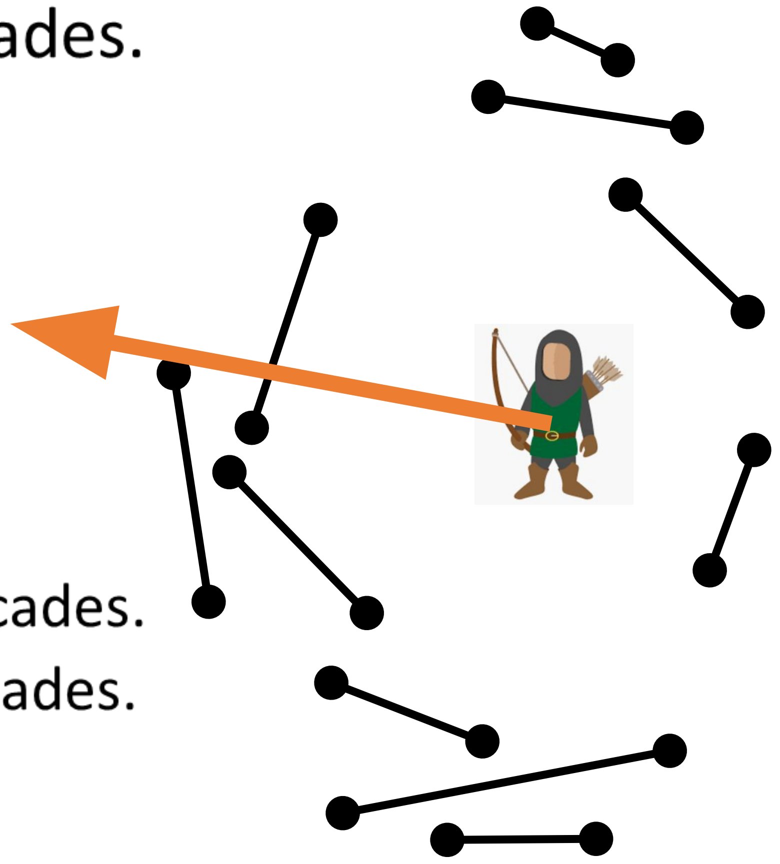
- An archer is surrounded by a set of barricades. What are his lines of sight?
- **Order:** Scan the segments by angle.
- **Status:** Number of 'active' barricades.
  - Init in $O(n)$.
- **Events:**
  - Start of a segment: increase number of barricades.
  - End of a segment: decrease number of barricades.
- Report angles with 0 barricades.



Sweeping: Example #2

Barricade counter: 2

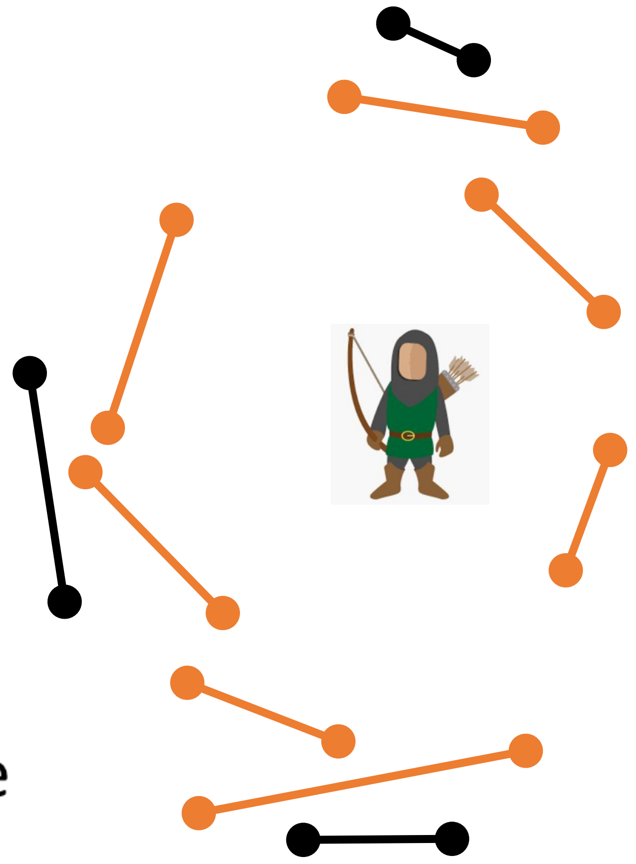
- An archer is surrounded by a set of barricades. What are his lines of sight?
- **Order:** Scan the segments by angle.
- **Status:** Number of 'active' barricades.
  - Init in $O(n)$.
- **Events:**
  - Start of a segment: increase number of barricades.
  - End of a segment: decrease number of barricades.
- Report angles with 0 barricades.
- Complexity: $O(n \log n)$



Sweeping: Example #3

Sweeping: Example #3

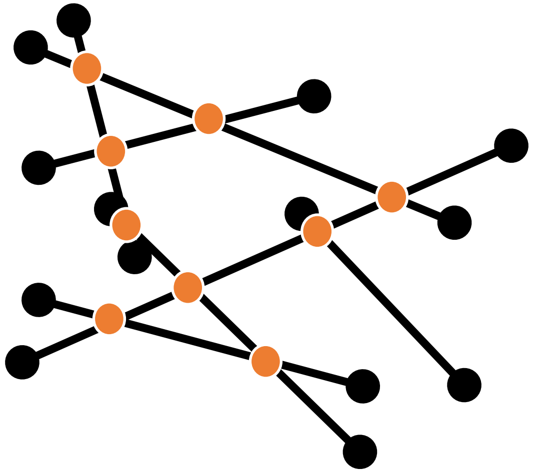
- An archer is surrounded by a set of barricades. Which barricades are visible to him?
- **Order:** Scan the segments by angle.
- **Status:** Set of active barricades, sorted by the distance from the archer.
- **Events:**
  - Start of a segment: Add segment to the status DS.
  - End of a segment: Remove segment from the status DS.
- Report all segments which was closest at some point.
- Complexity: $O(n \log n)$



Sweeping: Segment Intersection

Sweeping: Segment Intersection

- Given a set of n segments, report all intersection points.
- Naïve algorithm: Check all segment pairs, $O(n^2)$.
- Sweep line algorithm:
- **Order:** scan from left to right.
- **Status:** segments intersecting the sweep line.
  (Ordered by intersection point).
- **Events:** Segment start, Segment end and **Segments intersection.**
- **Check intersection only between adjacent segments in the status DS.**

