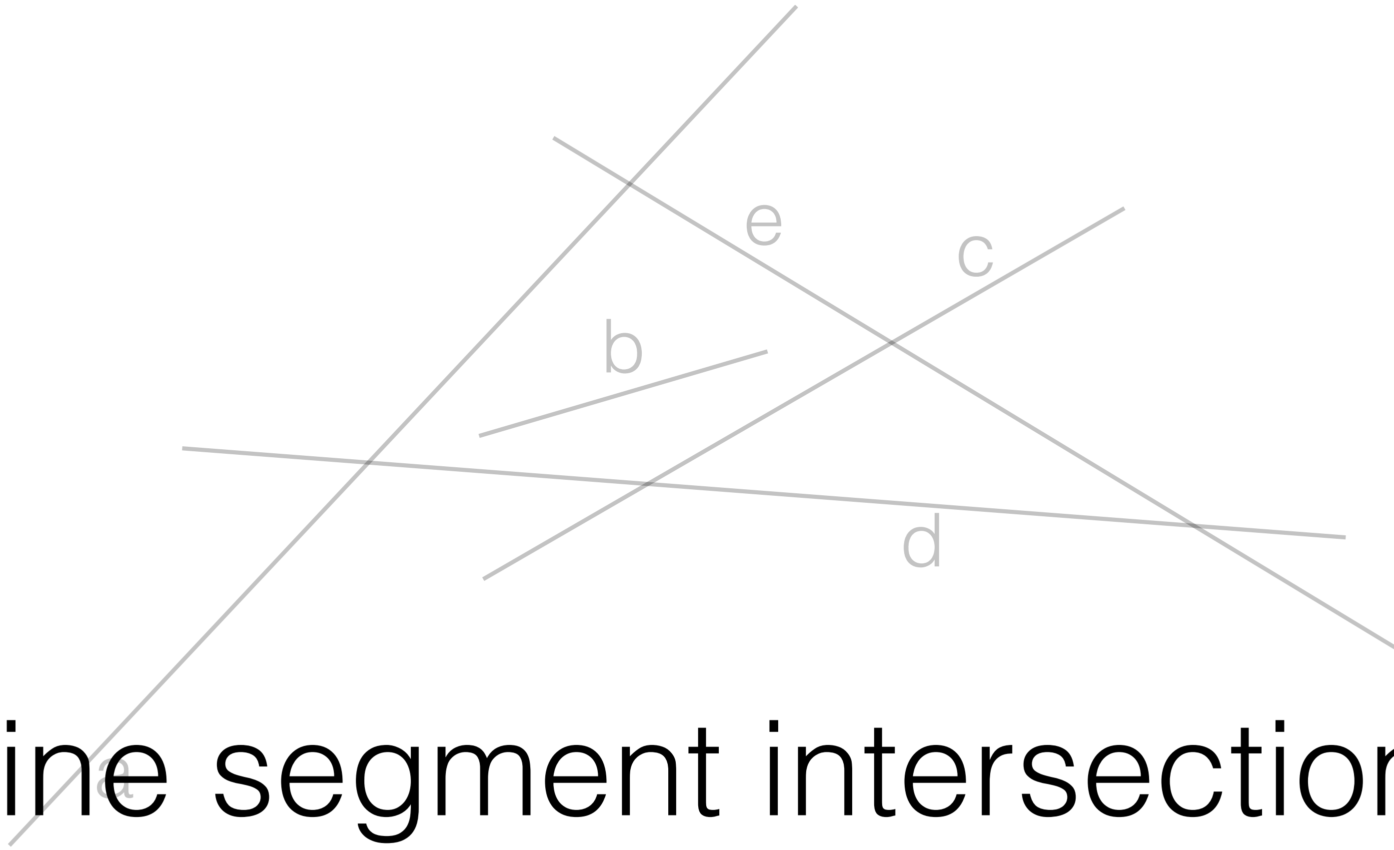
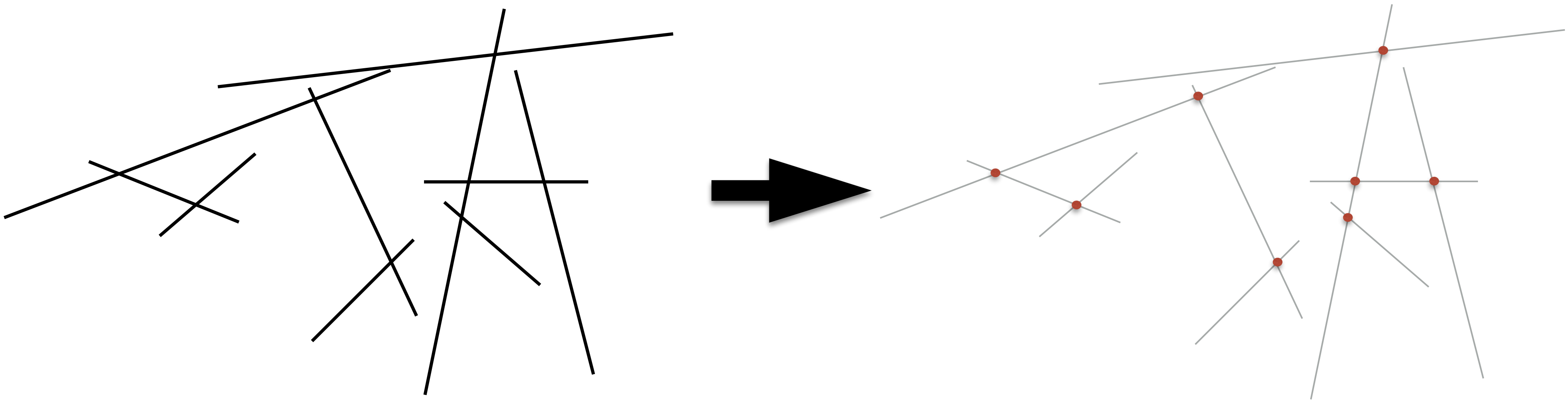


Line segment intersection

Computational Geometry [csci 3250]
Laura Toma
Bowdoin College

Line segment intersection

Given a set of n line segments in the plane → Find all their intersection points

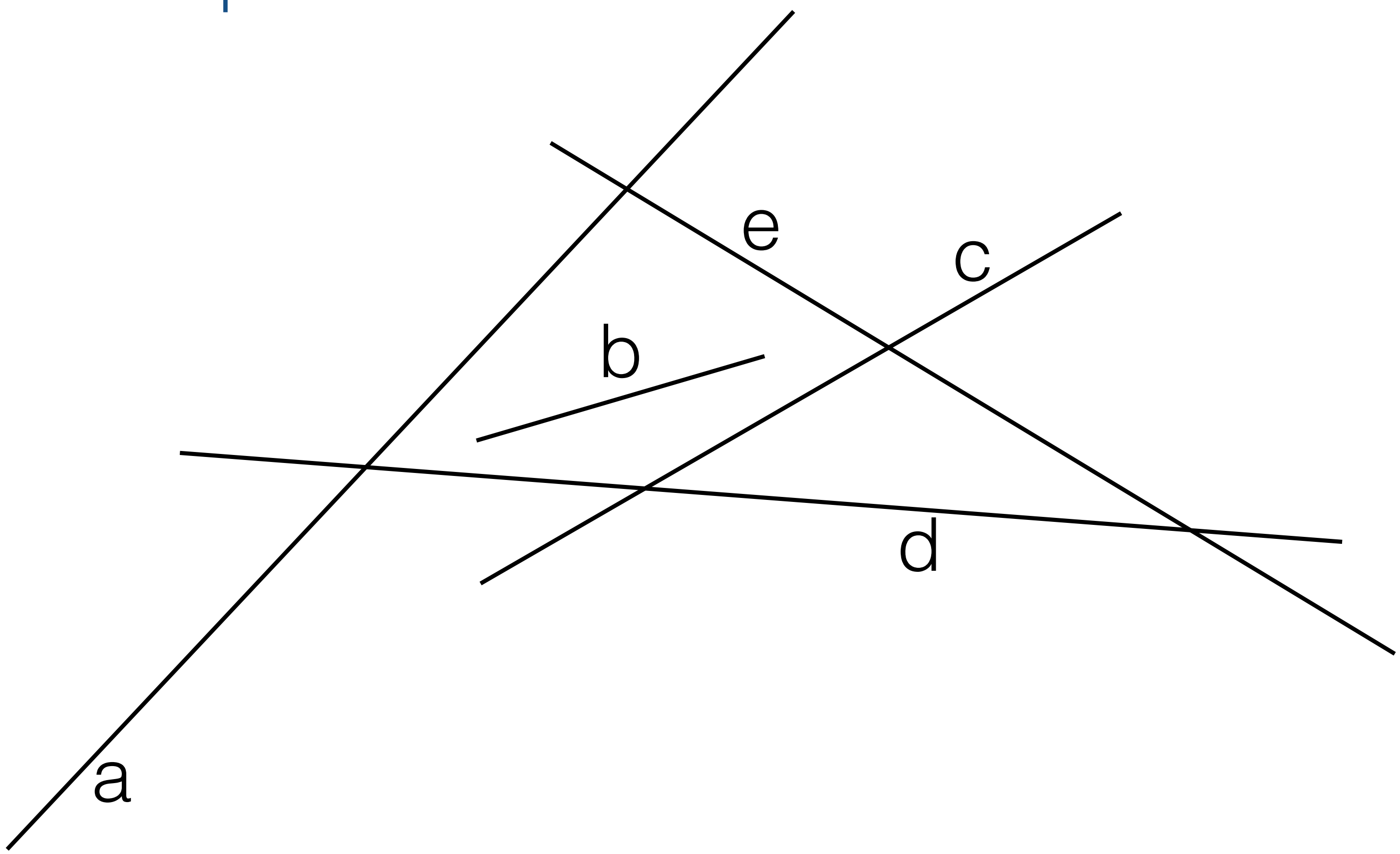


- n: size of the input (number of segments)
- k: size of output (number of intersections)

Overview

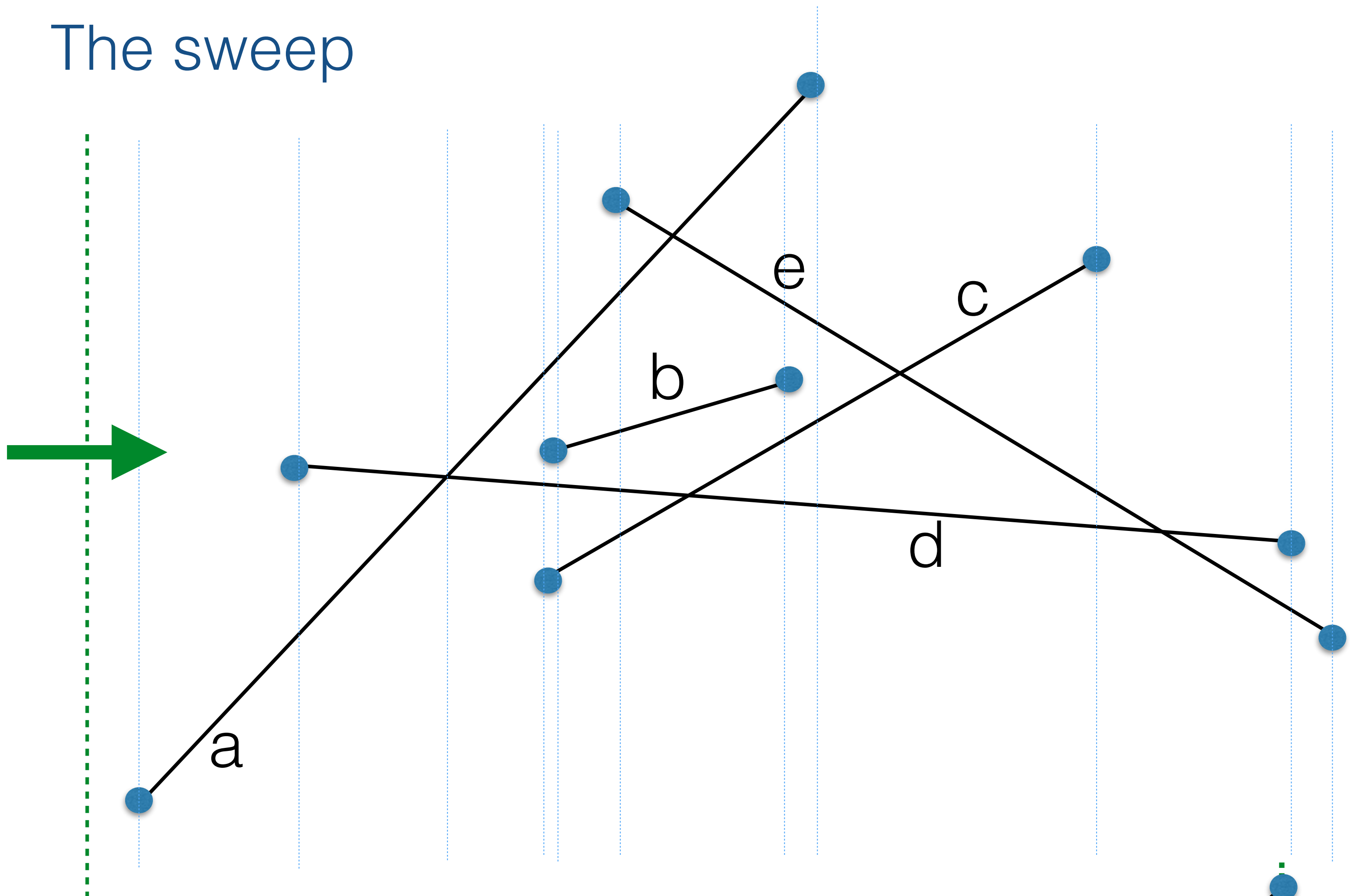
- Approach: line sweep
- We'll get an overall bound of $O(n \lg n + k \lg n)$
    - this improves on the naive $O(n^2)$ when k is small
- The algorithm was developed by Jon Bentley and Thomas Ottman in 1979
- Simple (..in retrospect!), elegant and practical

