The Sweep Line Paradigm

Computational Geometry - Recitation 2

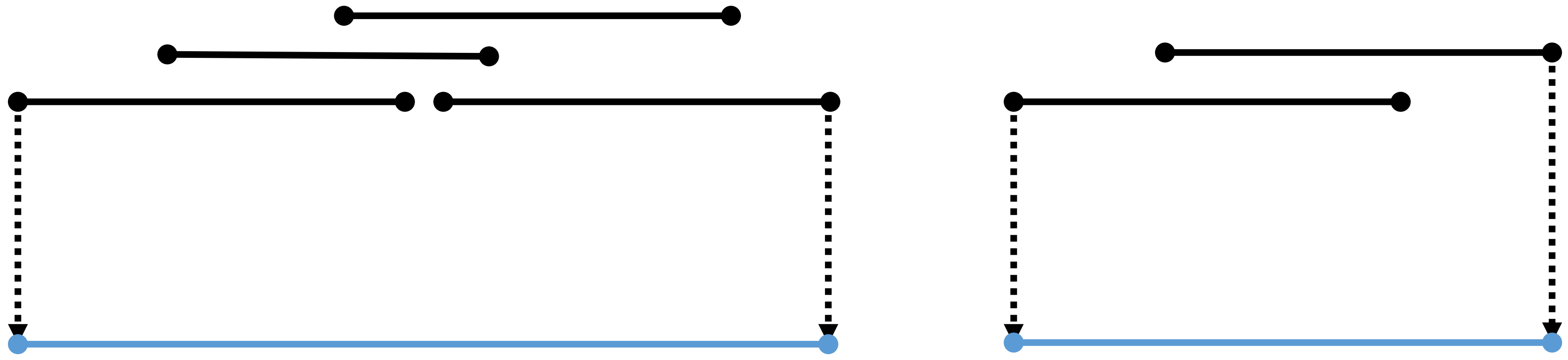
Agenda

- Toy examples
- Line segment intersection
- Area of union of rectangles
- Minimal distance pair

Sweeping: Example #1

Sweeping: Example #1

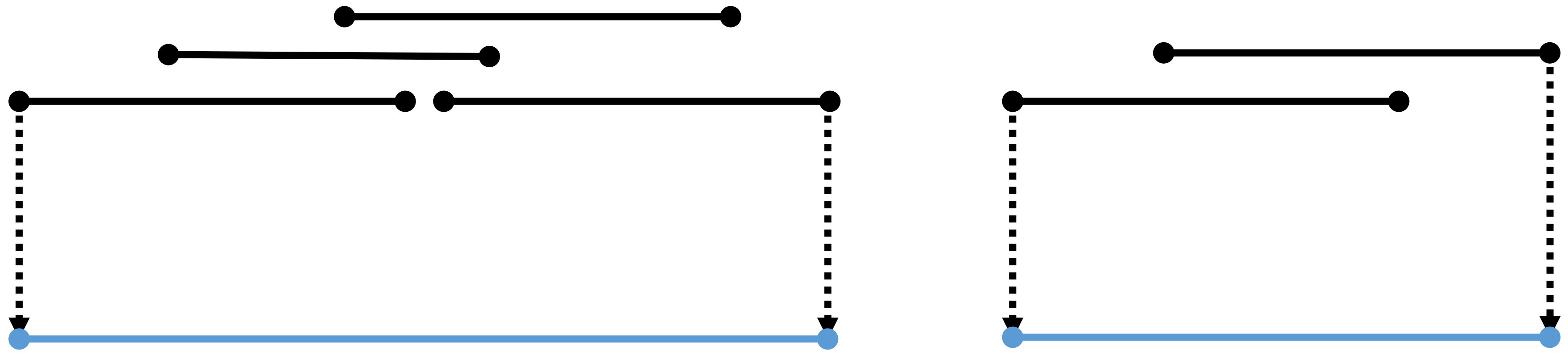
- Given a set of 1D segments, what is the union of them all?



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

Sweeping: Example #1

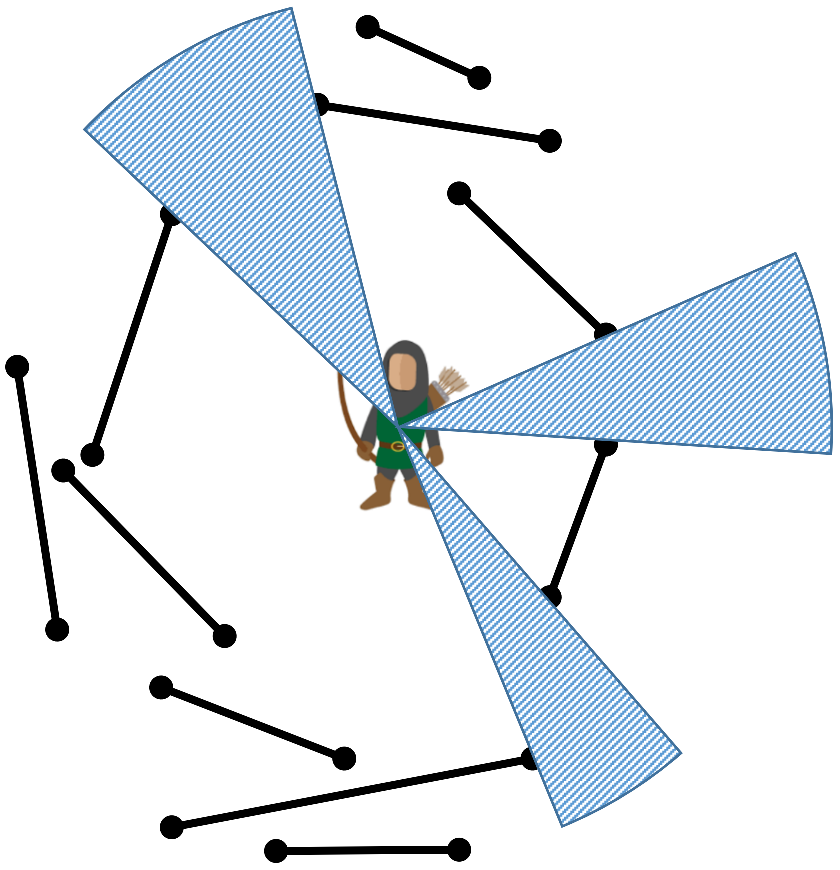
- We have traversed a discrete set of **Events**, in a certain **Order**, while maintaining some **Status** of the algorithm.
- **Events [What data was processed]:** start of segment, end of segment.
- **Order [In what order we traverse the events]:** From left to right
- **Status [Additional information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



Sweeping: Example #2

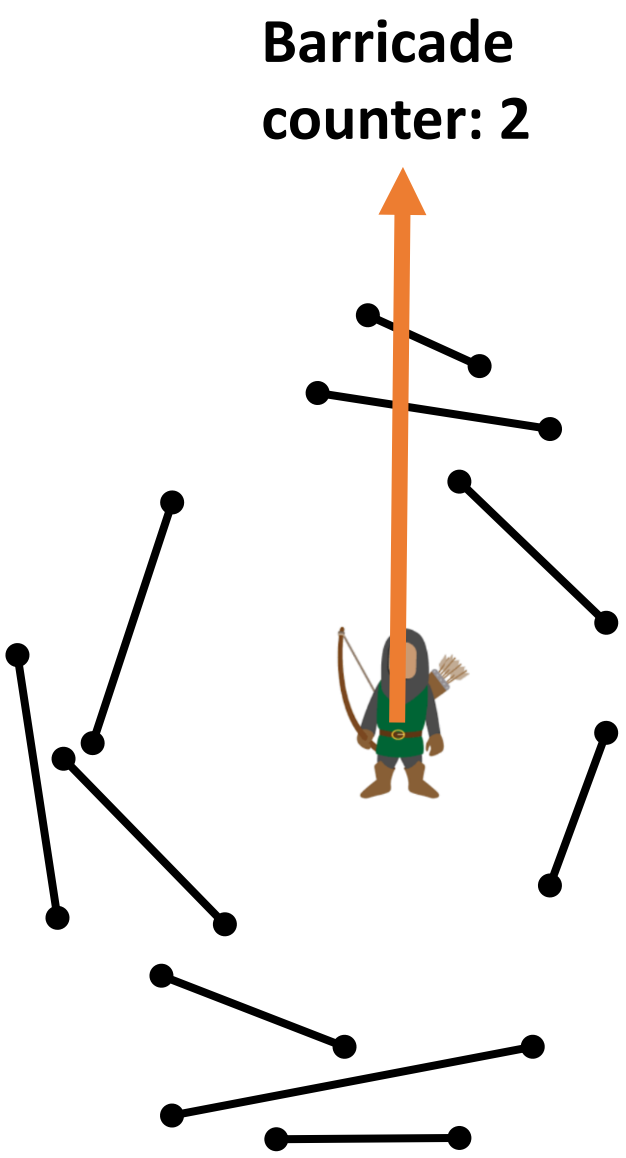
Sweeping: Example #2

- 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.



