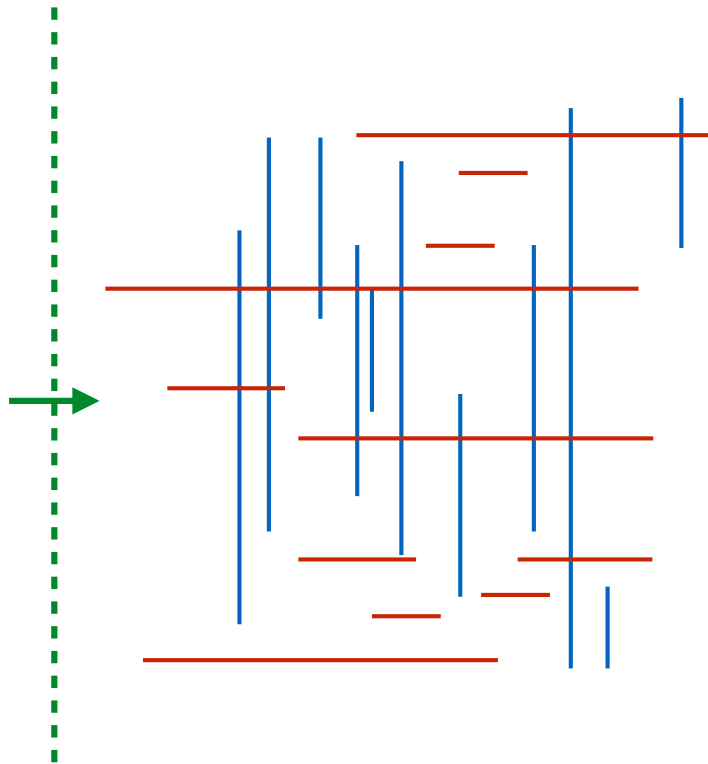


Line segment intersection

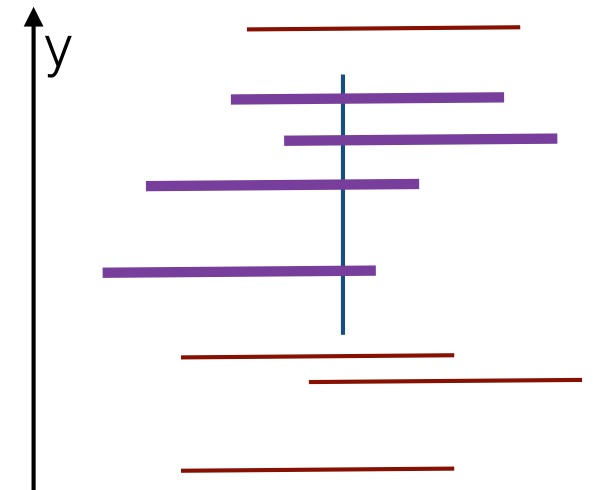
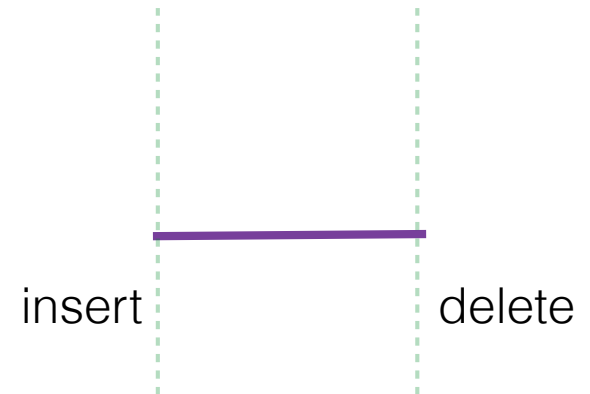
Last time we looked at a **special case** of line segment intersection

orthogonal segments



line sweep technique

solve the problem behind the line

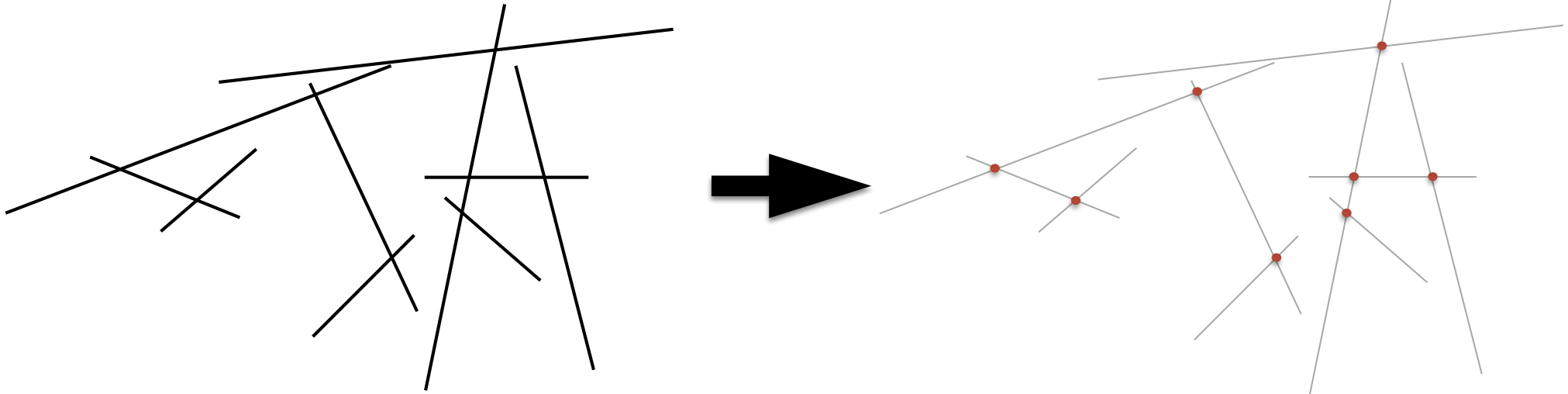


AS: segments in order of y

Result: The intersections of a set of n orthogonal segments in the plane can be found in $O(n \lg n + k)$ time.

General line segment intersection

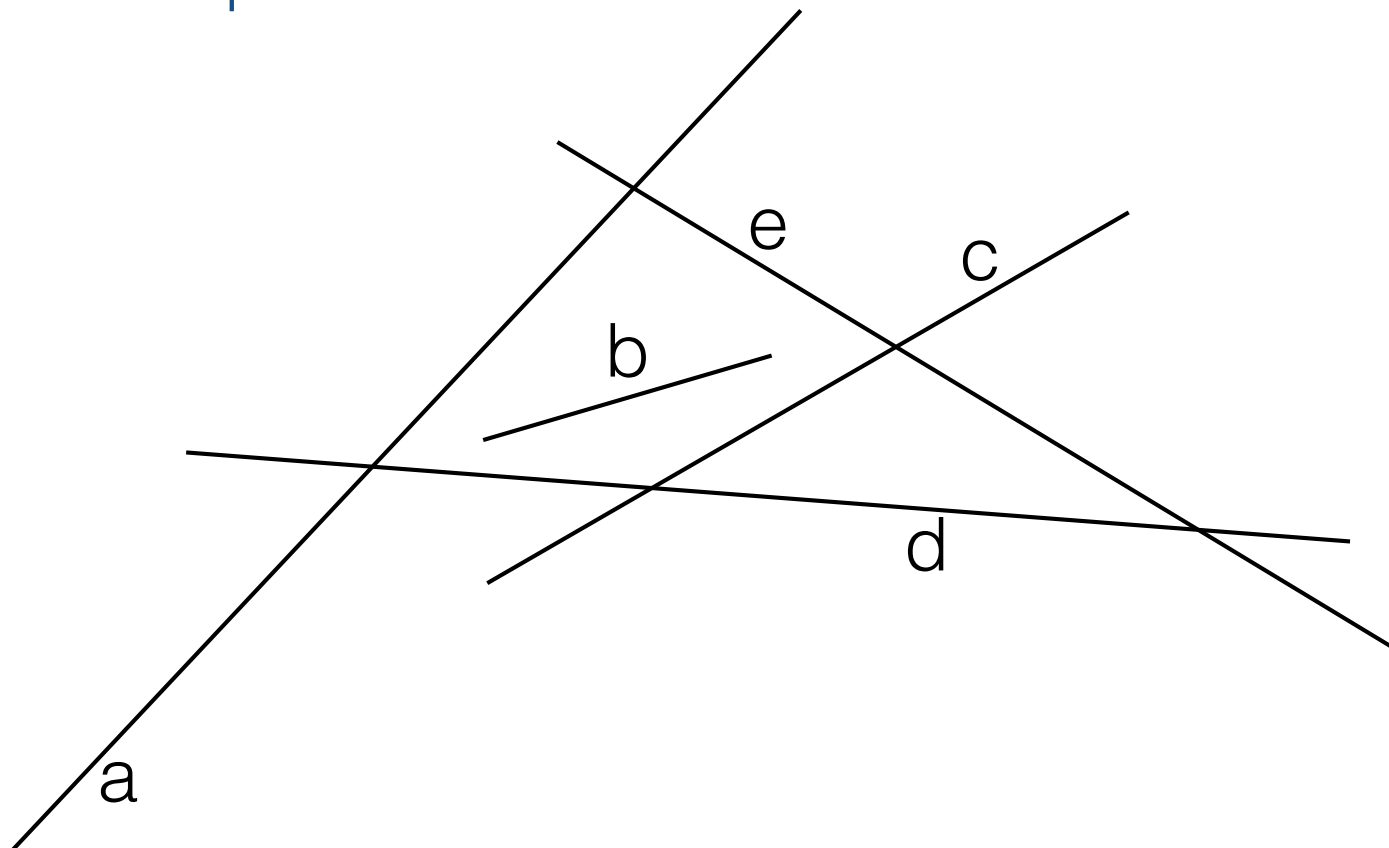
- n : size of the input (number of segments)
- k : size of output (number of intersections)



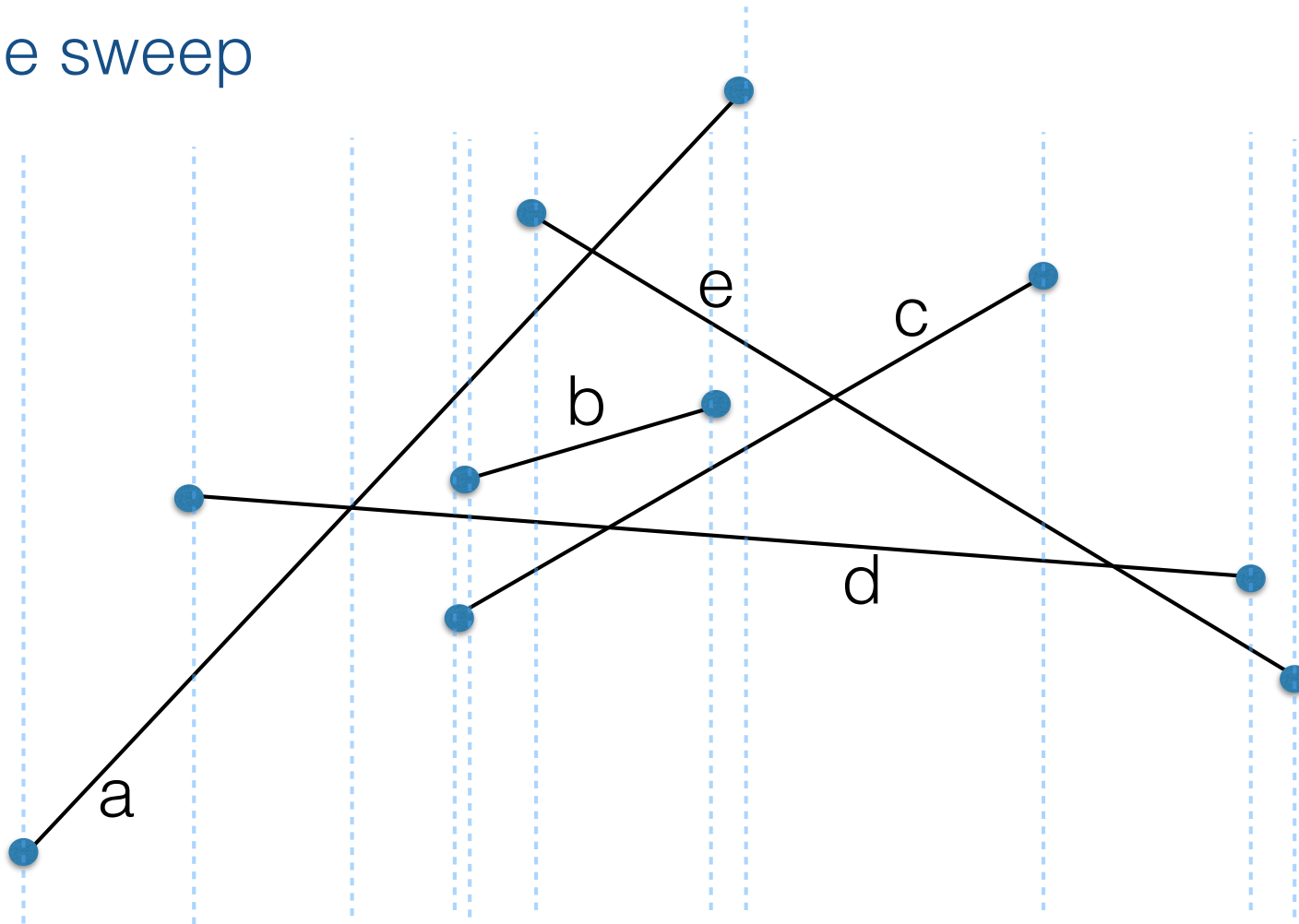
Overview

- How? we'll extend line sweep
- We'll get an overall bound of $O(n \lg n + k \lg n)$
 - this improves on the naive $O(n^2)$ when k is small
- The algorithm was developed by Jon Bentley and Thomas Ottman in 1979
- Simple (in retrospect!), elegant and practical

The sweep

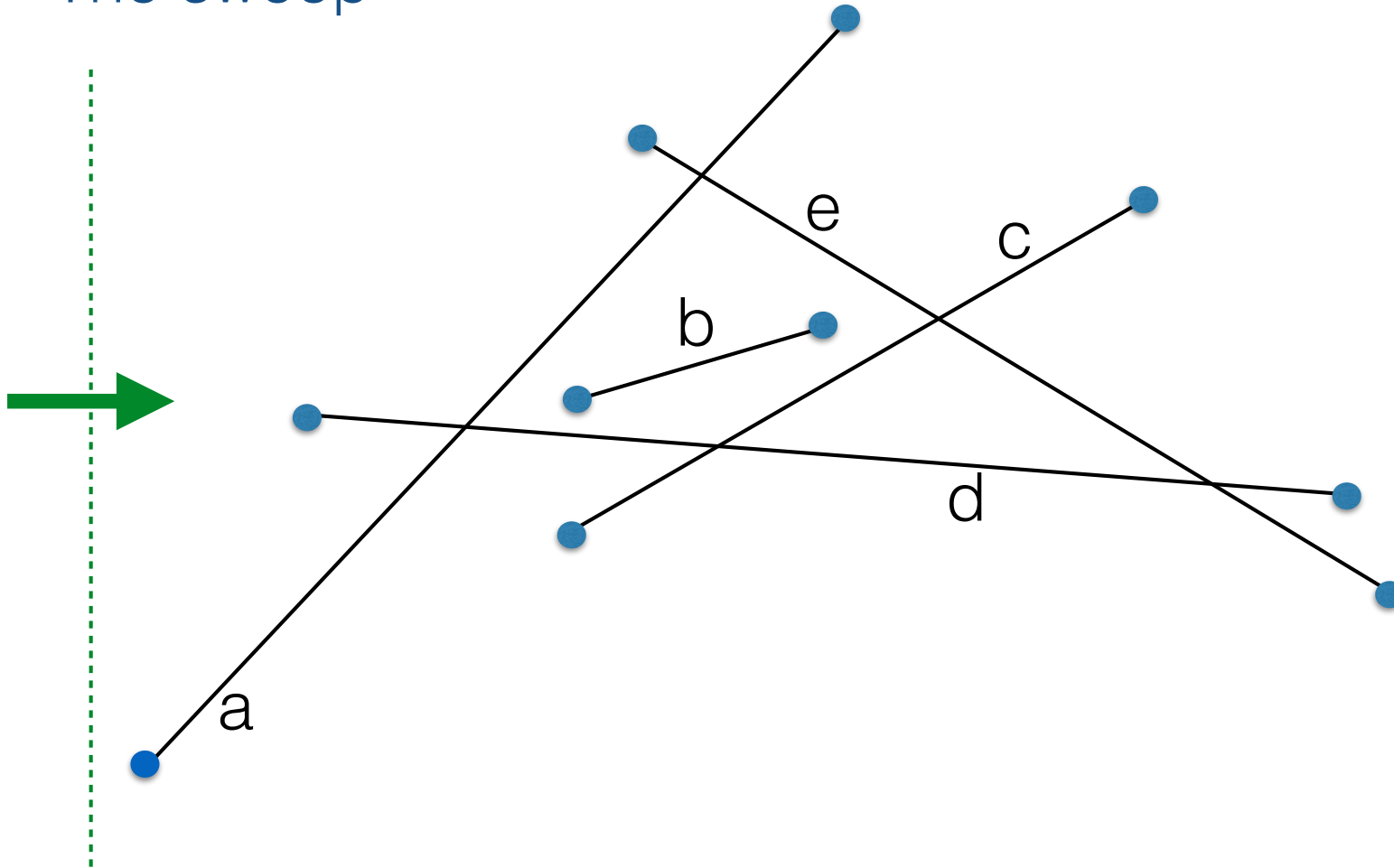


The sweep



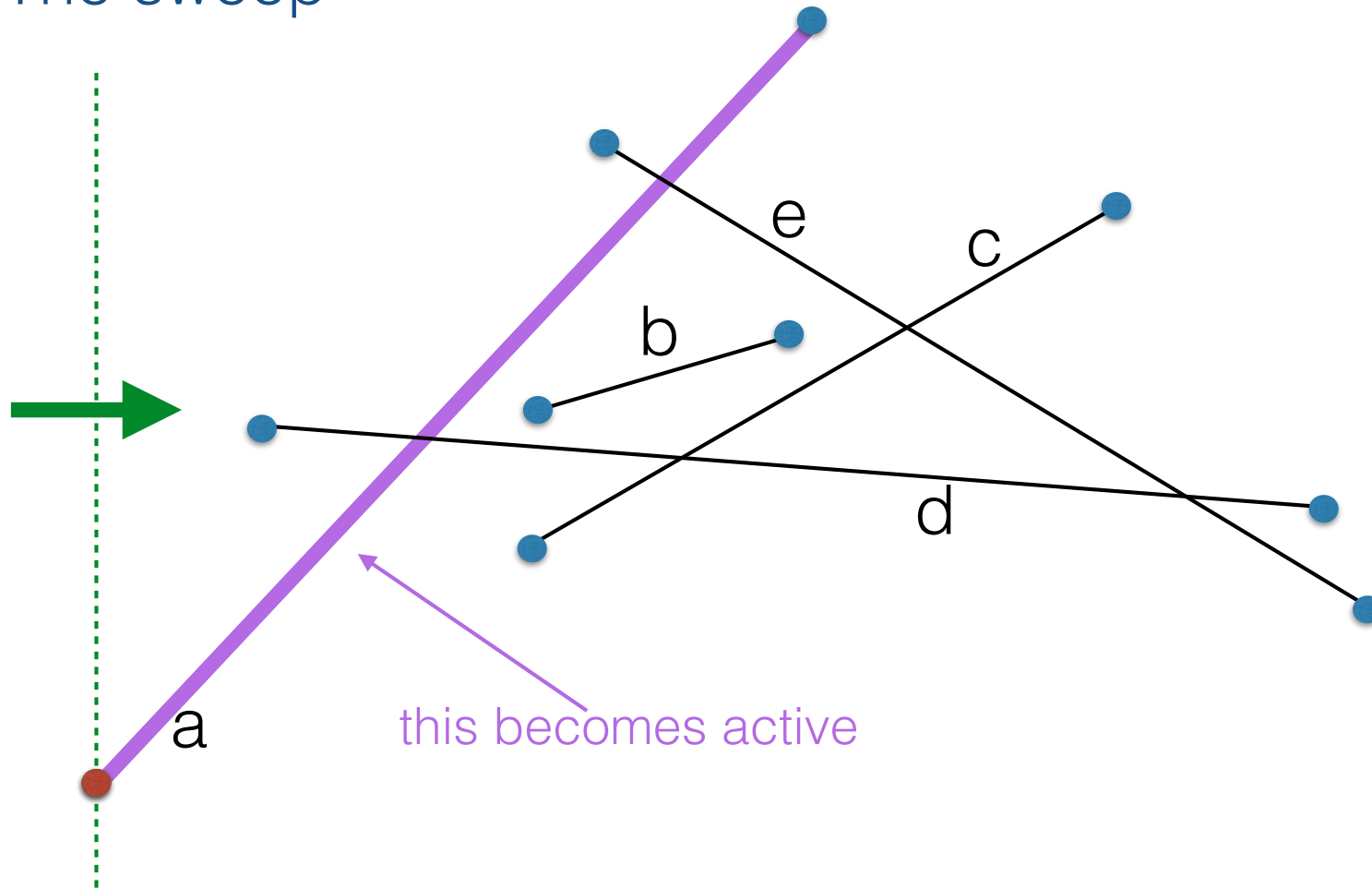
- Let X be the set of all x-coords of segments