The sweep

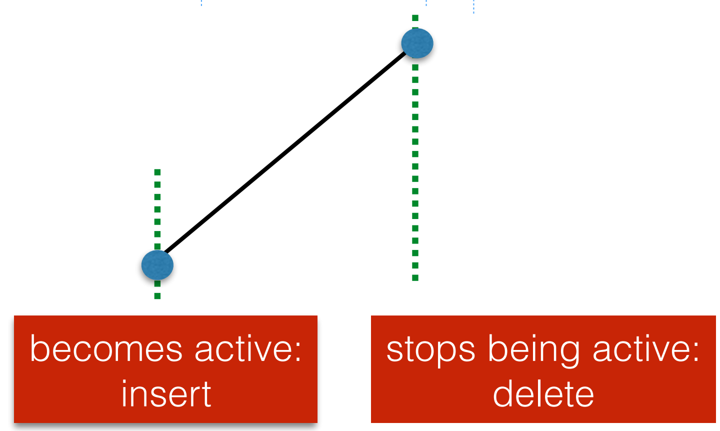


- Let X be the set of all x-coords of segments

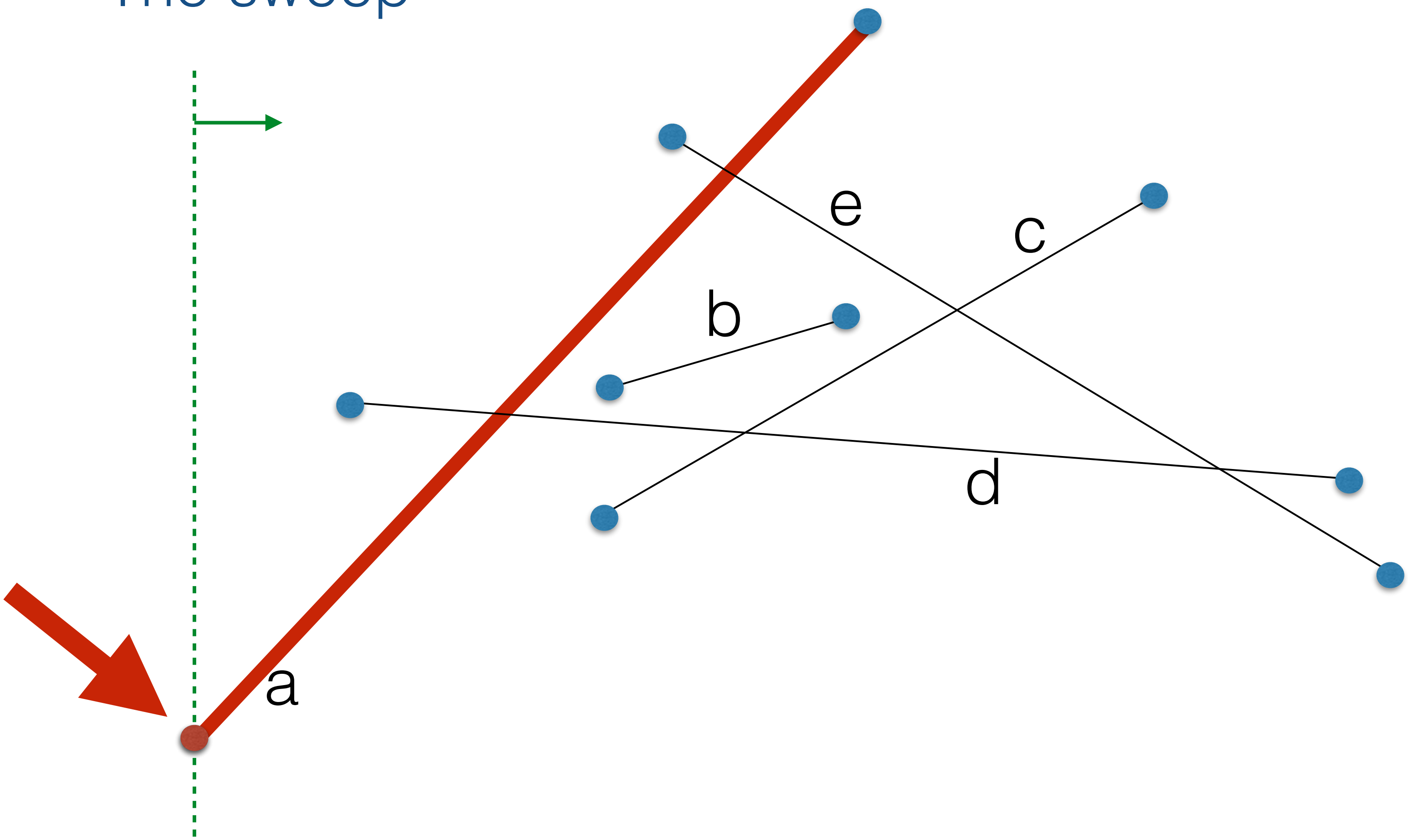
The sweep



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

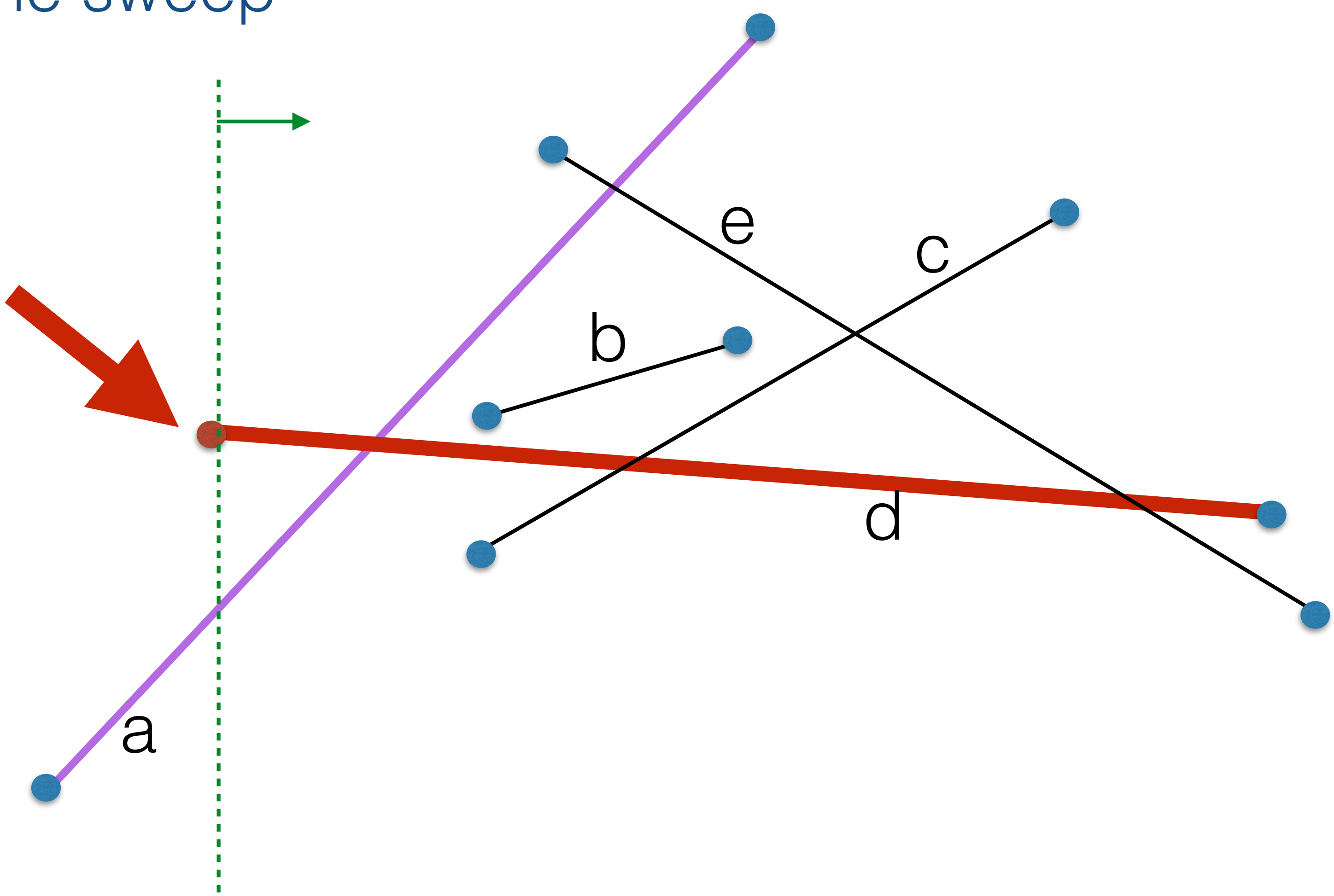


The sweep



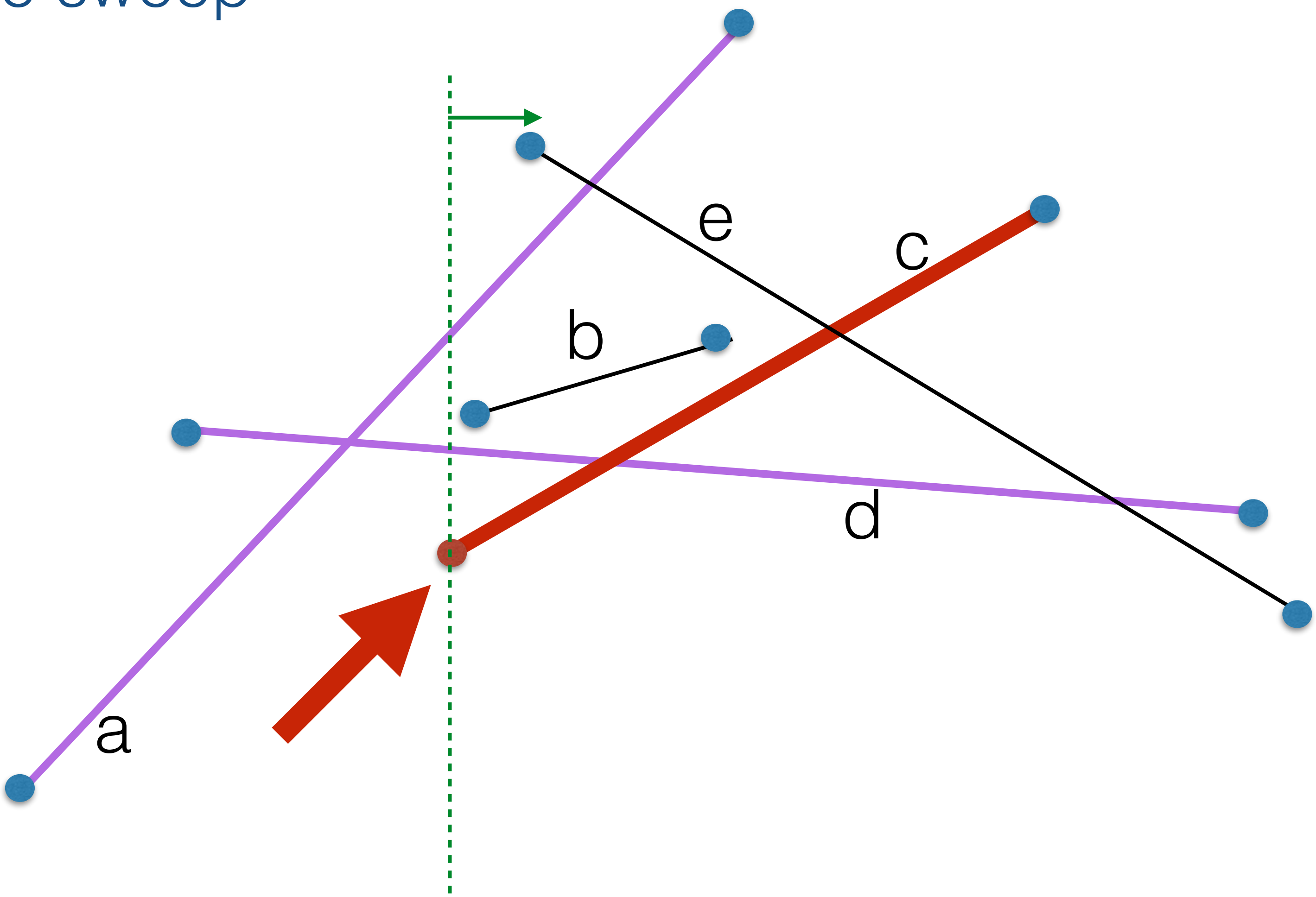
- a.start:
    - segment a becomes active
    - it will stay active until sweep line reaches a.end

The sweep



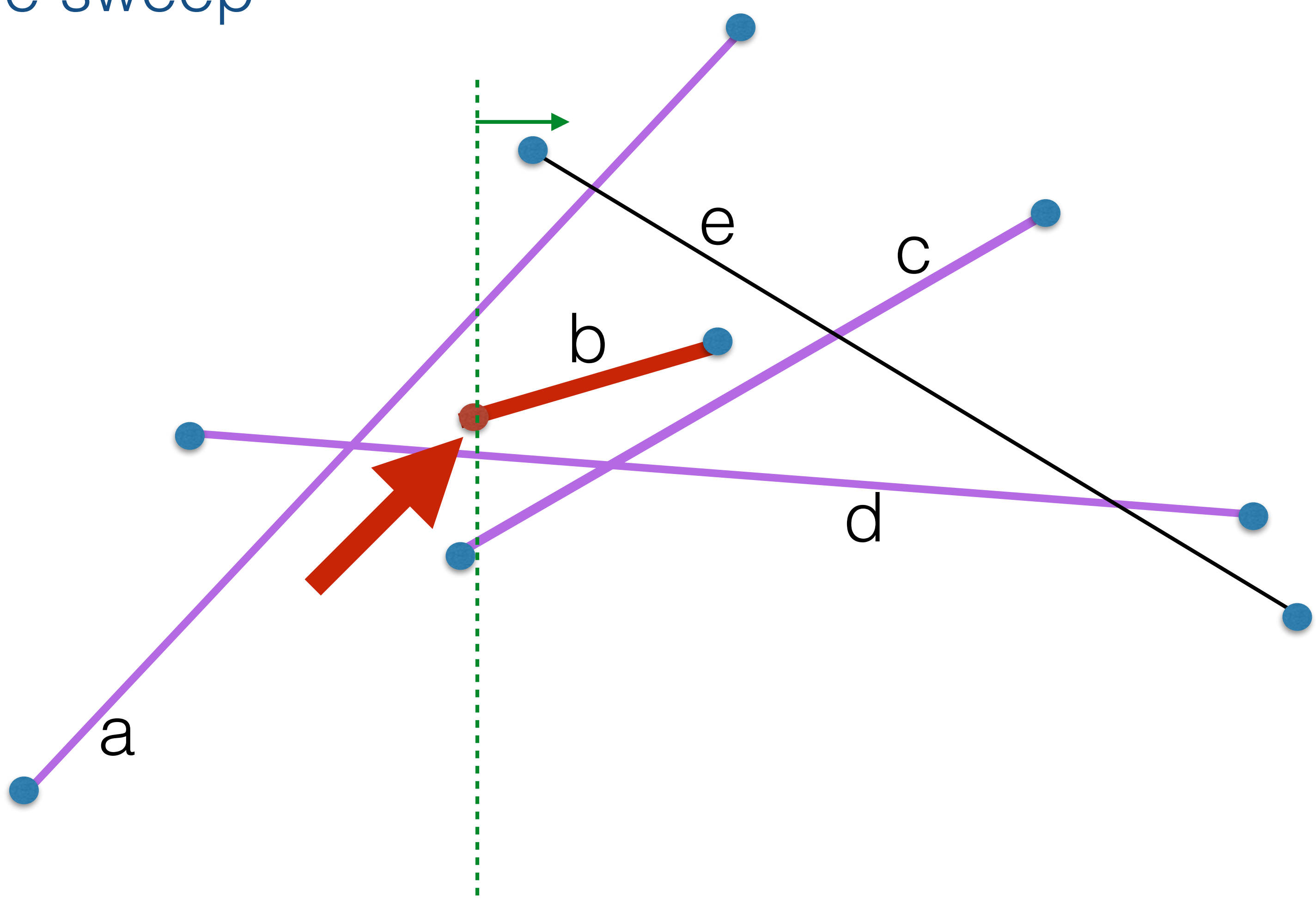
- d.start
  - segment d becomes active
  - it will stay active until sweep line reaches d.end