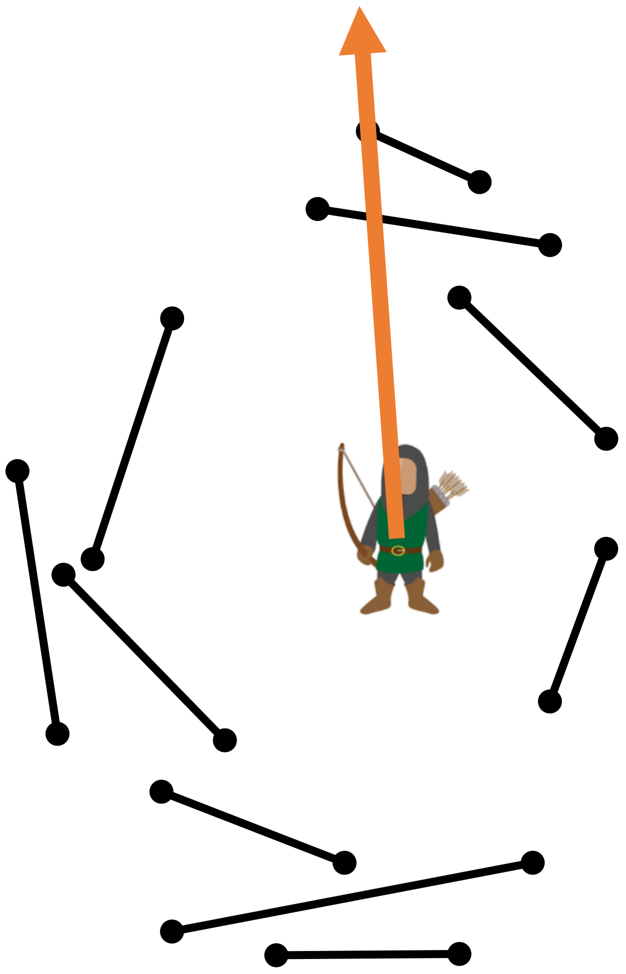
Sweeping: Example #2

- 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.

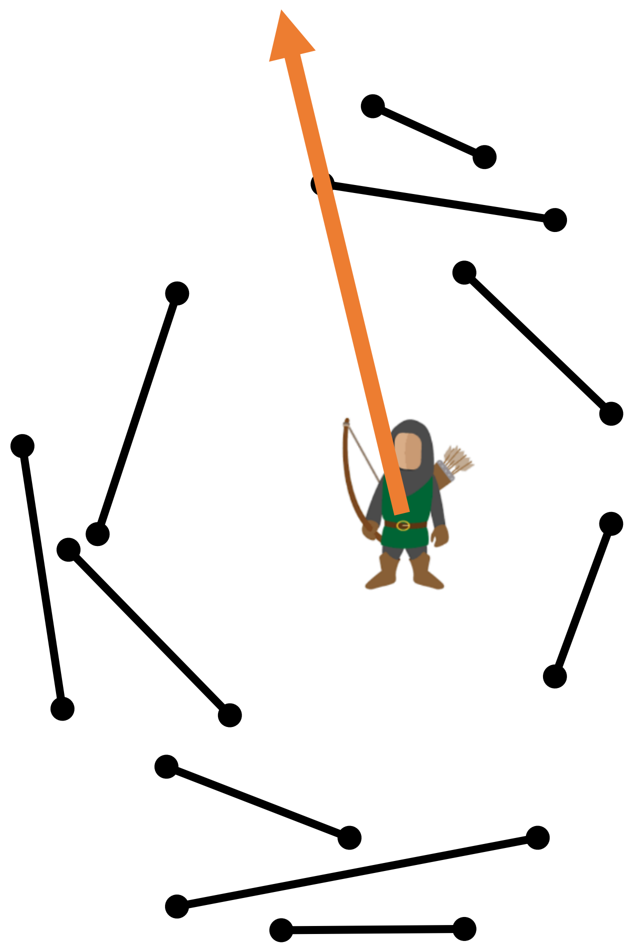


Sweeping: Example #2

- 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

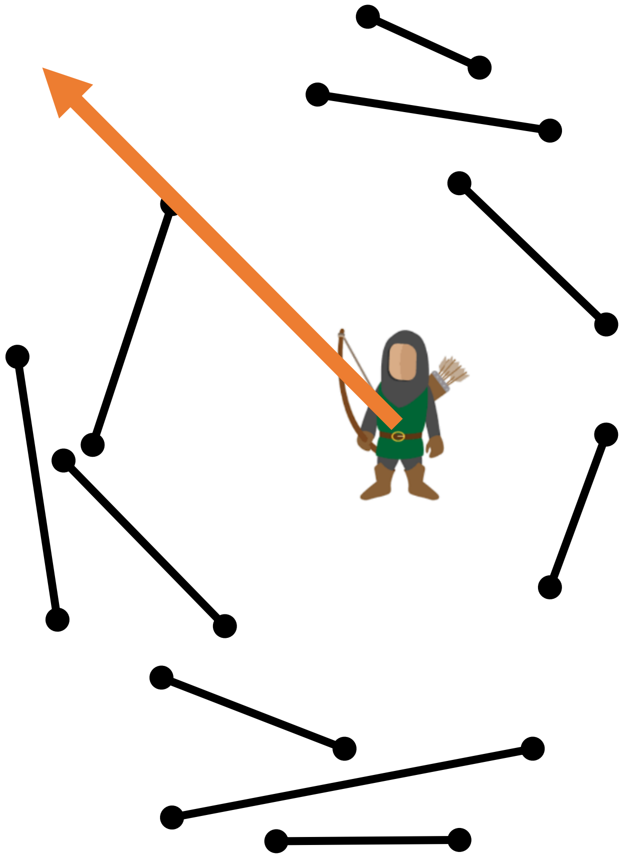


- An archer is surrounded by a set of barricades. What are his lines of sight?
- **Order:** Scan the segments by angle.
- **Status:** Number of ‘active’ barricades.
  - Init in $O(n)$.
- **Events:**
  - Start of a segment: increase number of barricades.
  - End of a segment: decrease number of barricades.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

Sweeping: Example #2

Barricade counter: 1

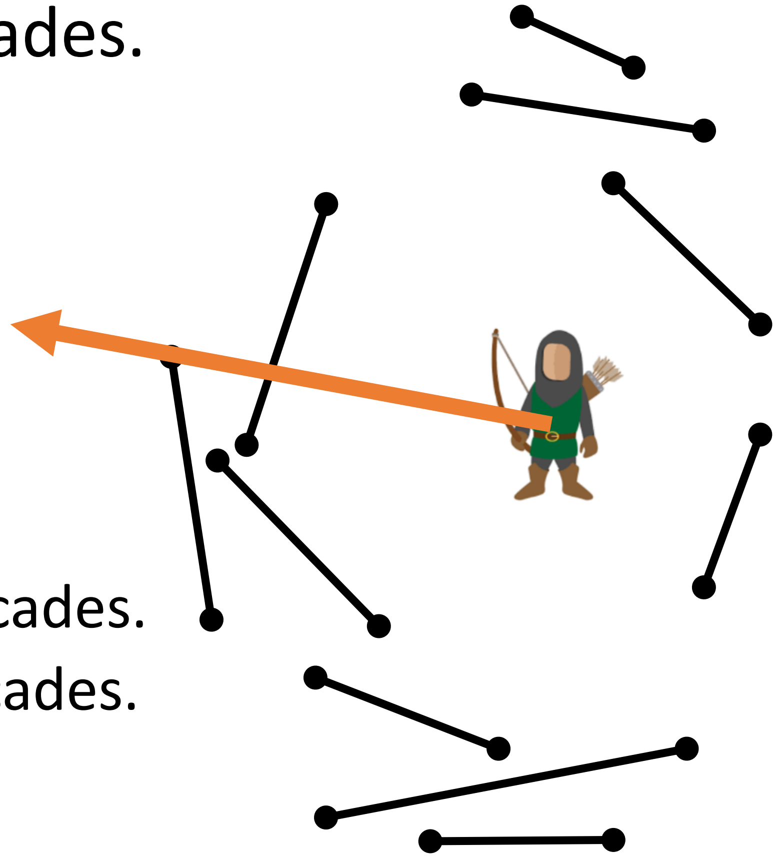
- An archer is surrounded by a set of barricades. What are his lines of sight?
- **Order:** Scan the segments by angle.
- **Status:** Number of 'active' barricades.
  - Init in $O(n)$.
- **Events:**
  - Start of a segment: increase number of barricades.
  - End of a segment: decrease number of barricades.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



Sweeping: Example #2

Barricade counter: 2

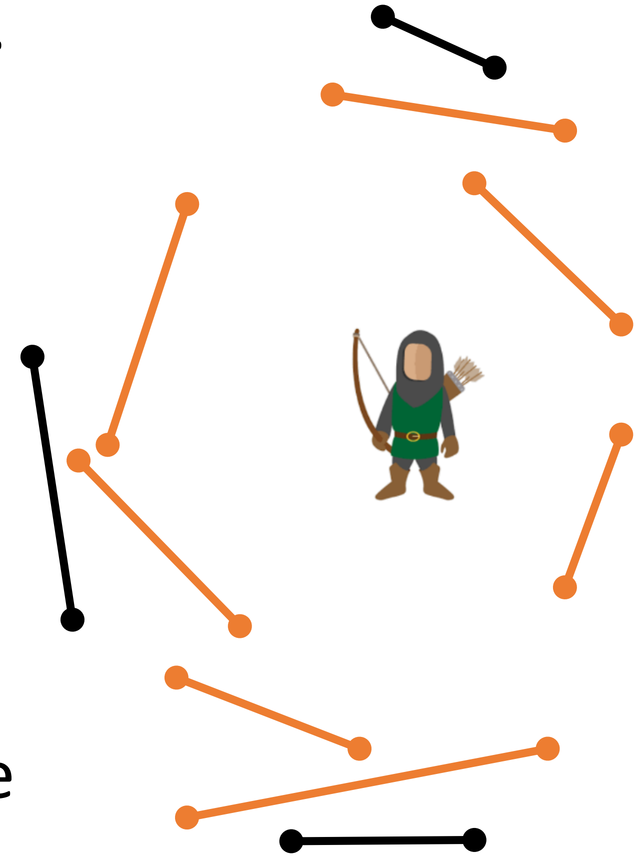
- An archer is surrounded by a set of barricades. What are his lines of sight?
- **Order:** Scan the segments by angle.
- **Status:** Number of ‘active’ barricades.
  - Init in $O(n)$.
- **Events:**
  - Start of a segment: increase number of barricades.
  - End of a segment: decrease number of barricades.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