The sweep



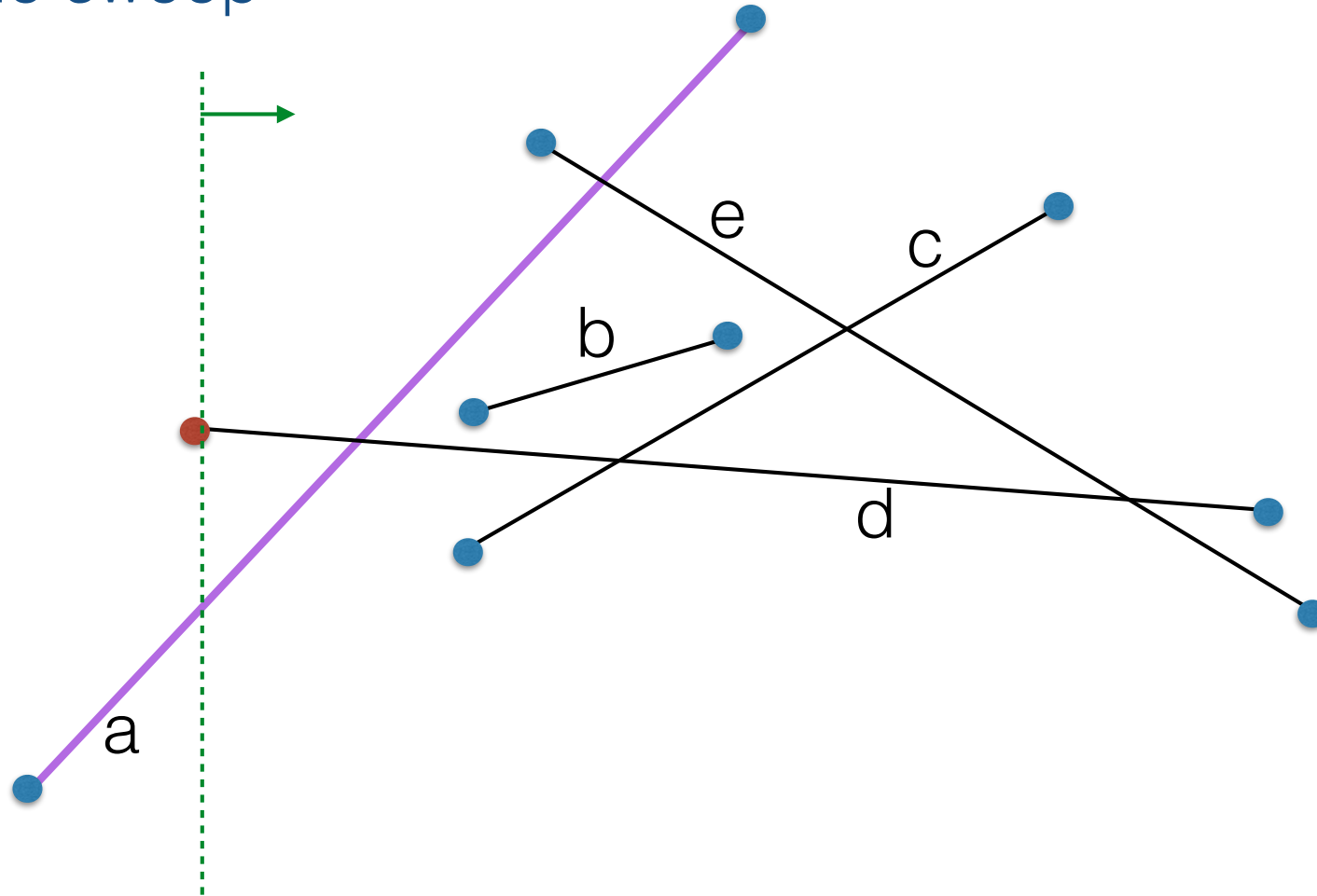
- Let X be the set of all x-coords of segments
- Traverse the events in X in order

The sweep



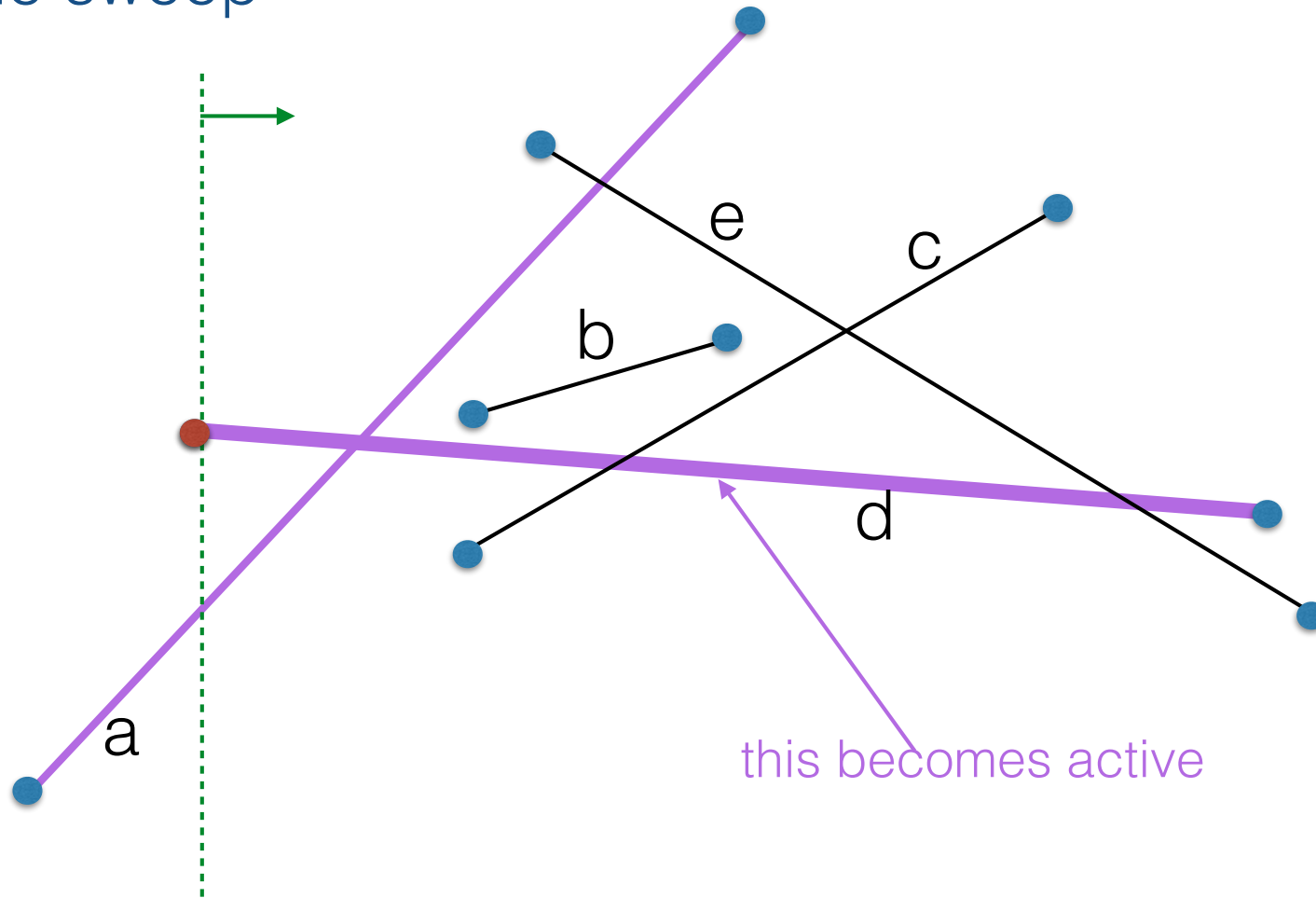
- Let X be the set of all x-coords of segments
- Traverse the events in X in order

The sweep



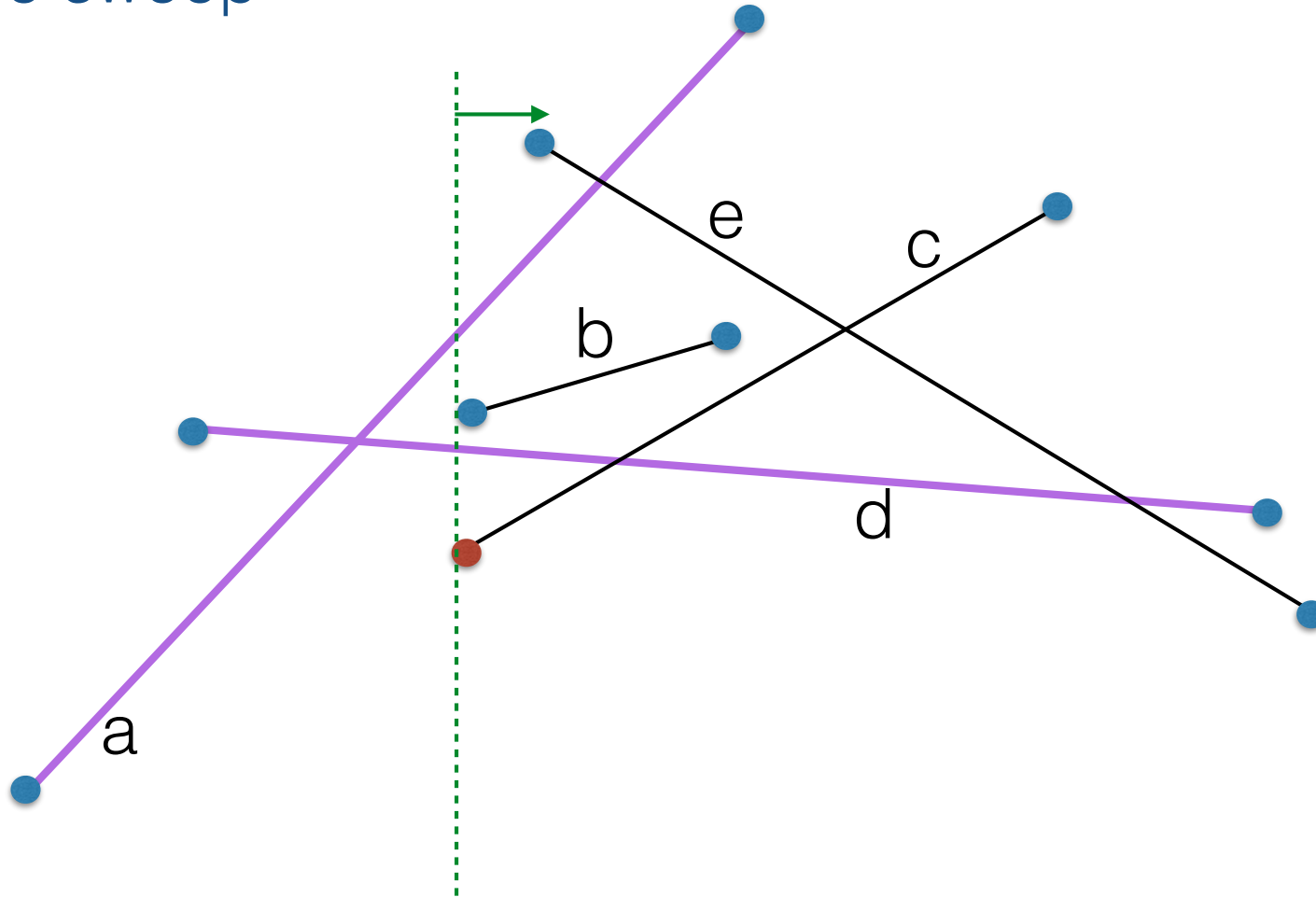
- Let X be the set of all x -coords of segments
- Traverse the events in X in order

The sweep



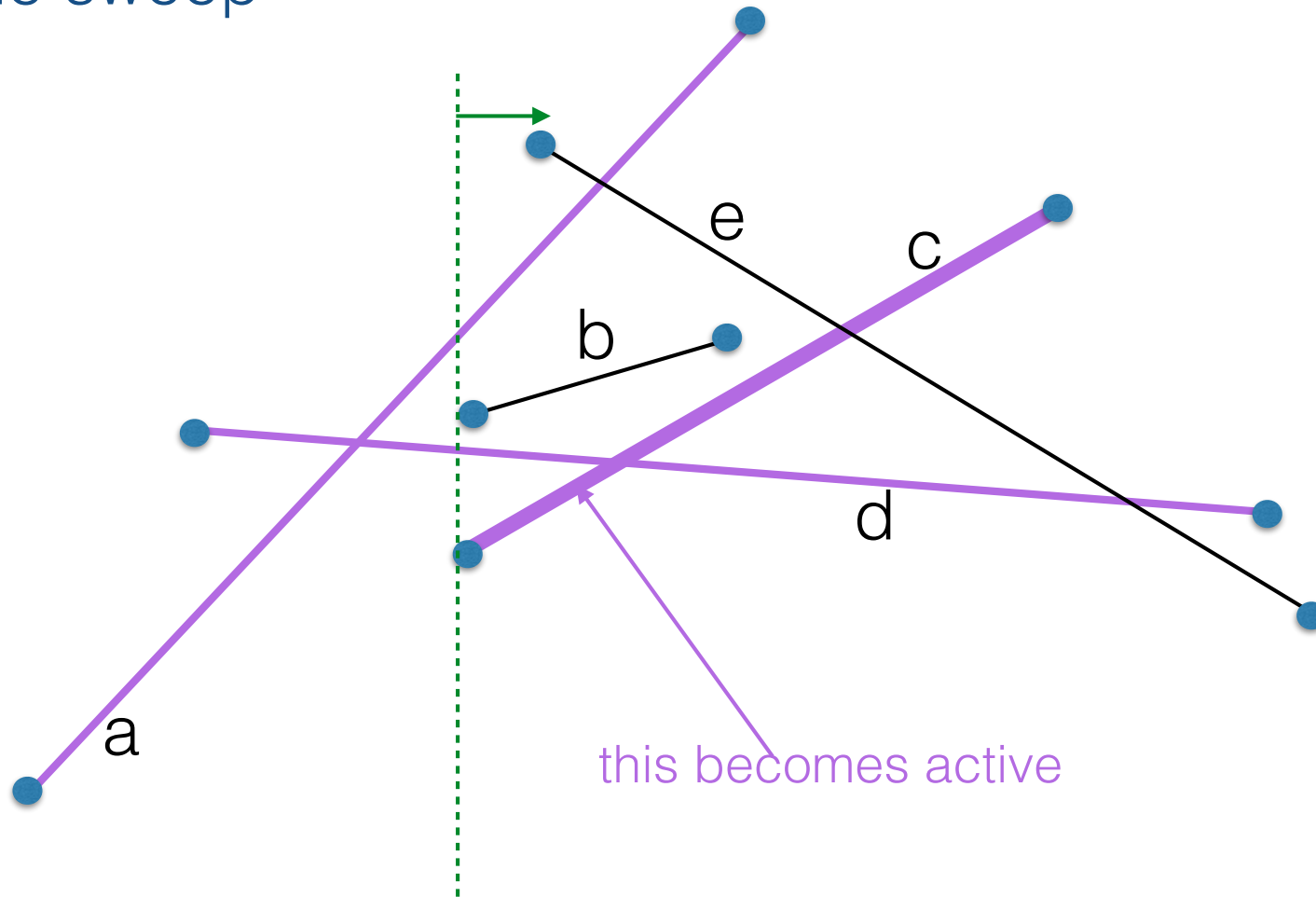
- Let X be the set of all x-coords of segments
- Traverse the events in X in order

The sweep



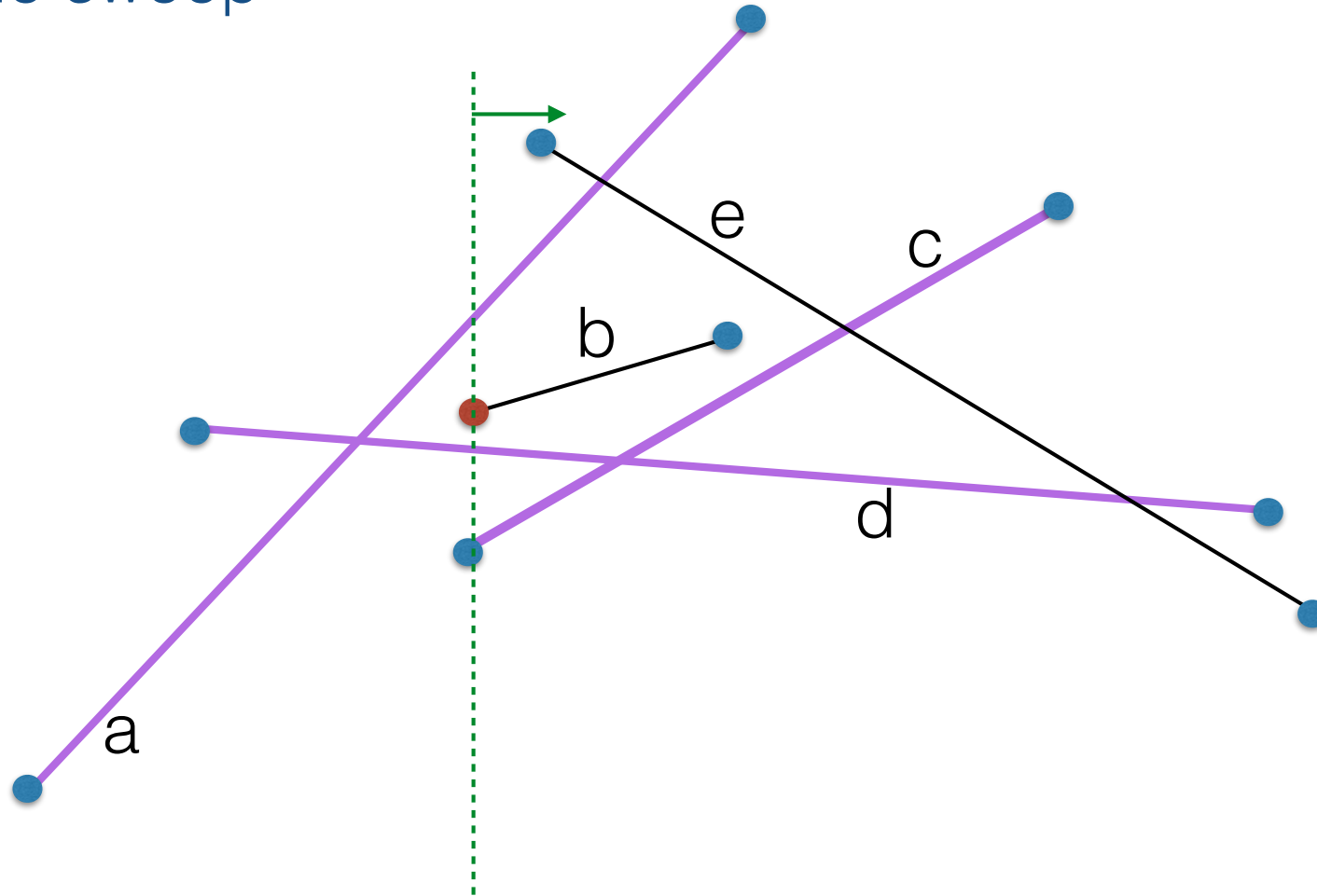
- Let X be the set of all x -coords of segments
- Traverse the events in X in order

The sweep



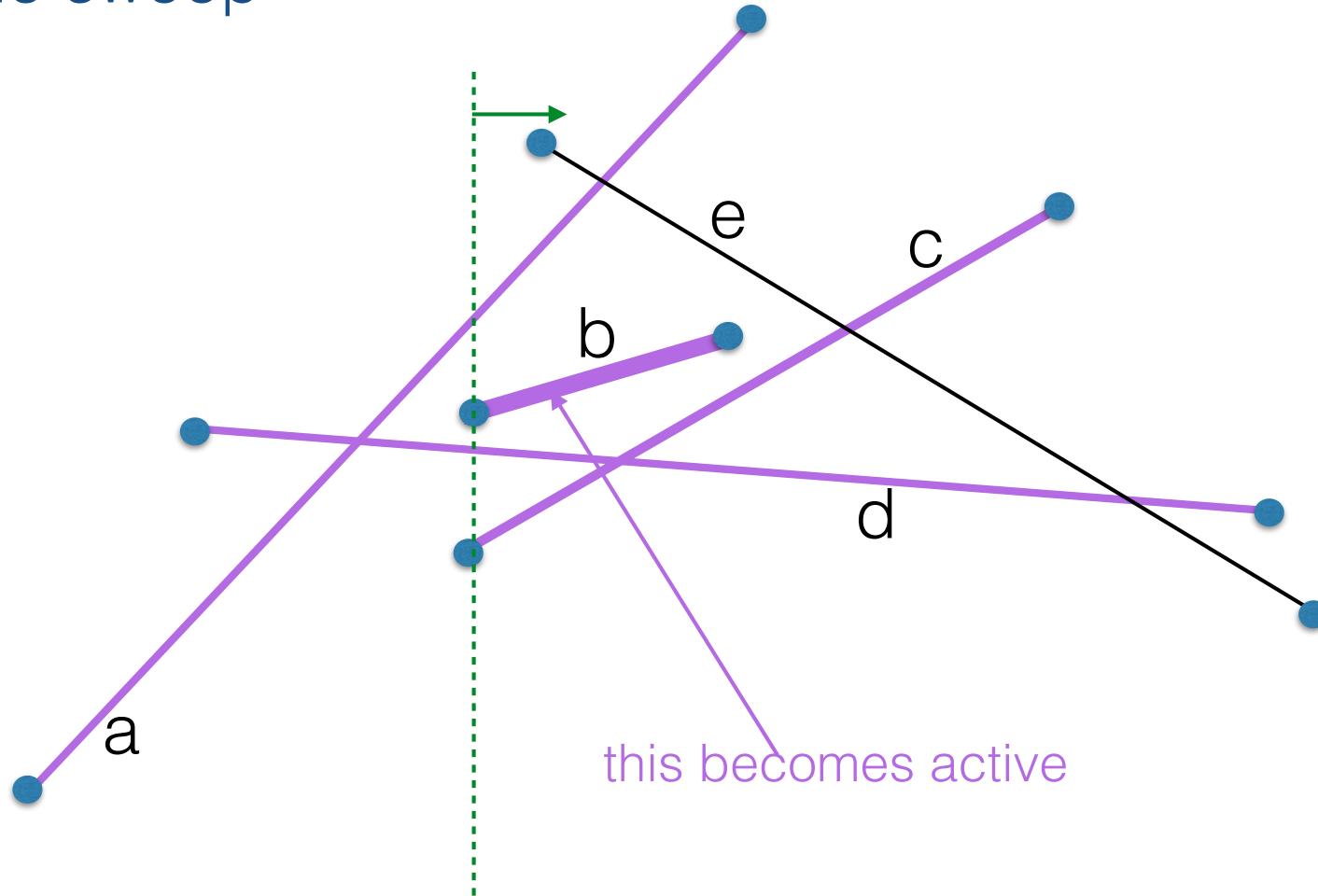
- Let X be the set of all x-coords of segments
- Traverse the events in X in order