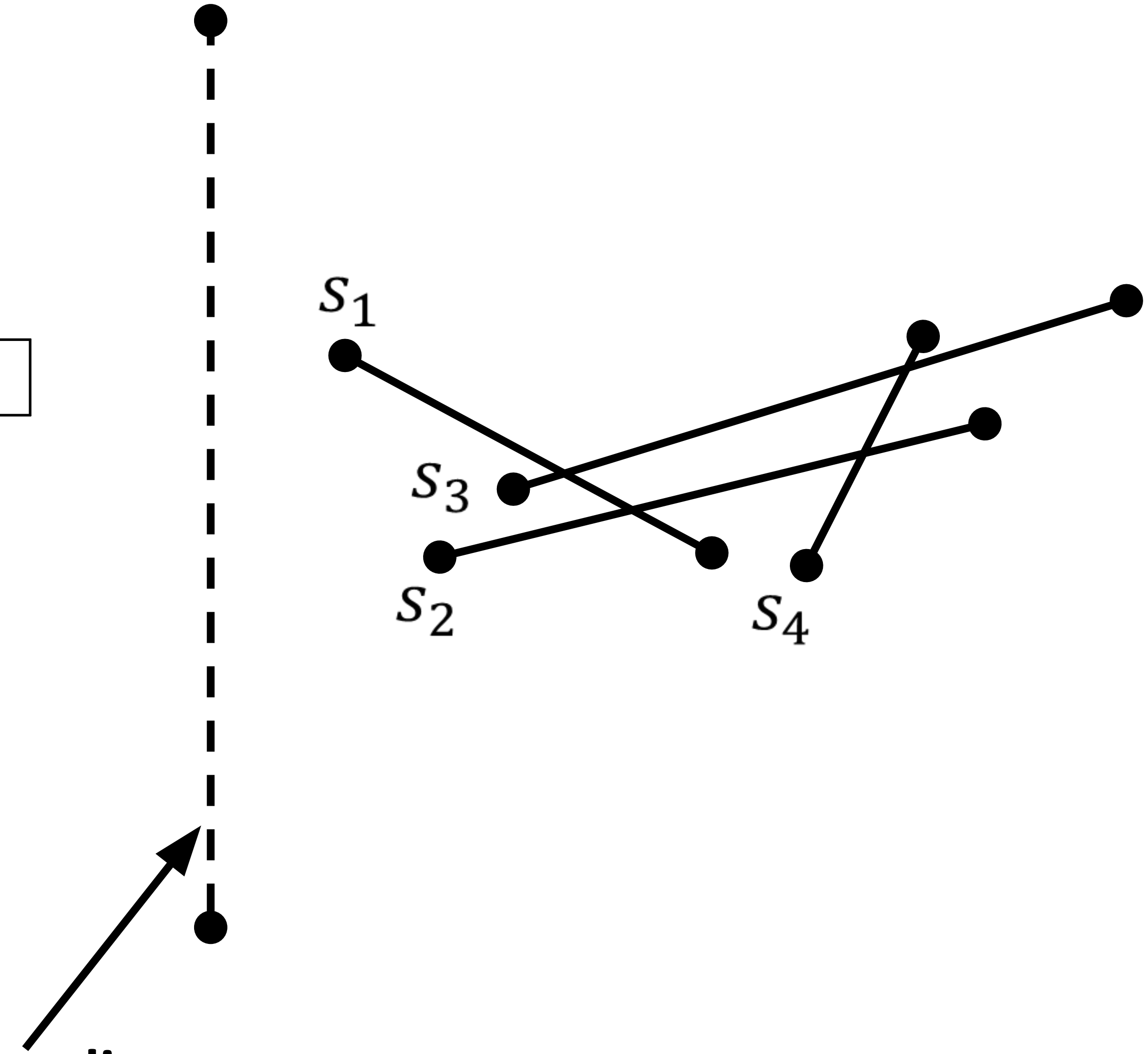


Dynamic events!