Sweeping: Example #3

Sweeping: Example #3

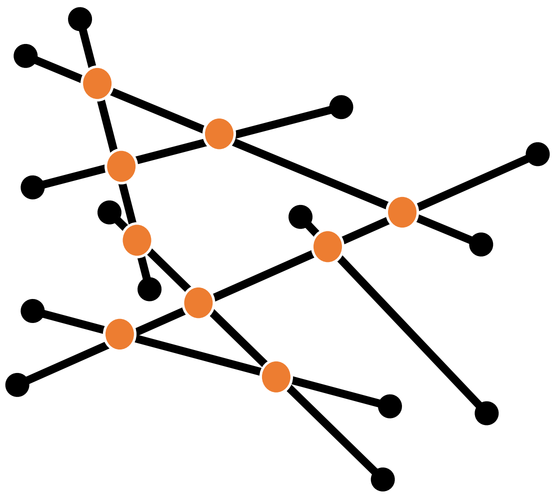
- An archer is surrounded by a set of barricades. Which barricades are visible to him?
- **Order:** Scan the segments by angle.
- **Status:** Set of active barricades, sorted by the distance from the archer.
- **Events:**
  - Start of a segment: Add segment to the status DS.
  - End of a segment: Remove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



Segment Intersection

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.**

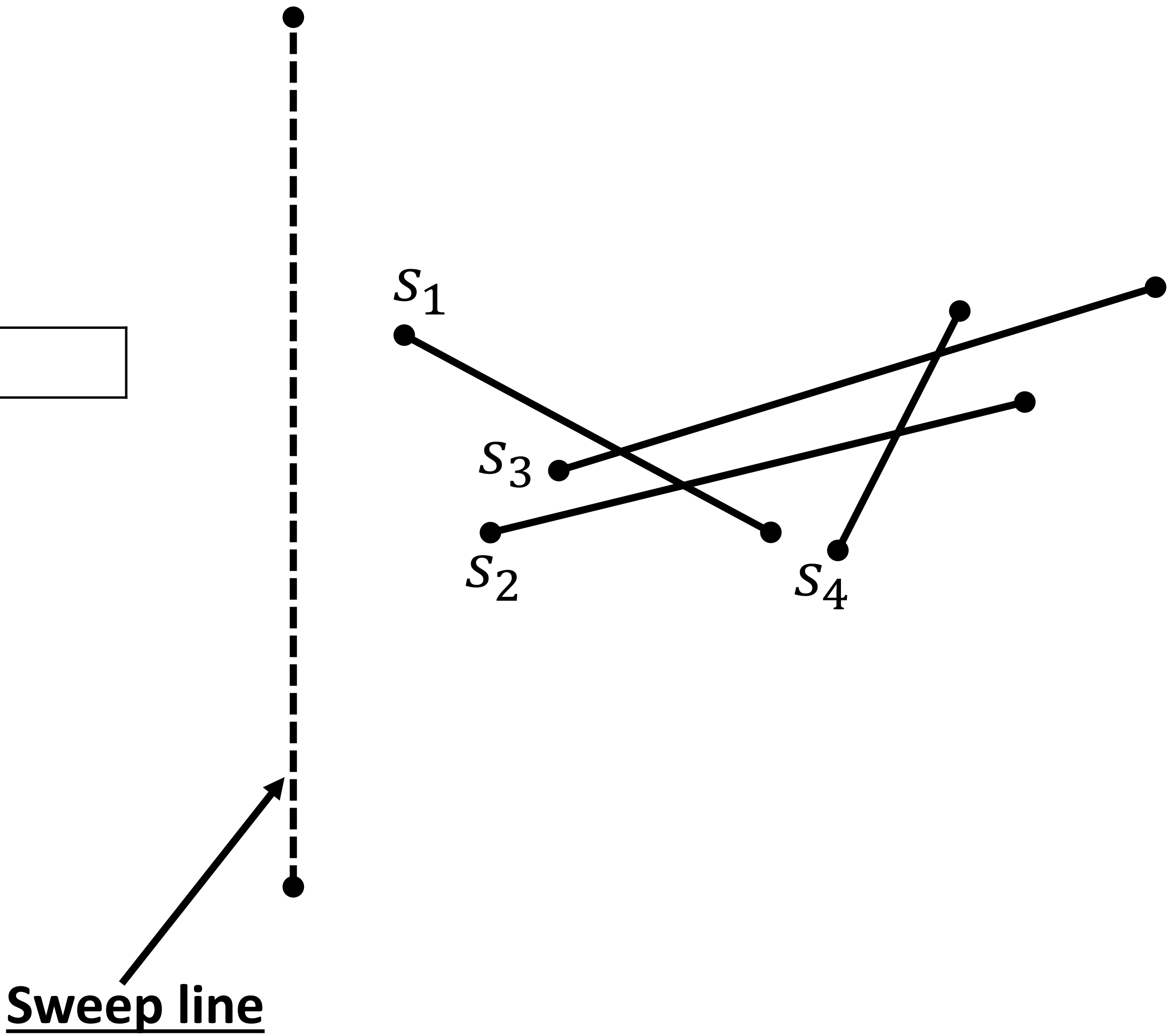


Dynamic events!

Handle event: $None$

Events
$Start(S_1)$
$Start(S_2)$
$Start(S_3)$
$End(S_1)$
$Start(S_4)$
$End(S_4)$
$End(S_2)$
$End(S_3)$

Status
$\emptyset$



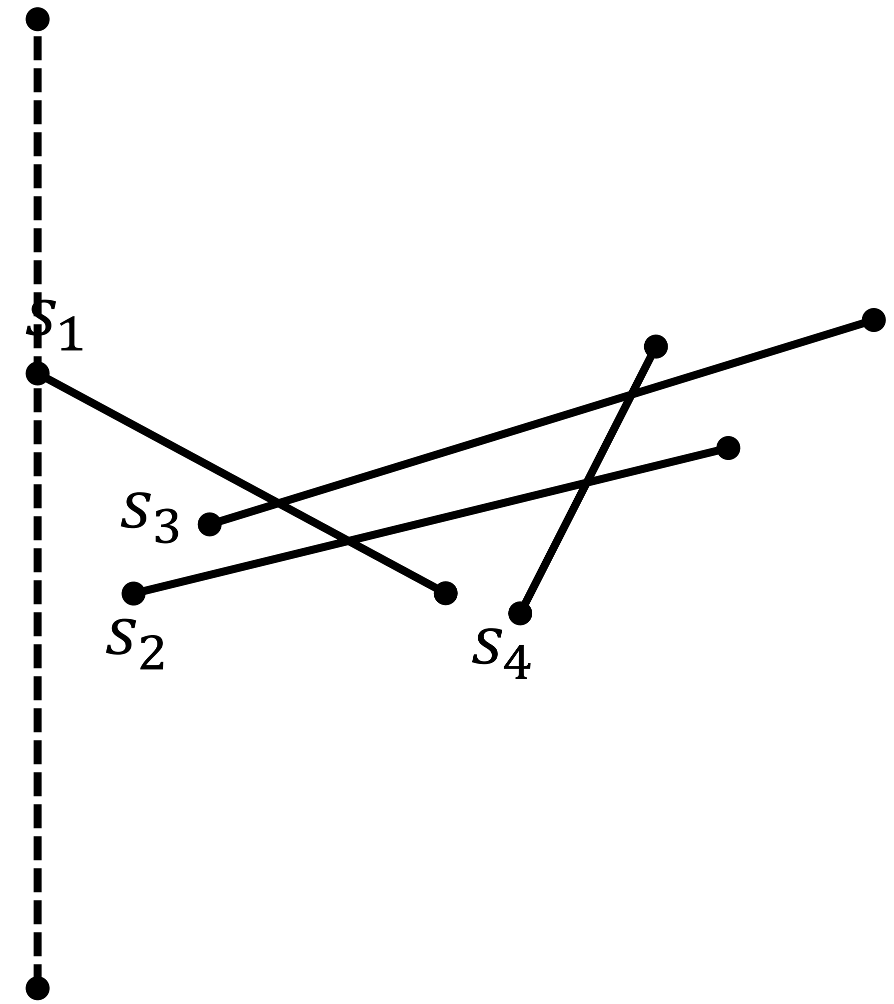
Handle event: $Start(S_1)$

Events

$Start(S_2)$
$Start(S_3)$
$End(S_1)$
$Start(S_4)$
$End(S_4)$
$End(S_2)$
$End(S_3)$