The sweep



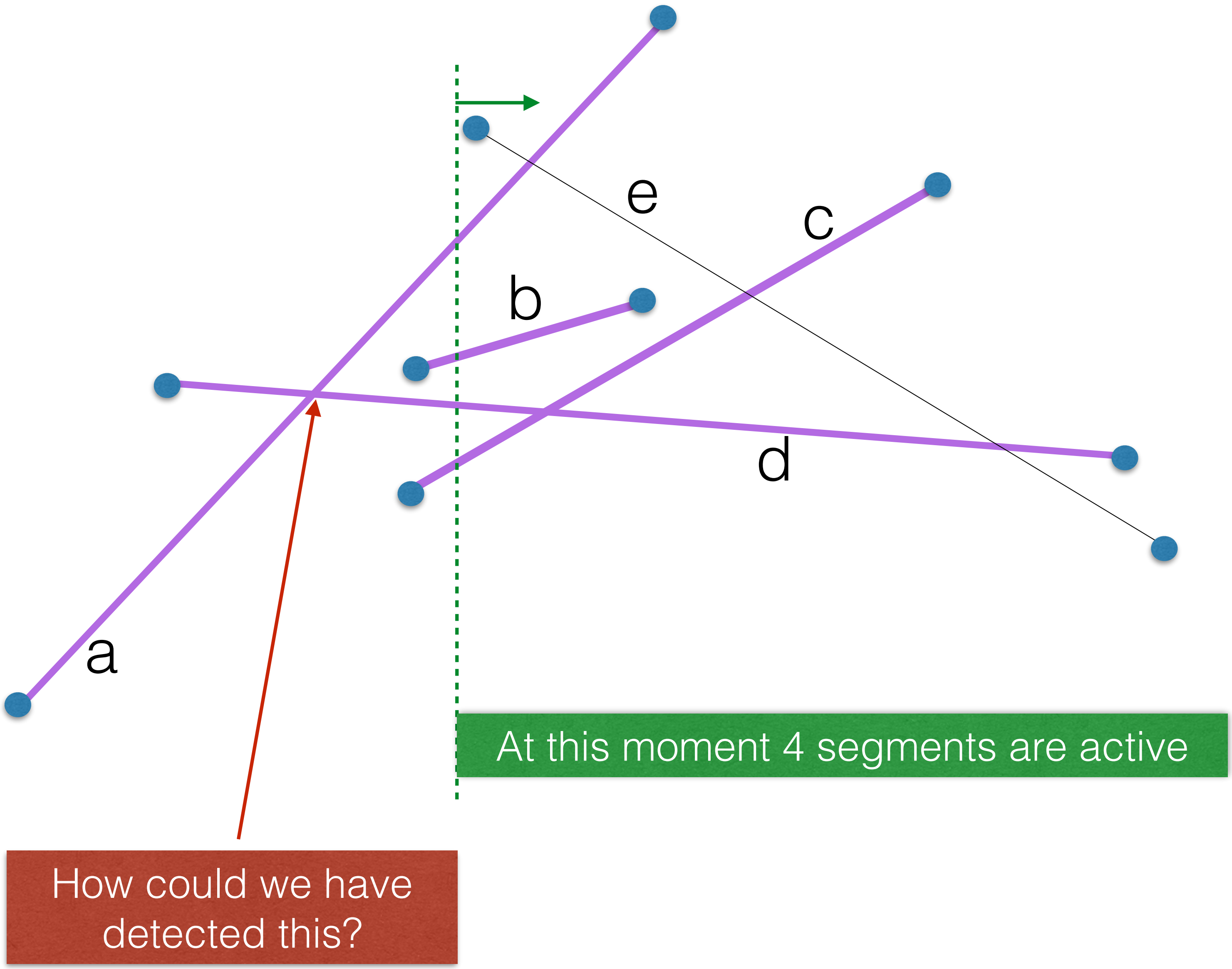
- c.start
  - segment c becomes active
  - it will stay active until sweep line reaches c.end

The sweep



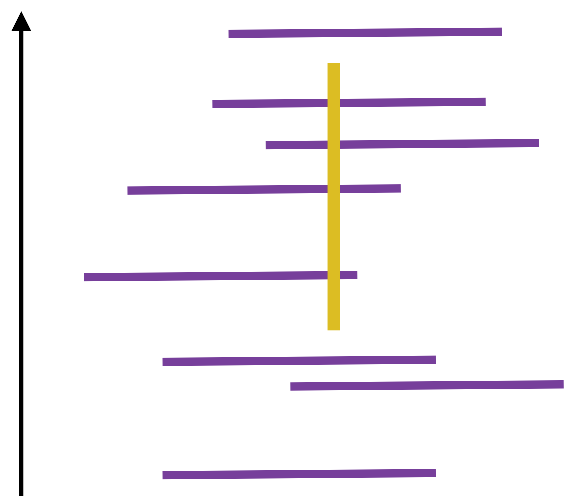
- b.start

How do we detect intersections during the sweep?



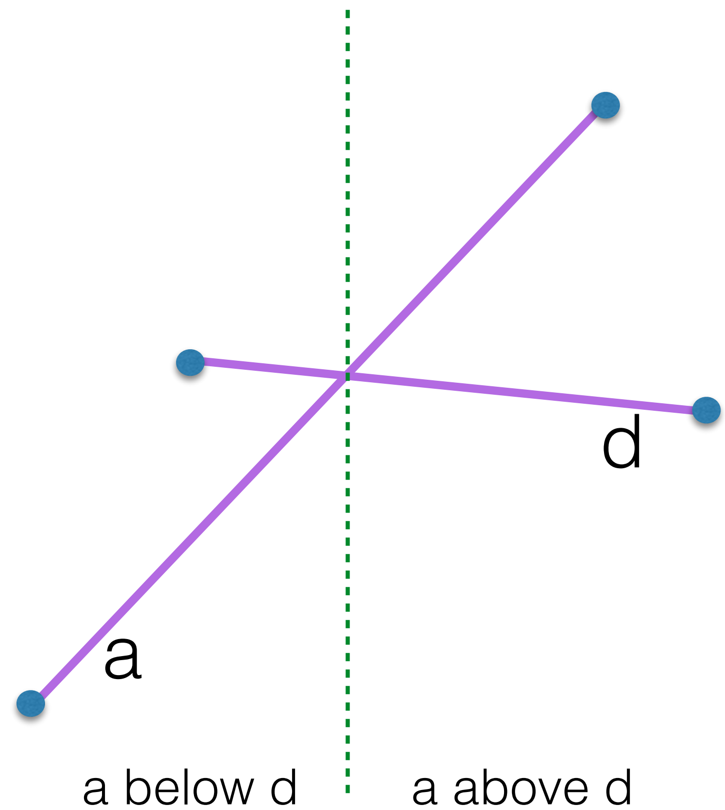
Key idea #1

orthogonal segments



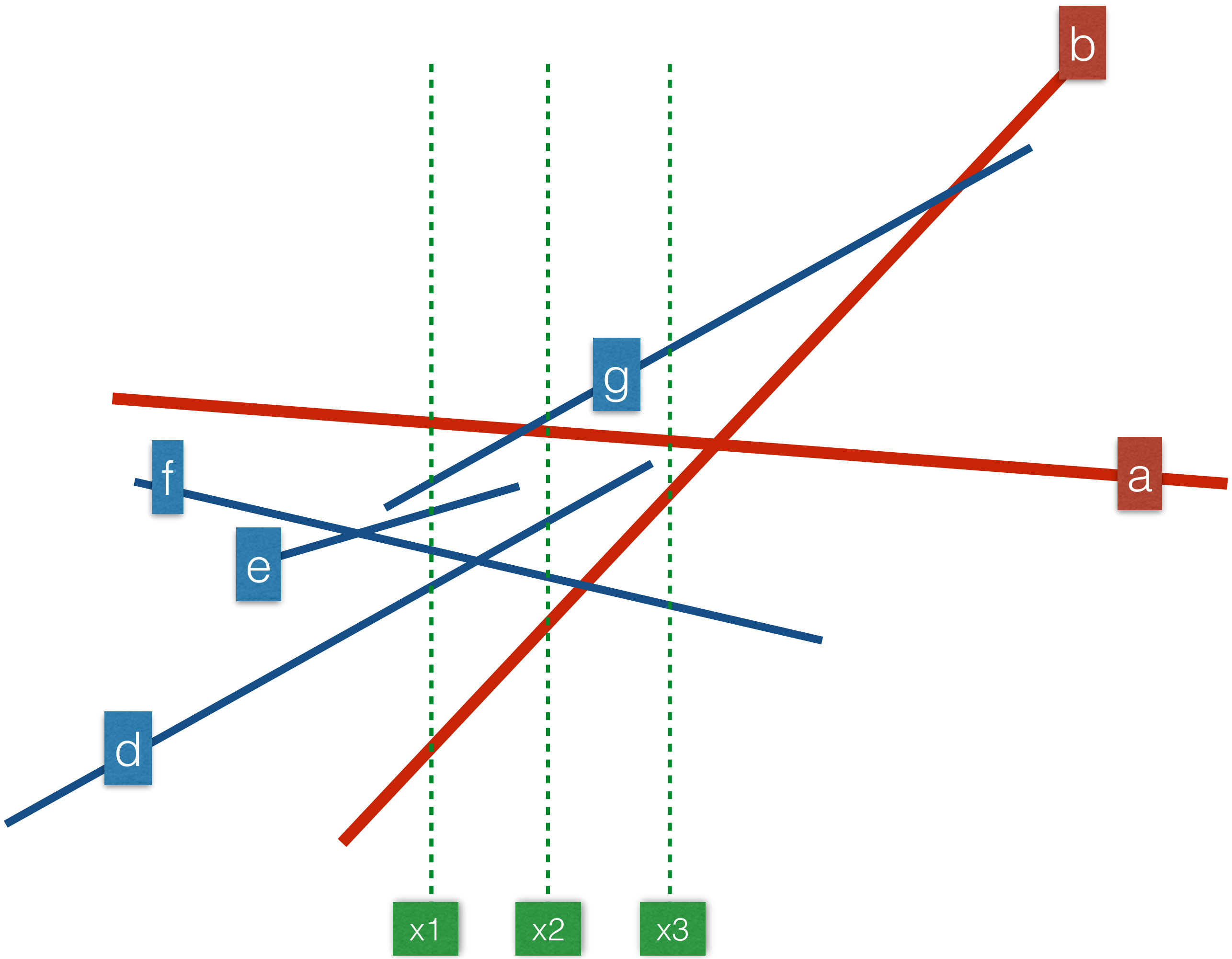
horizontal segments in y-order

general segments



above-below order flips at intersection point!