Handle event: $None$

Events

Status

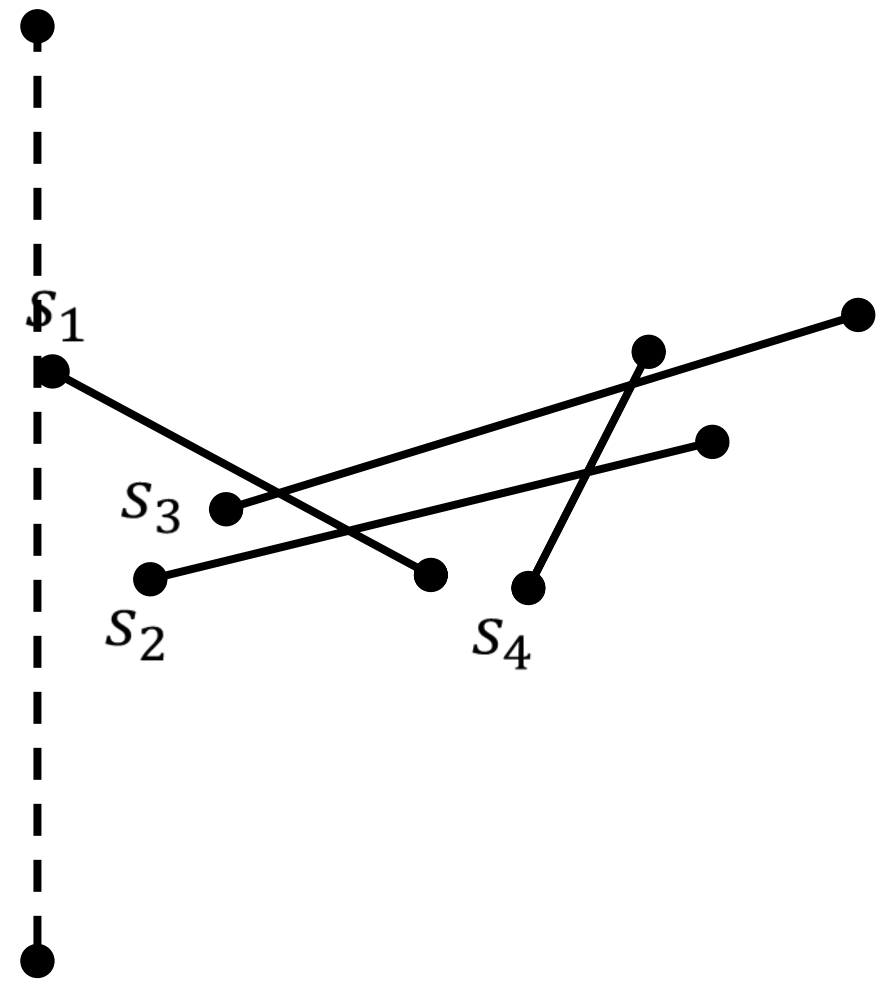


Sweep line

Handle event: $Start(S_1)$

Events

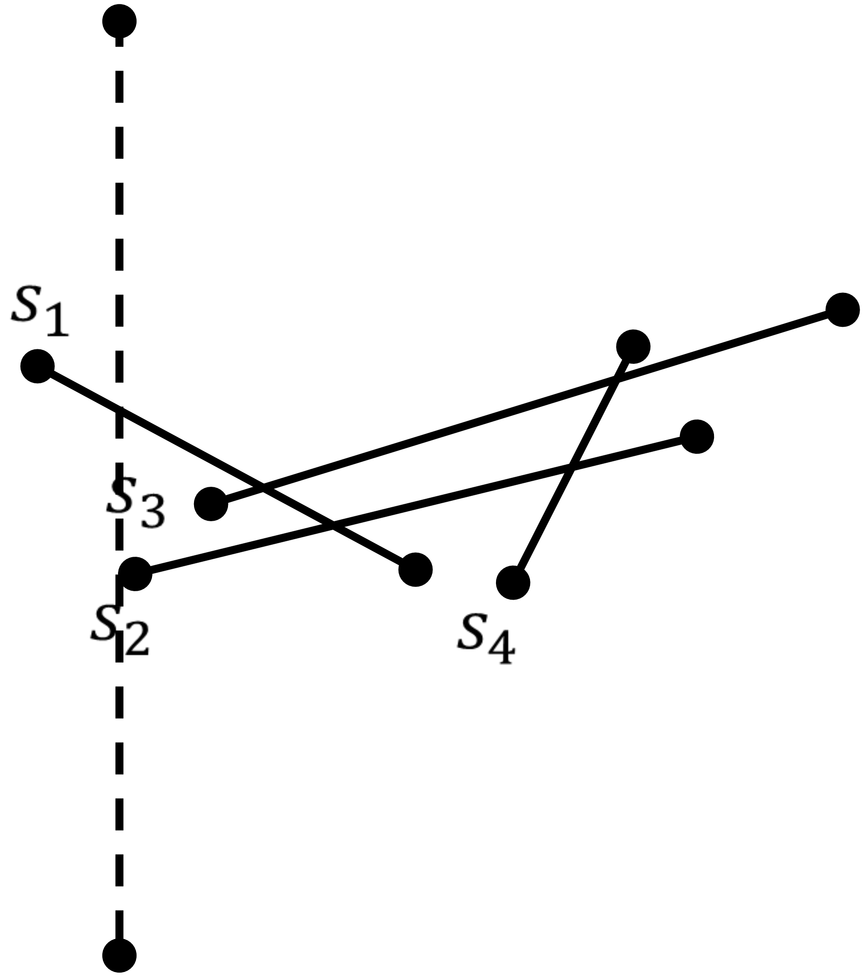
Status



Handle event: $Start(S_2)$

Events

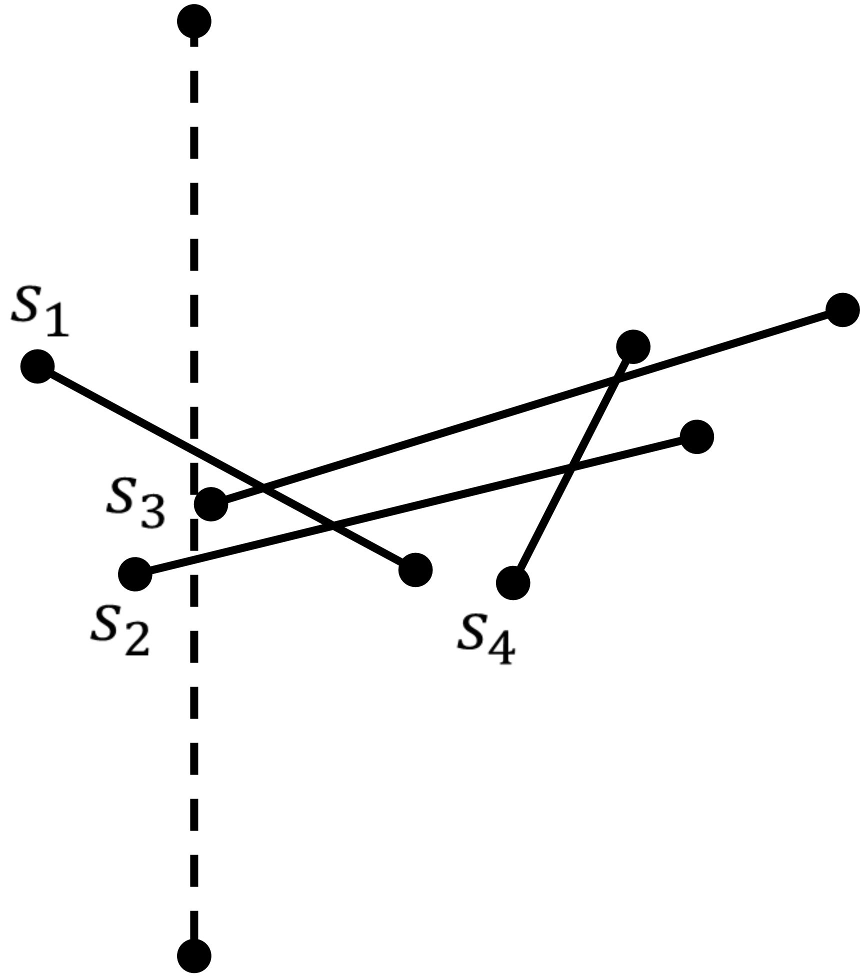
Status



Handle event: $Start(S_3)$

<u>Events</u>

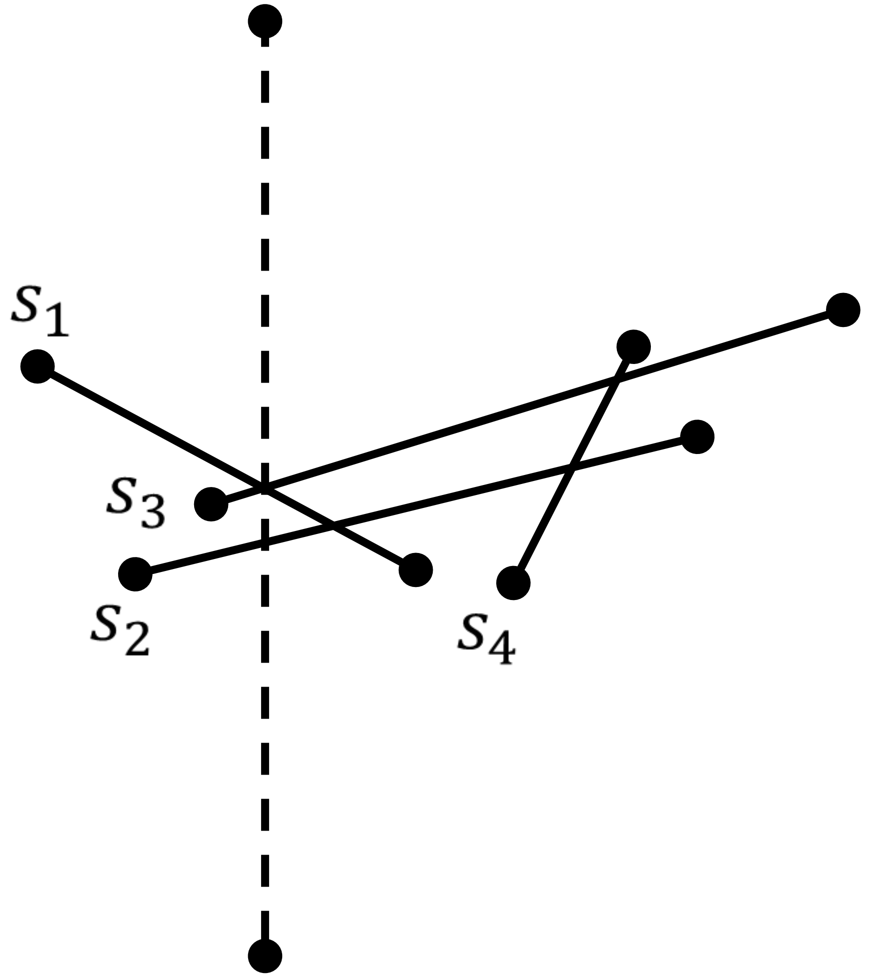