Status

S_1



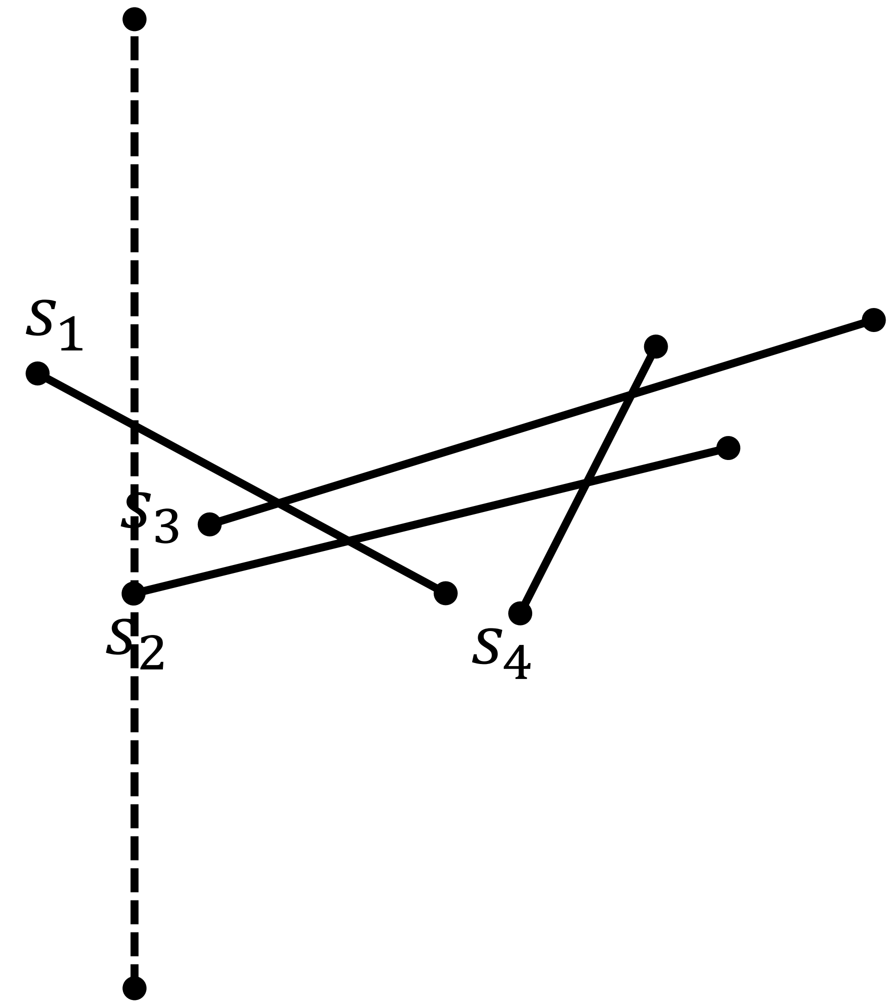
Handle event: $Start(S_2)$

Events

$Start(S_3)$
$Intersection(S_1, S_2)$
$End(S_1)$
$Start(S_4)$
$End(S_4)$
$End(S_2)$
$End(S_3)$

Status

S_1
S_2



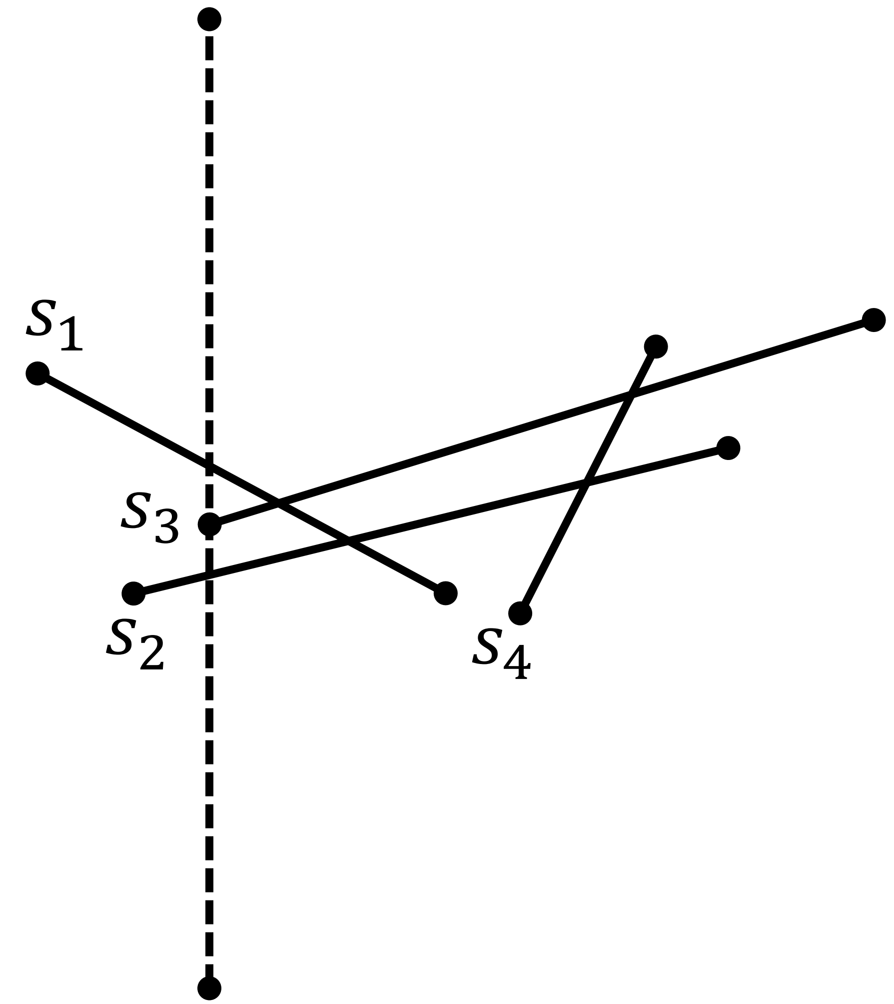
Handle event: $Start(S_3)$

Events

$Intersection(S_1, S_3)$
$Intersection(S_1, S_2)$
$End(S_1)$
$Start(S_4)$
$End(S_4)$
$End(S_2)$
$End(S_3)$

Status

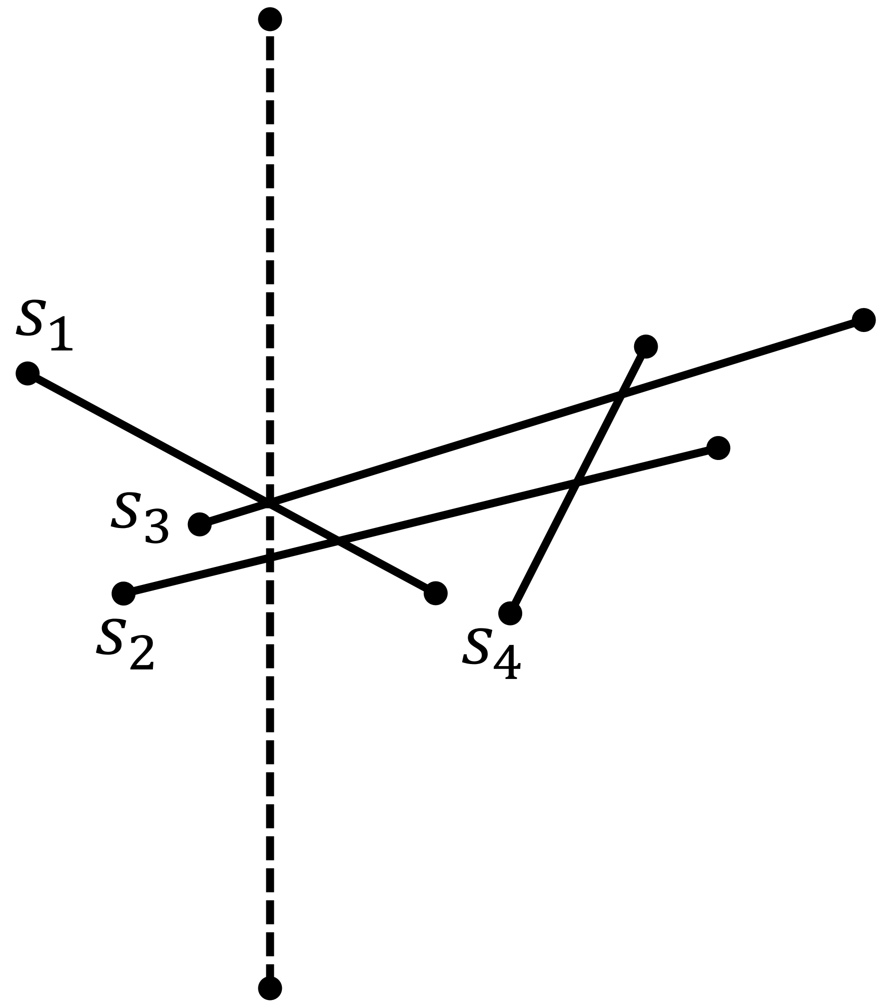
S_1
S_3
S_2



Handle event: $Intersection(S_1, S_3)$

Events
$Intersection(S_1, S_2)$
$End(S_1)$
$Start(S_4)$
$End(S_4)$
$End(S_2)$
$End(S_3)$