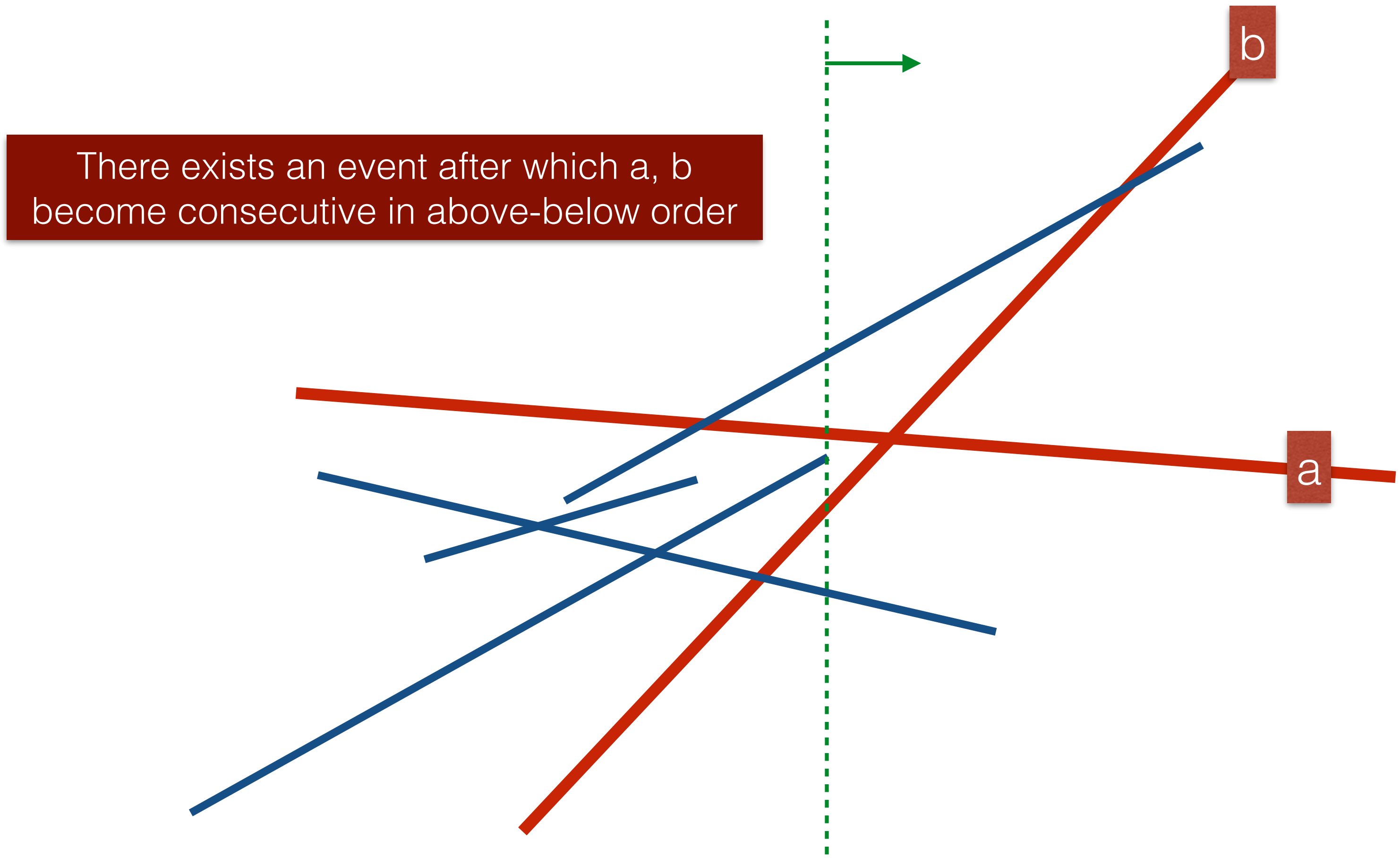
Key idea #2



- Write the segments in above-below order at x1, x2 and x3

Key idea #2

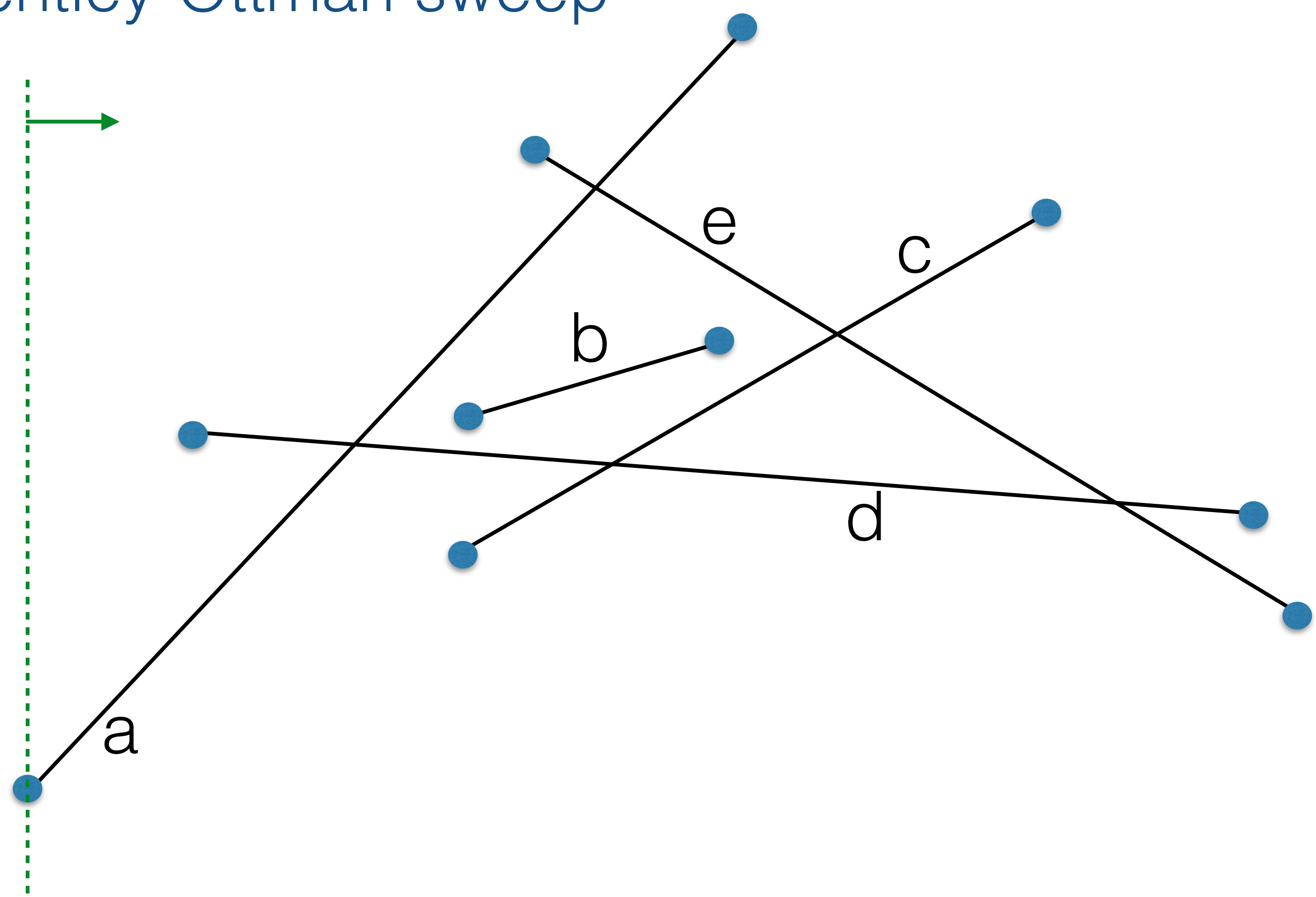
- Segments that intersect are consecutive in above-below order just before they intersect



- Strategy: Throughout the sweep, we'll check for intersection all pairs of segments that are consecutive in above-below order. This way we cannot miss any intersection!

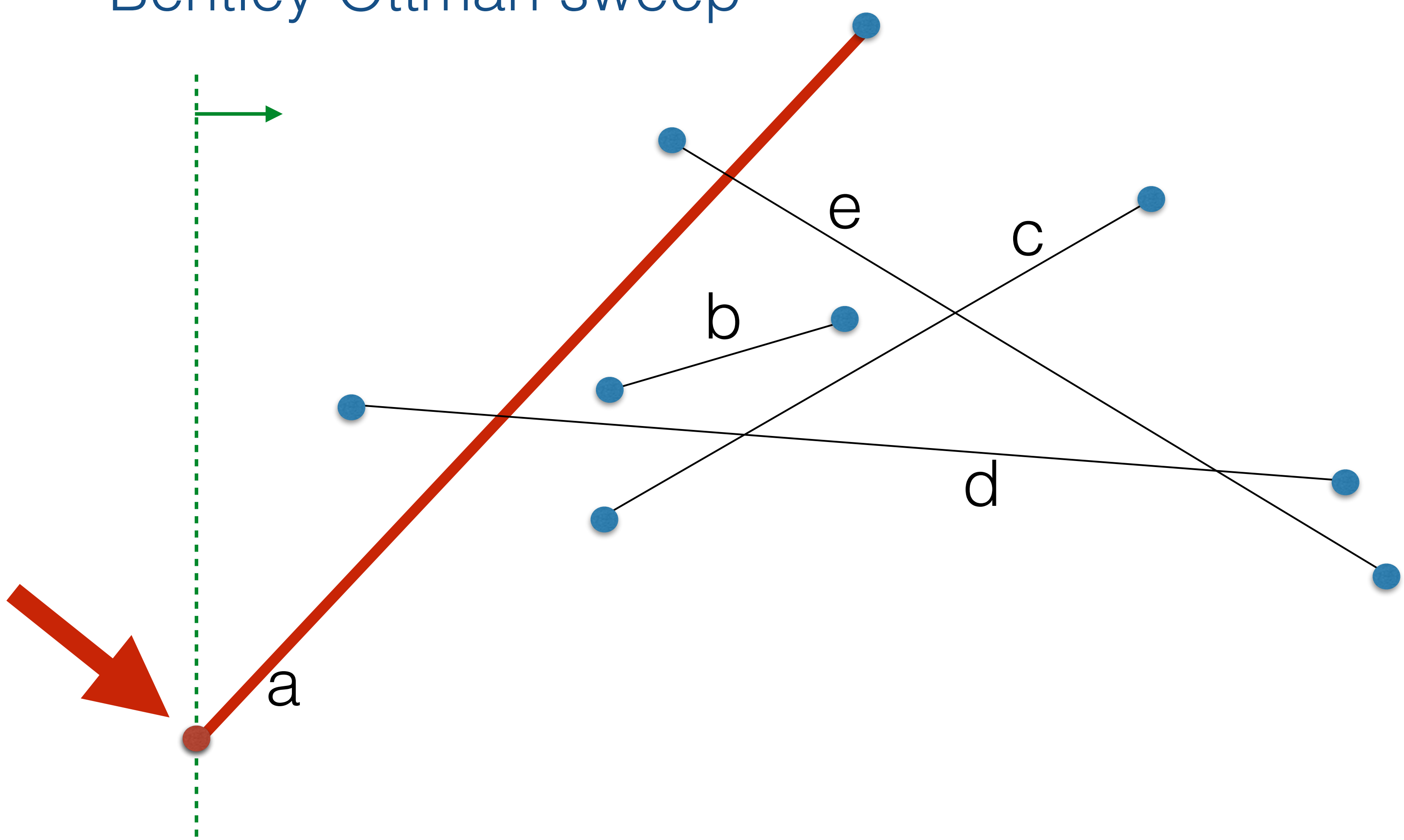
Let’s start over..

Bentley-Ottman sweep



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

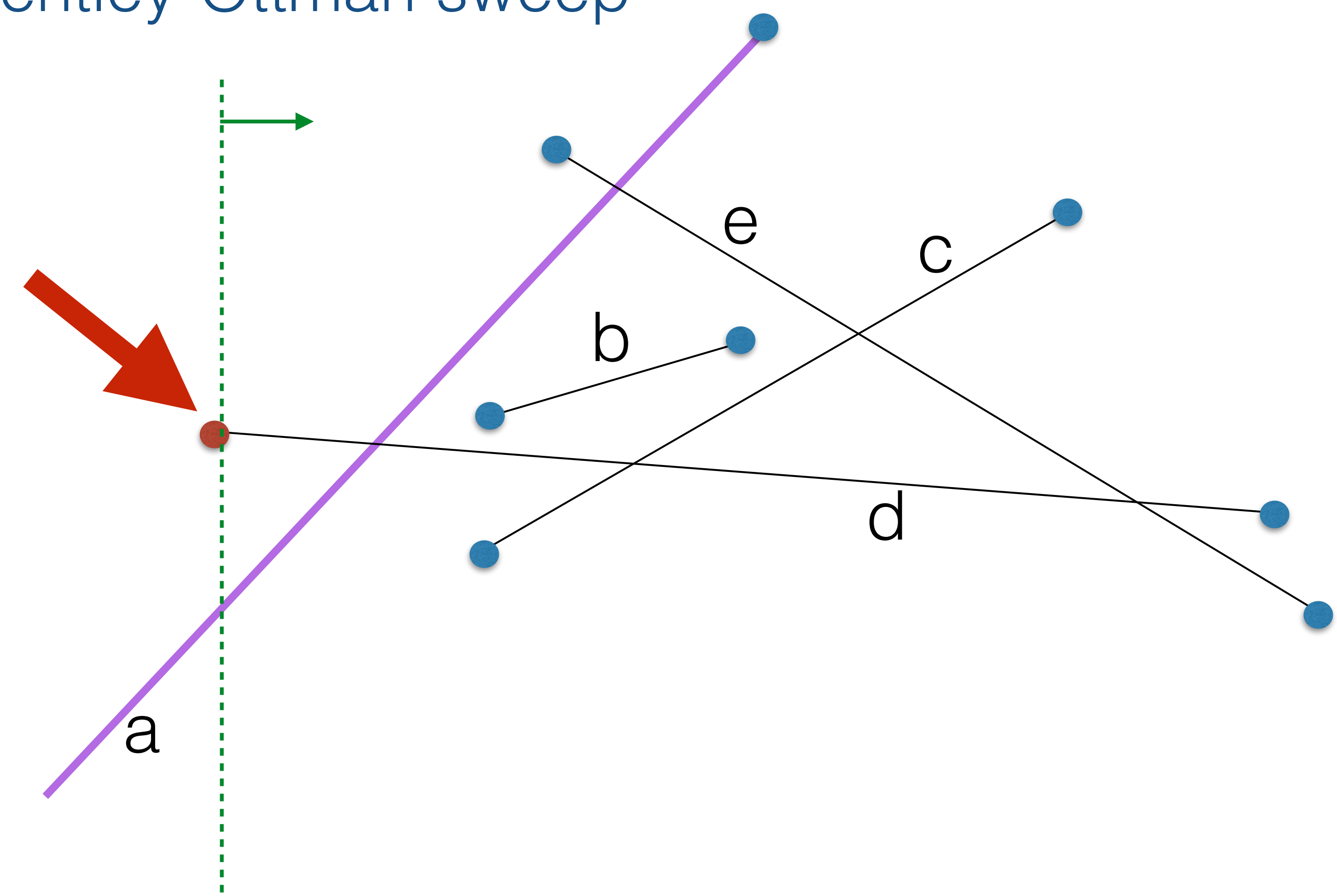
Bentley-Ottman sweep



a.start:

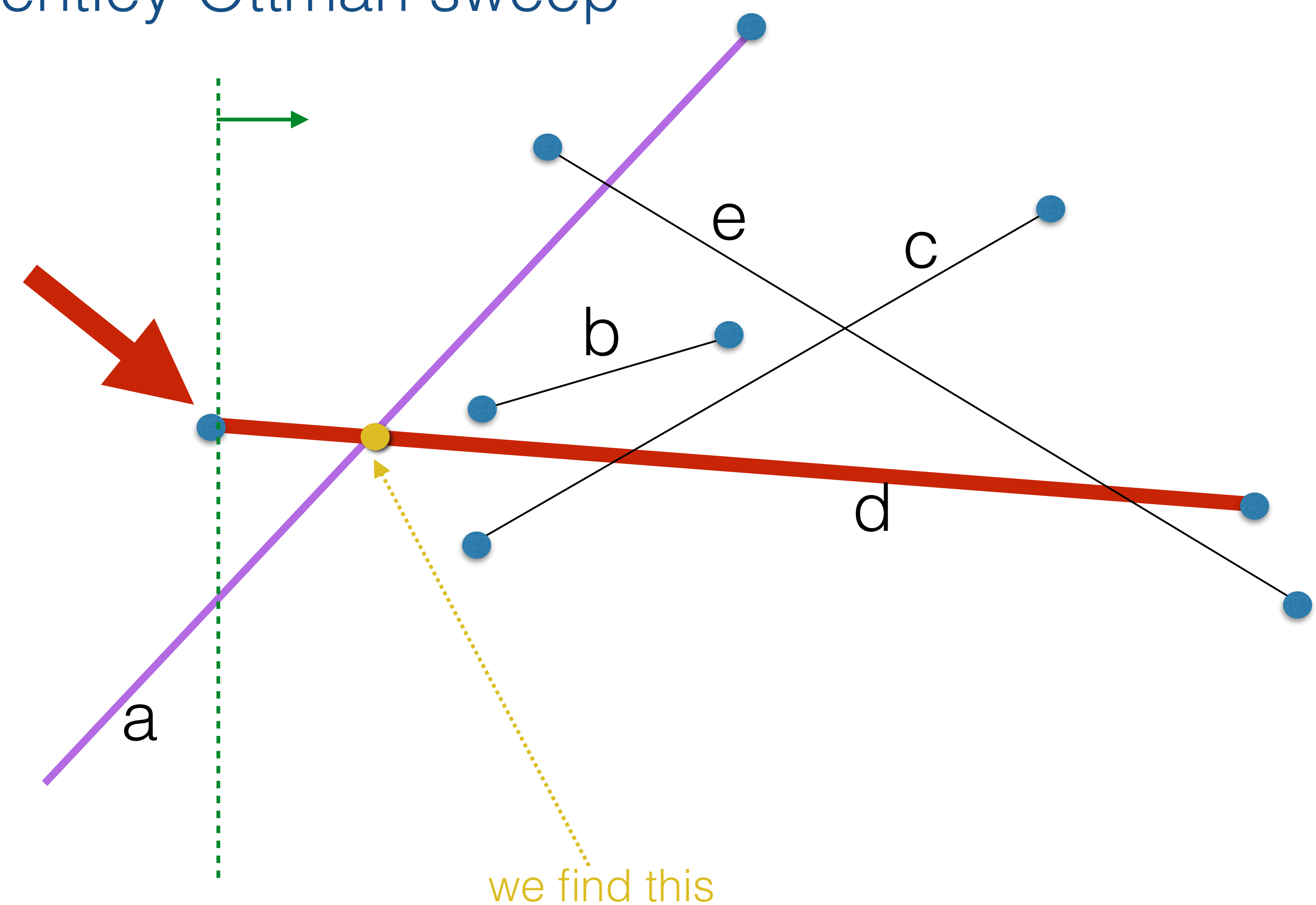
- insert a in AS: a

Bentley-Ottman sweep



d.start:

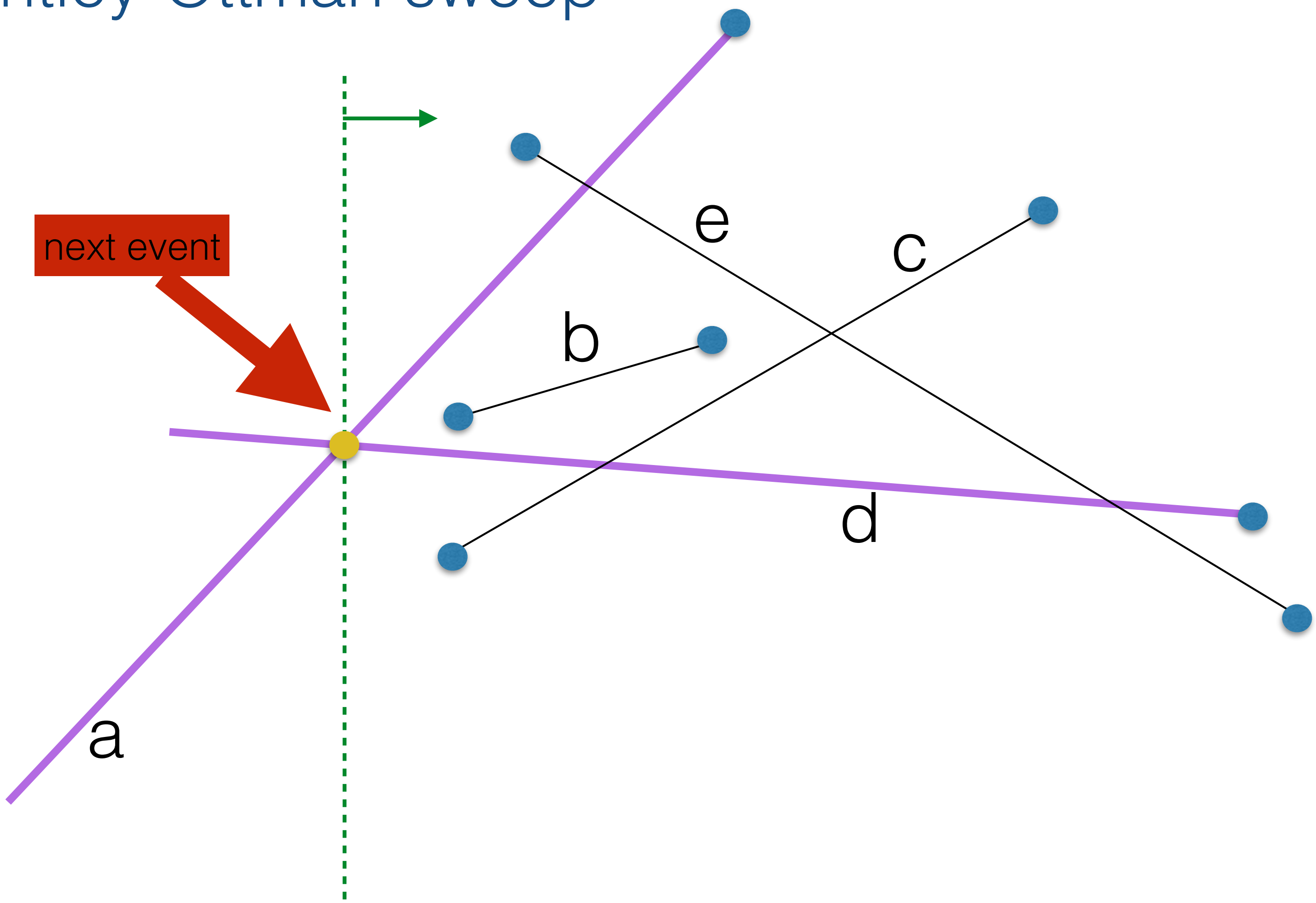
Bentley-Ottman sweep



d.start:

- insert d in AS: **a** < d
- a,d consecutive: check if (a,d) intersect to the right of the line; they do; report point and insert it in the list of future events

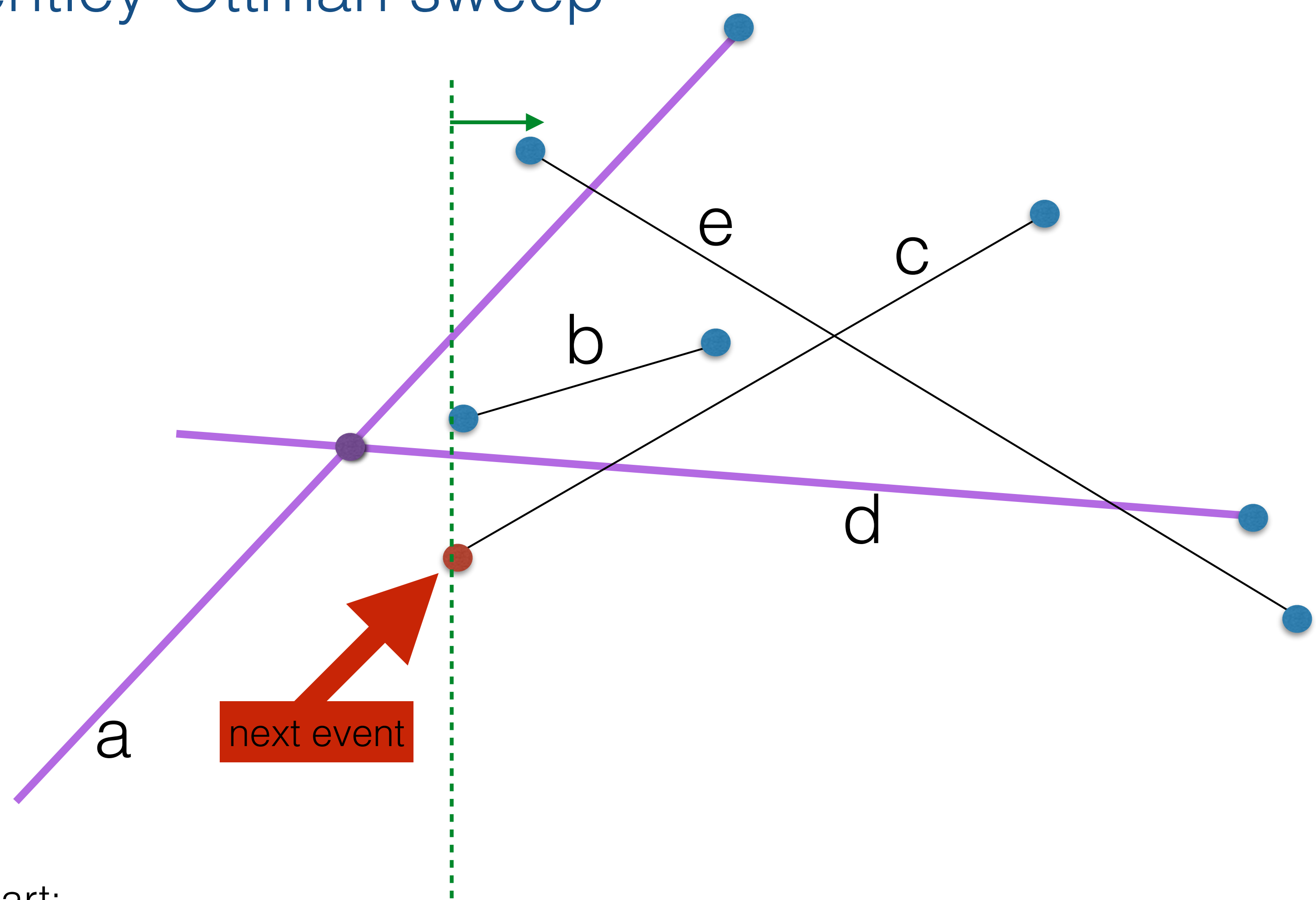
Bentley-Ottman sweep



this event is intersection of (a,d):

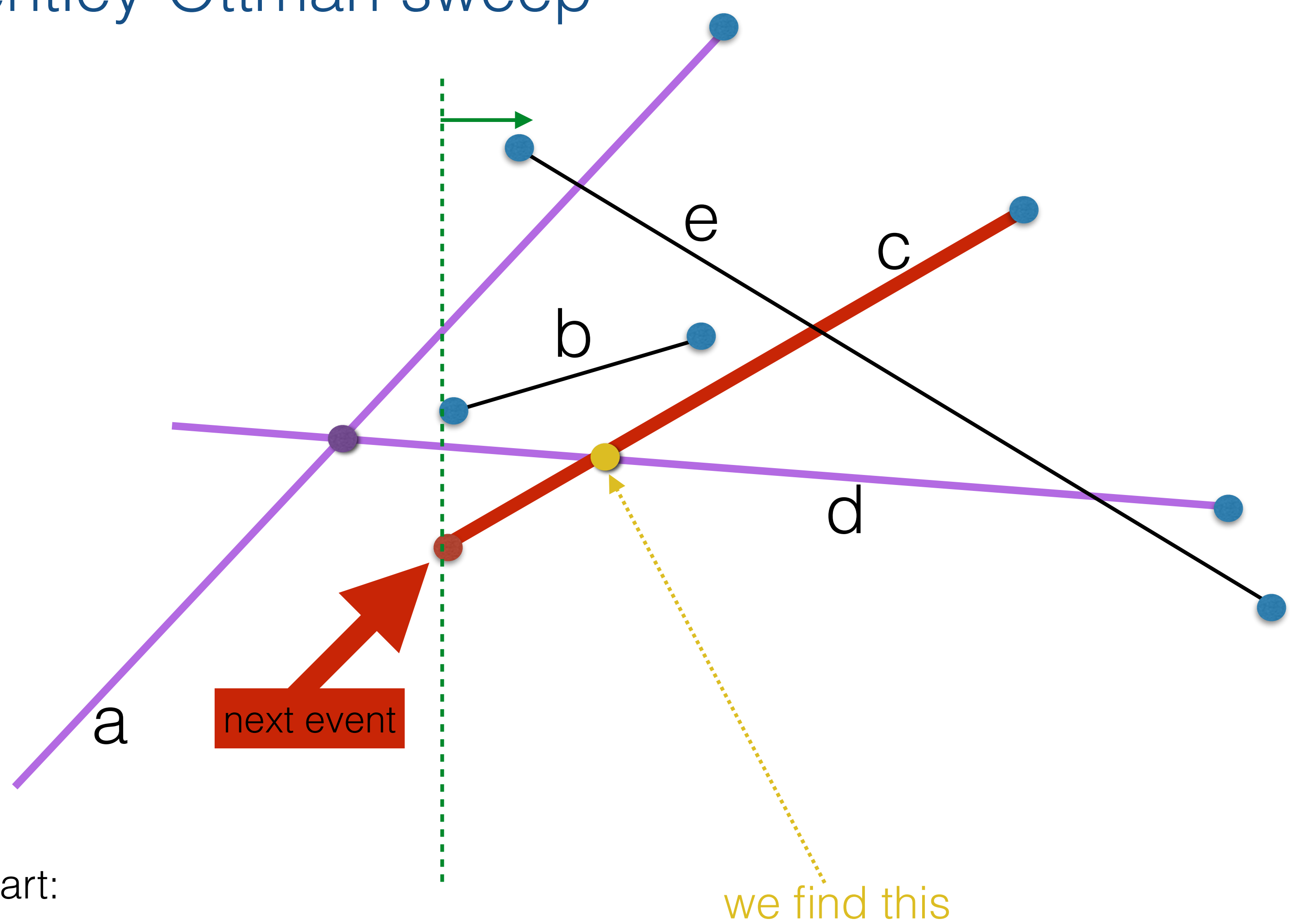
- flip a and d is AS: a is now above d (d < a)