<u>Status</u>



Handle event: $Intersection(S_1, S_3)$

Events

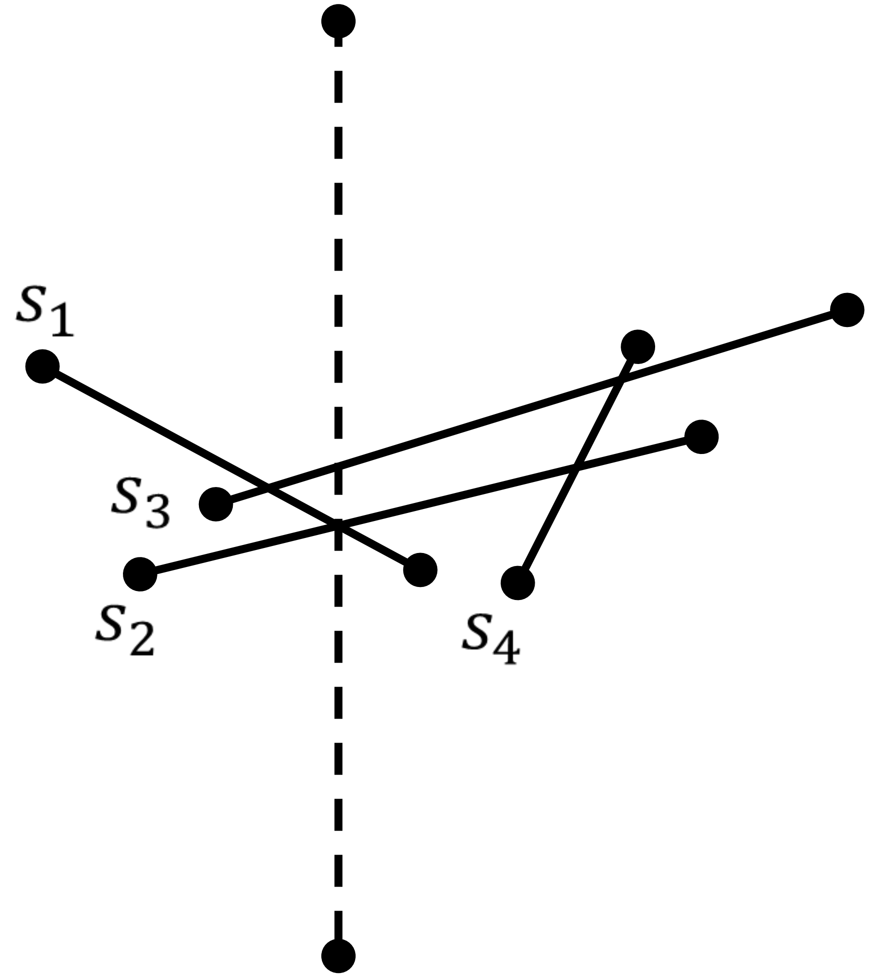
Status



Handle event: $Intersection(S_1, S_2)$

Events

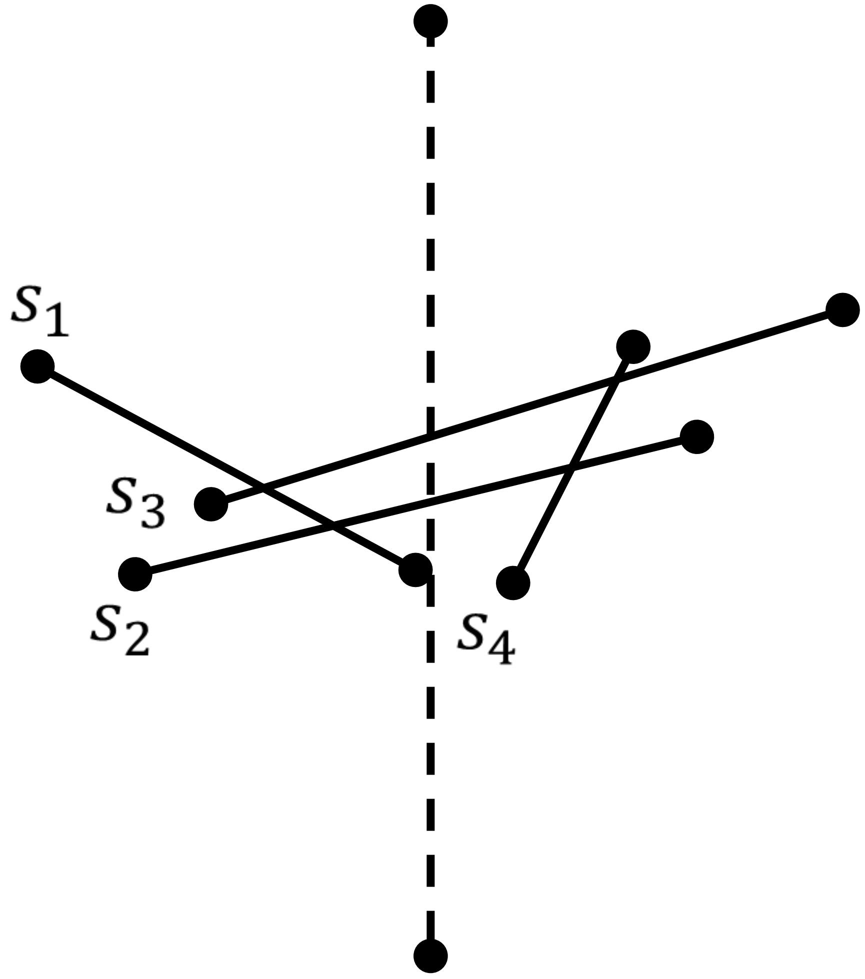
Status



Handle event: $End(S_1)$

Events

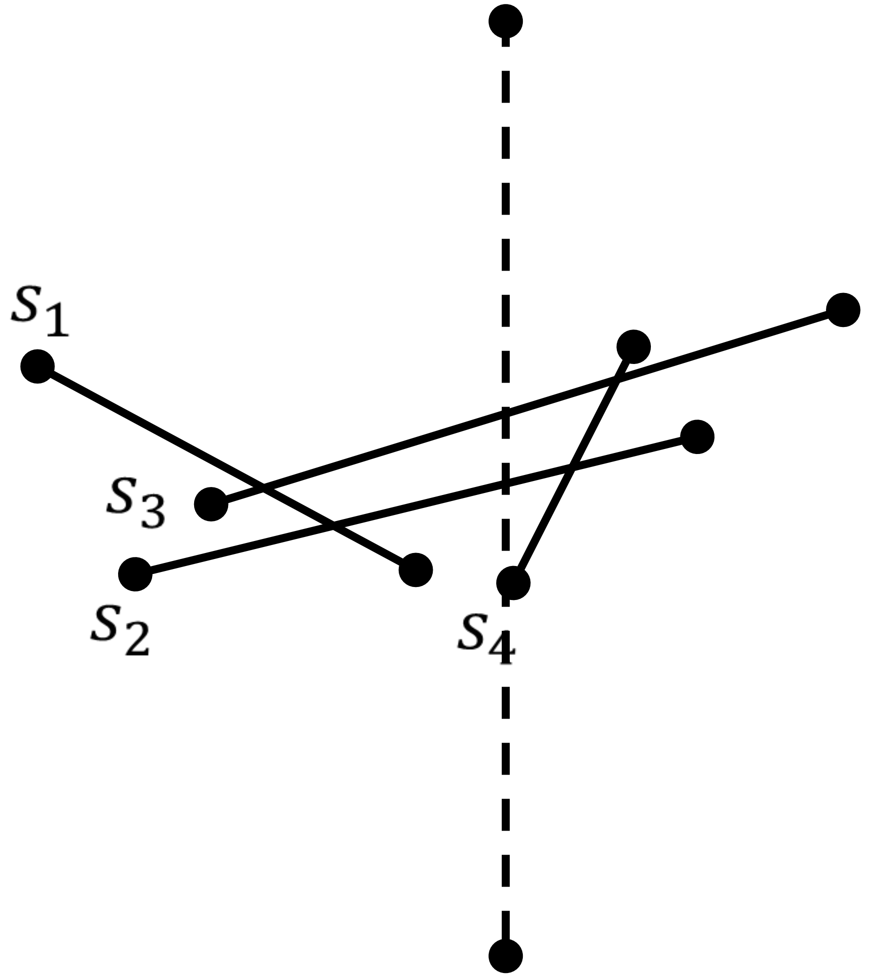
Status



Handle event: $Start(S_4)$

Events