The sweep



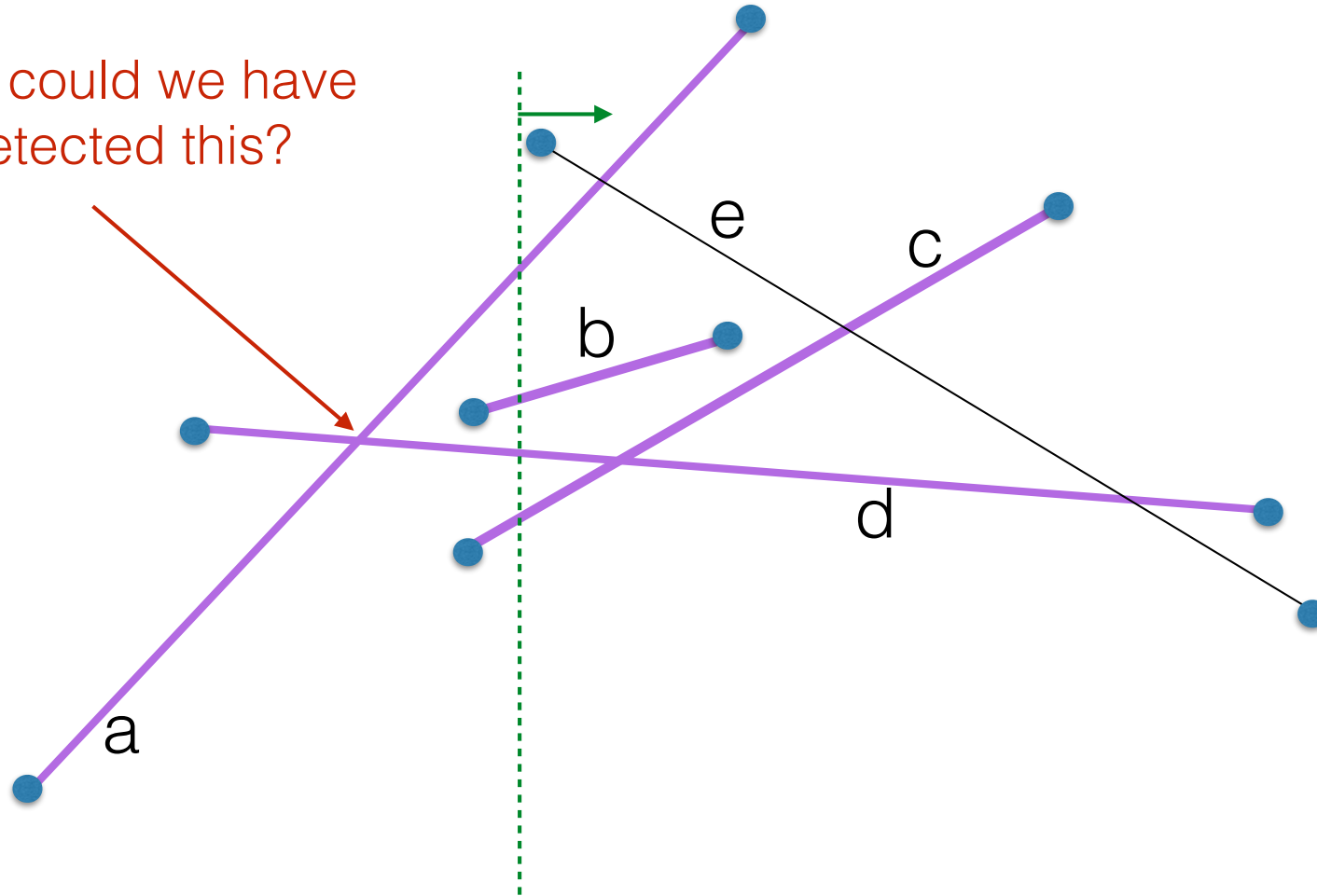
- Let X be the set of all x-coords of segments
- Traverse the events in X in order

The sweep



- Let X be the set of all x-coords of segments
- Traverse the events in X in order

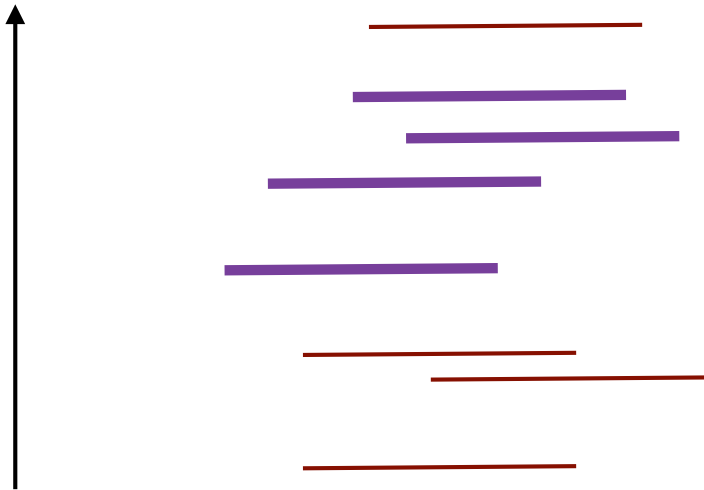
How could we have detected this?



- At this moment 4 segments are active
- How do we detect intersections?

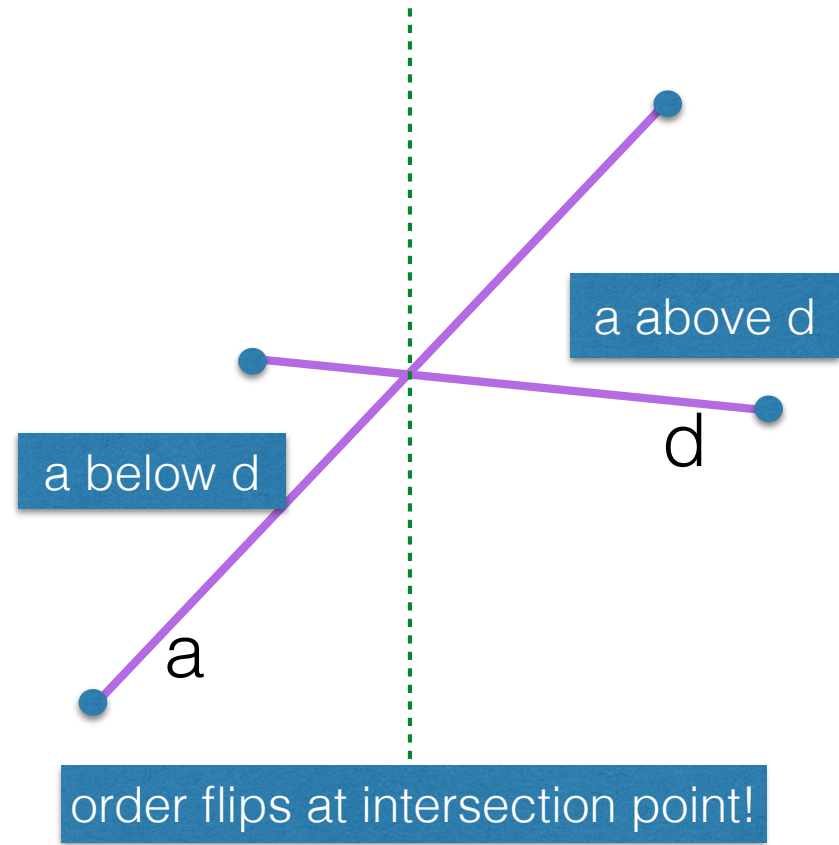
Let's step back and remember the orthogonal case....

orthogonal segments



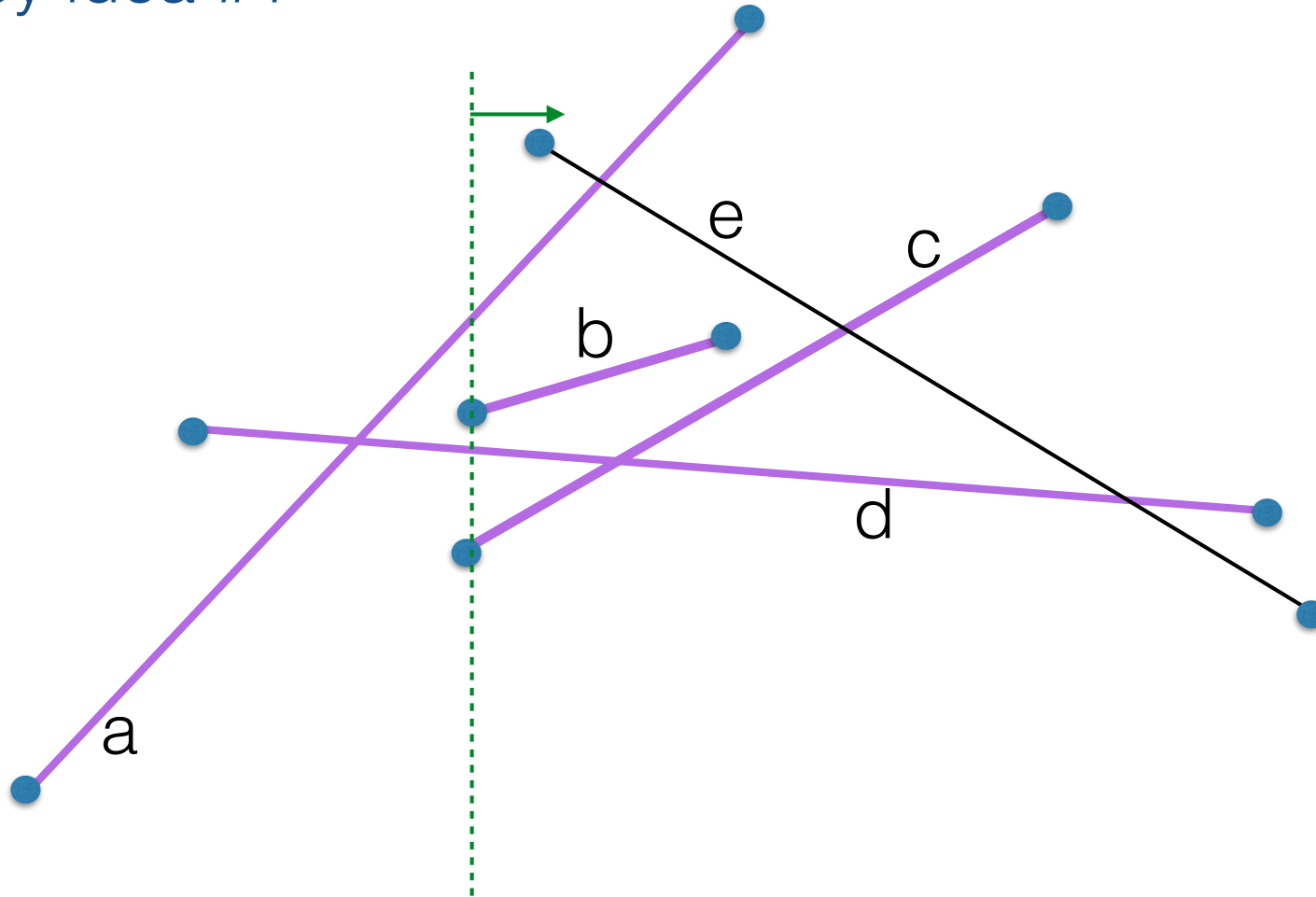
we ordered horizontal segments by y-values

general segments



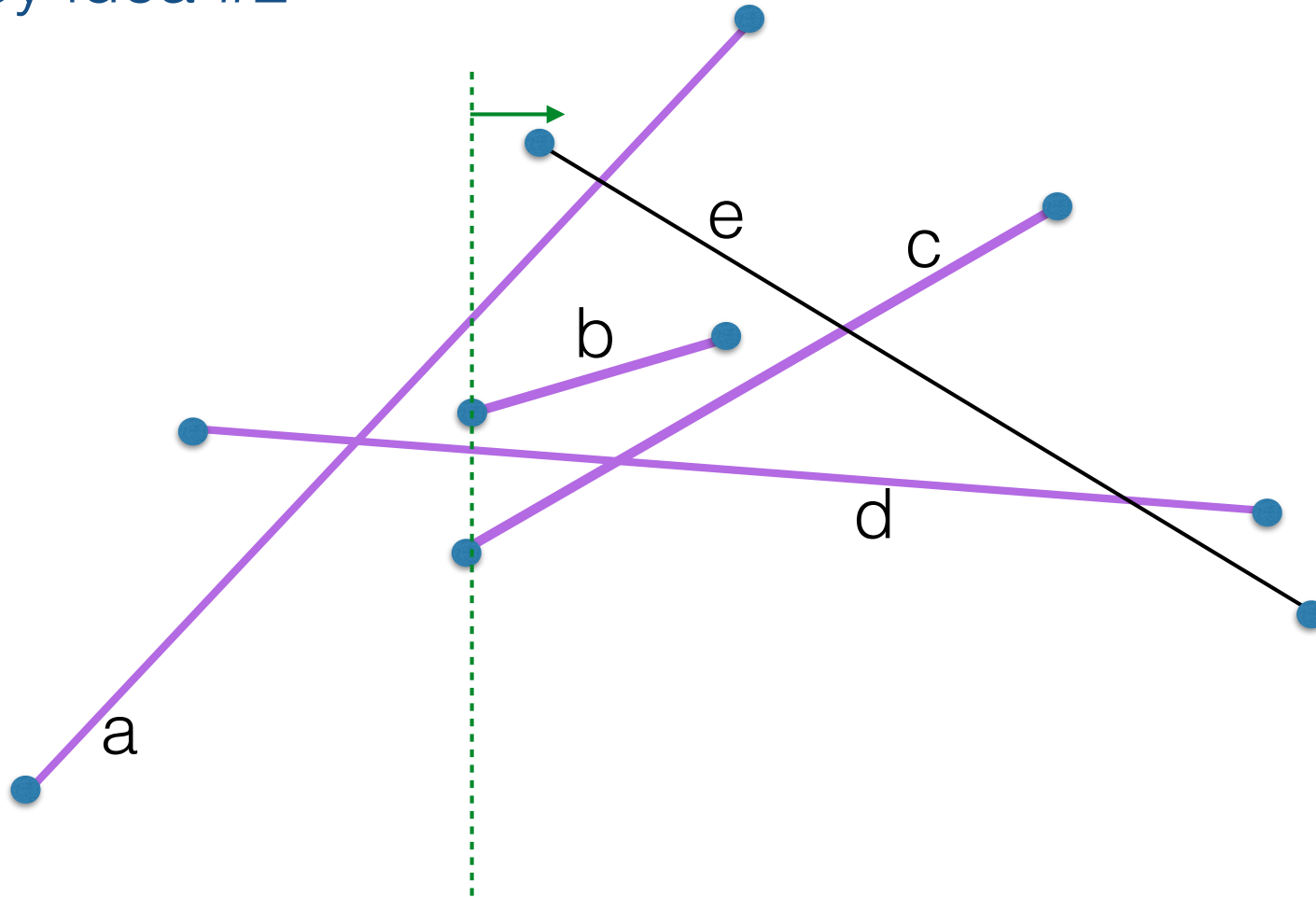
we can use above-below order

Key idea #1



- Use above-below order
- Order will flip at intersection point

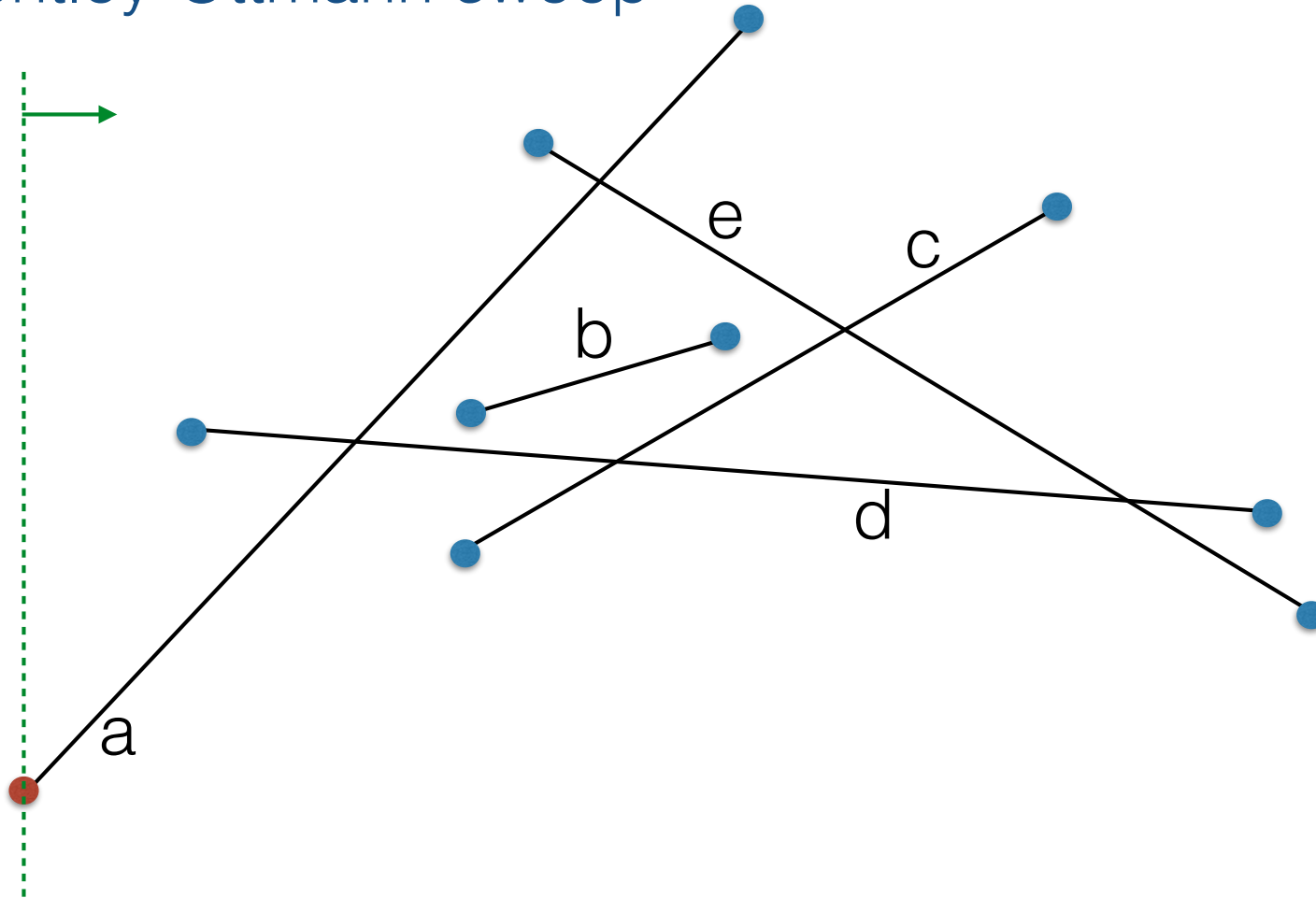
Key idea #2



- Segments that intersect are consecutive in above-below order just before they intersect

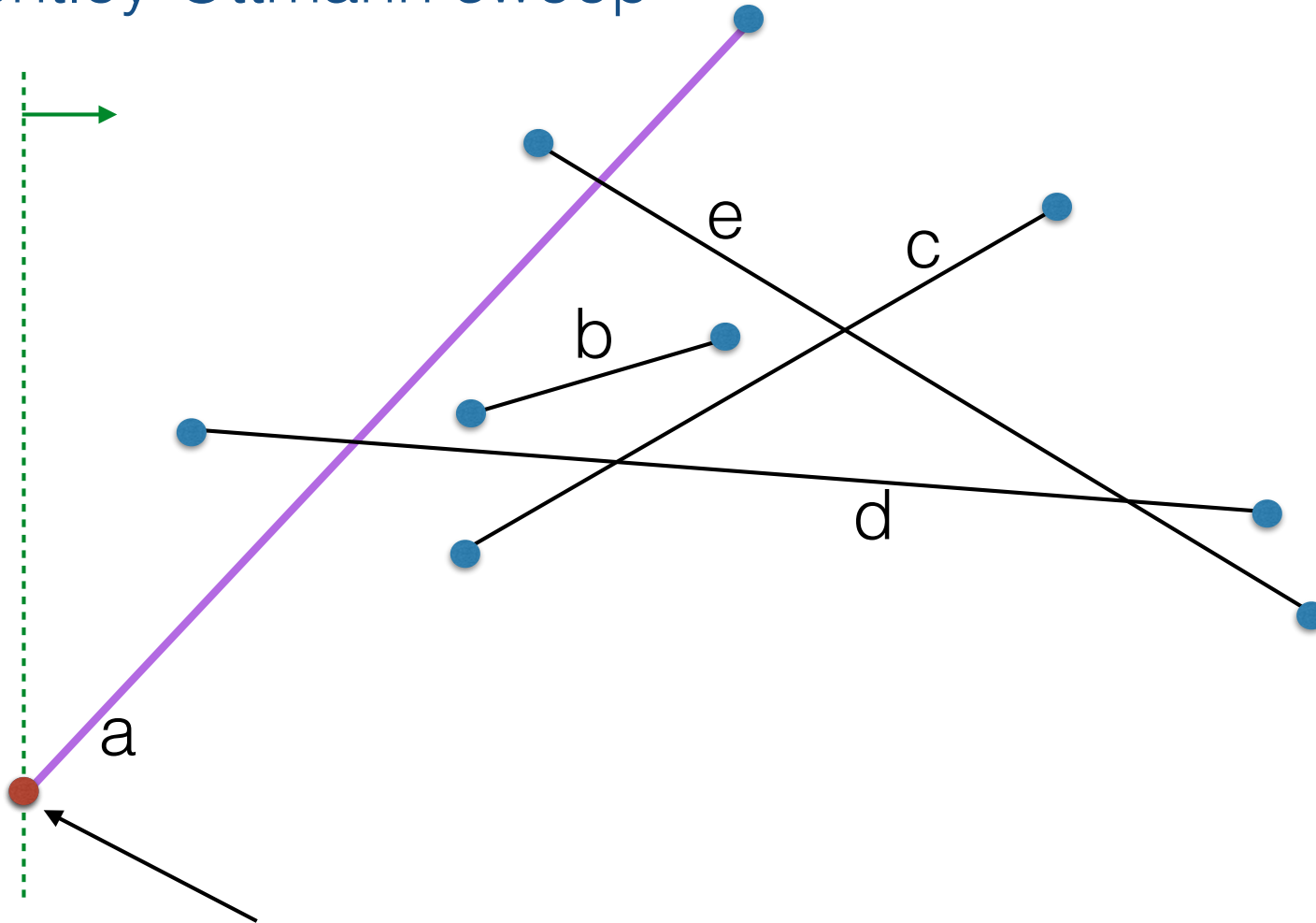
Let's start over..