Bentley-Ottman sweep



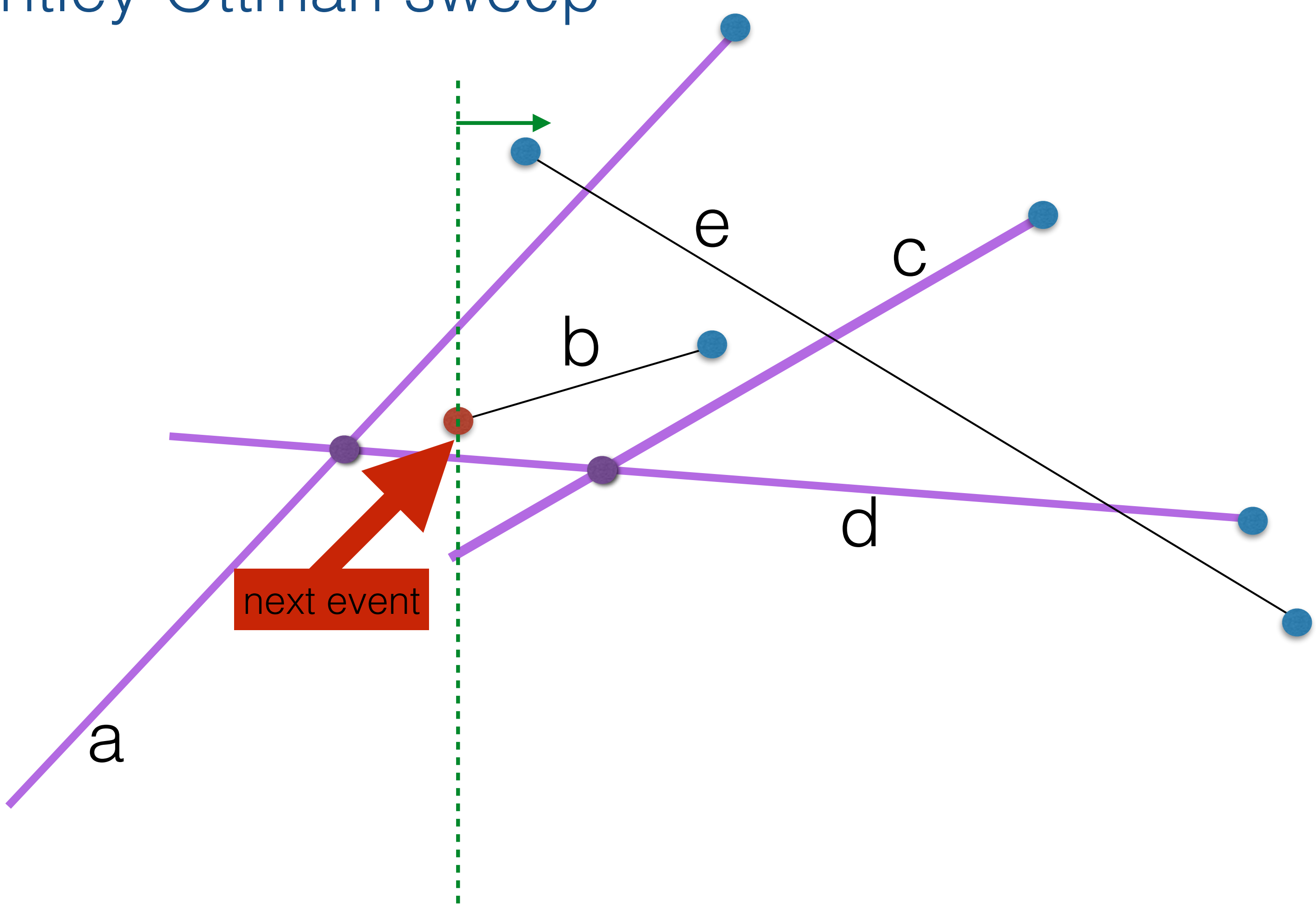
- c.start:

Bentley-Ottman sweep



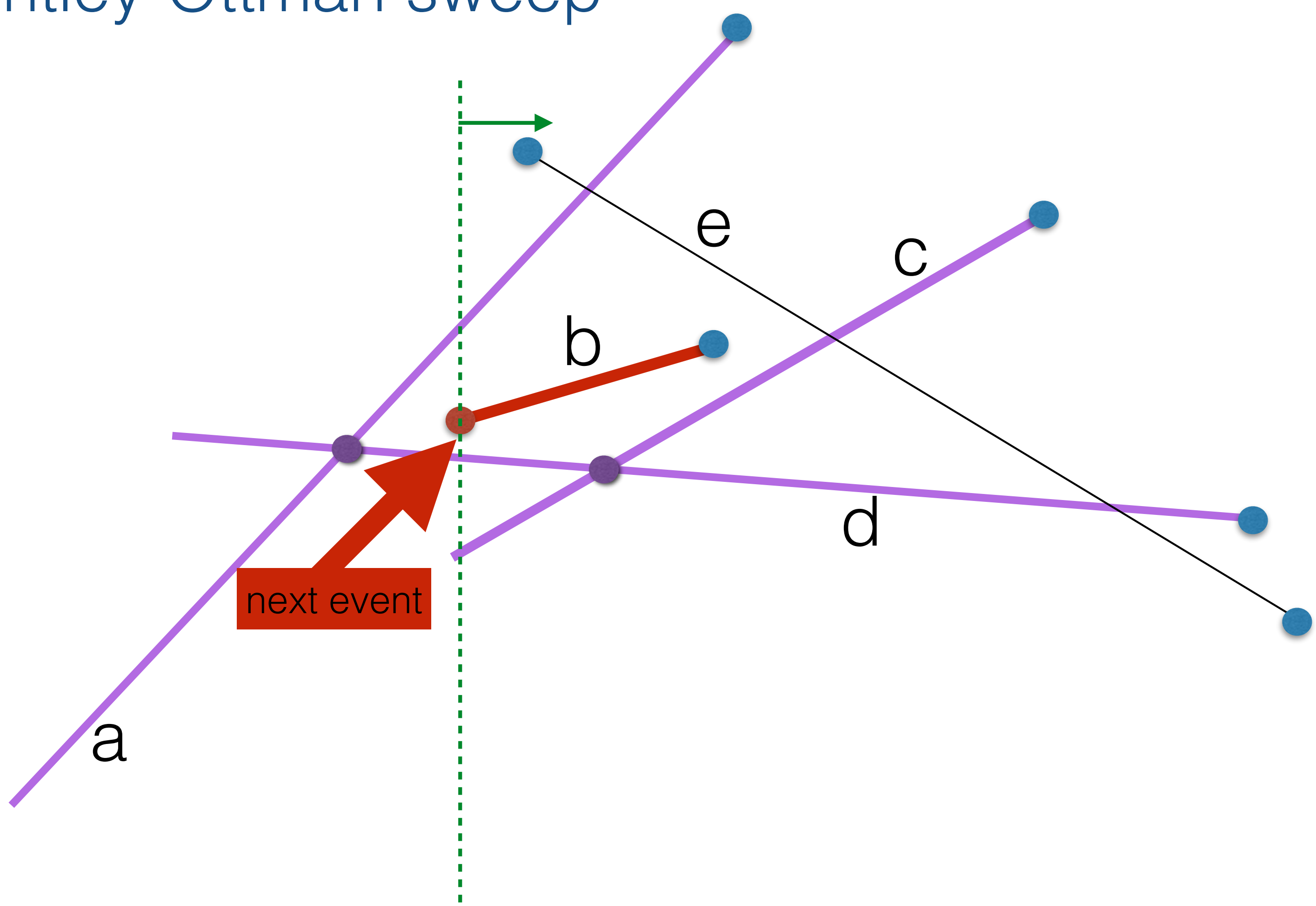
- c.start:
  - insert c in AS: c < d < a
  - check c with its above and below neighbors for intersection to the right of the sweep line; this detects the intersection point of c and d; report it and insert it as future event

Bentley-Ottman sweep



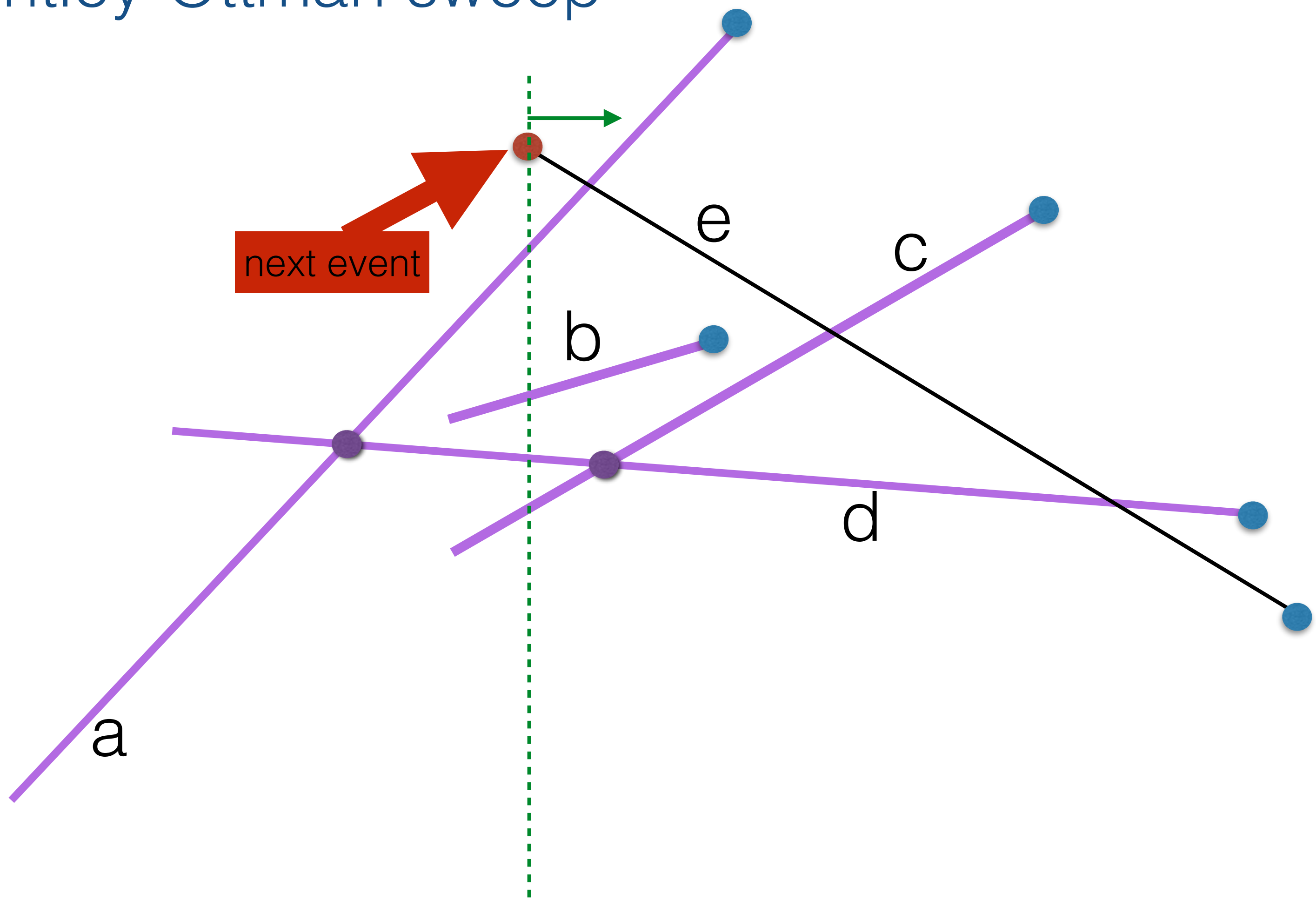
- b.start:

Bentley-Ottman sweep



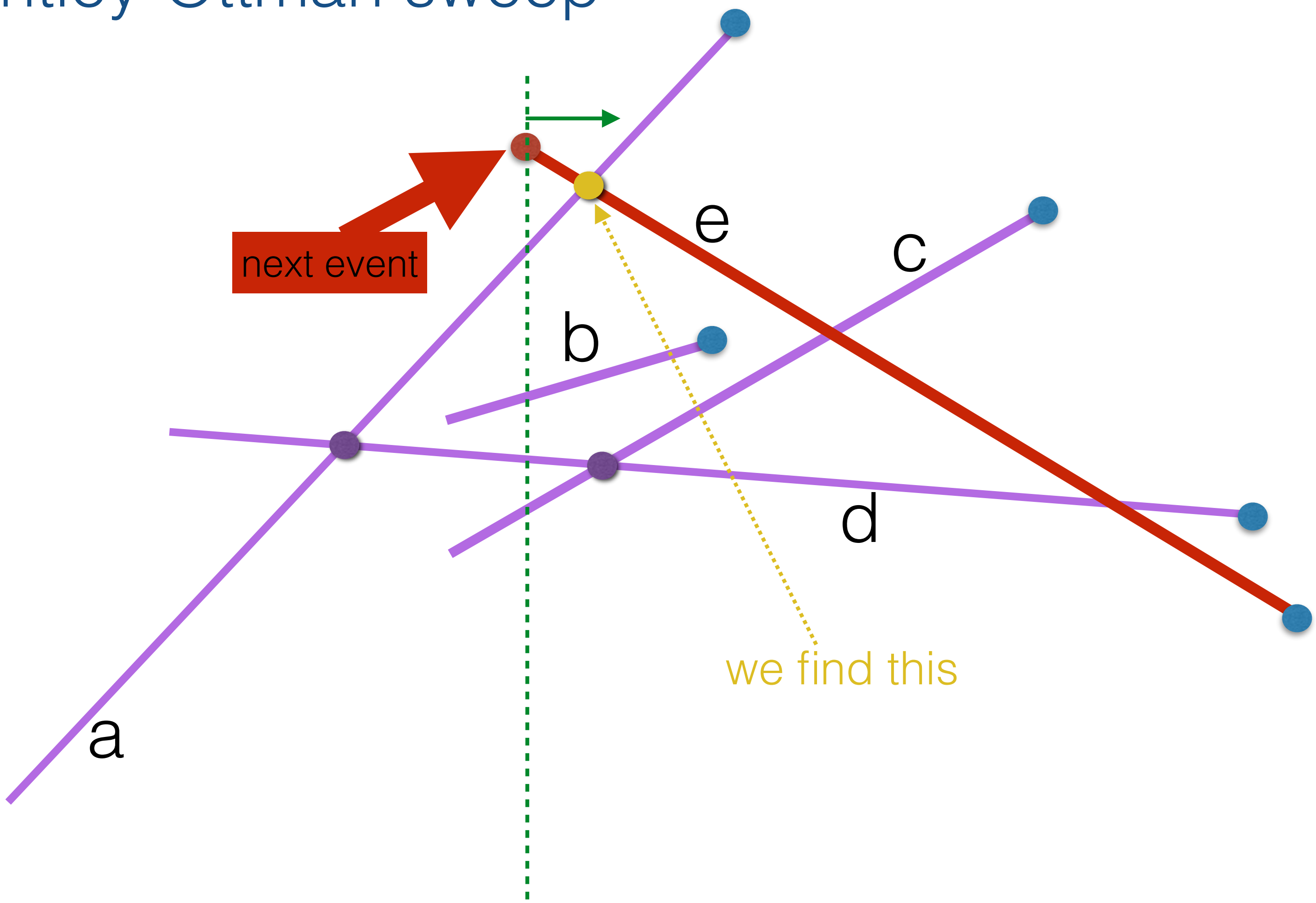
- b.start:
  - insert b in AS; c < d < **b** < a
  - check b with its above and below neighbors for intersection to the right of the sweep line; (d,b) don't intersect; (b, a) don't intersect