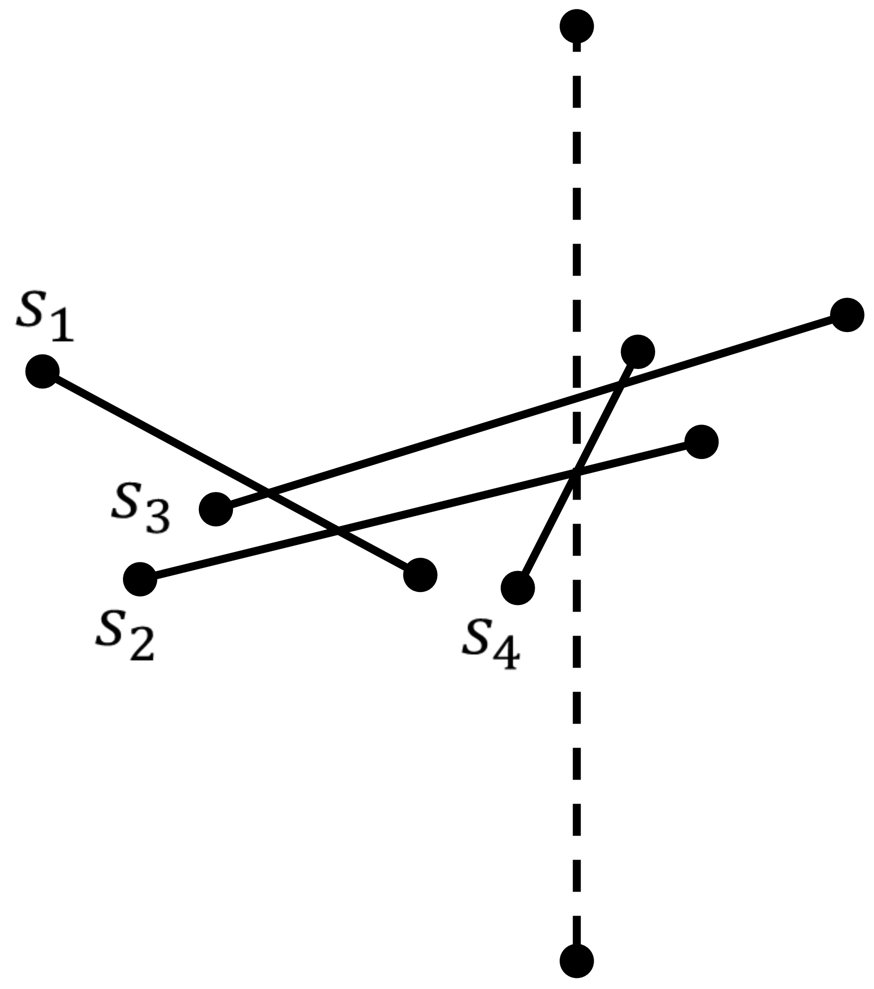
Status



Handle event: $Intersection(S_2, S_4)$

Events

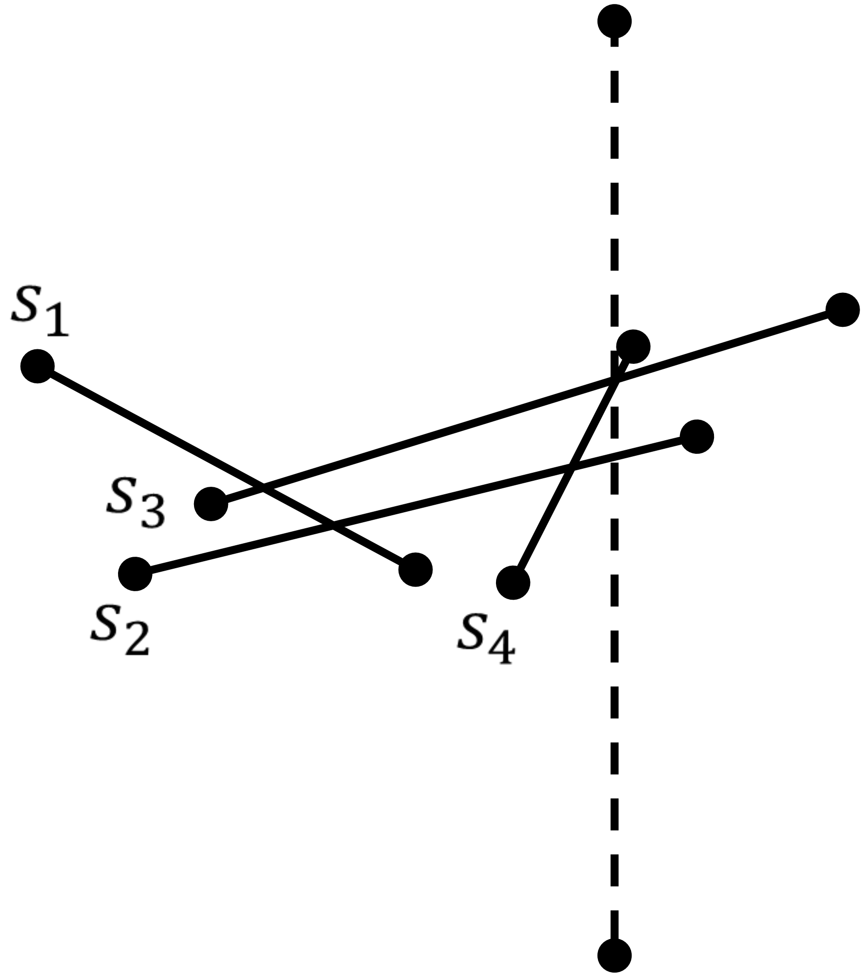
Status



Handle event: $Intersection(S_3, S_4)$

Events

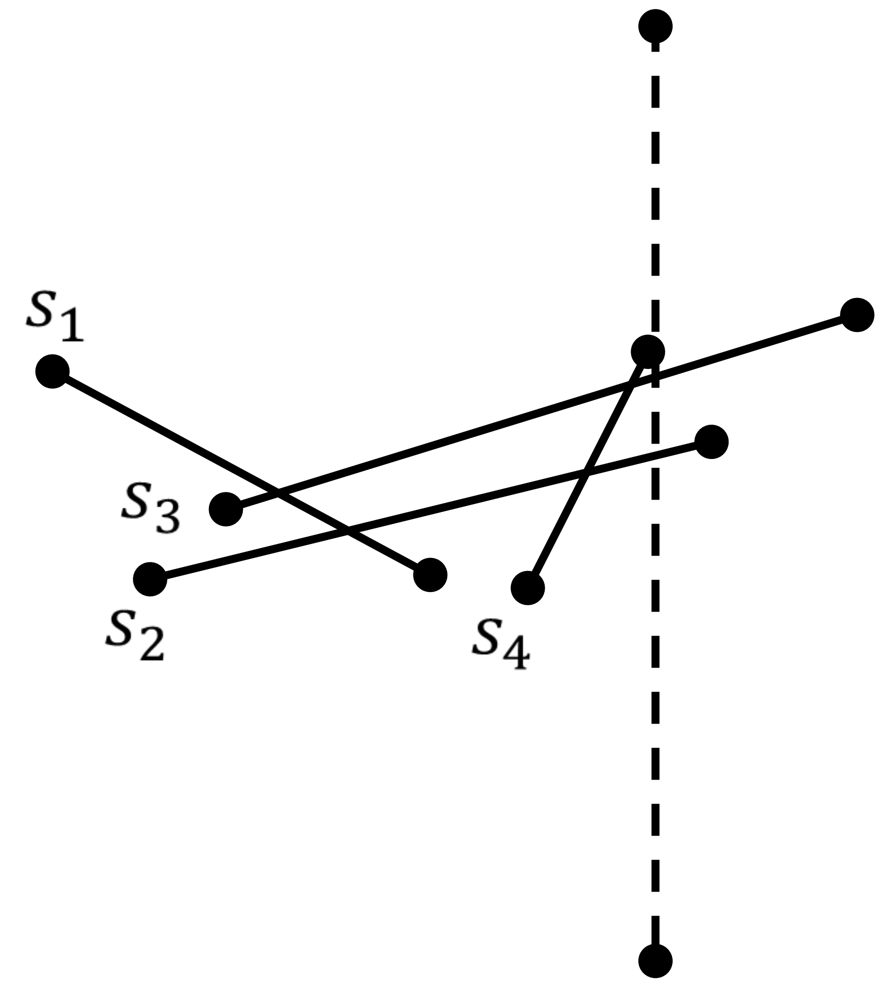
Status



Handle event: $End(S_4)$

Events

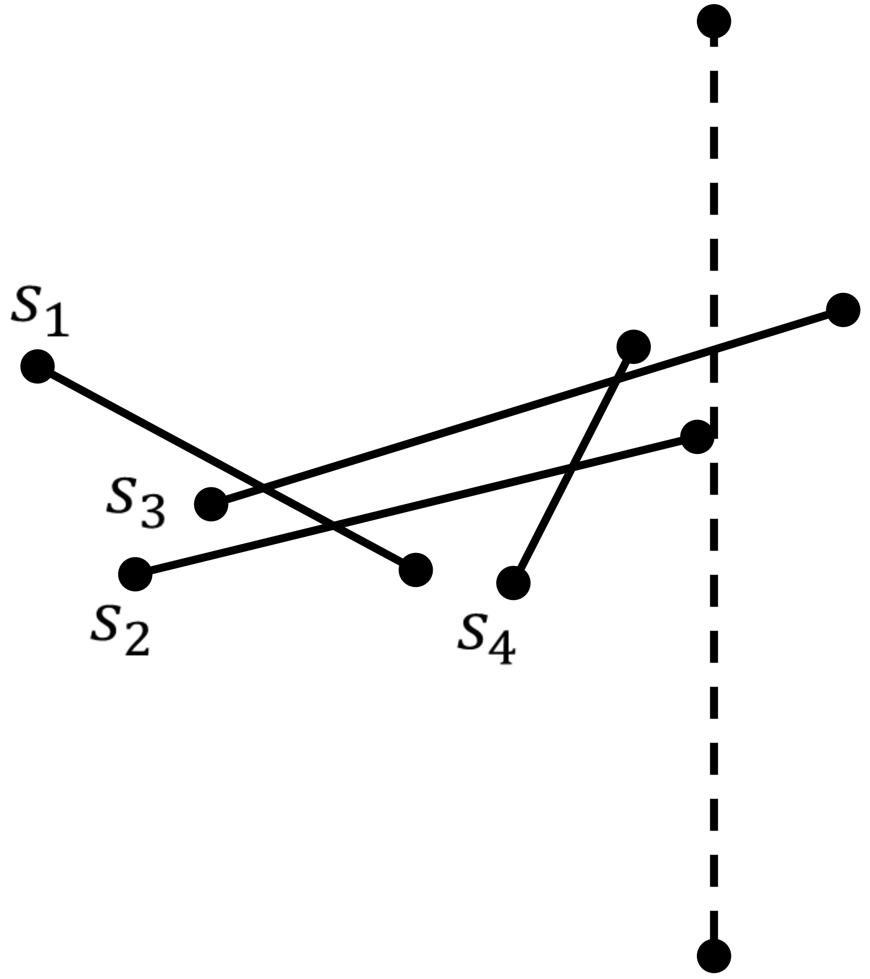
Status



Handle event: $End(S_2)$