Bentley-Ottmann sweep



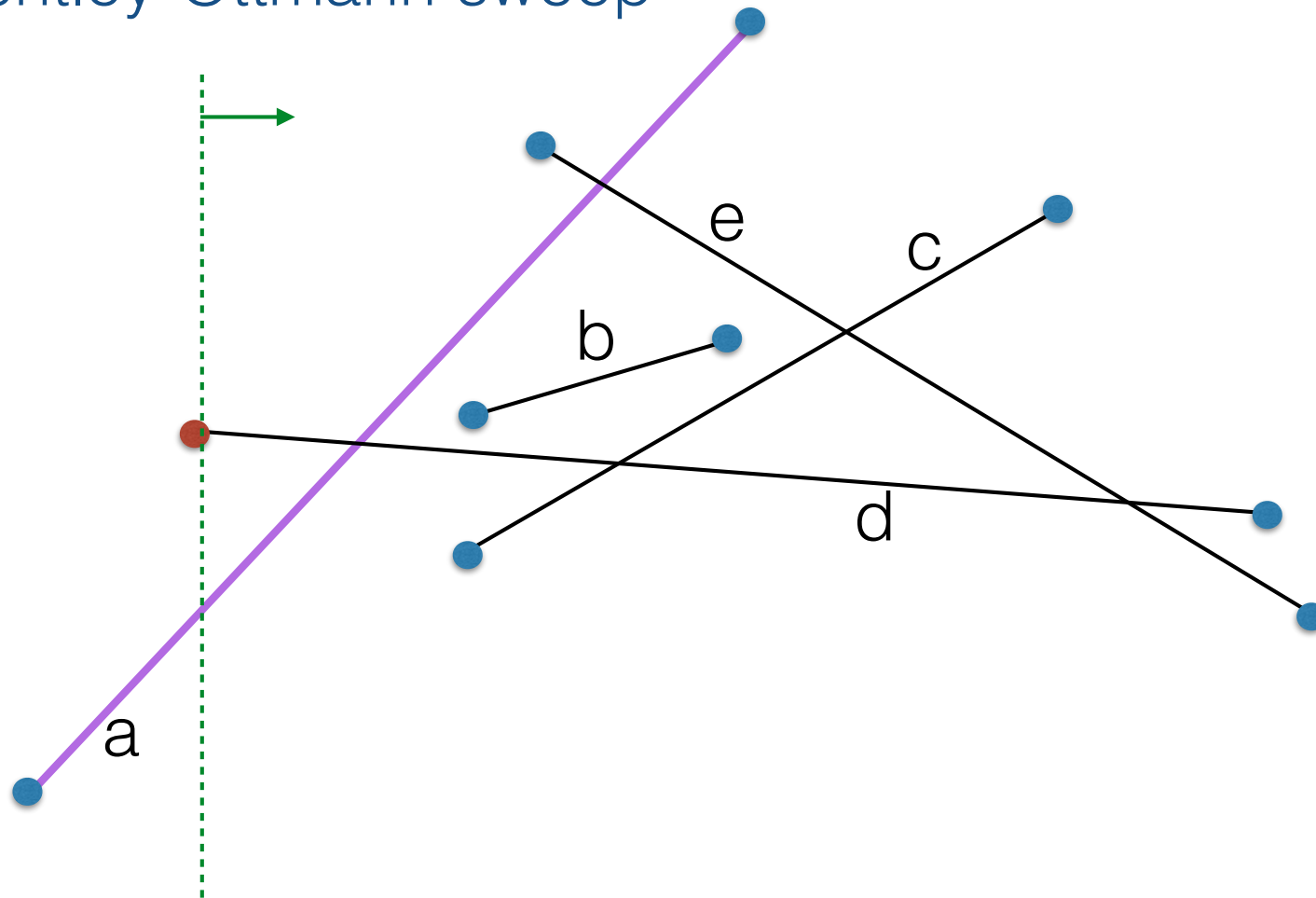
- Let X be the set of all x-coords of segments
- Initialize $AS = \{\}$
- Traverse events in order

Bentley-Ottmann sweep

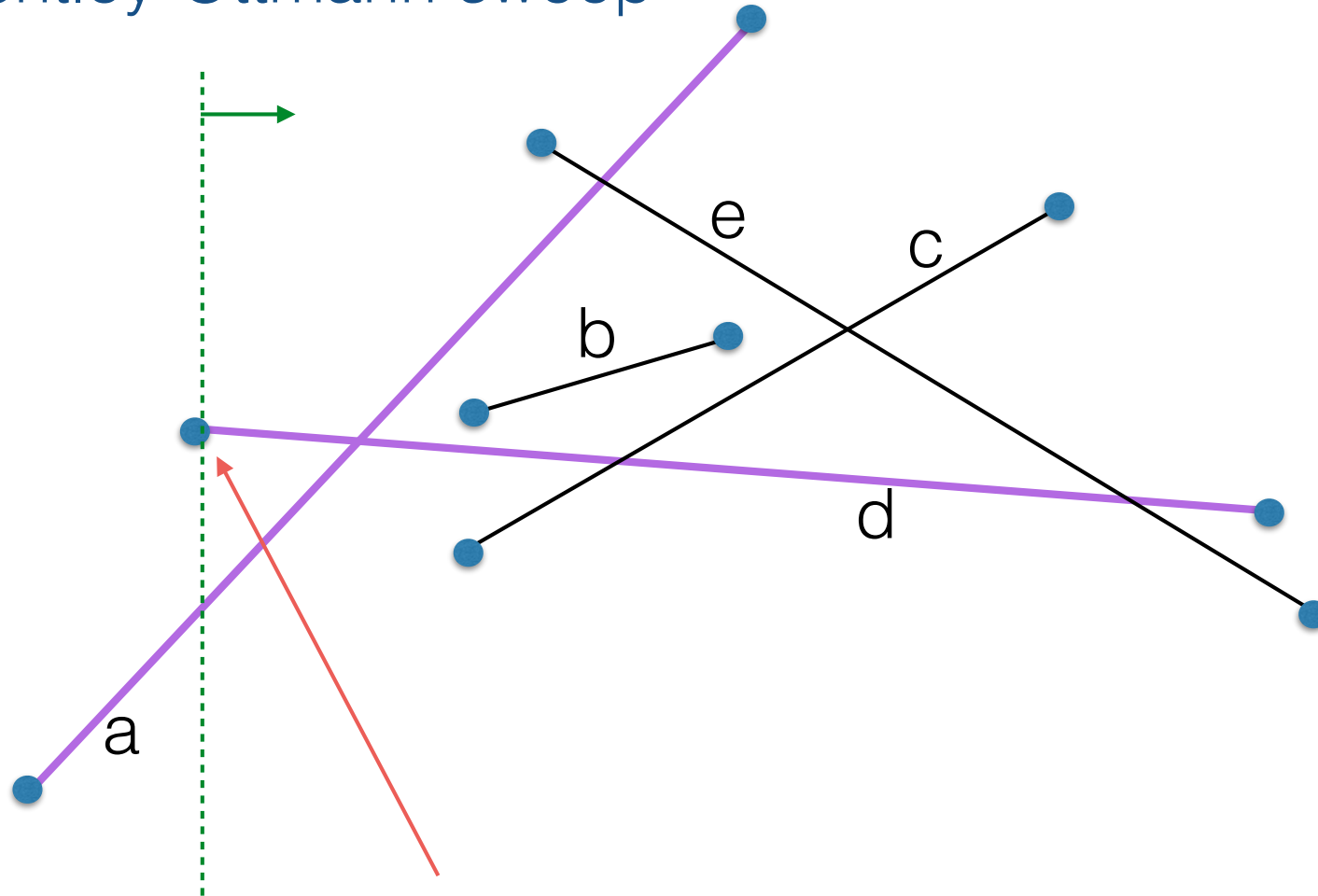


this event is start of segment a: insert a in AS

Bentley-Ottmann sweep

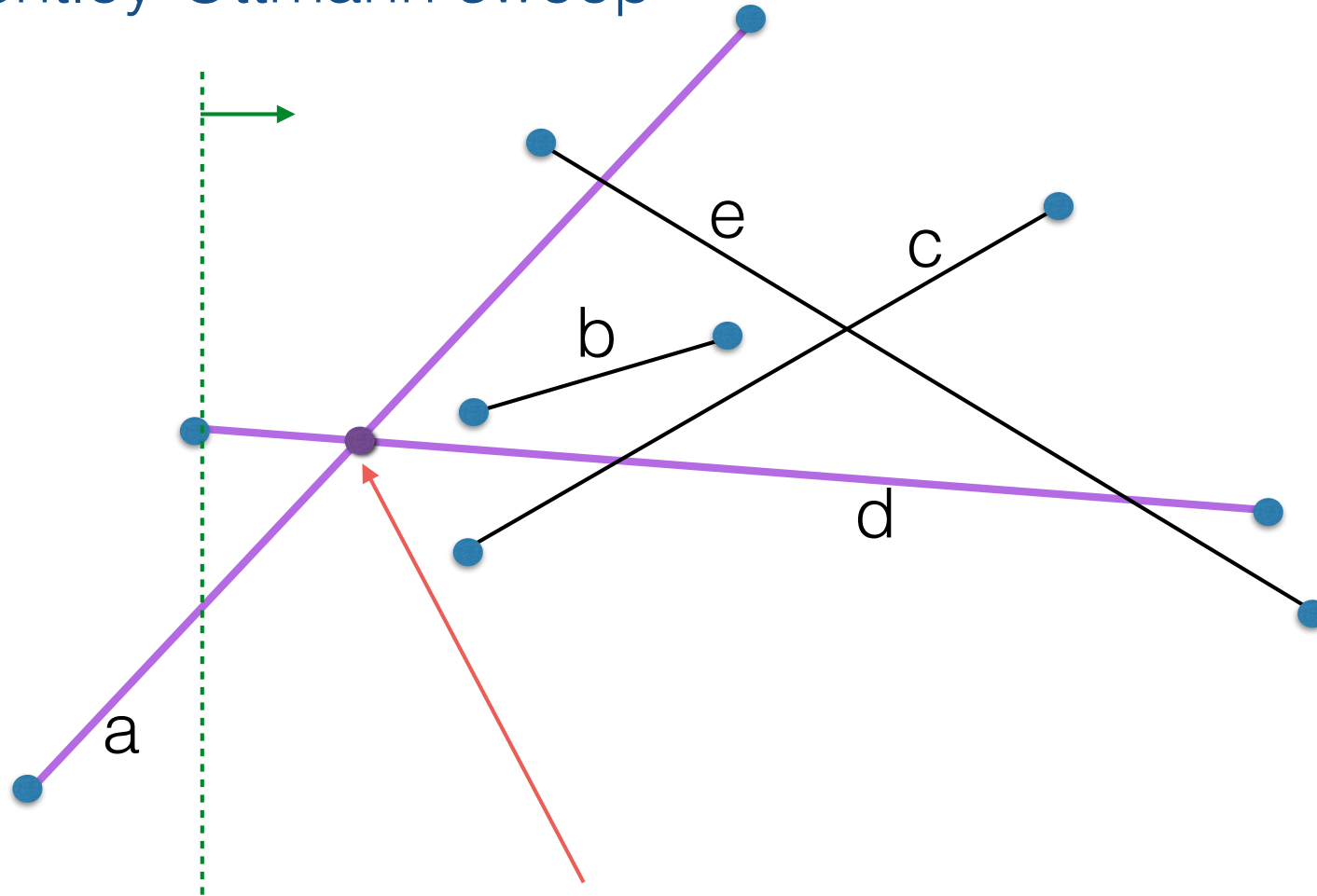


Bentley-Ottmann sweep



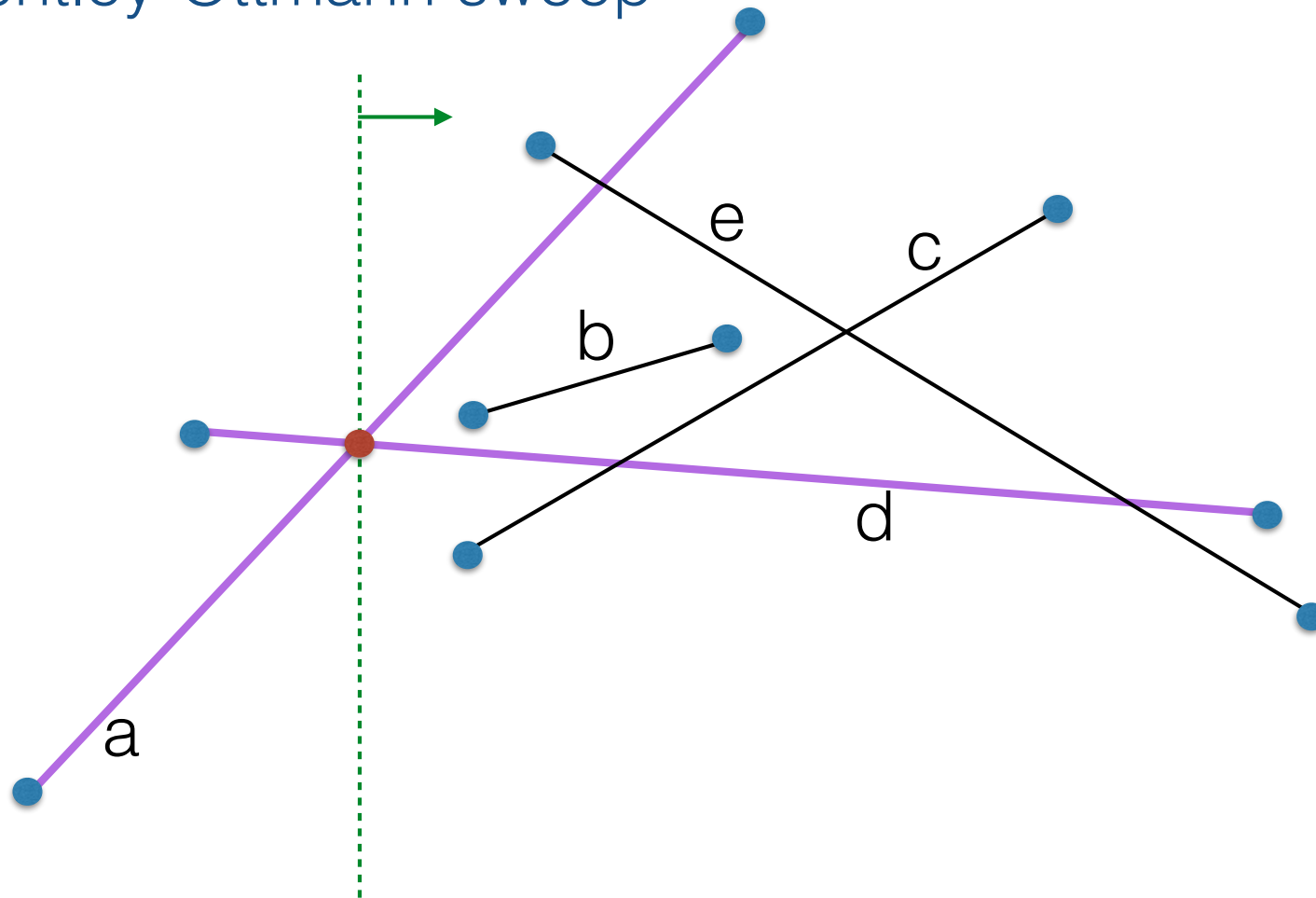
this event is start of segment d
insert d in AS: $a < d$
check if (d,a) intersect to the right of the line; they do; report point and
insert it in the list of future events