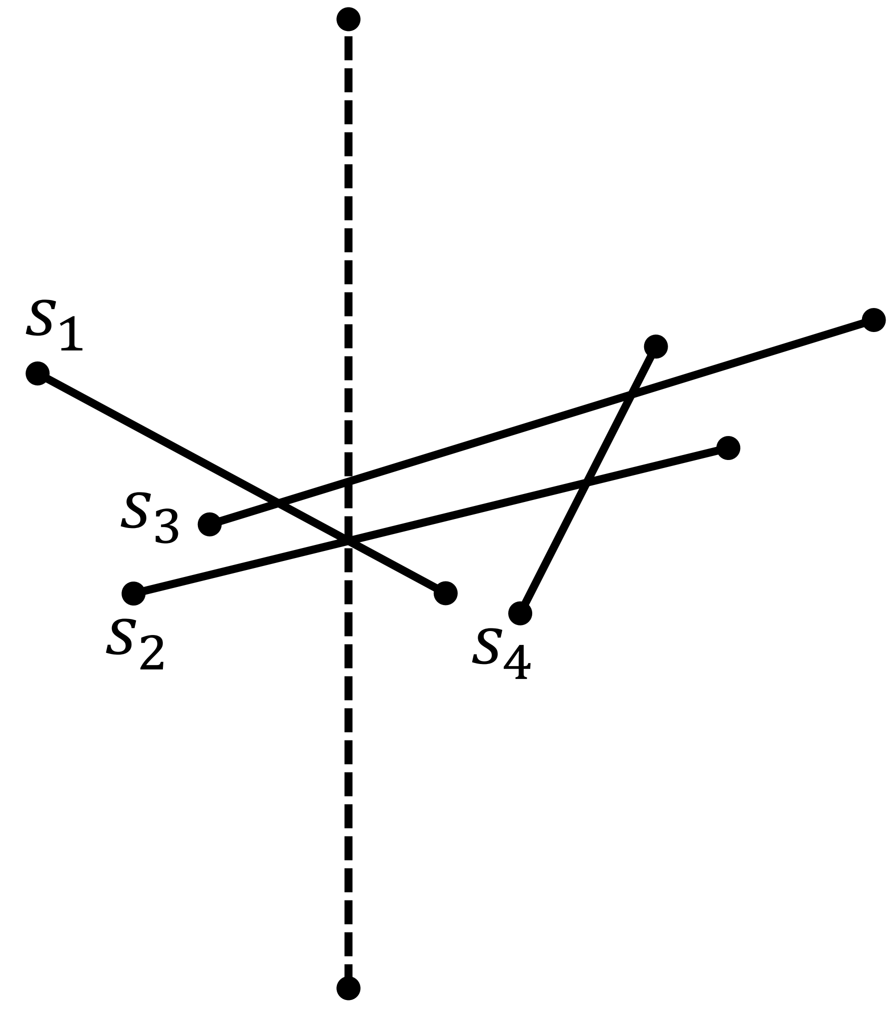
Status
S_3
S_1
S_2



Handle event: $Intersection(S_1, S_2)$

Events
$End(S_1)$
$Start(S_4)$
$End(S_4)$
$End(S_2)$
$End(S_3)$

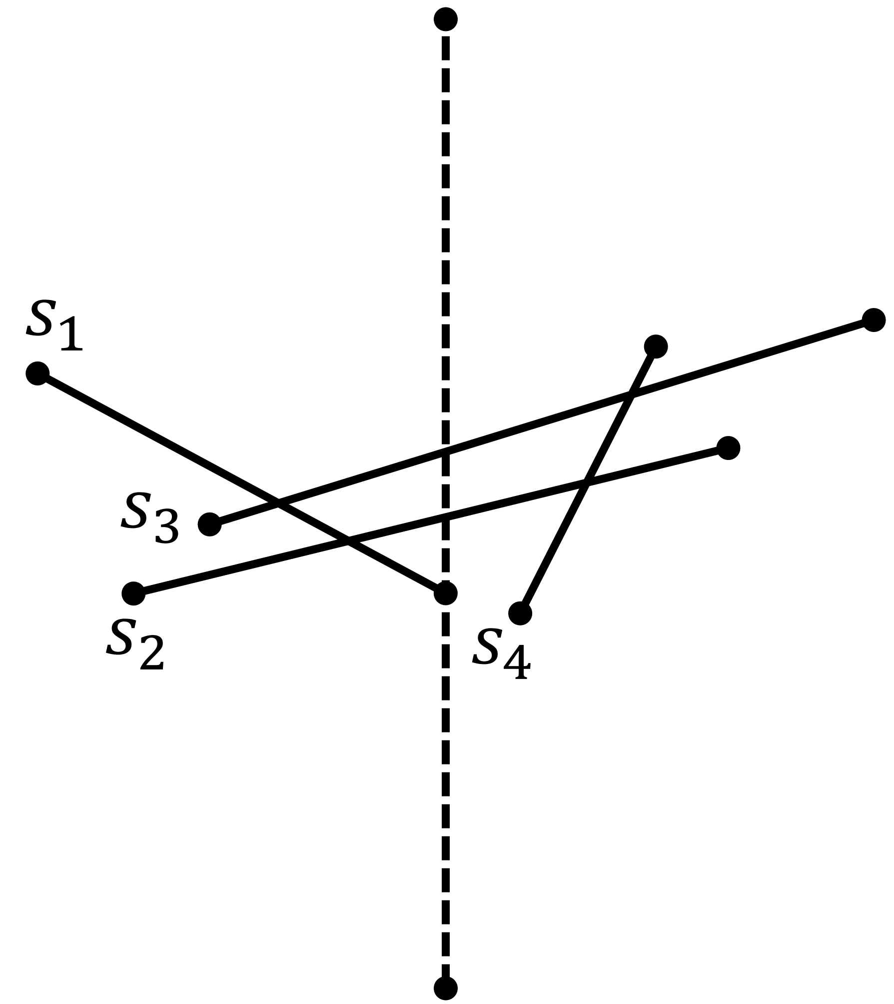
Status
S_3
S_2
S_1



Handle event: $End(S_1)$

Events
$Start(S_4)$
$End(S_4)$
$End(S_2)$
$End(S_3)$

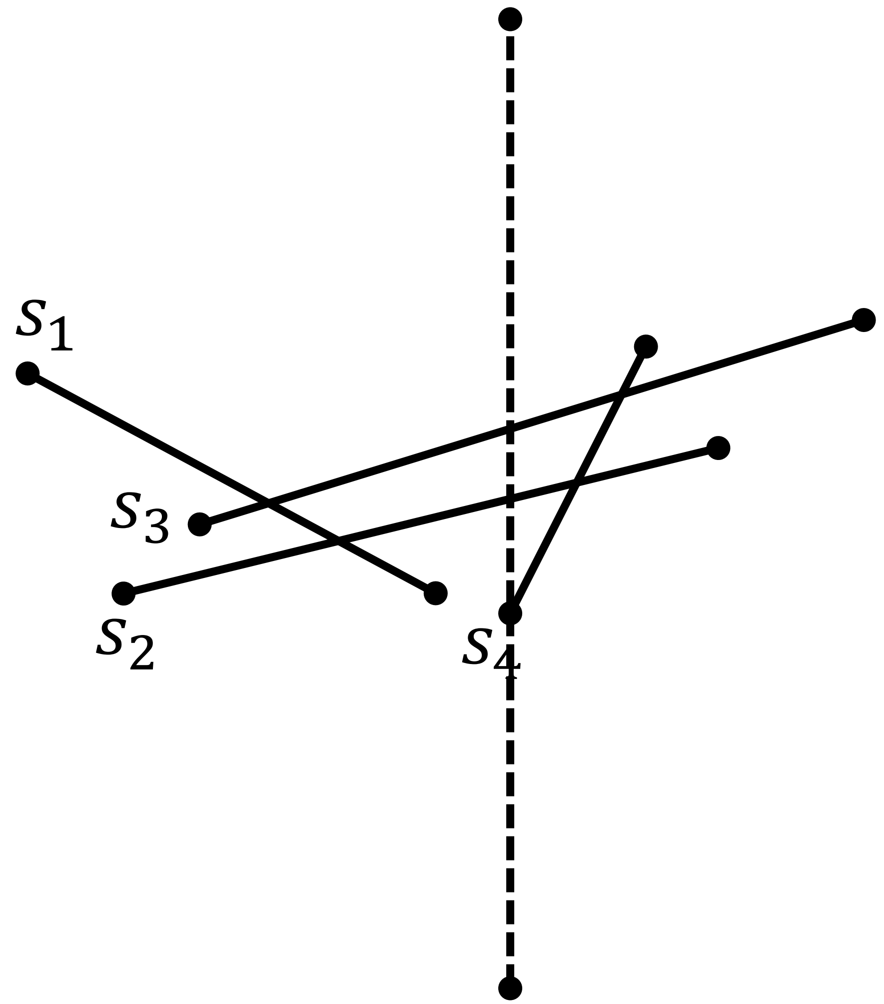
Status
S_3
S_2



Handle event: $Start(S_4)$

Events
$Intersection(S_2, S_4)$
$End(S_4)$
$End(S_2)$
$End(S_3)$

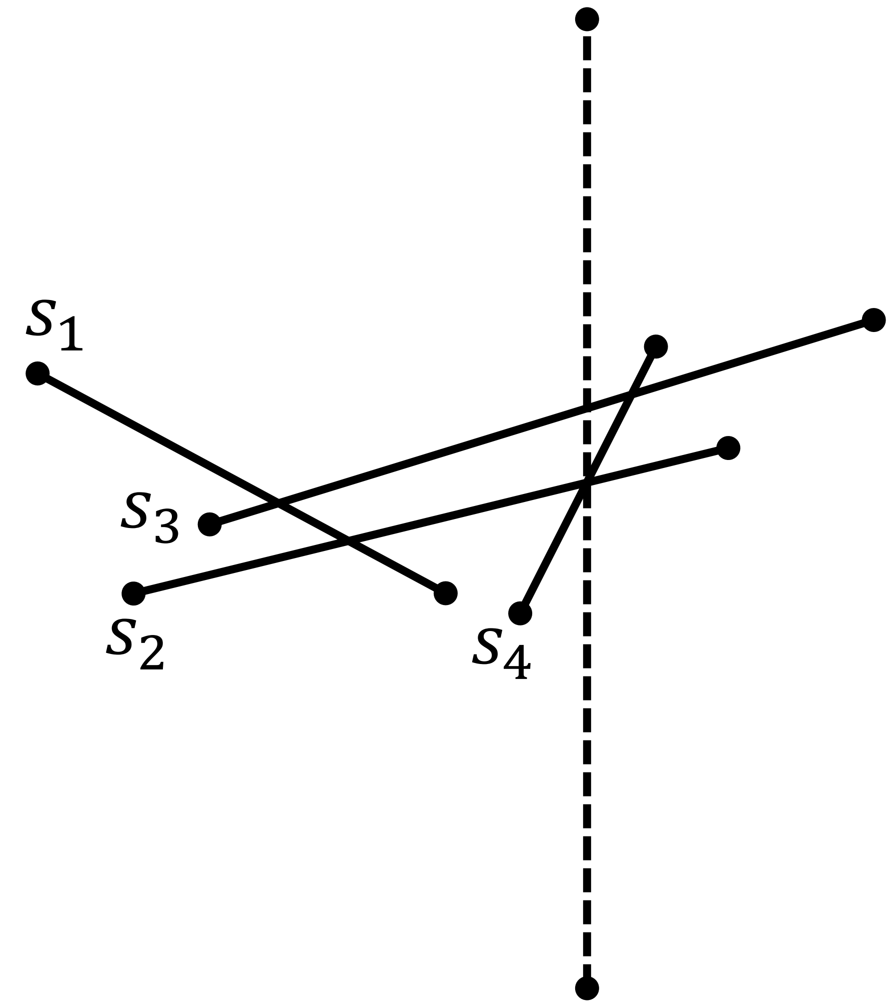
Status
S_3
S_2
S_4



Handle event: $Intersection(S_2, S_4)$