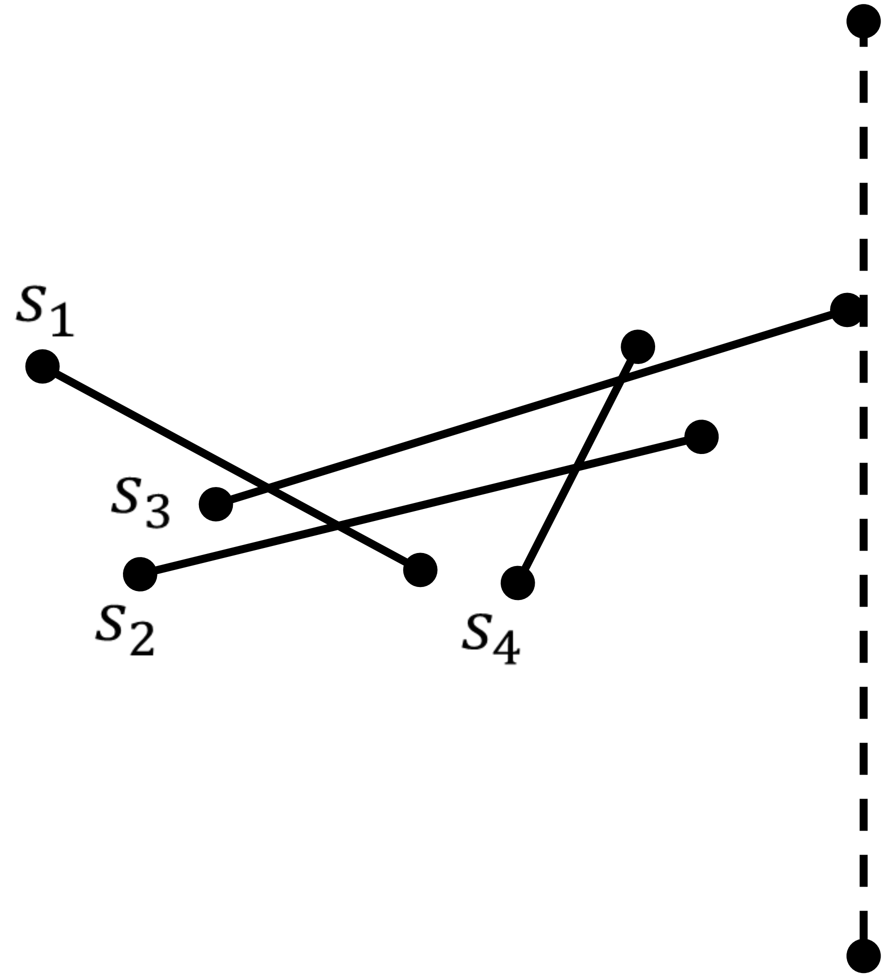
Events

Status



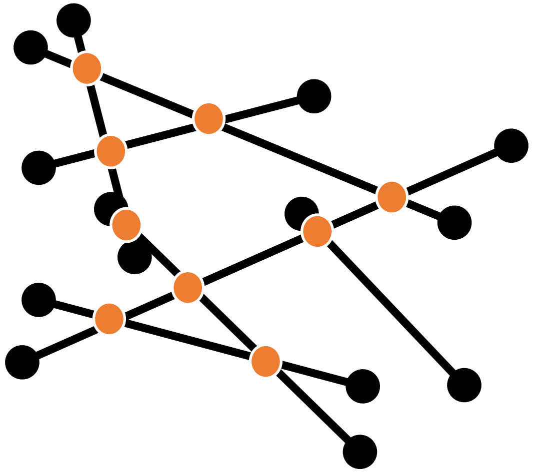
Handle event: $End(S_3)$

Events	**Status**



Sweeping: Segment Intersection

- Given a set of n segments, report all intersection points.
- Naïve algorithm: Check all segment pairs, $O(n^2)$.
- Sweep line algorithm:
- **Order:** scan from left to right.
- **Status:** segments intersecting the sweep line.
  (Ordered by intersection point).
- **Events:** Segment start, Segment end and **Segments intersection.**
- **Check intersection only between adjacent segments in the status DS.**
- Complexity: $O(n \log n)$

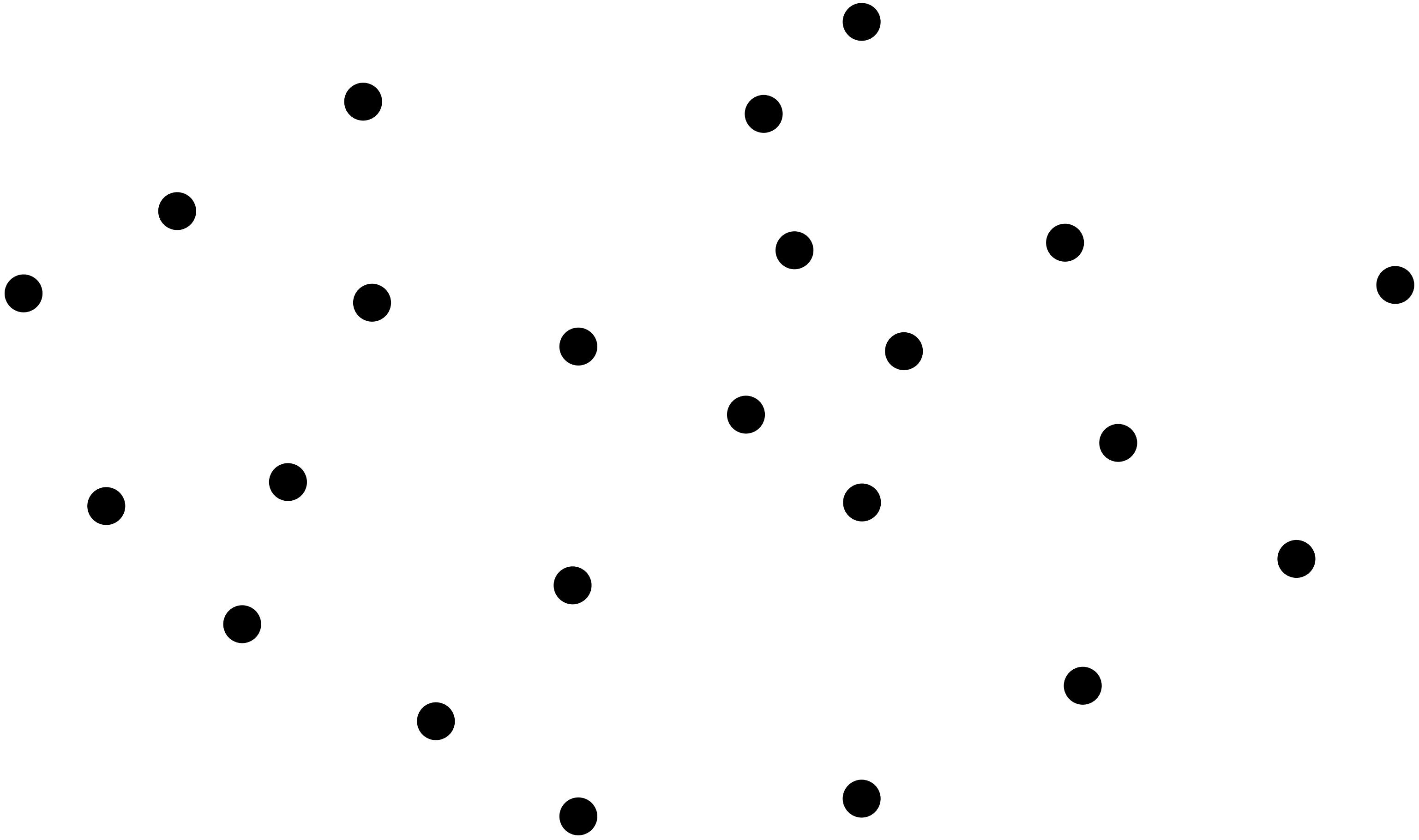


Dynamic events!