Bentley-Ottmann sweep

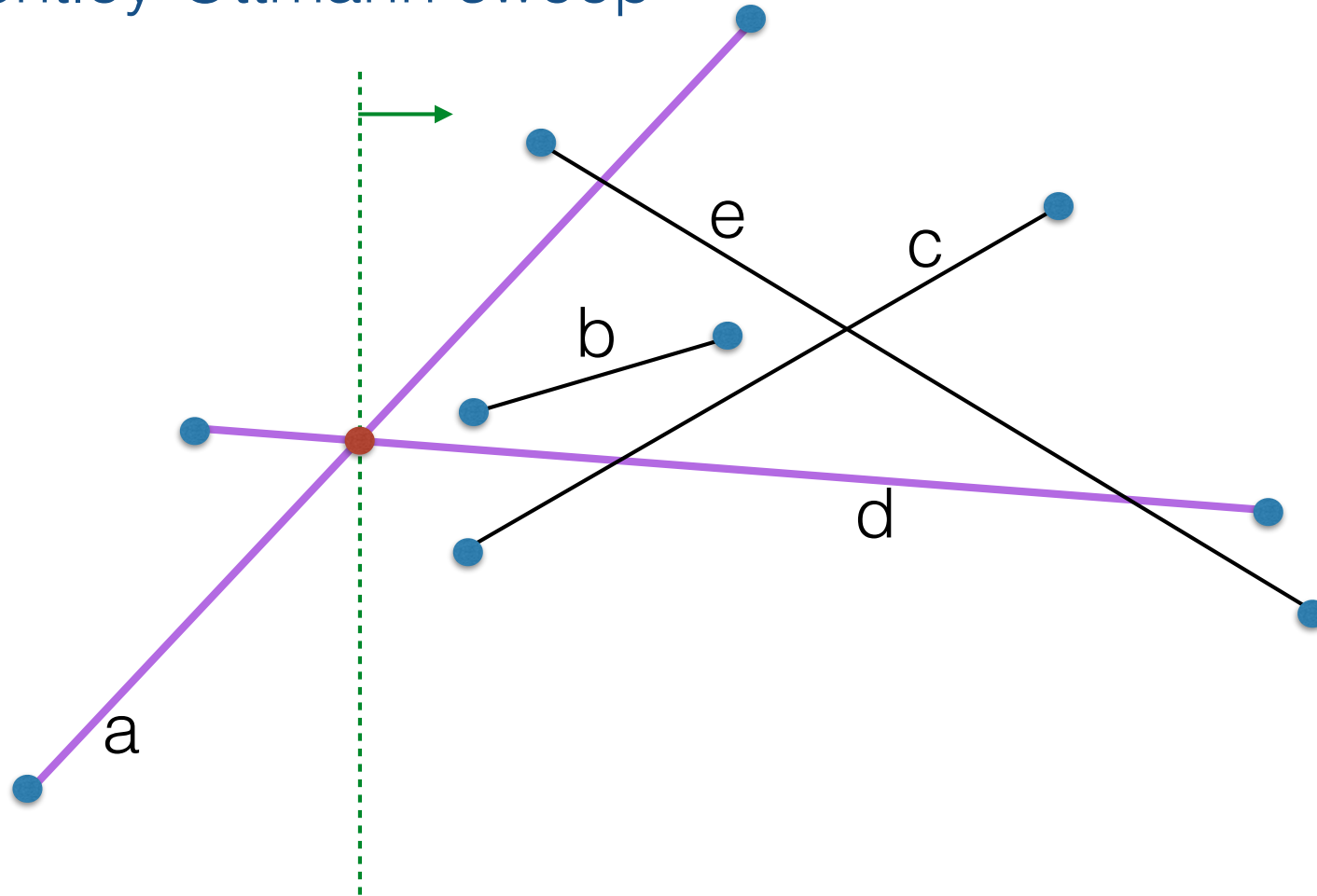


we found this when d becomes active
we insert it in event list

Bentley-Ottmann sweep

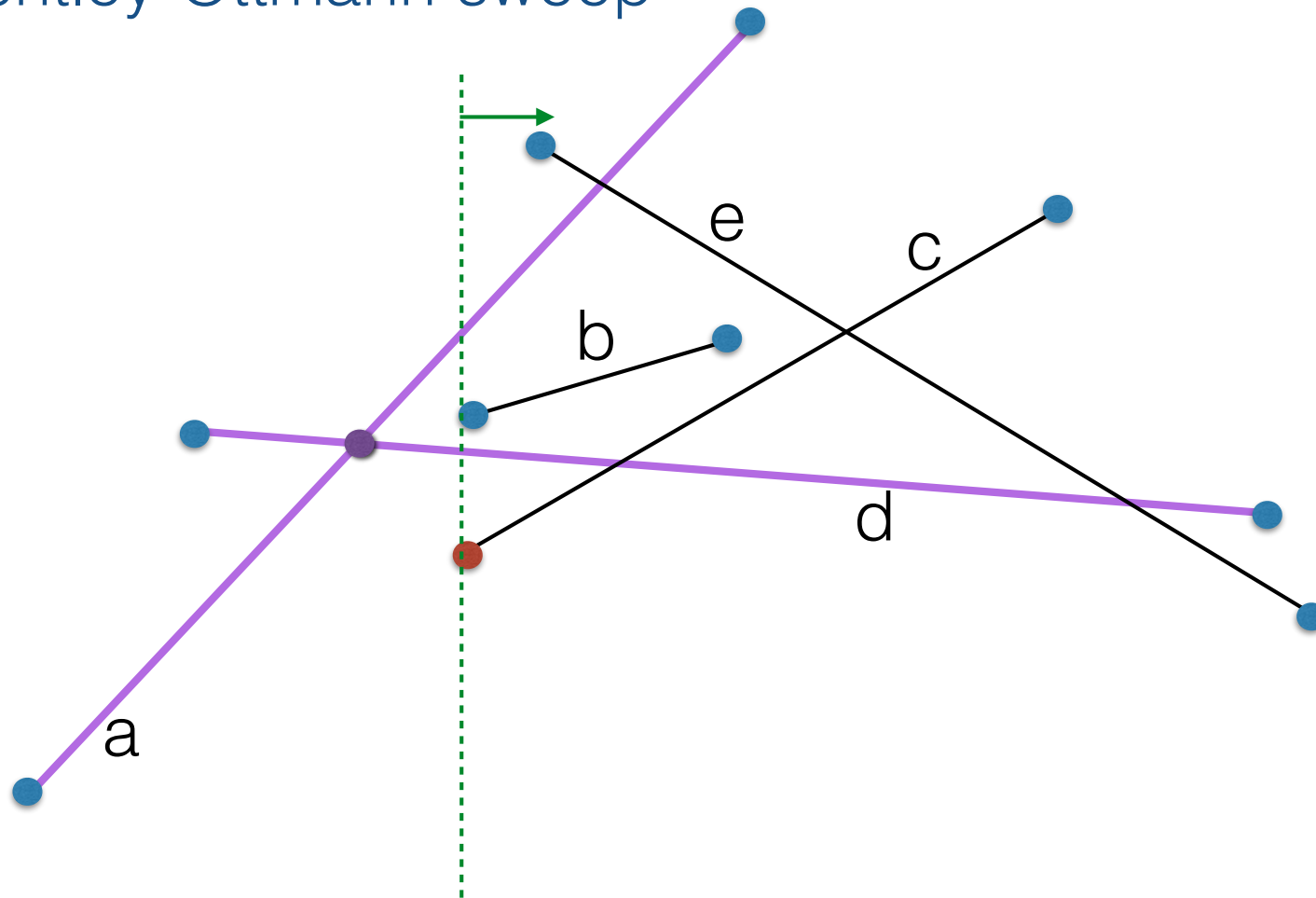


Bentley-Ottmann sweep

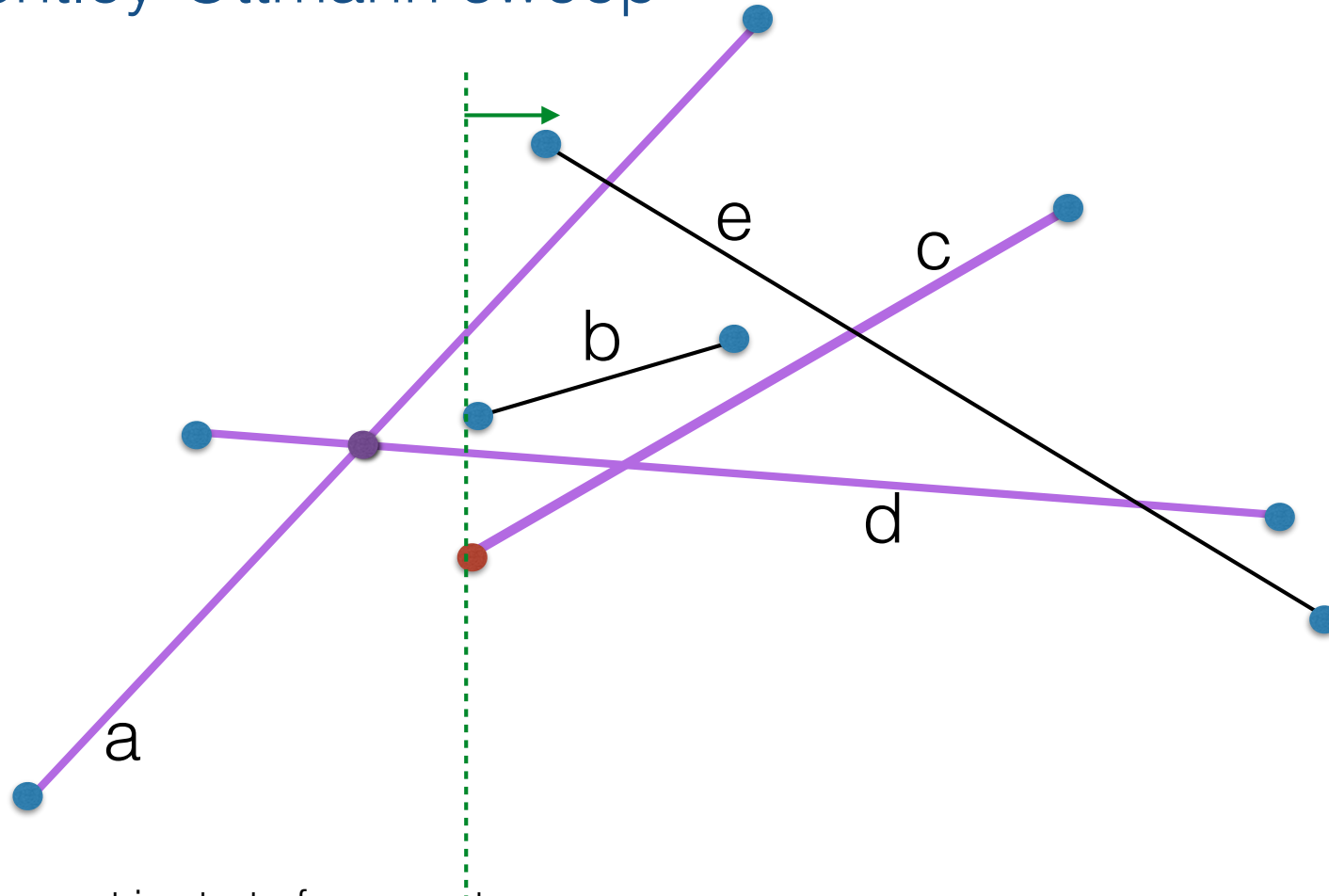


- this event is an intersection point of (a,d):
 - flip a and d in AS: a is now above d ($d < a$)

Bentley-Ottmann sweep

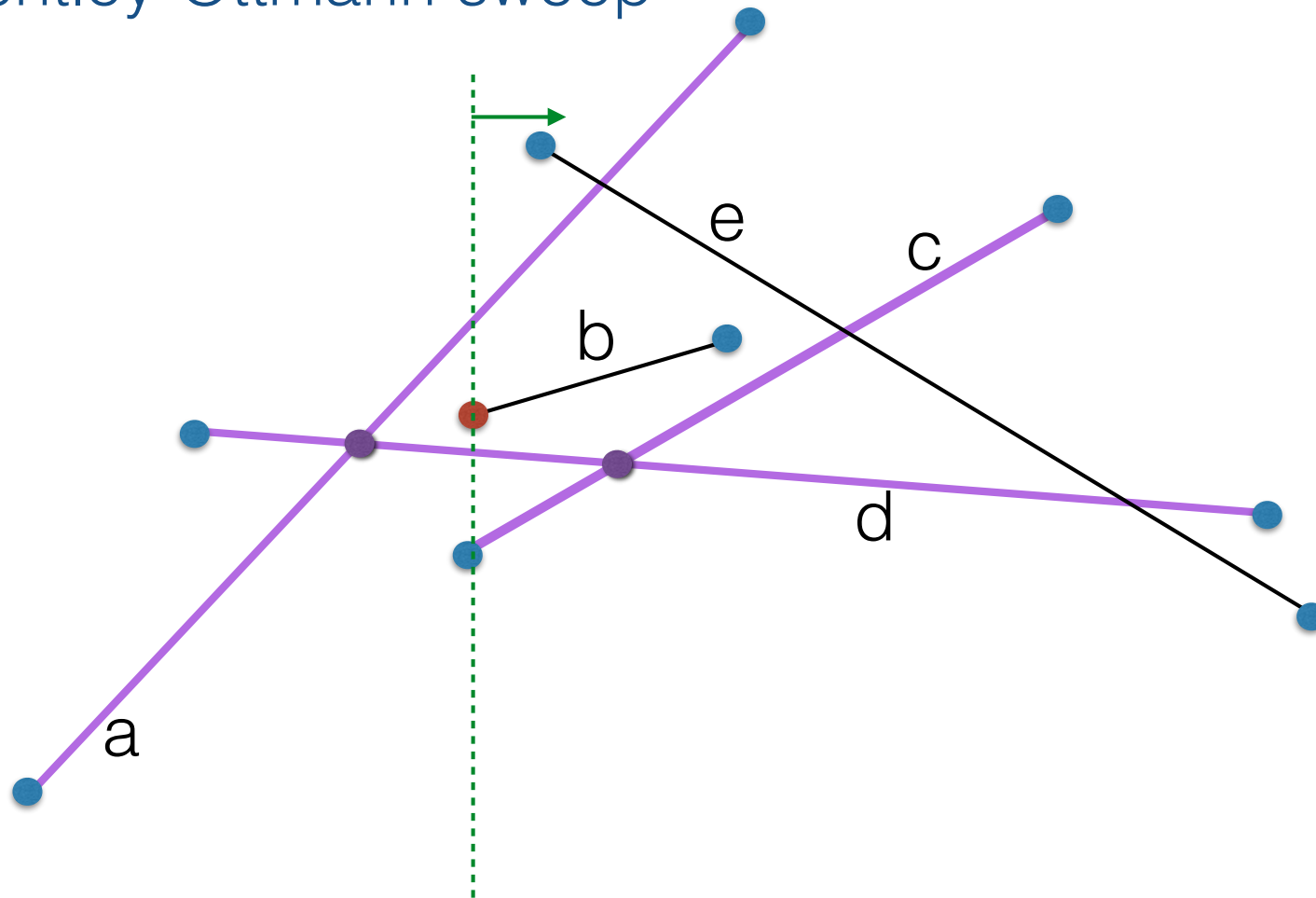


Bentley-Ottmann sweep



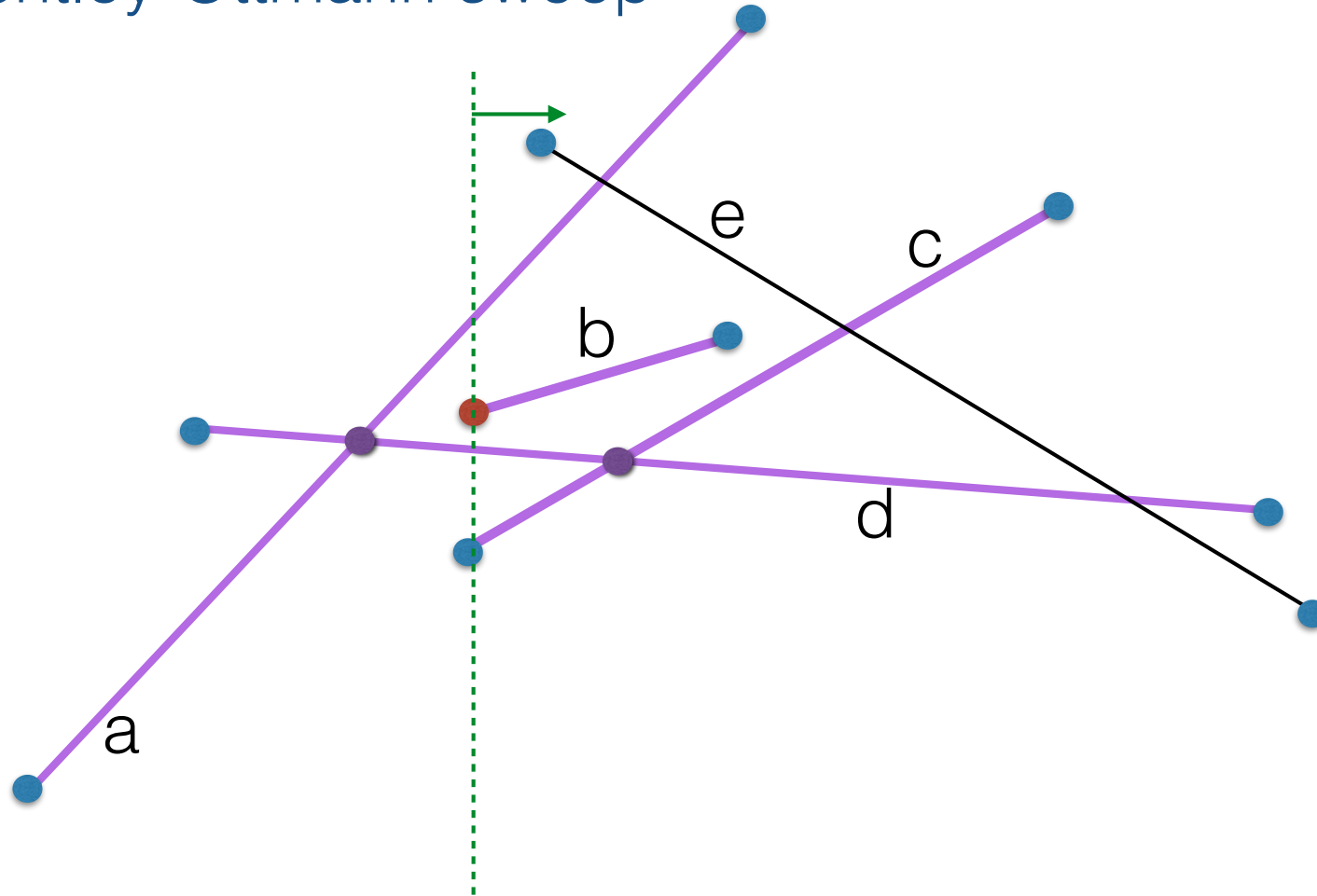
- this event is start of segment c:
 - insert c in AS; c is below d ($c < d < a$)
 - check c with its above and below neighbors for intersection to the right of the sweep line; this detects the intersection point of c and d ; report it and insert it as future event

Bentley-Ottmann sweep



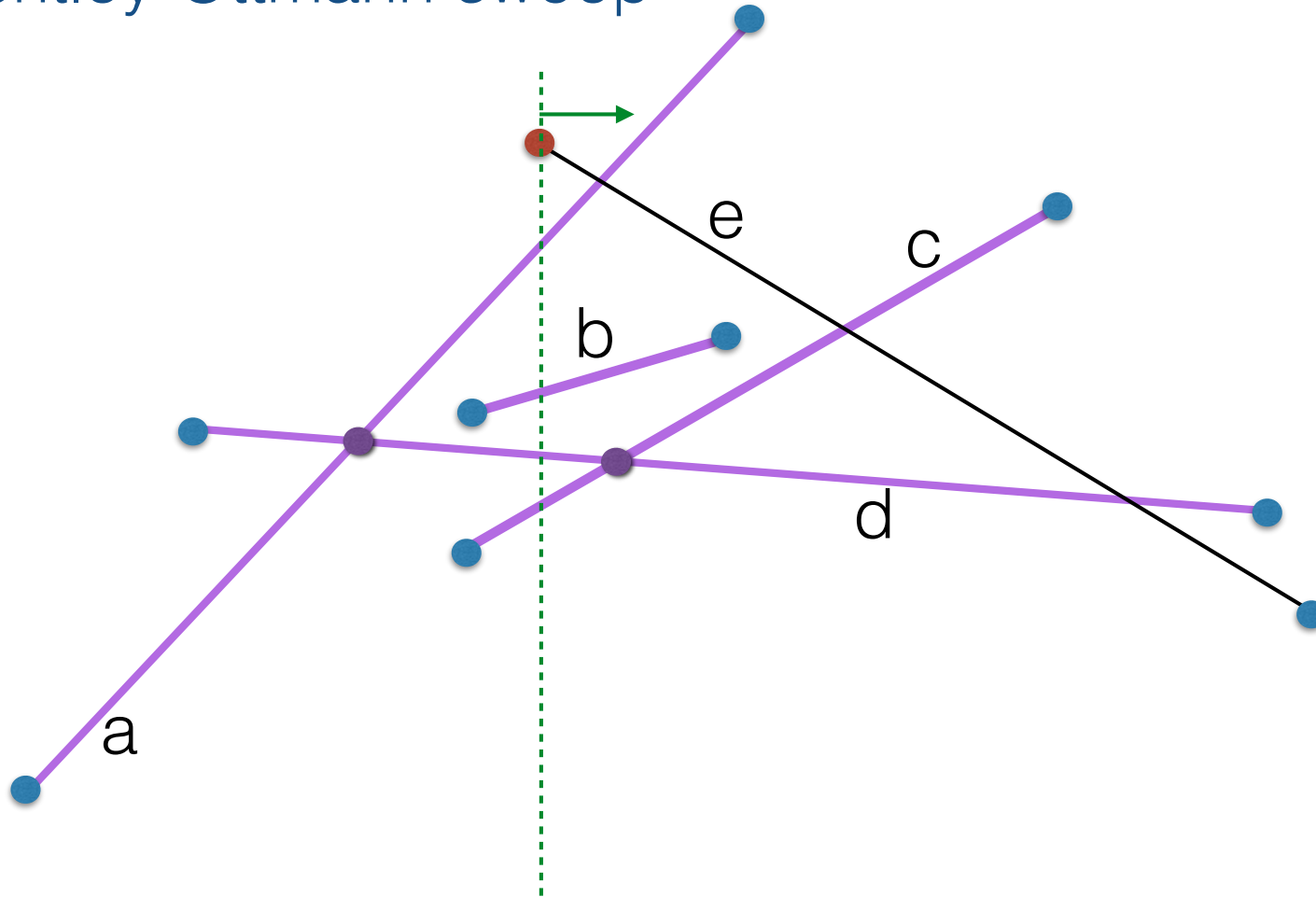
- this event is start of segment b:

Bentley-Ottmann sweep



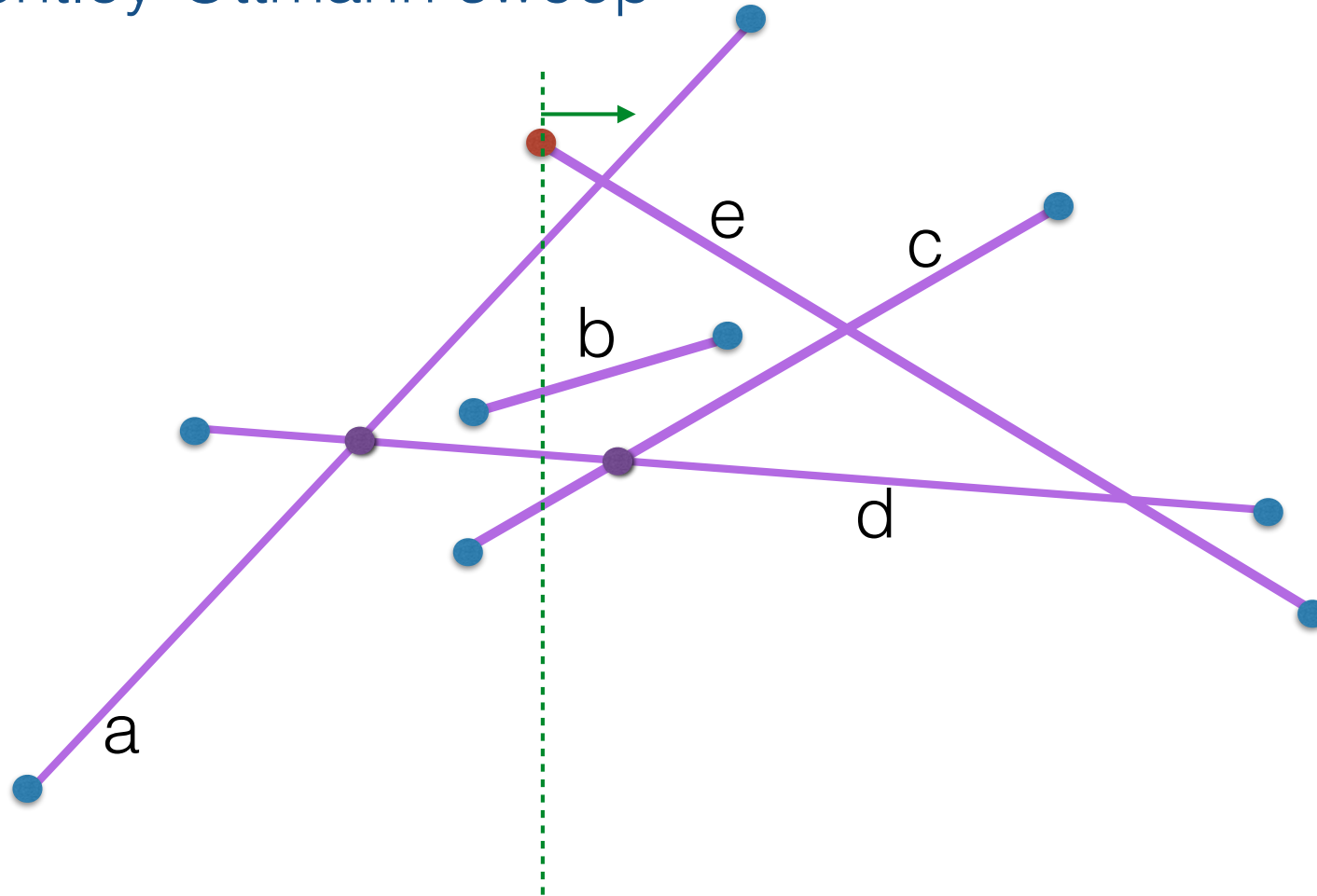
- this event is start of segment b:
 - insert b in AS; $c < d < b < a$
 - check b with its above and below neighbors for intersection to the right of the sweep line; (d,b) don't intersect; (b, a) don't intersect

Bentley-Ottmann sweep



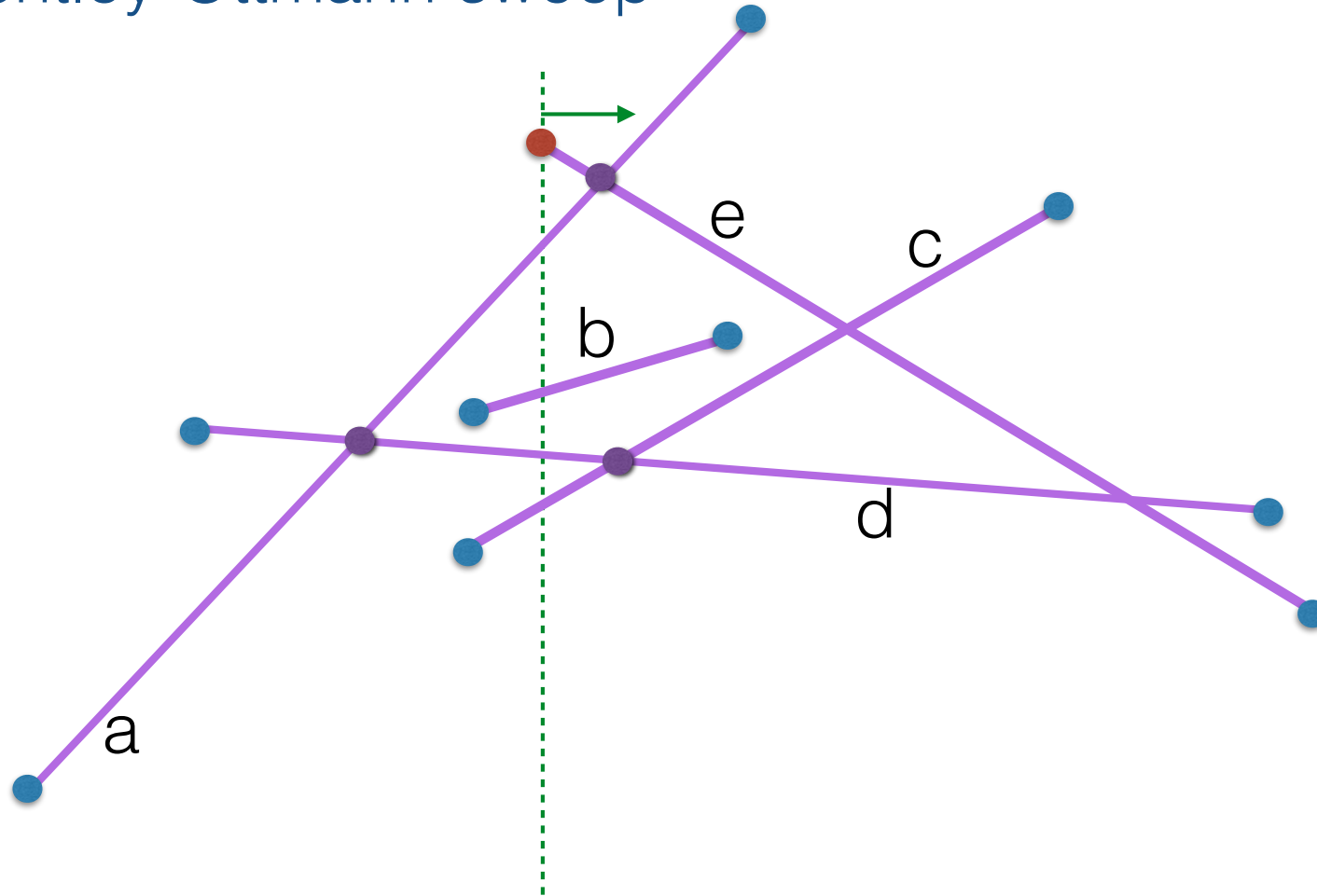
- this event is start of segment e:

Bentley-Ottmann sweep



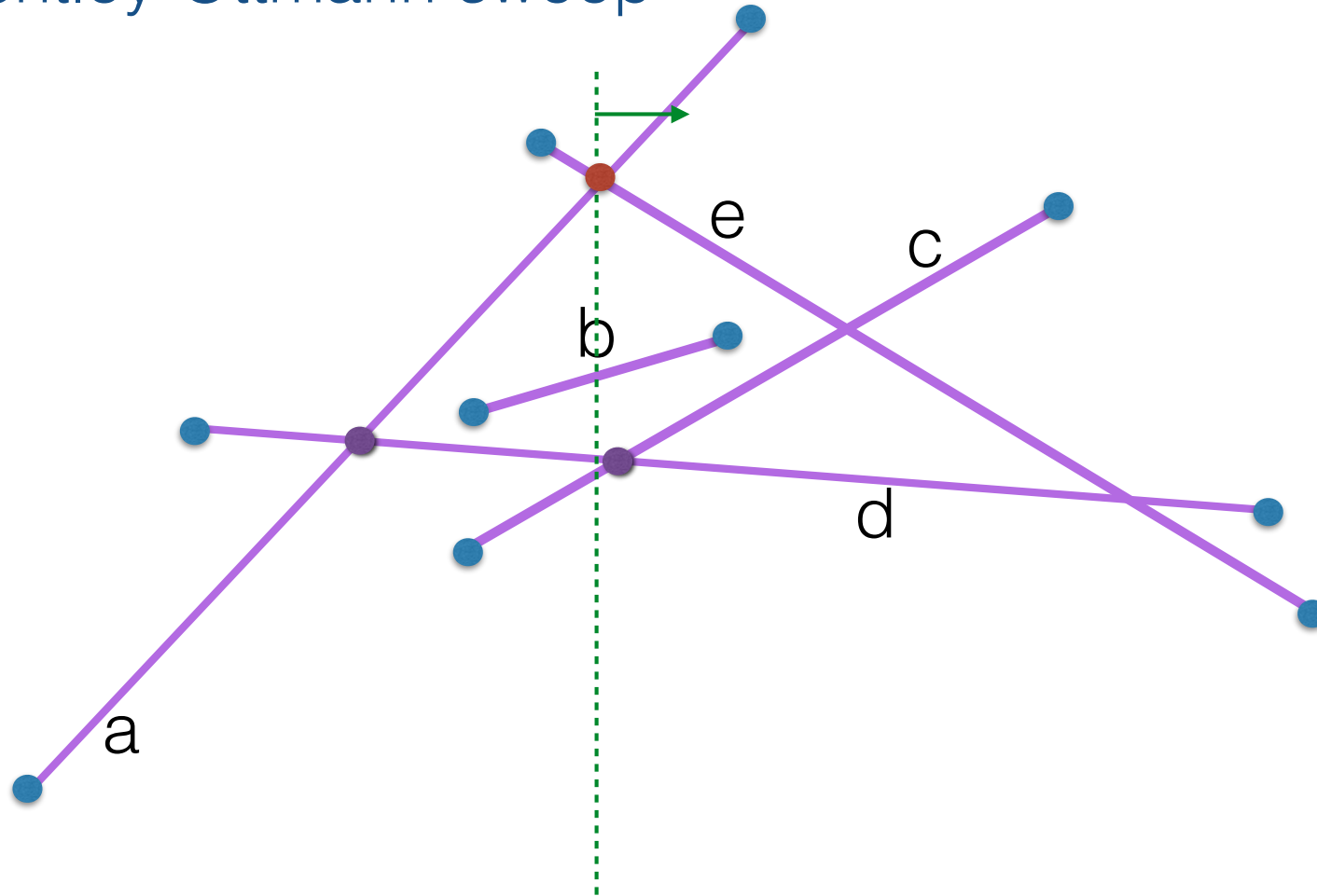
- this event is start of segment e:
 - insert e in AS: $c < d < b < a < e$
 - check e with its above and below neighbors for intersection to the right of the sweep line; this detects intersection point of (a,e); report it and insert it as future event

Bentley-Ottmann sweep



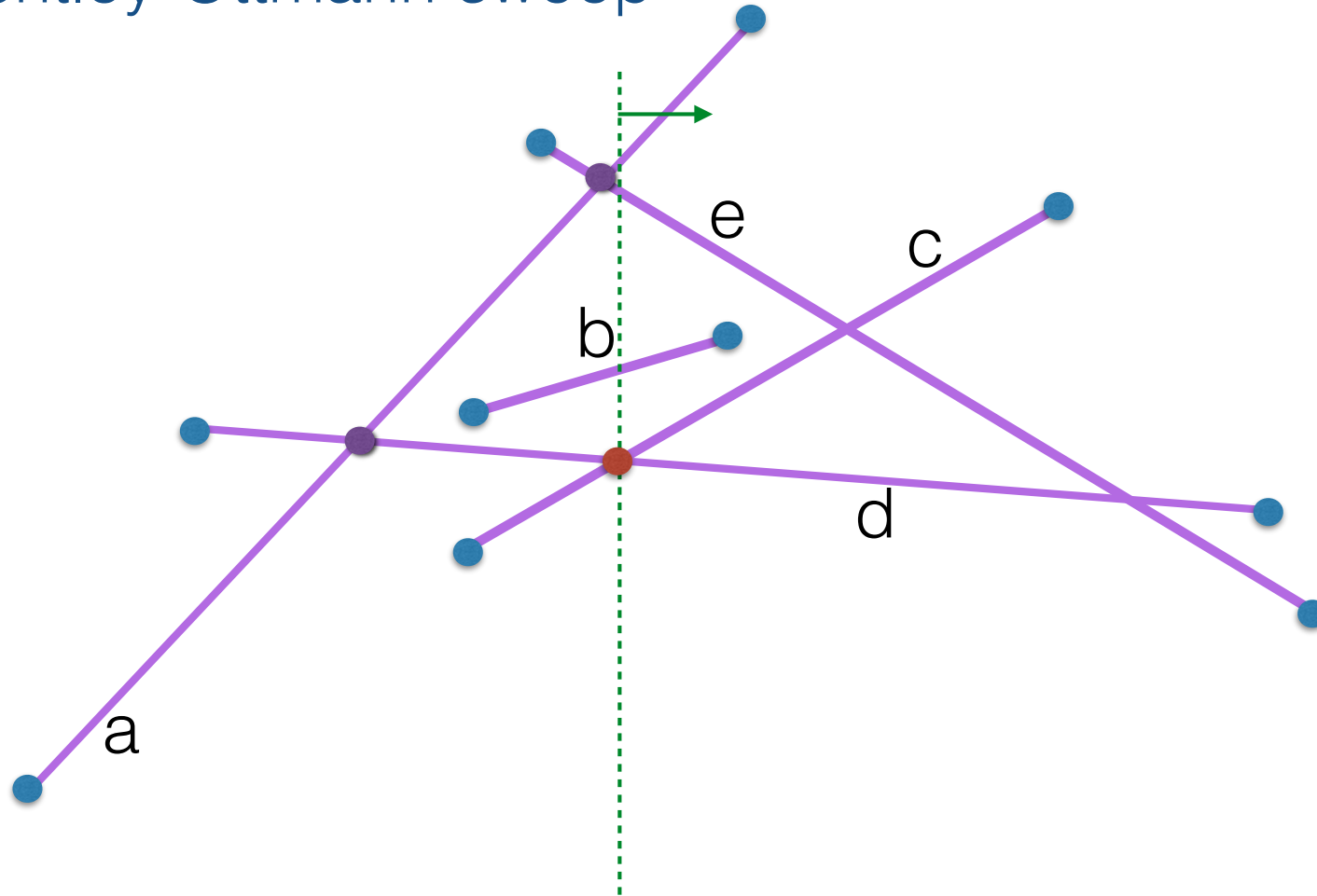
- this event is start of segment e:
 - insert e in AS: $c < d < b < a < e$
 - check e with its above and below neighbors for intersection to the right of the sweep line; this detects intersection point of (a,e); report it and insert it as future event

Bentley-Ottmann sweep



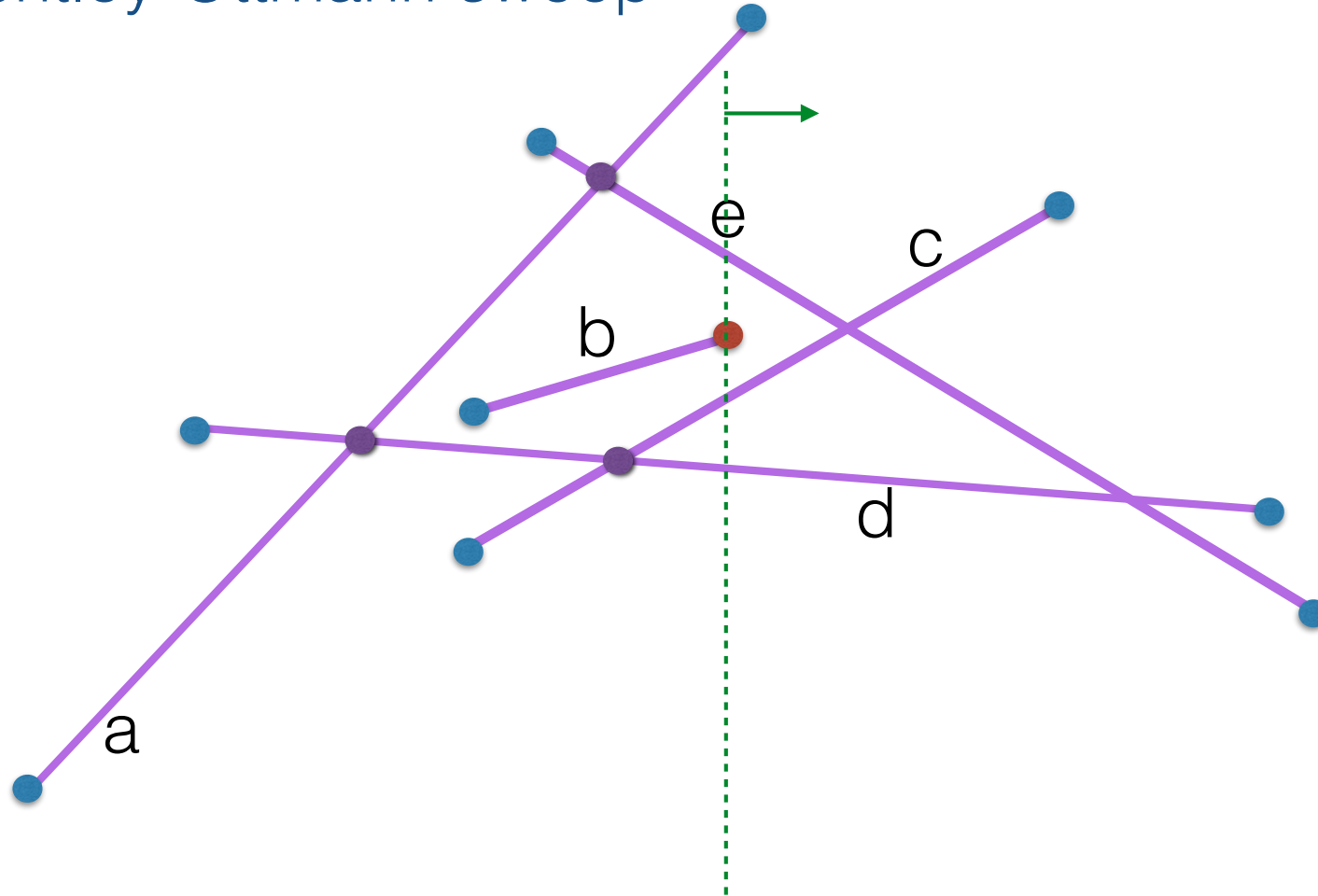
- this event is intersection of (a,e):
 - flip a and e: $c < d < b < e < a$
 - check new neighbors (e,b) for intersection to the right of the sweep line; (e,b) don't intersect

Bentley-Ottmann sweep



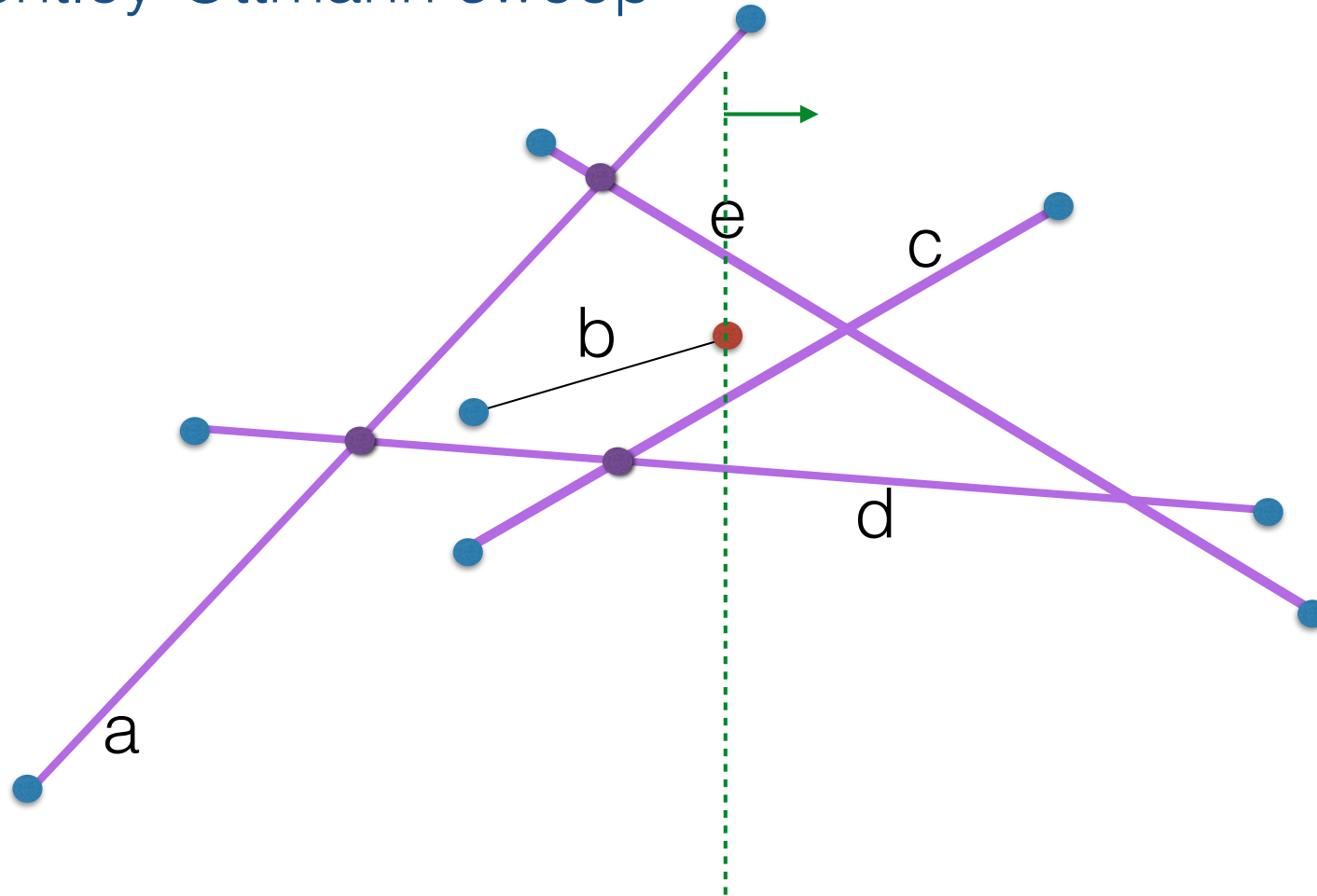
- this event is intersection of (c,d):
 - flip c and d: $d < c < b < e < a$
 - check new neighbors (c,b) for intersection to the right of the sweep line; (c,b) don't intersect

Bentley-Ottmann sweep



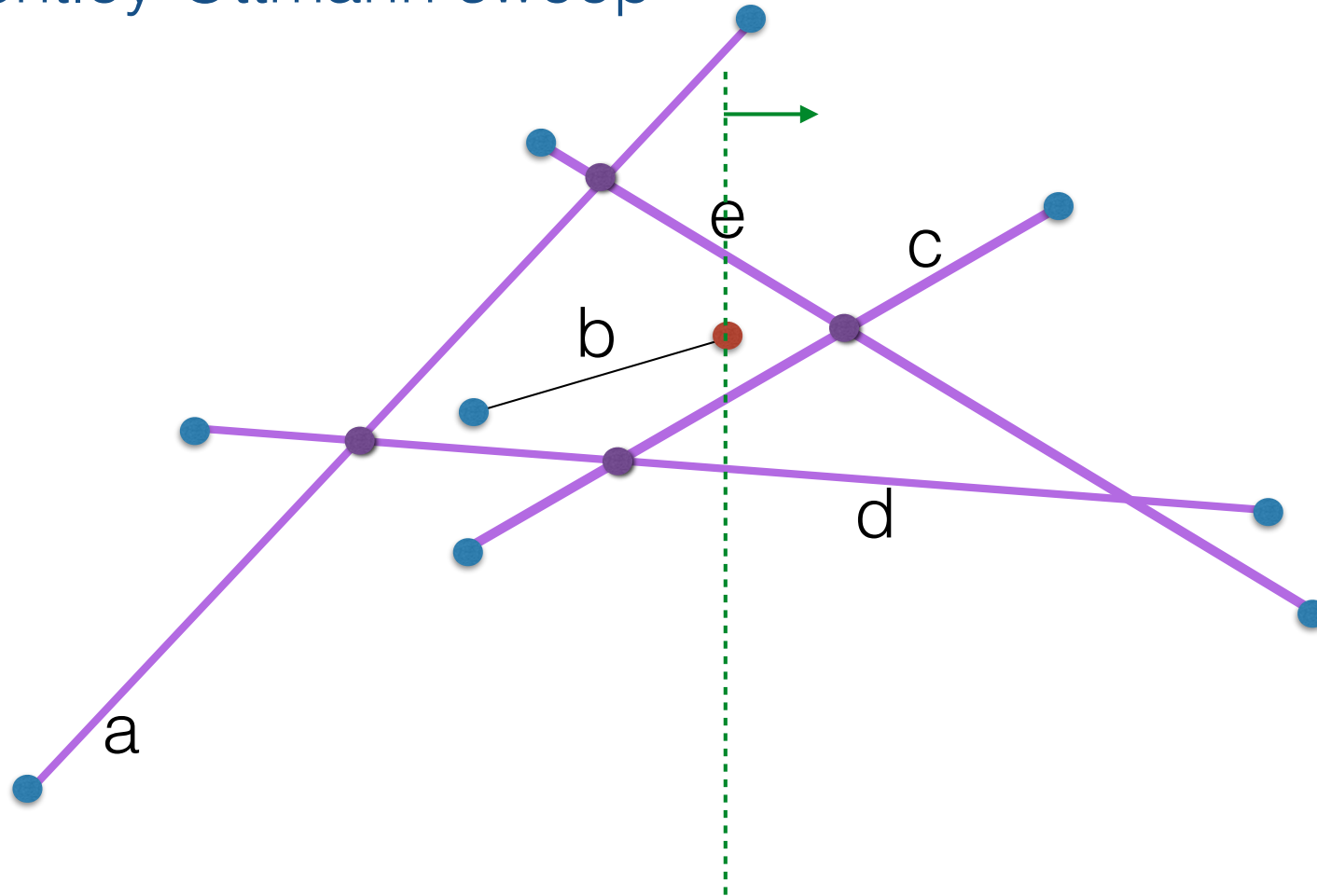
- this event is end of segment b:
 - delete b from AS: $d < c < e < a$
 - check new neighbors (c,e) for intersection to the right of the sweep line; this detects the intersection point of (c,e); report it and insert it as future event

Bentley-Ottmann sweep



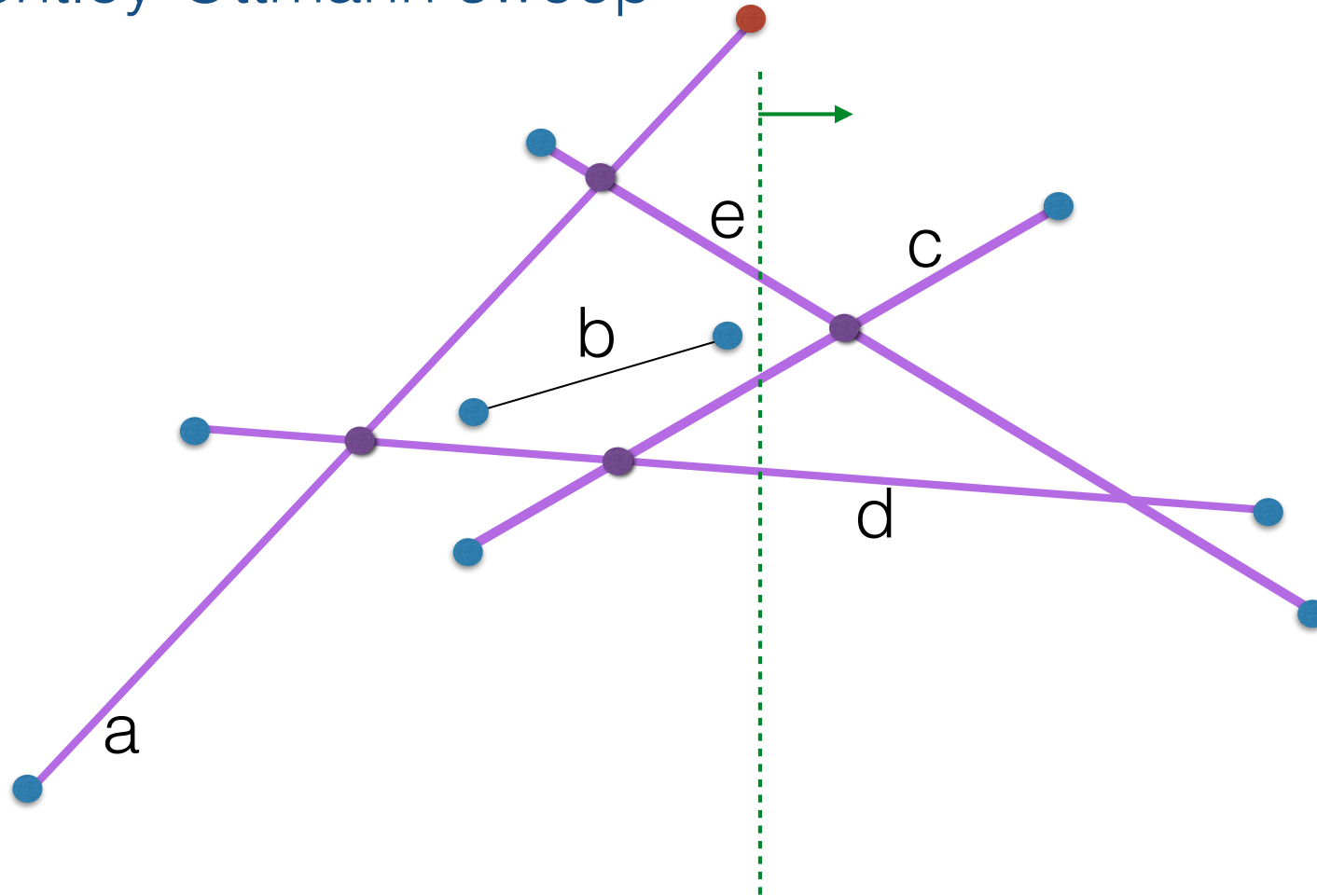
- this event is end of segment b:
 - delete b from AS: $d < c < e < a$
 - check new neighbors (c,e) for intersection to the right of the sweep line; this detects the intersection point of (c,e); report it and insert it as future event