Events
$Intersection(S_3, S_4)$
$End(S_4)$
$End(S_2)$
$End(S_3)$

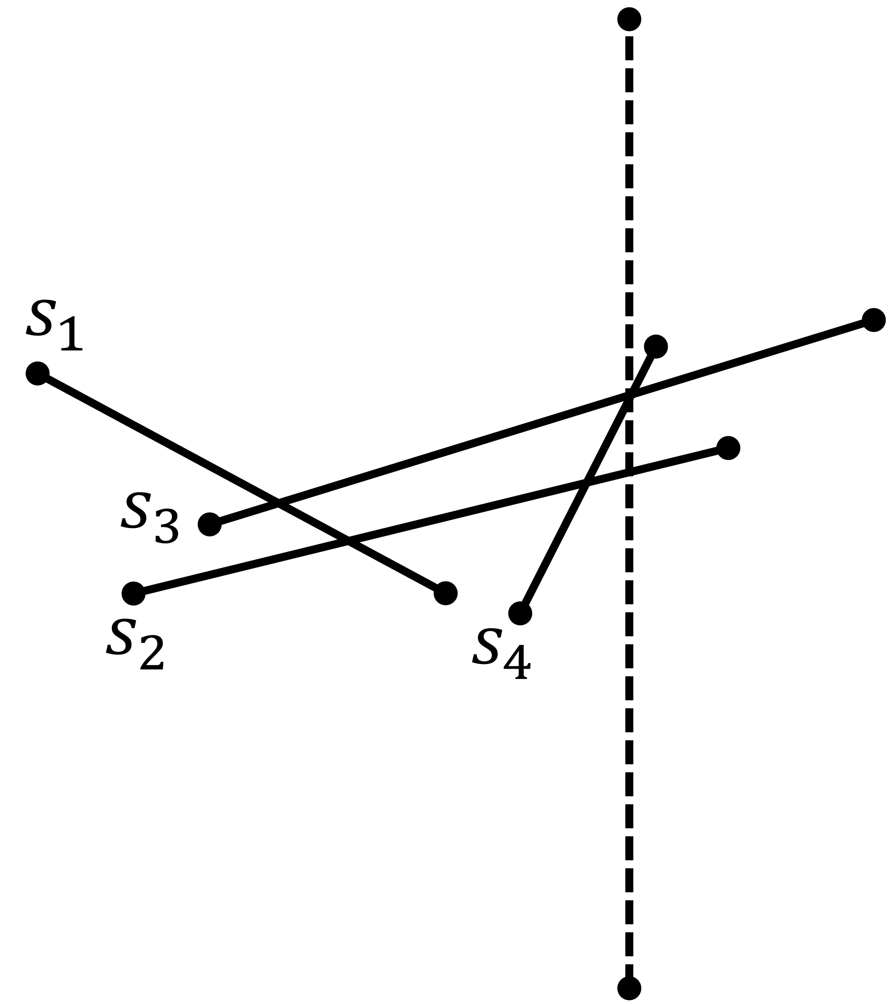
Status
S_3
S_4
S_2



Handle event: $Intersection(S_3, S_4)$

Events
$End(S_4)$
$End(S_2)$
$End(S_3)$

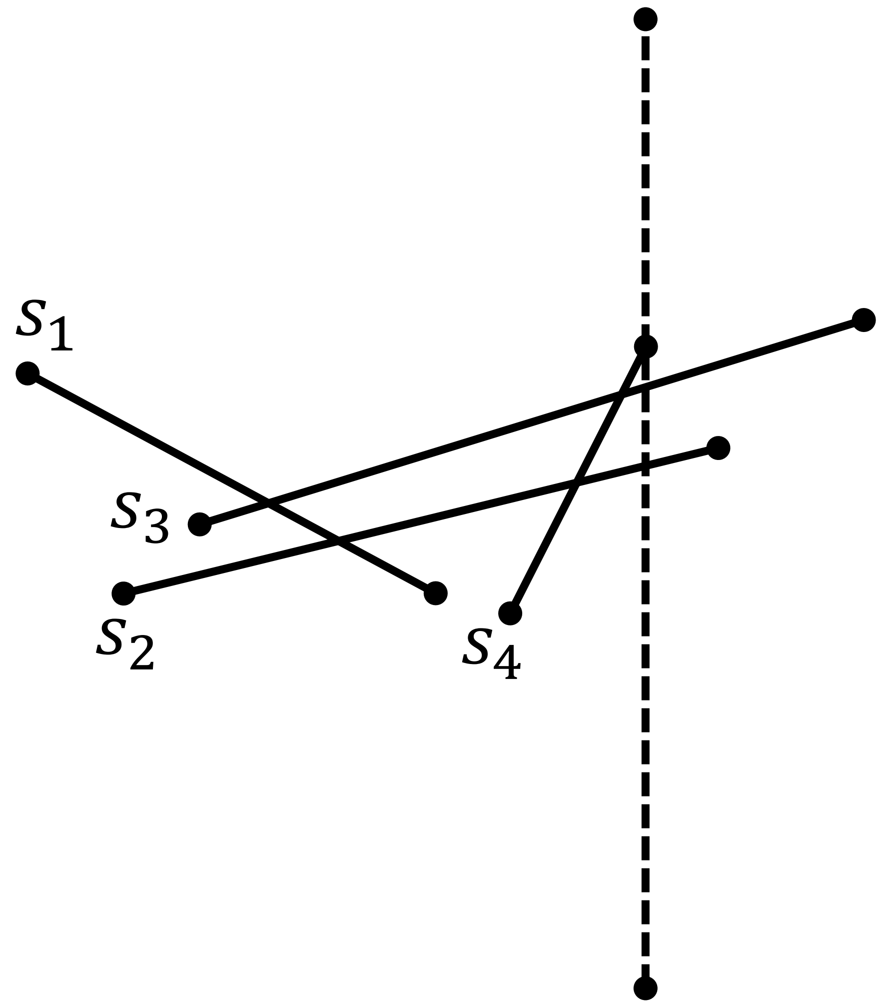
Status
S_4
S_3
S_2



Handle event: $End(S_4)$

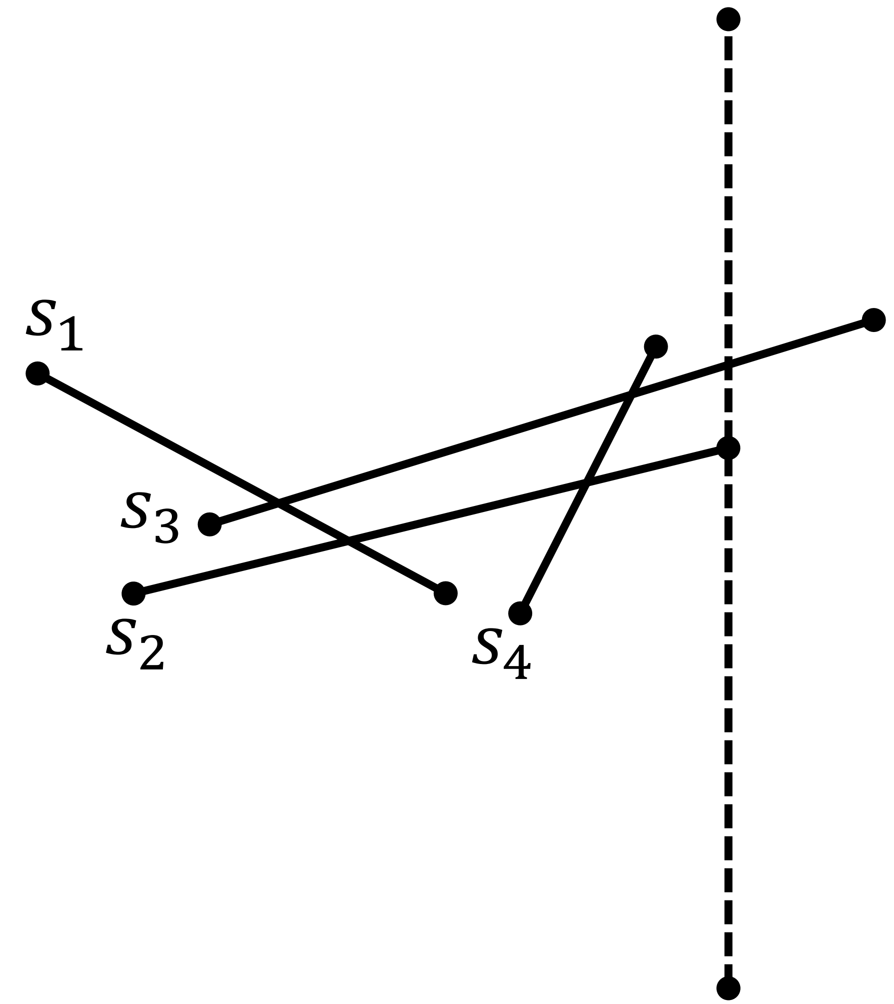
Events
$End(S_2)$
$End(S_3)$

Status
S_3
S_2



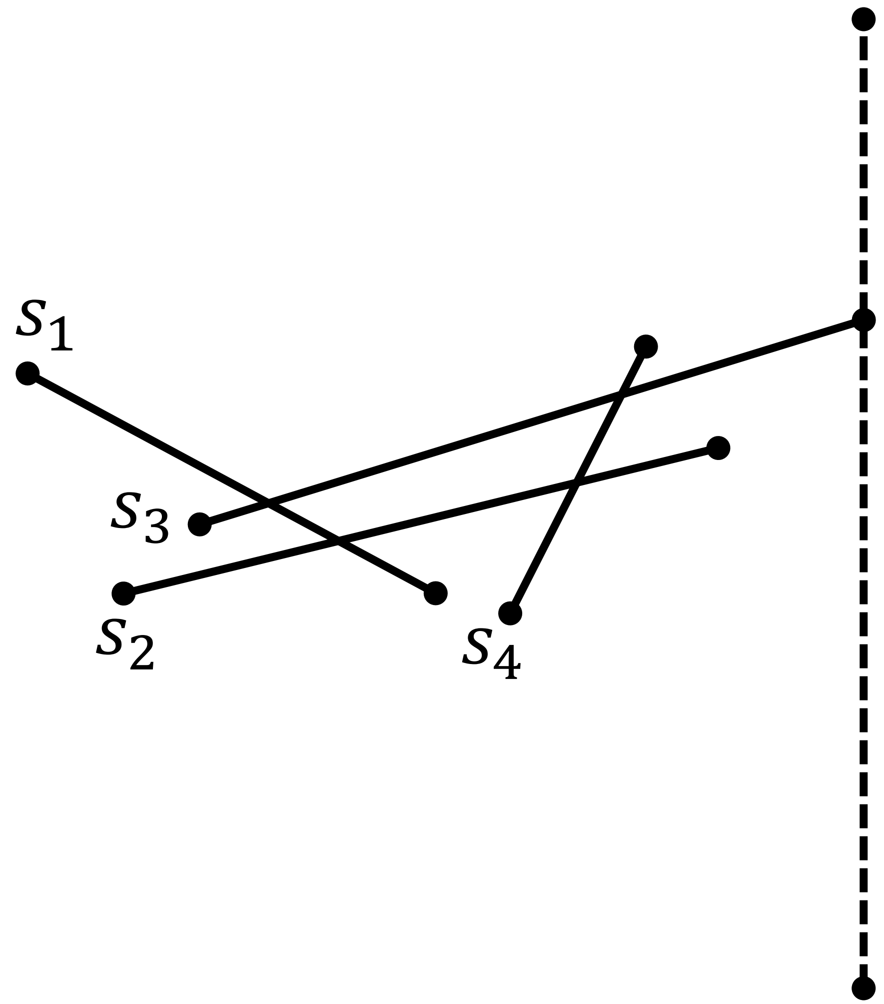
Handle event: $End(S_2)$

Events	Status
$End(S_3)$	S_3



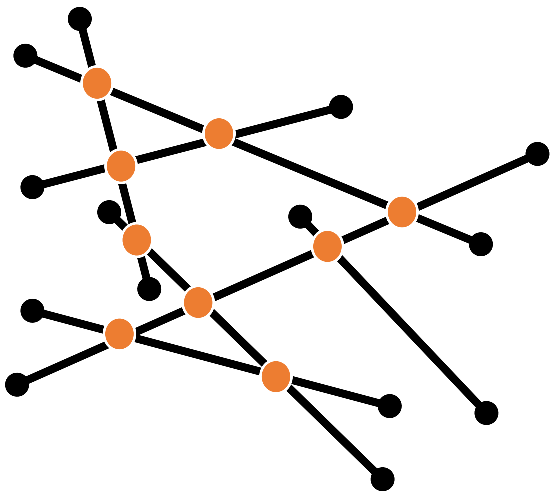
Handle event: $End(S_3)$