Bentley-Ottman sweep



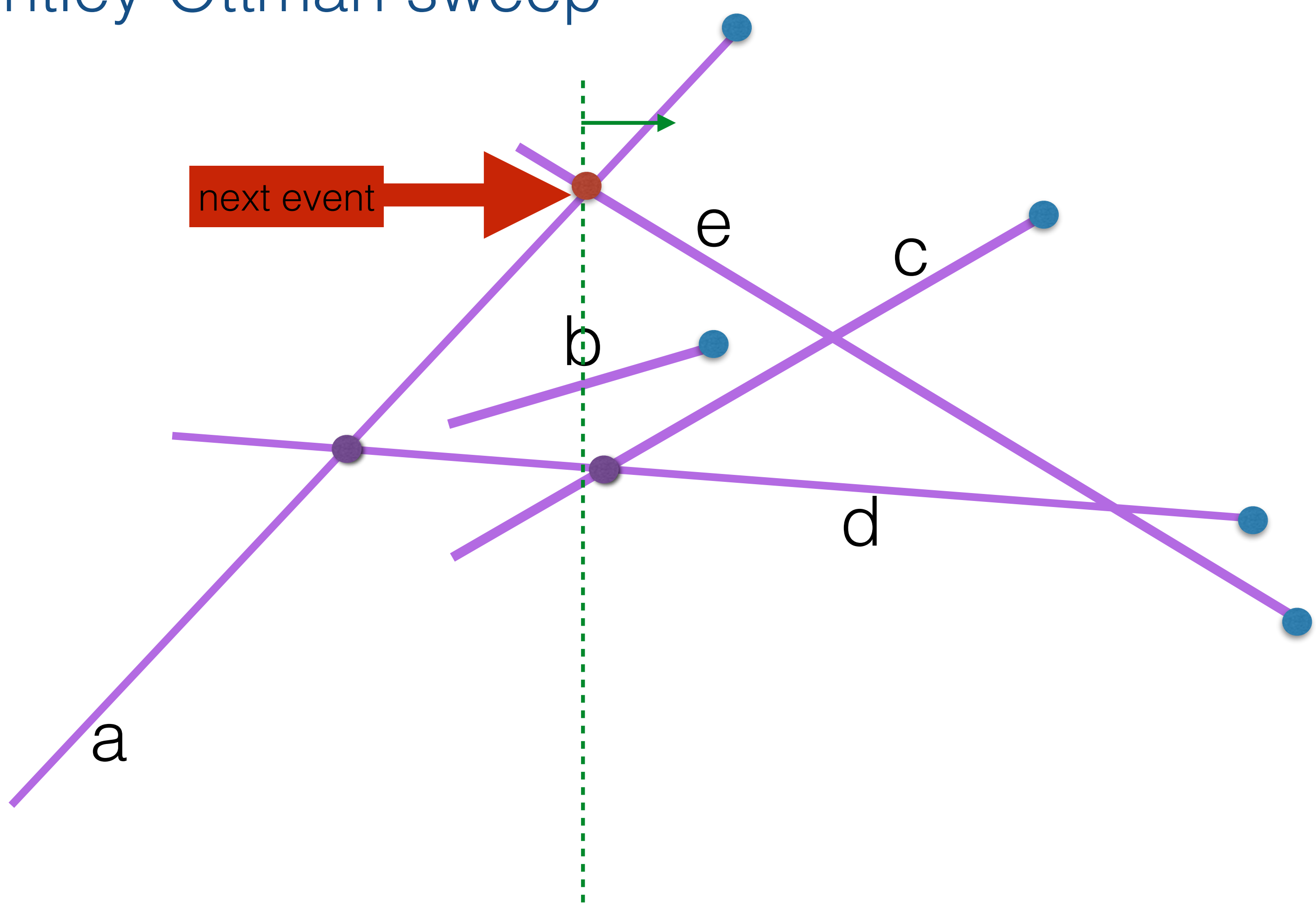
- e.start:

Bentley-Ottman sweep



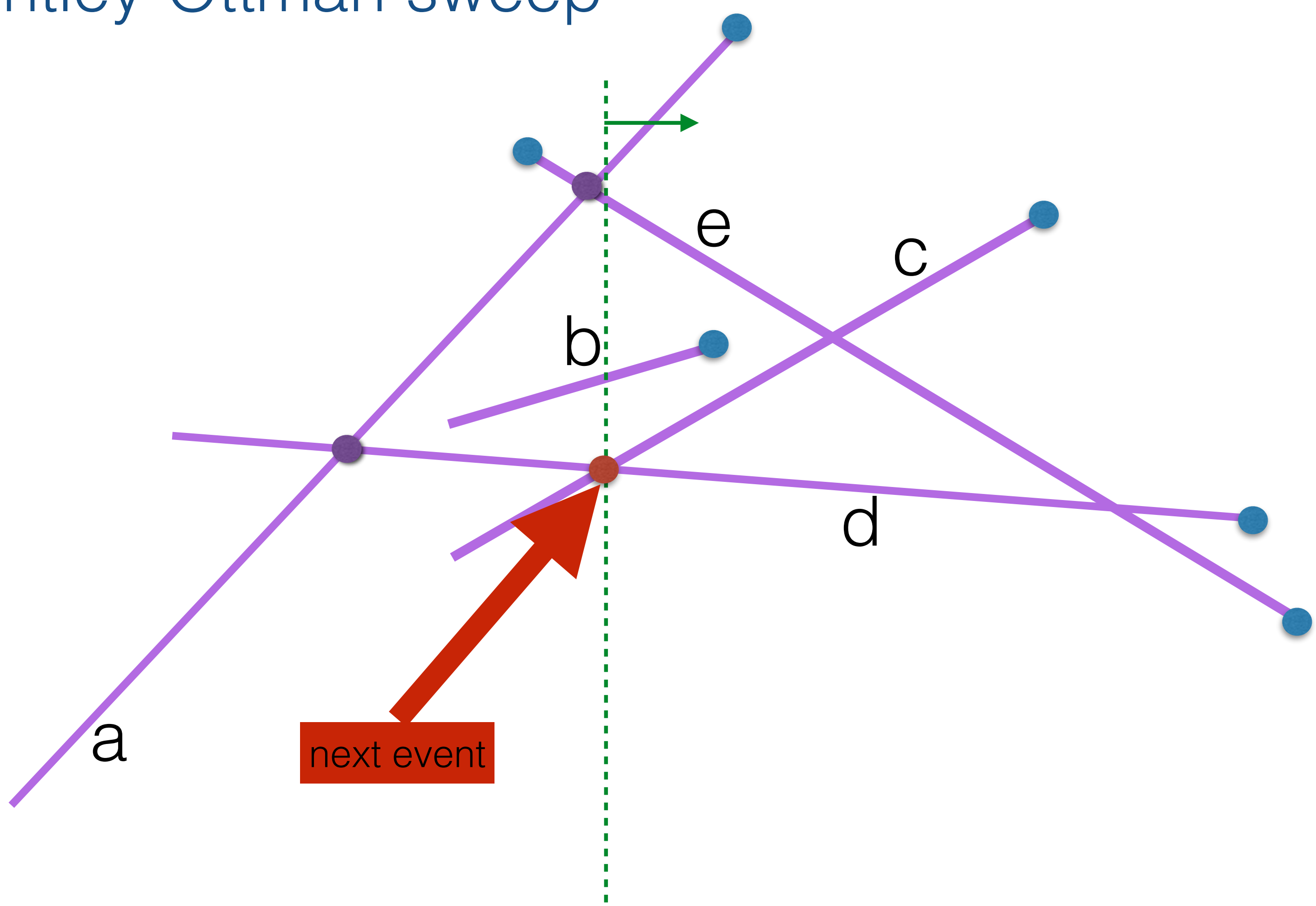
- e.start:
  - insert e in AS: c < d < b < a < **e**
  - check e with its above and below neighbors for intersection to the right of the sweep line; this detects intersection point of (a,e); report it and insert it as future event

Bentley-Ottman sweep



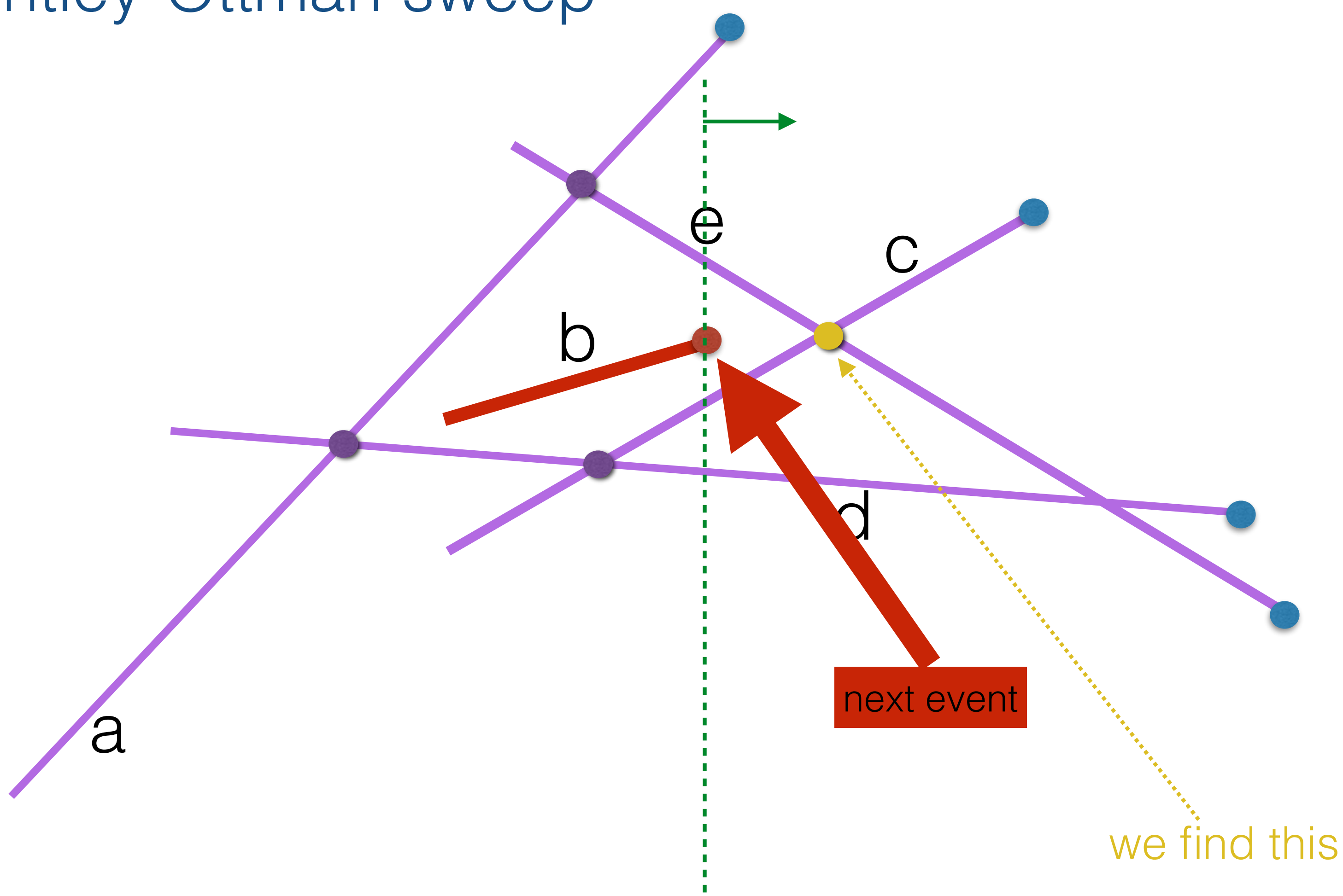
- next event is the intersection of (a,e):
  - flip a and e: c < d < b < **e < a**
  - check new neighbors (e,b) for intersection to the right of the sweep line; (e,b) don’t intersect