Bentley-Ottmann sweep



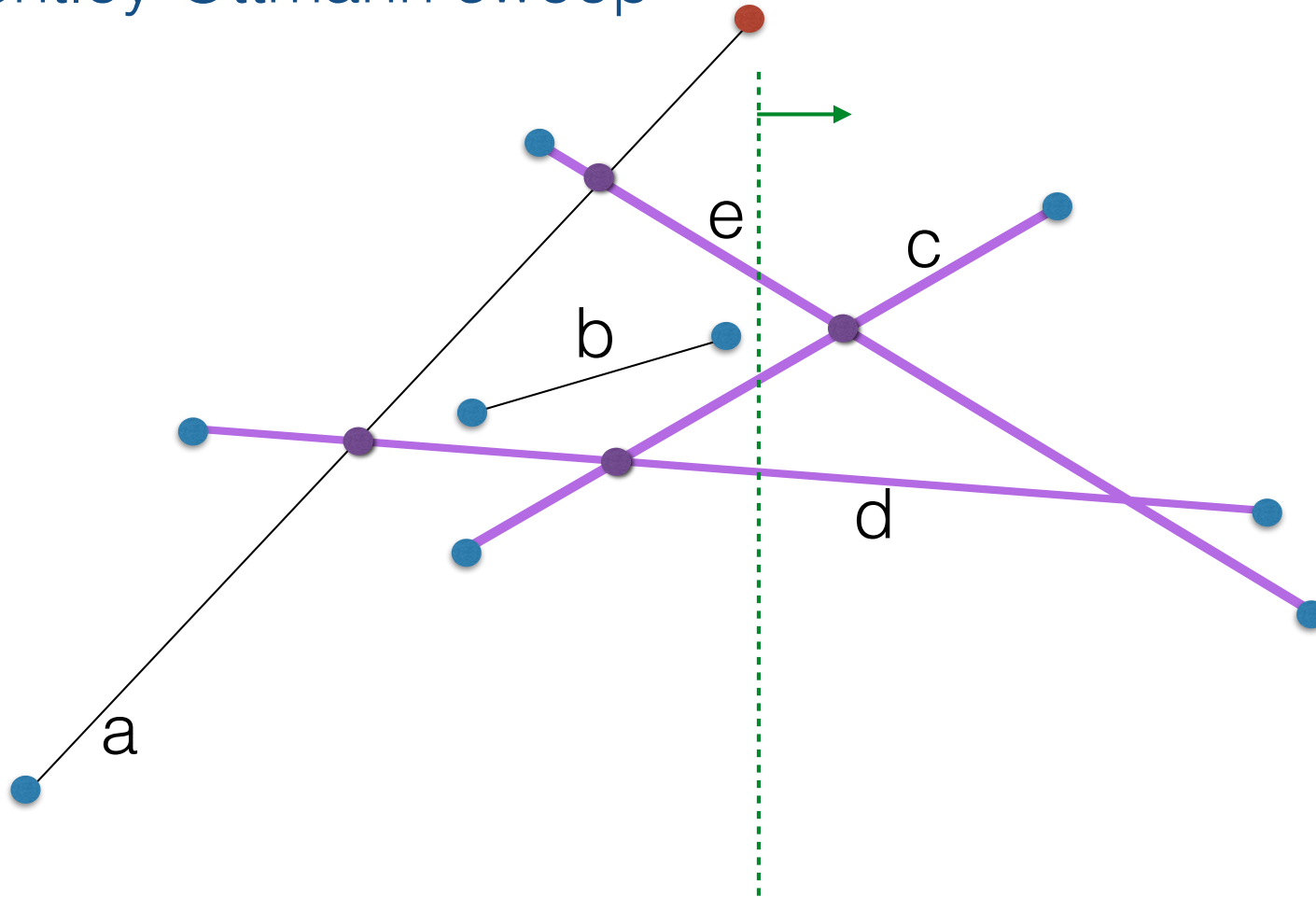
- this event is end of segment b:
 - delete b from AS: $d < c < e < a$
 - check new neighbors (c,e) for intersection to the right of the sweep line; this detects the intersection point of (c,e); report it and insert it as future event

Bentley-Ottmann sweep



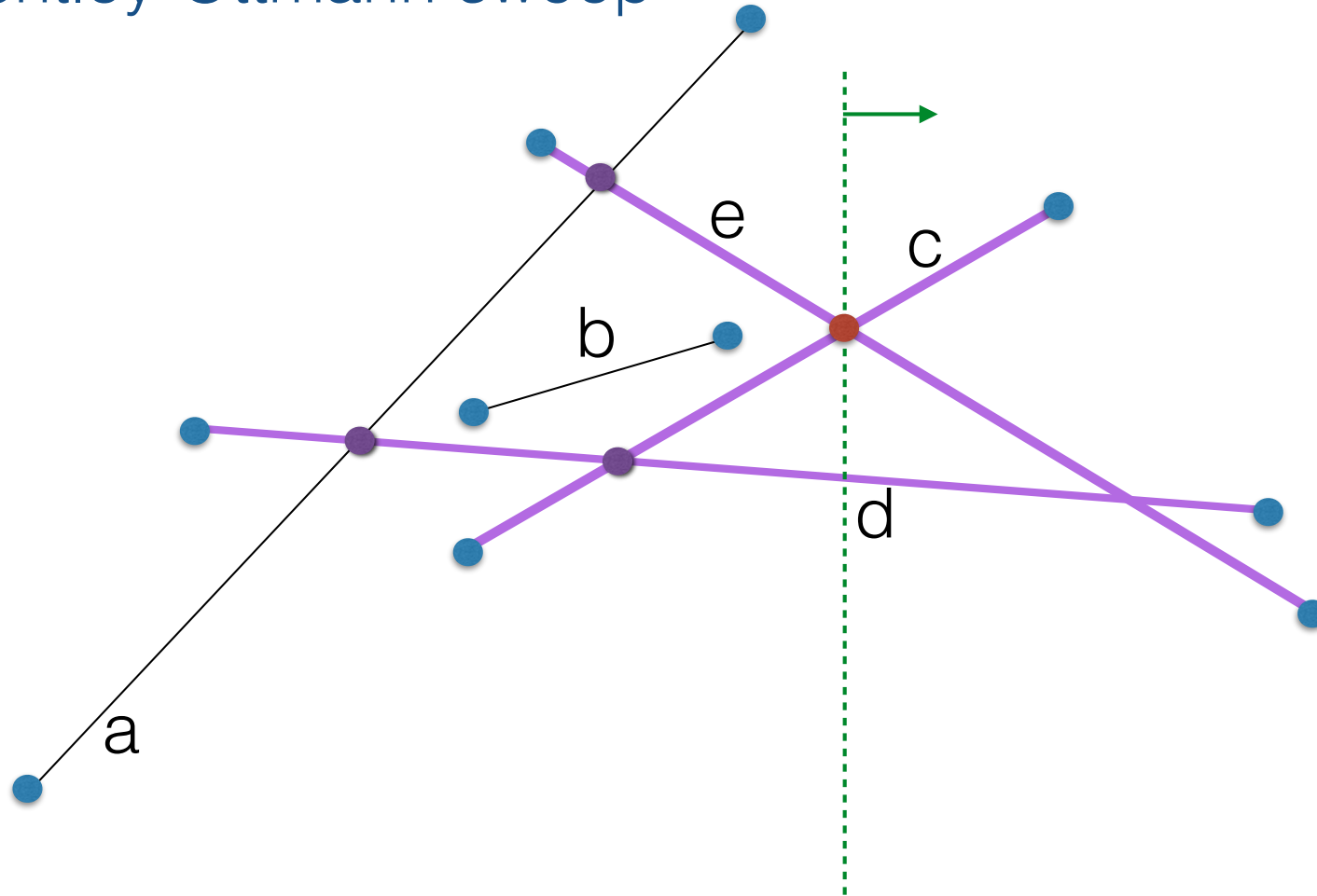
- this event is end of segment a:
 - delete a from AS: $d < c < e$
 - no new neighbors

Bentley-Ottmann sweep



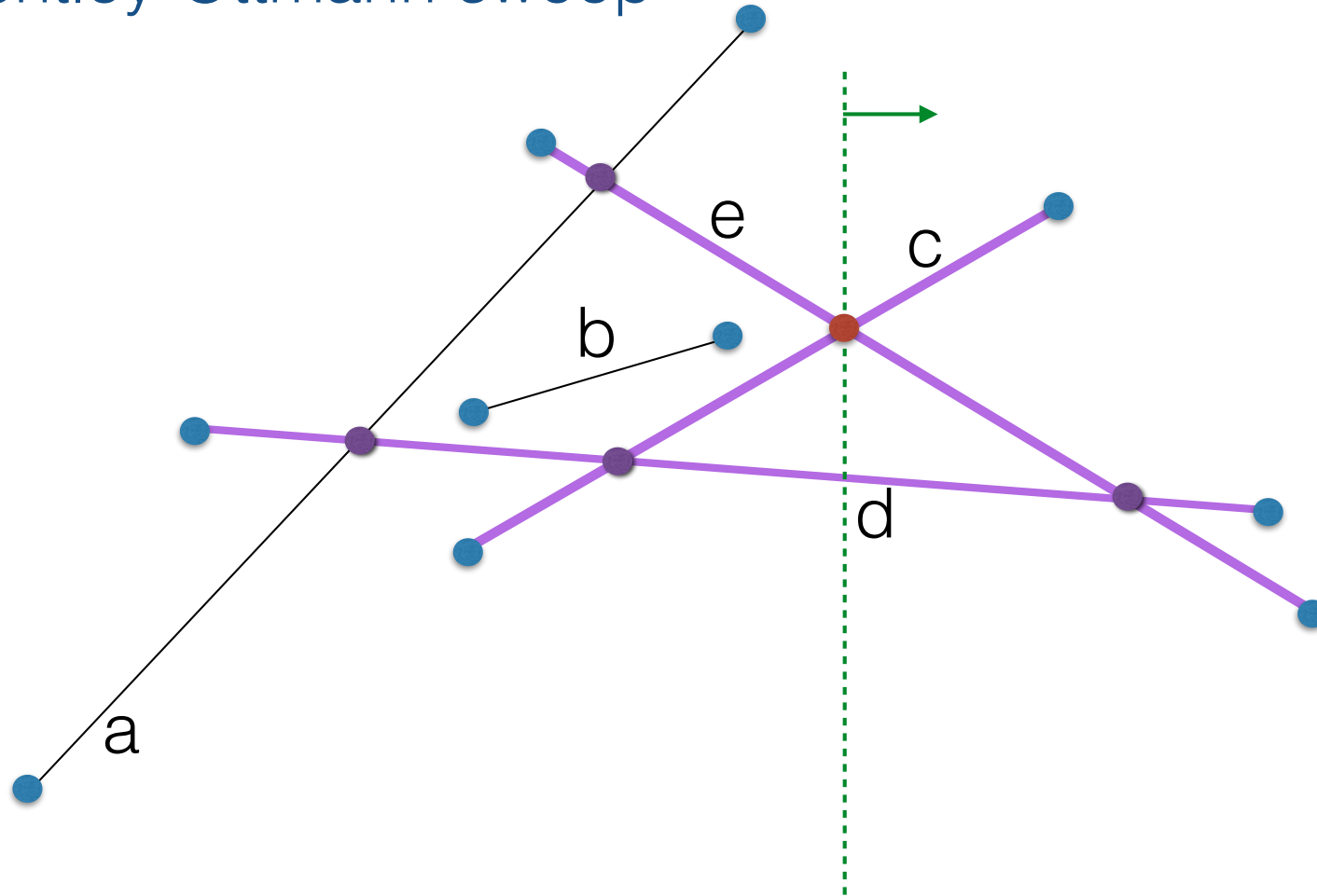
- this event is end of segment a:
 - delete a from AS: $d < c < e$
 - no new neighbors

Bentley-Ottmann sweep



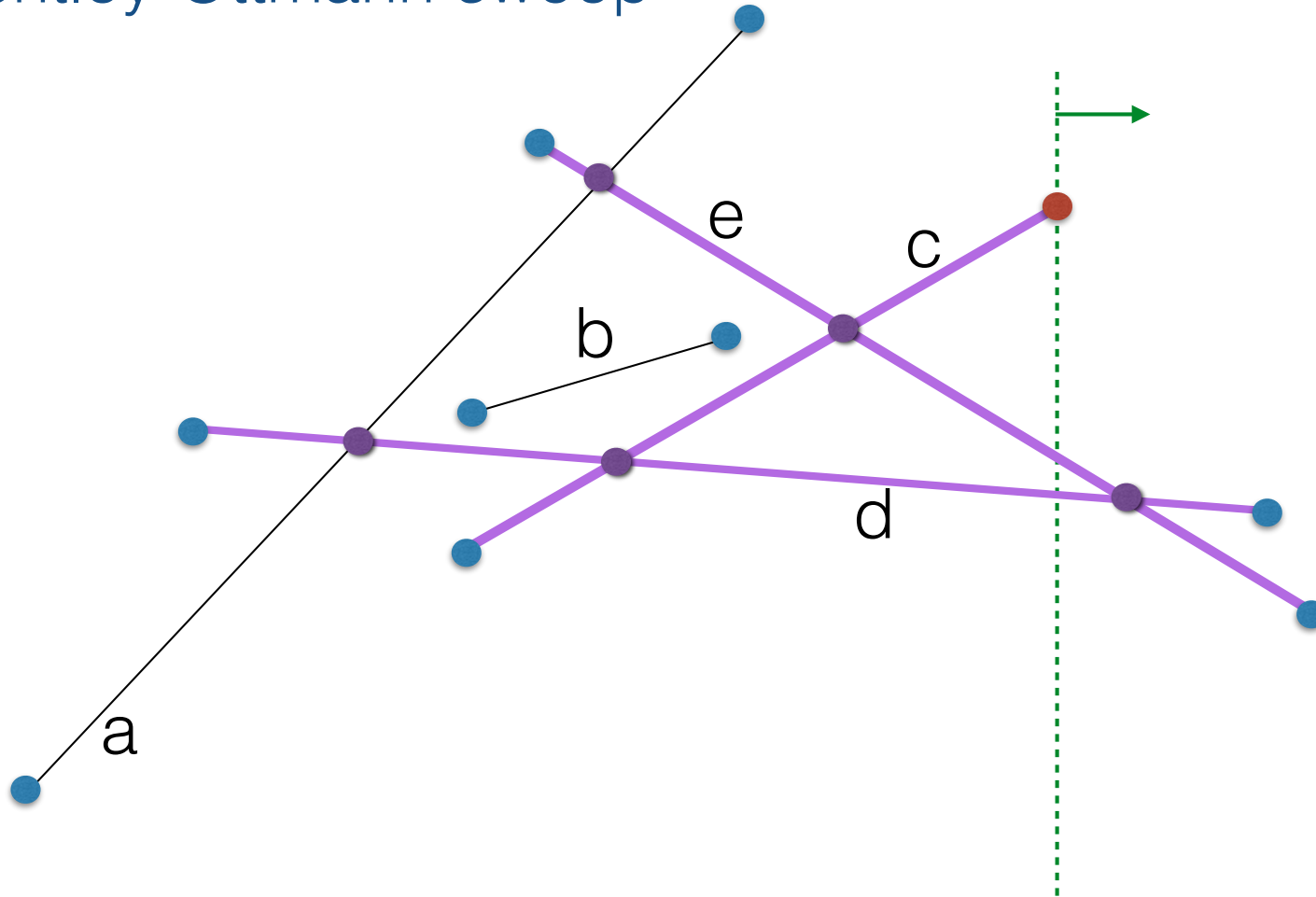
- this event is the intersection of (c,e):
 - flip c,e in AS: $d < e < c$
 - check new neighbors (d,e) for intersection to the right of the sweep line; this detects the intersection of (d,e); report it and insert it as future event

Bentley-Ottmann sweep



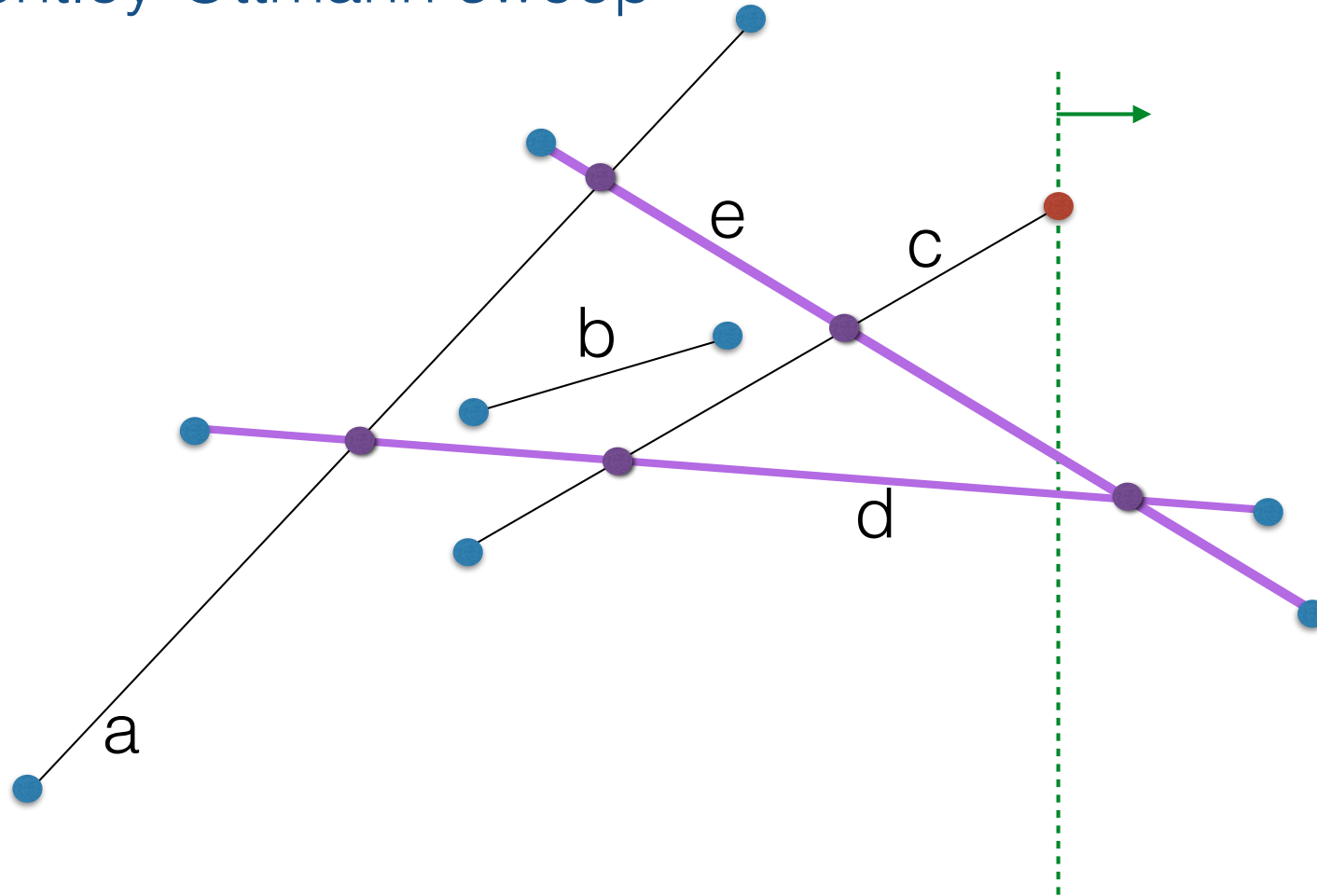
- this event is the intersection of (c,e):
 - flip c,e in AS: $d < e < c$
 - check new neighbors (d,e) for intersection to the right of the sweep line; this detects the intersection of (d,e); report it and insert it as future event