Events	**Status**
$\emptyset$	$\emptyset$



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: $O(n \log n)$

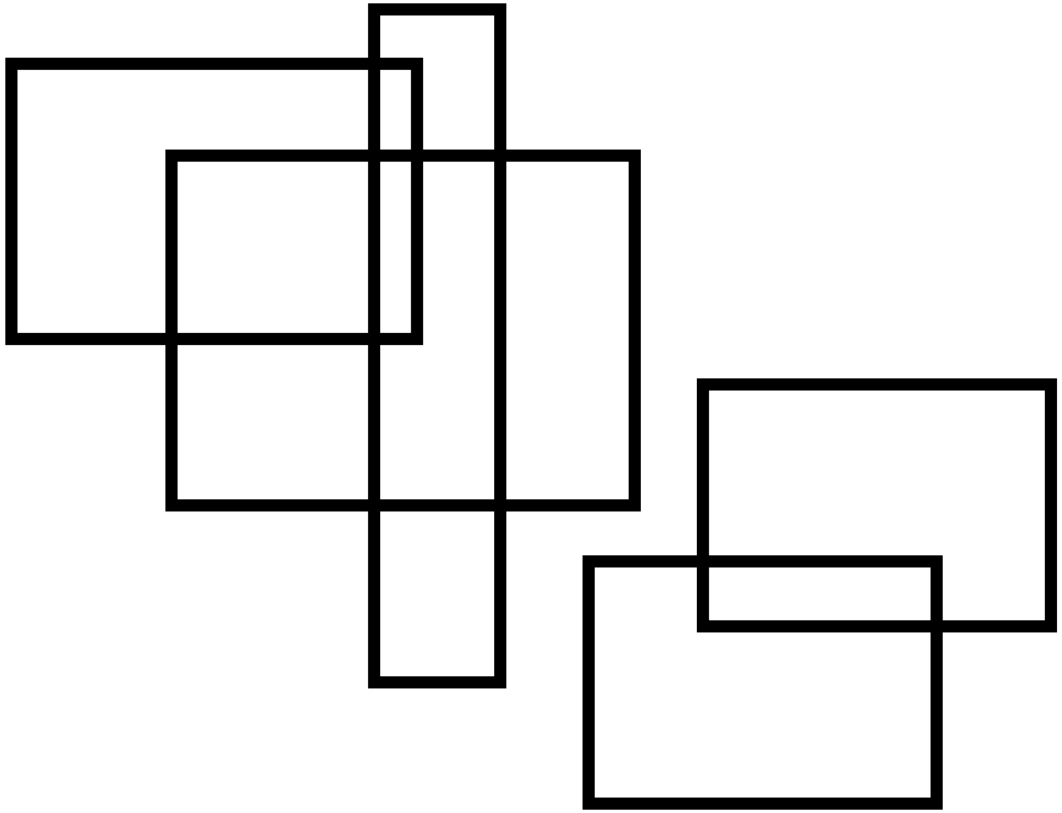


Dynamic events!

Area of union of Rectangles

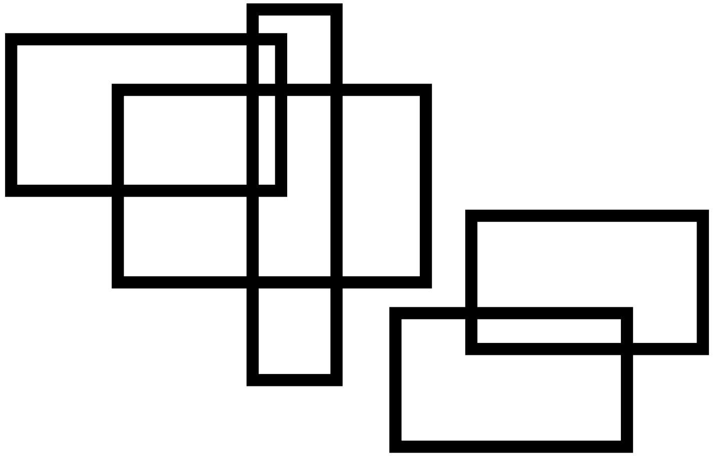
Area of union of Rectangles

- What is the total area covered by a set of rectangles?



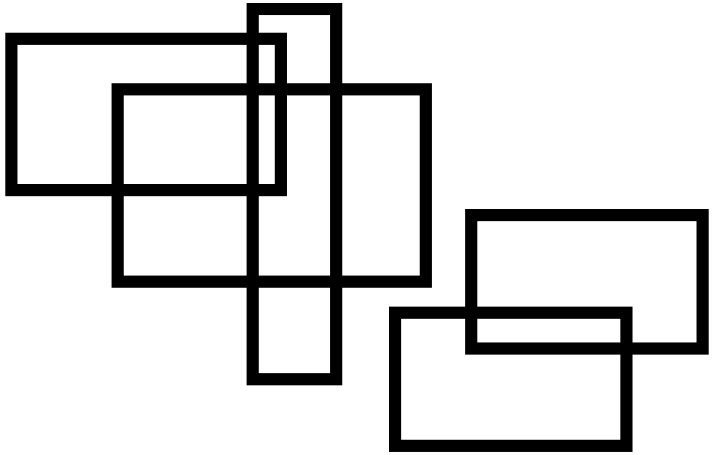
Area of union of Rectangles

- What is the total area covered by a set of rectangles?
- **Order:** left to right
- **Events:** begin and end of a rectangle
- **Status:** active rectangles