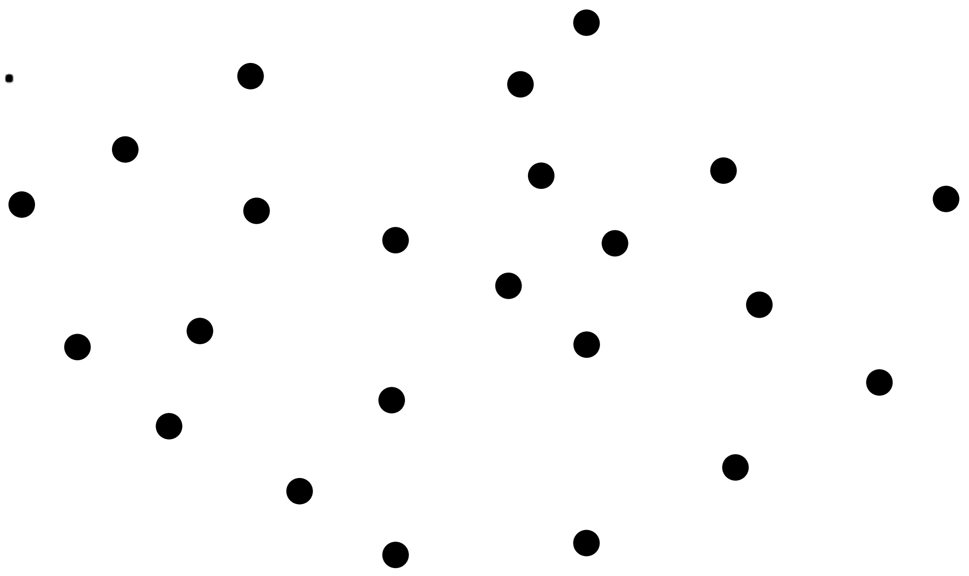
Sweeping: Minimal Distance Pair

Sweeping: Minimal Distance Pair

- Problem: Find the closest pair of points.

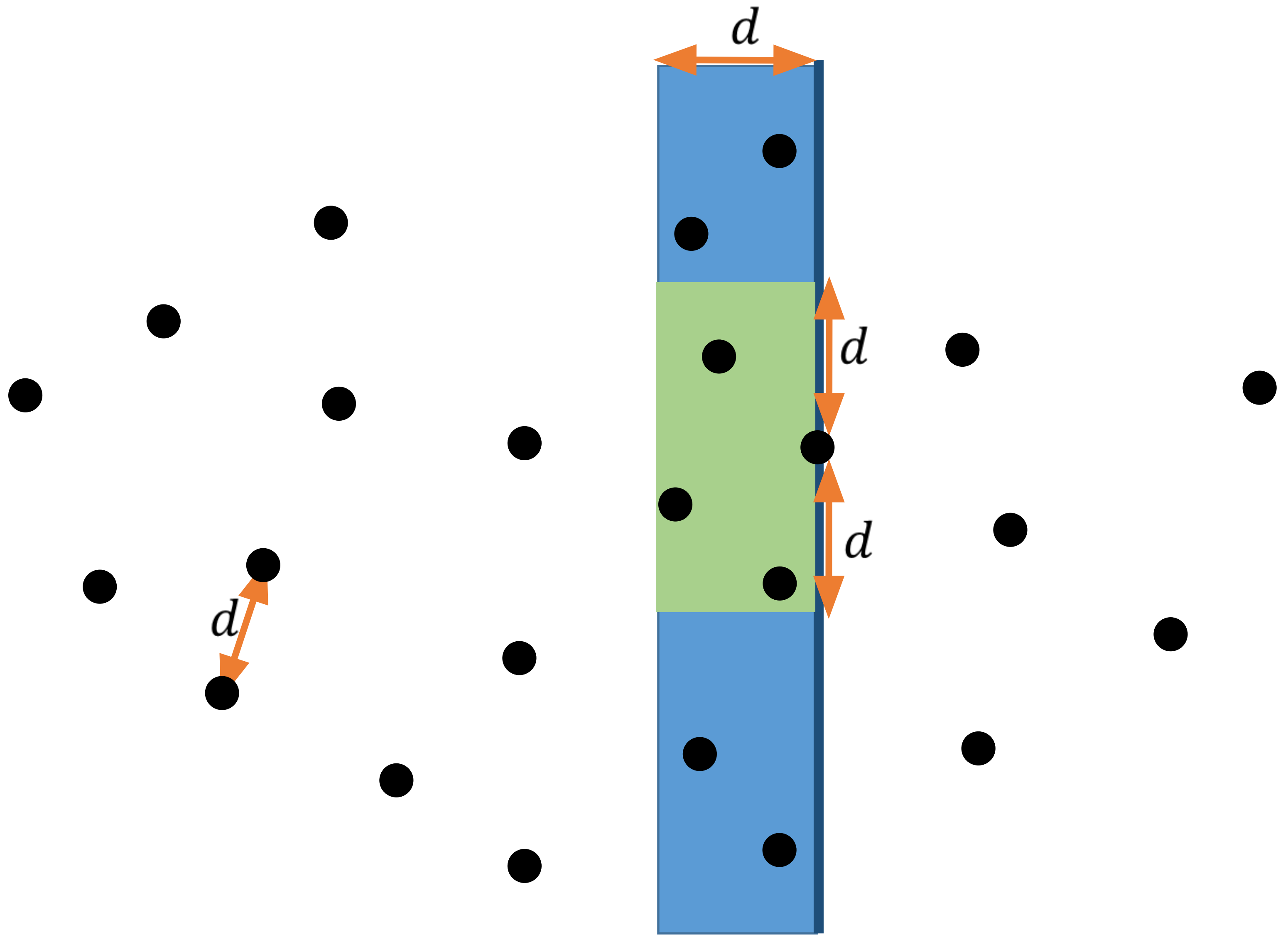


Sweeping: Minimal Distance Pair



- Problem: Find the closest pair of points.
- Naïve algorithm: Check all pairs, $O(n^2)$
- Sweeping idea:
- **Events:** All the points
- **Order:** left to right
- **Status:** minimal distance seen so far, d.
  And two BSTs ofall the points in a strip of width d**.**
  one sorted by the y coordinate,
  and another sorted by the x coordinate.

Sweeping: Minimal Distance Pair



Sweeping: Minimal Distance Pair

- Handle event:
- Compare the distance with the relevant points.
  - Using the sorted by y tree, lower bound and upper bound are useful.
- Update d if needed.
- Remove from both trees the points that now are not part of the strip.
  - Using the sorted by x tree.