Bentley-Ottmann sweep



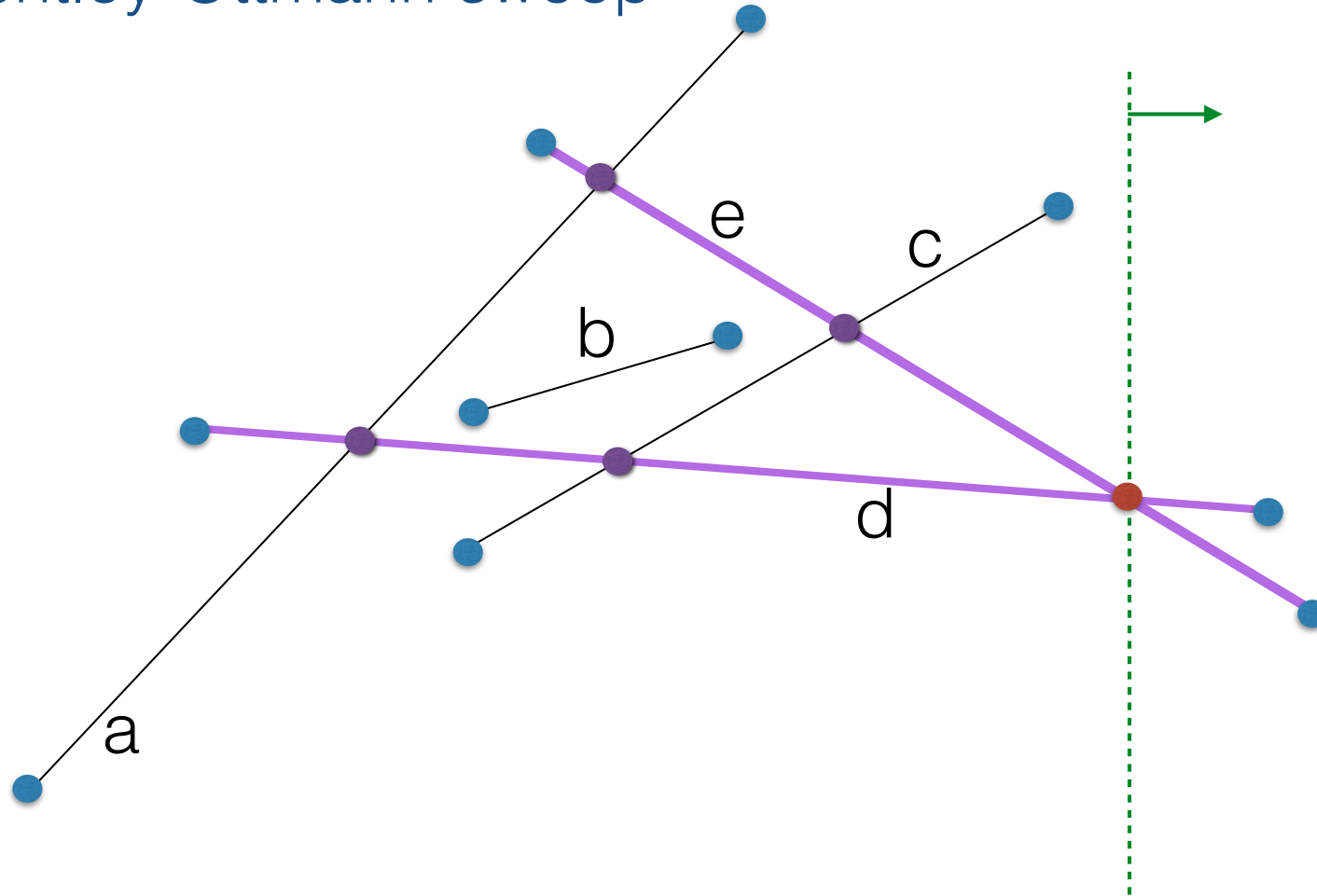
- this event is end of segment c:
 - delete c in AS: $d < e$
 - no new neighbors

Bentley-Ottmann sweep



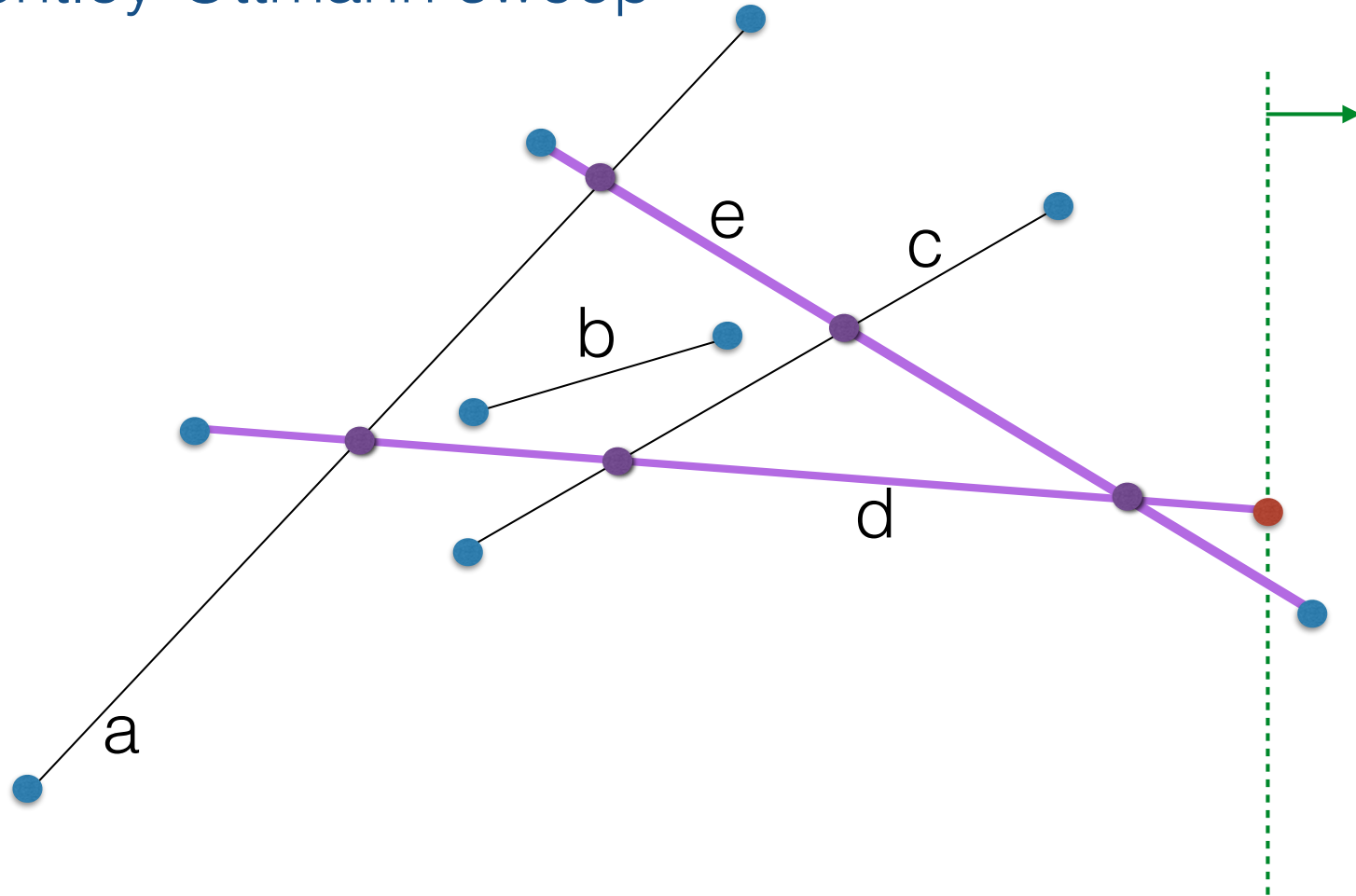
- this event is end of segment c:
 - delete c in AS: $d < e$
 - no new neighbors

Bentley-Ottmann sweep



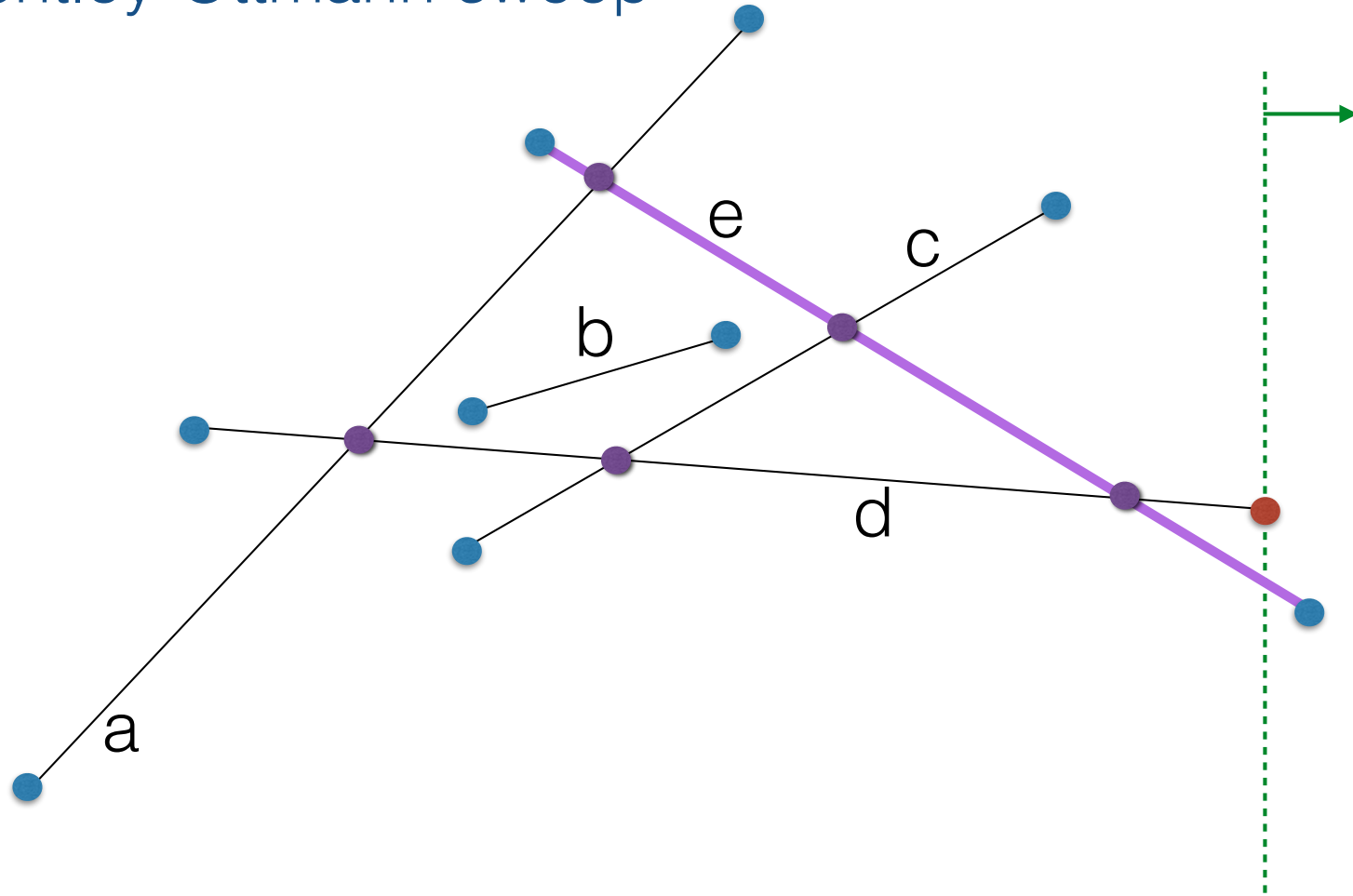
- this event is the intersection of (d,e):
 - flip d,e in AS: $e < d$
 - no new neighbors

Bentley-Ottmann sweep



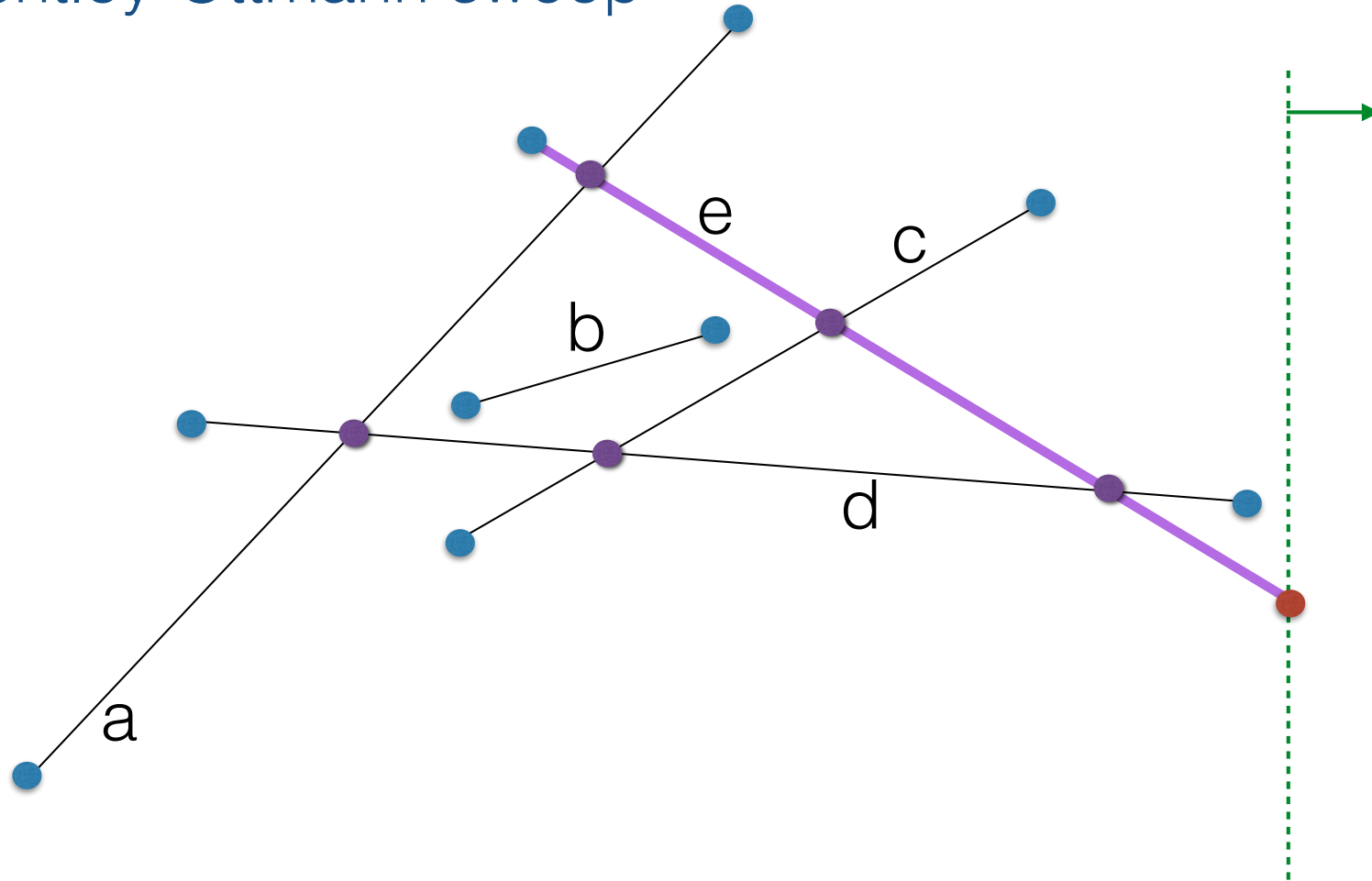
- this event is the end of d:
 - delete d in AS: e
 - no new neighbors

Bentley-Ottmann sweep



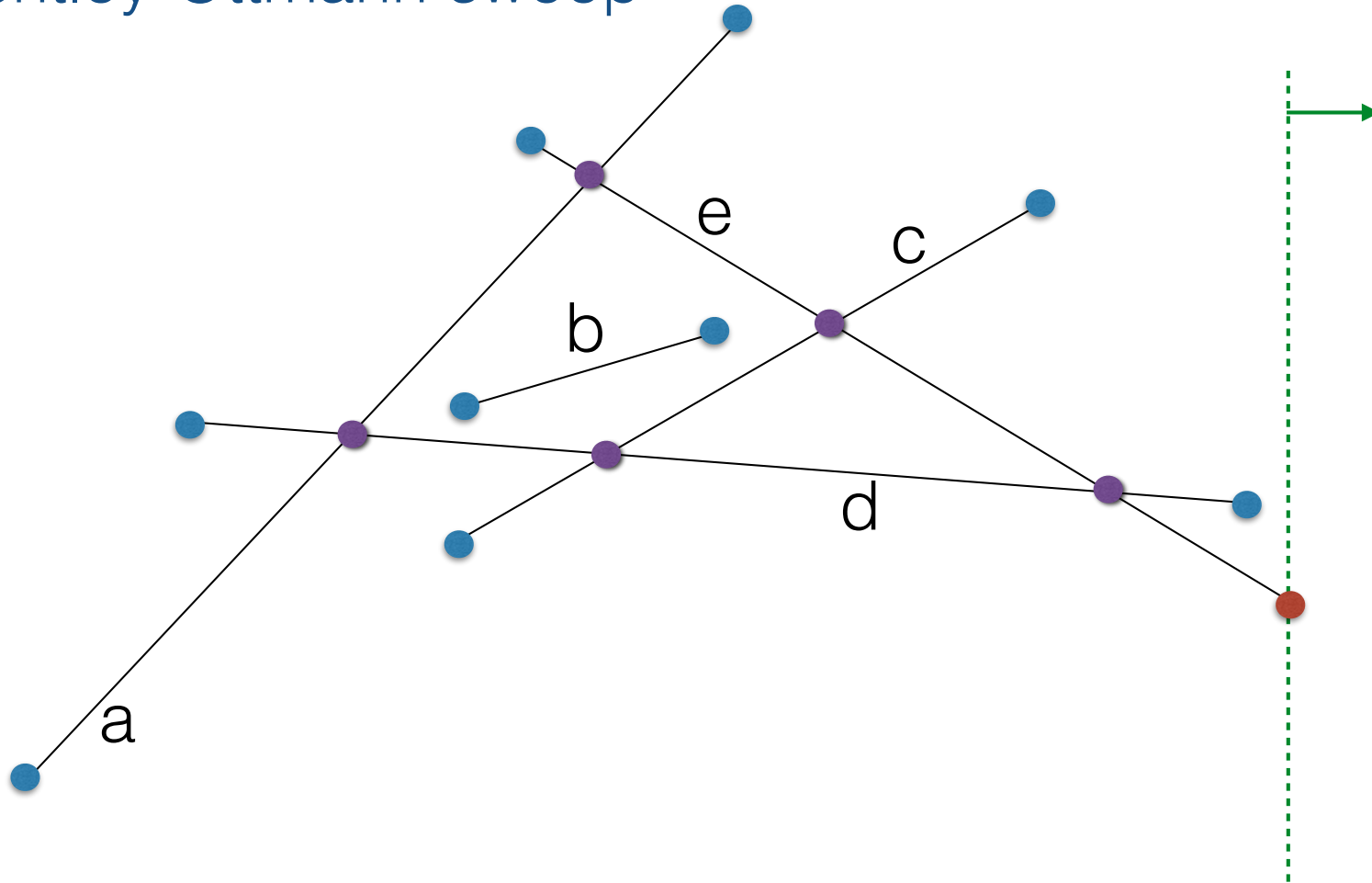
- this event is the end of d:
 - delete d in AS: e
 - no new neighbors

Bentley-Ottmann sweep



- this event is the end of e:
 - delete e in AS:
 - no new neighbors

Bentley-Ottmann sweep



- this event is the end of e:
 - delete e in AS:
 - no new neighbors

Bentley-Ottmann sweep

- Simplifying assumptions
 - no vertical segments
 - no two segments intersect at their endpoints
 - no three (or more) segments have a common intersection
 - all endpoints (of segments) and all intersection points have different x-coordinates
 - no segments overlap
- These assumptions are not realistic for real data..
- But, they don't provide insight into the plane sweep technique, so we omit them
- Real data challenges
 - dealing with degenerate cases
 - dealing with finite precision arithmetic and precision problems

Bentley-Ottmann sweep

We'll maintain the following invariants during the algorithm:

Bentley-Ottmann sweep

We'll maintain the following invariants during the algorithm:

- Active structure AS:
 - For any position of the sweep line SL, AS contains all active segments (ie segments that start before SL and end after SL)
 - AS is sorted by their y-coordinates of their intersection with SL

Bentley-Ottmann sweep

We'll maintain the following invariants during the algorithm:

- **Active structure AS:**
 - For any position of the sweep line SL, AS contains all active segments (ie segments that start before SL and end after SL)
 - AS is sorted by their y-coordinates of their intersection with SL
- **Event list EL:**
 - For any position of SL, EL contains segment endpoints to the right of SL, and also the intersections to the right of SL of active segments that were/are neighbors in SL
 - EL is sorted by x-coordinate

Bentley-Ottmann sweep

We'll maintain the following invariants during the algorithm:

- Active structure AS:
 - For any position of the sweep line SL, AS contains all active segments (ie segments that start before SL and end after SL)
 - AS is sorted by their y-coordinates of their intersection with SL
- Event list EL:
 - For any position of SL, EL contains segment endpoints to the right of SL, and also the intersections to the right of SL of active segments that were/are neighbors in SL
 - EL is sorted by x-coordinate
- For any position of the sweep line SL, all pairs of intersecting dead segments have been reported.

Algorithm Bentley-Ottmann (S)

//S is a set of n line segments in the plane

- initialize AS= {}
- sort 2n endpoints of all segments in S by x-coord and store them in EventList
- while EventList not empty
 - let e be the next event from EventList; delete it from EL
 - //sweep line moves to x=e.x
 - if e is left endpoint of a segment l
 - //l becomes active
 - insert l in AS in the right place
 - check if l intersects with l->prev and l->succ in AS to the right of the sweep line; if they do, insert their intersection point in the EventList
 - //optional: since l.prev and l.succ are not neighbors anymore, we check if they intersect and if they do, delete that intersection point from the EventList
 - if e is the right endpoint of a segment
 - ...
 - if e is the intersection of two segments
 - ...
- end.

Bentley-Ottmann sweep

Questions

- AS
 - What operations do we do on AS?
 - What data structure should we use for AS?
- EL
 - Note that we know a priori the $2n$ events corresponding to start and end-points of segments, but EL is not static; the events corresponding to intersection points are generated on the fly
 - What operations do we do on EL?
 - What data structure should we use for EL?

Bentley-Ottmann sweep

Running time

- **AS**
 - What is the size of AS?
 - $O(n)$
 - How many operations?
 - $O(n+k)$
 - Overall time?
 - $O((n+k)\lg n)$
- **EventList**
 - What is the size of EventList?
 - $O(n+k)$
 - How many operations?
 - $O(n+k)$
 - Overall time?
 - $O((n+k)\lg n)$

Bentley-Ottmann sweep

Running time

- **AS**
 - What is the size of AS?
 - $O(n)$
 - How many operations?
 - $O(n+k)$
 - Overall time?
 - $O((n+k)\lg n)$
- **EventList**
 - What is the size of EventList?
 - $O(n+k)$
 - How many operations?
 - $O(n+k)$
 - Overall time?
 - $O((n+k)\lg n)$

Bentley-Ottmann sweep runs in $O((n+k)\lg n)$ time.