Area of union of Rectangles

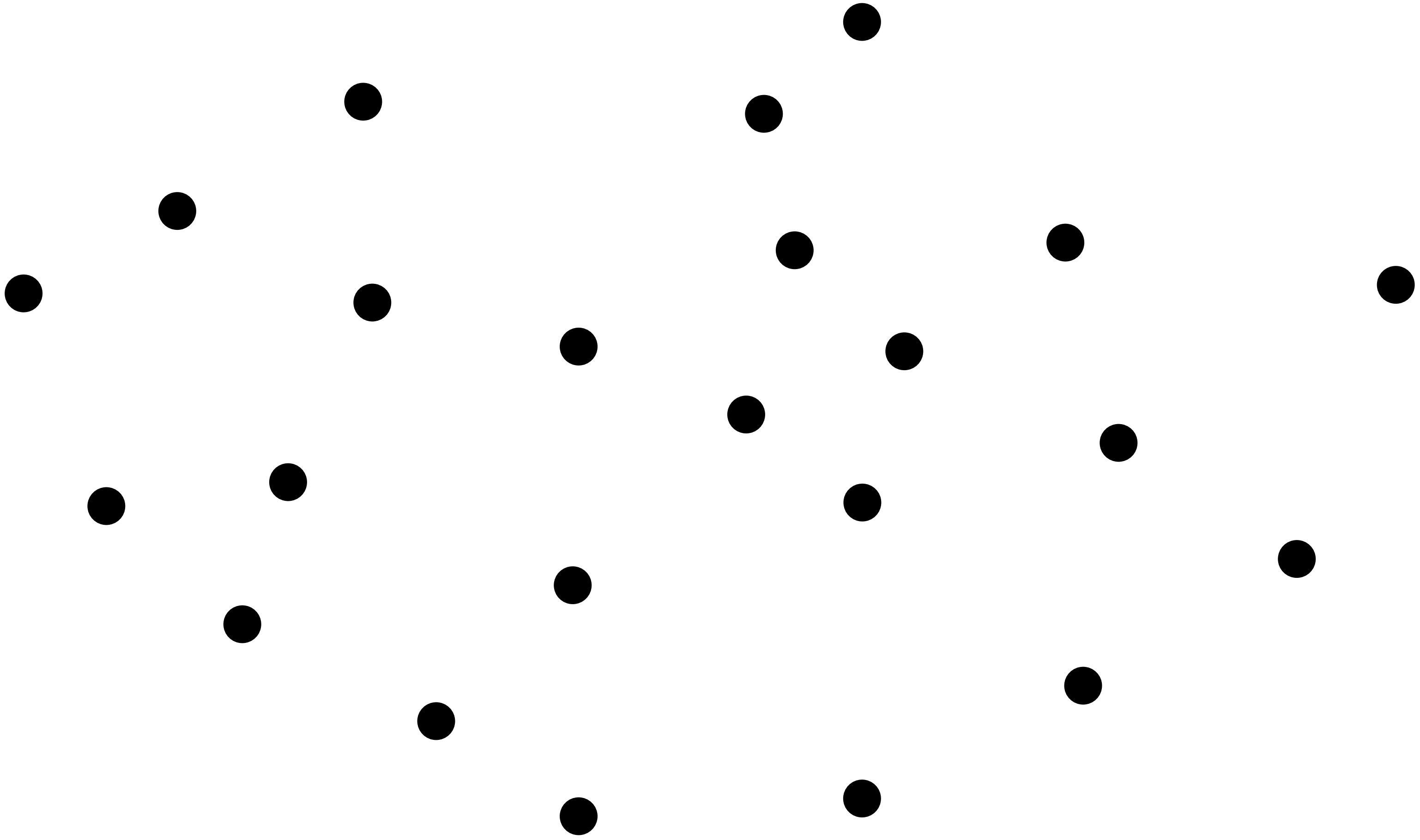
- **Status:** active rectangles
- How do we maintain the active rectangle set?
- More importantly, how do we find the total length covered by the active rectangle?
- Naïve implementation: Recalculate the union each time (using example #1). Complexity: $O(n^2)$.
- Better implementation: Use augmented BST (classic DS exercise). Complexity: $O(n \log n)$.



Minimal Distance Pair

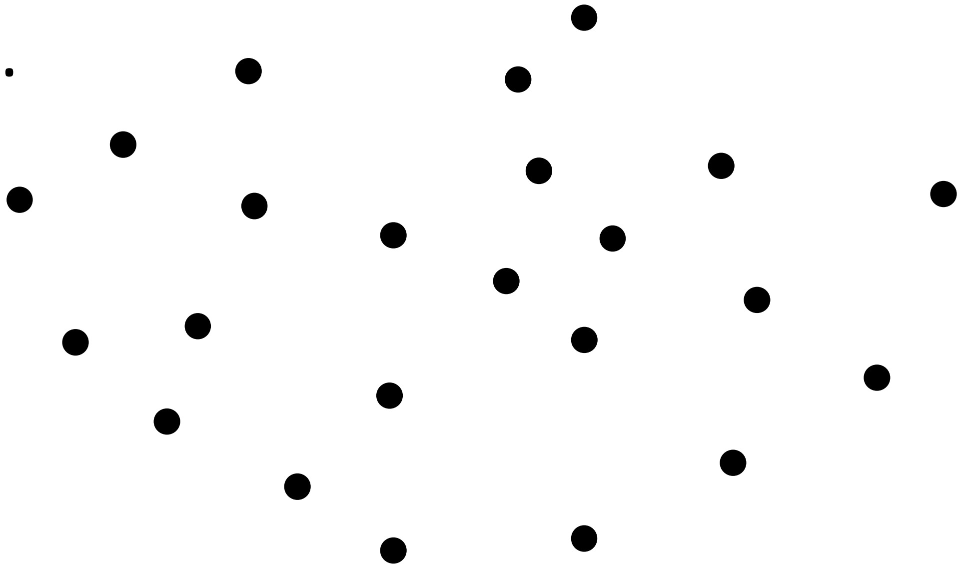
Minimal Distance Pair

- Problem: Find the closest pair of points.

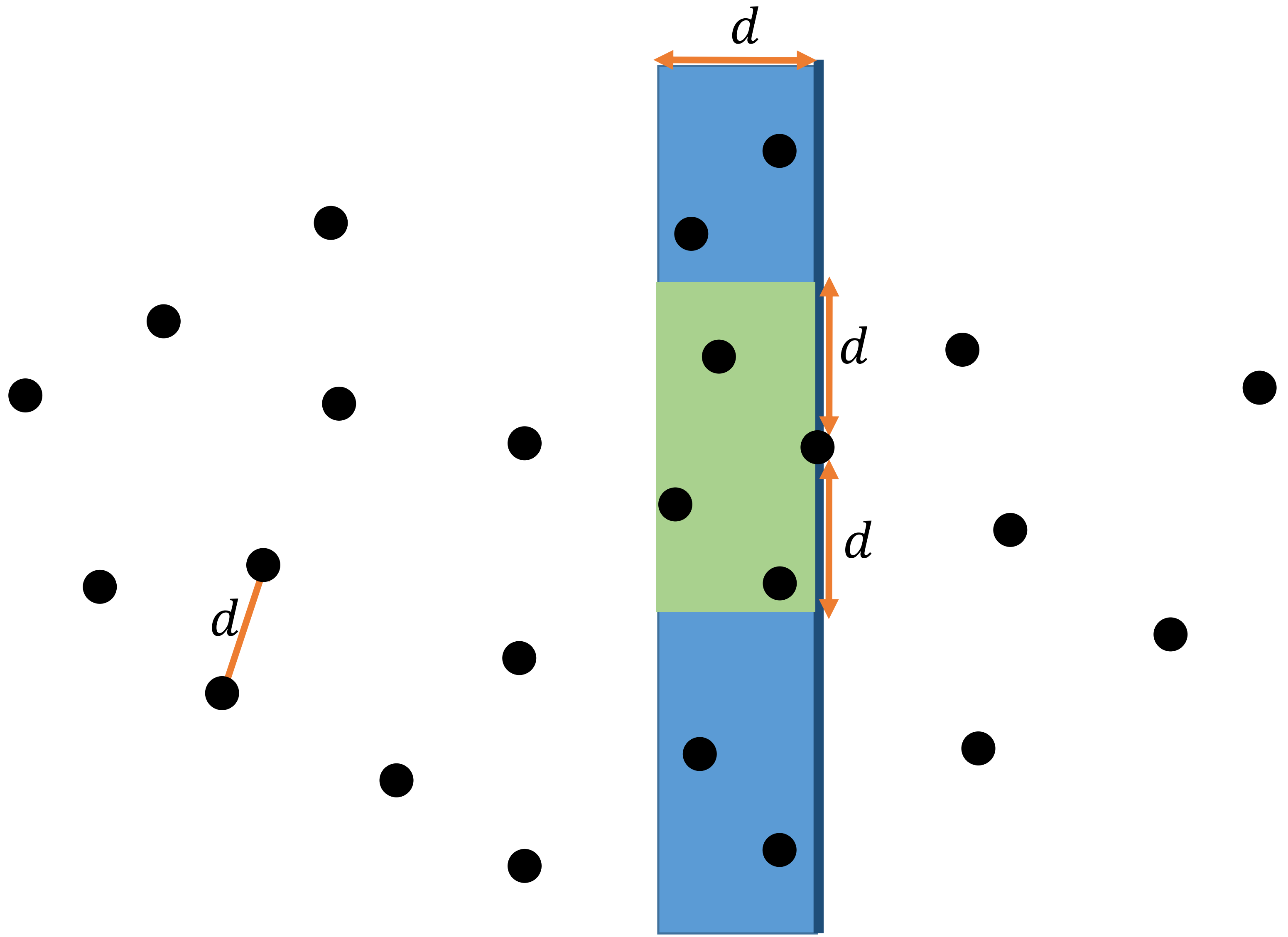


Minimal Distance Pair

- 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 of all the points in a strip of width d.
  one sorted by the y coordinate,
  and another sorted by the x coordinate.



Minimal Distance Pair



Minimal Distance Pair

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