Bentley-Ottman sweep



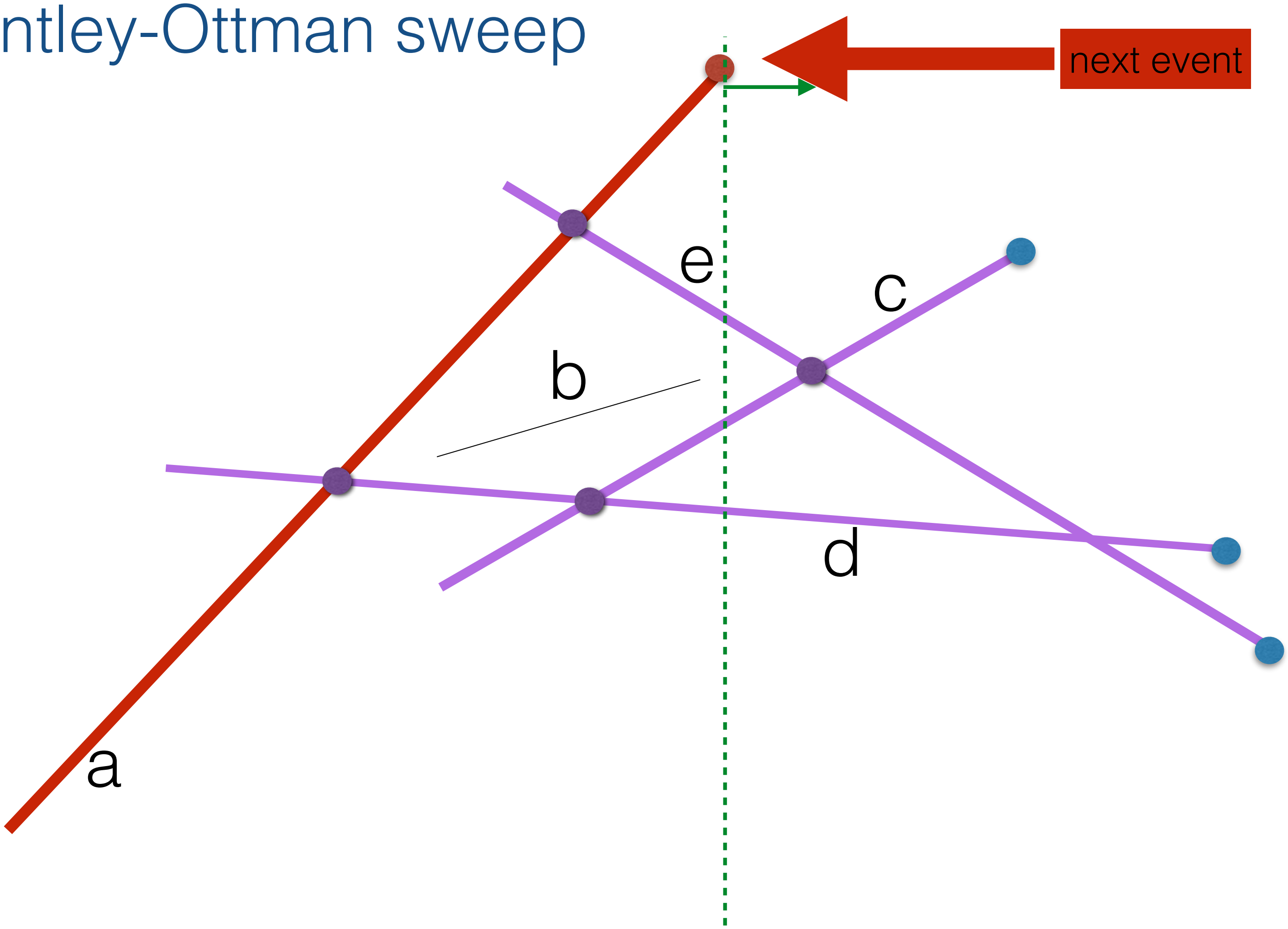
- next event is intersection of (c,d):
  - flip c and d: **d < c** < b < e < a
  - check new neighbors (c,b) for intersection to the right of the sweep line; (c,b) don't intersect

Bentley-Ottman sweep



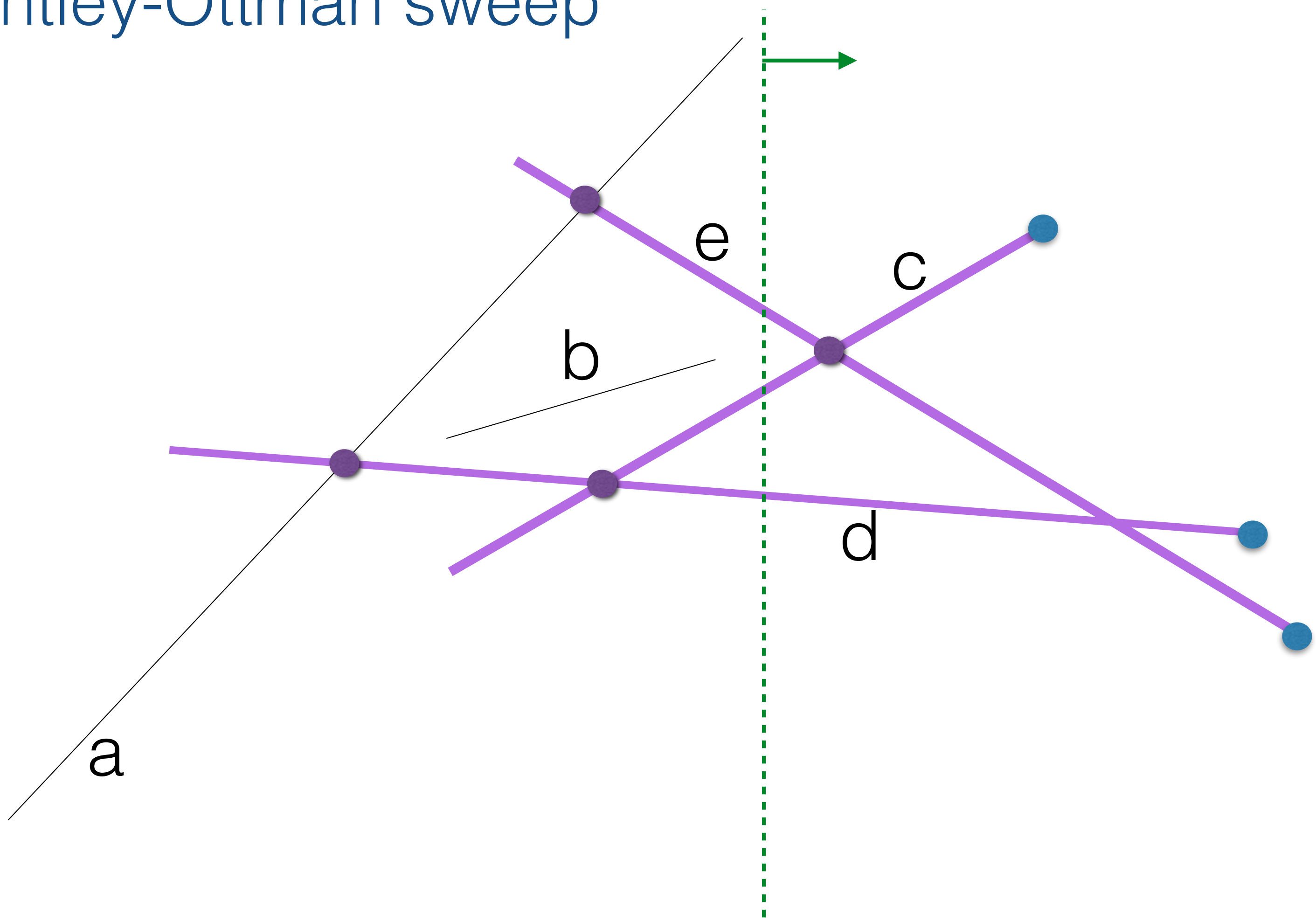
- b.end:
    - delete b from AS: d < c < b < e < a
    - check new neighbors (c,e) for intersection to the right of the sweep line; this detects the intersection point of (c,e); report it and insert it as future event

Bentley-Ottman sweep



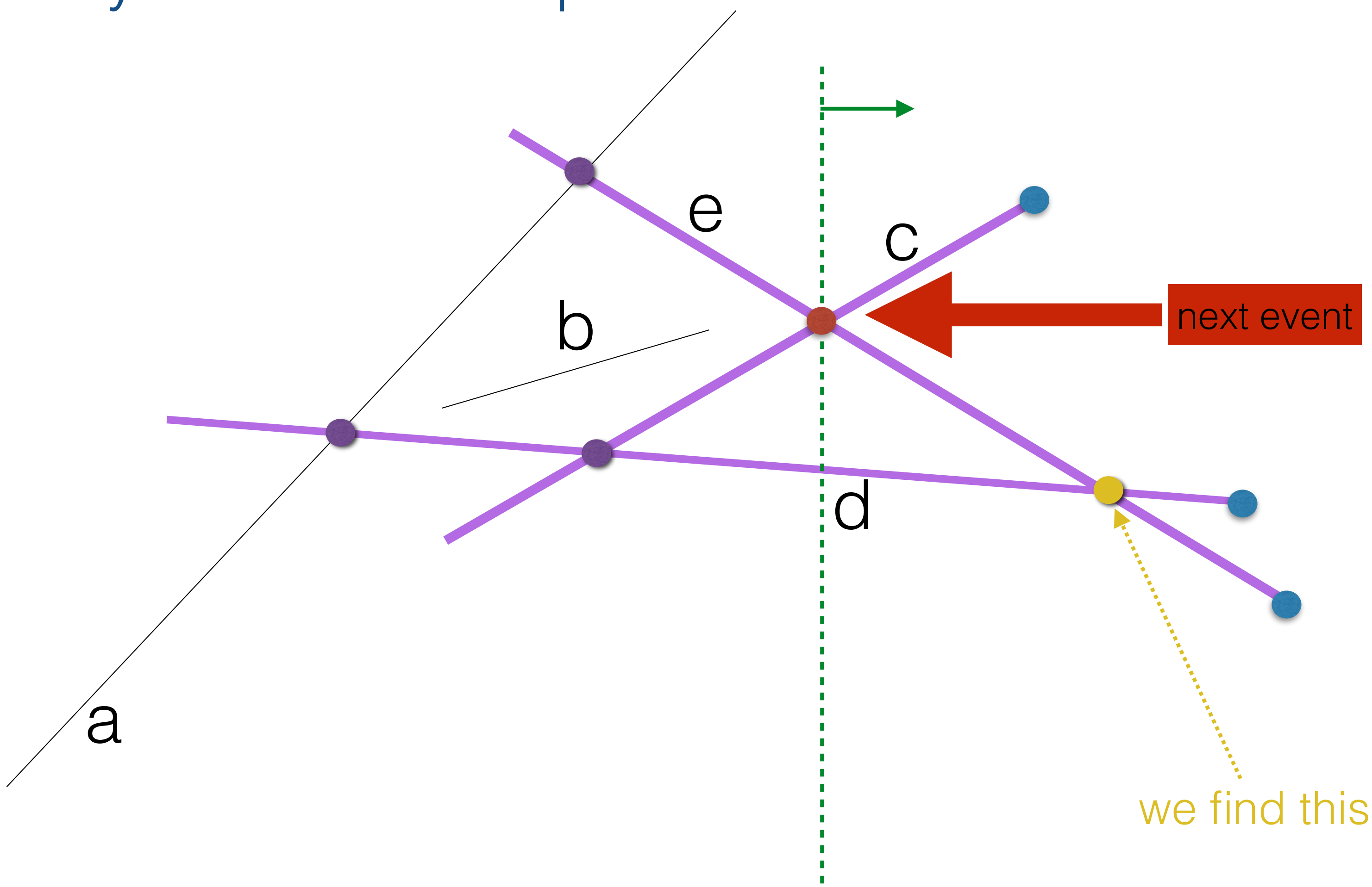
- a.end:
  - delete a from AS: d < c < e < a
  - no new neighbors

Bentley-Ottman sweep



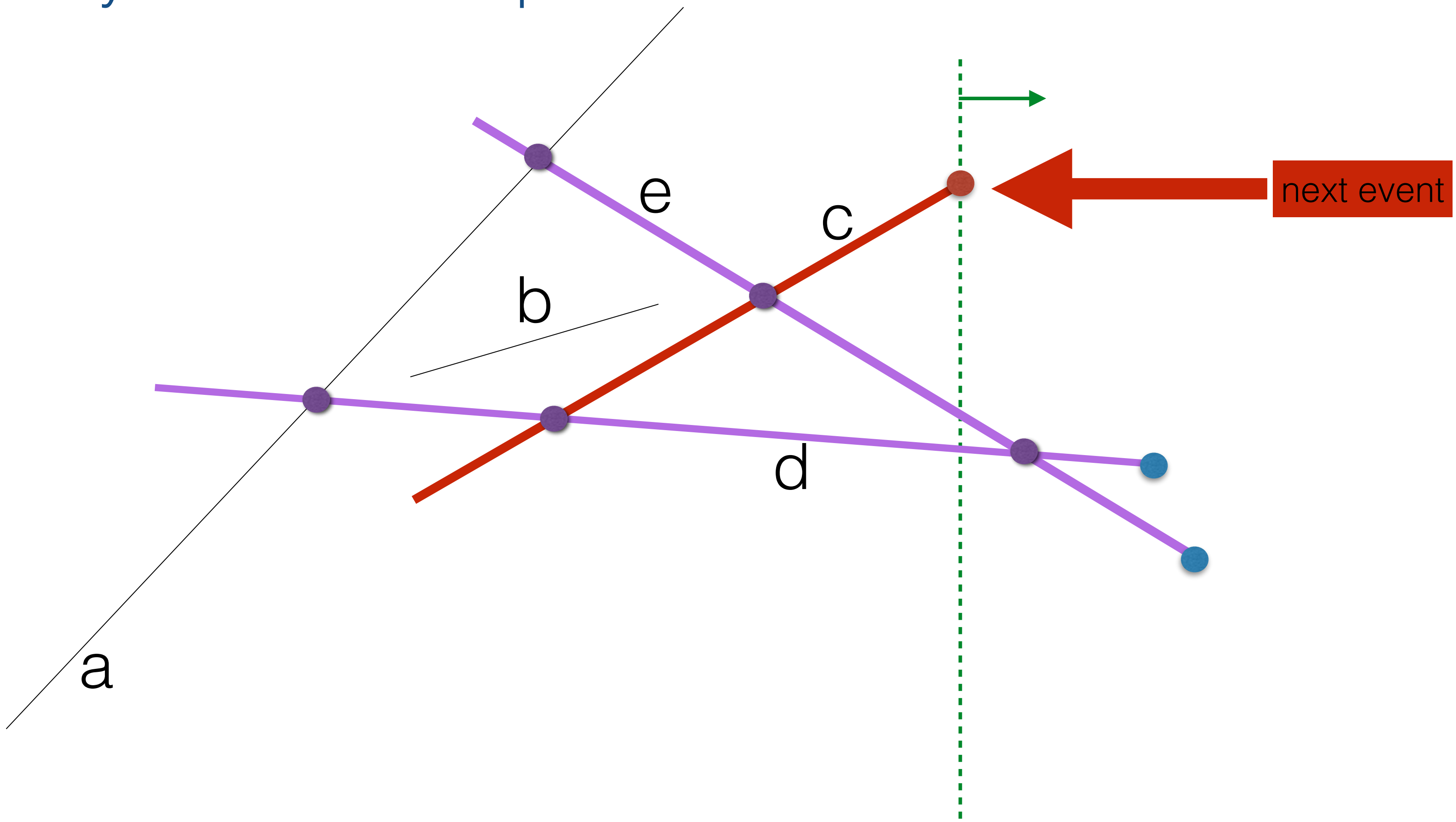
- a.end:
  - delete a from AS: d < c < e < a
  - no new neighbors

Bentley-Ottman sweep



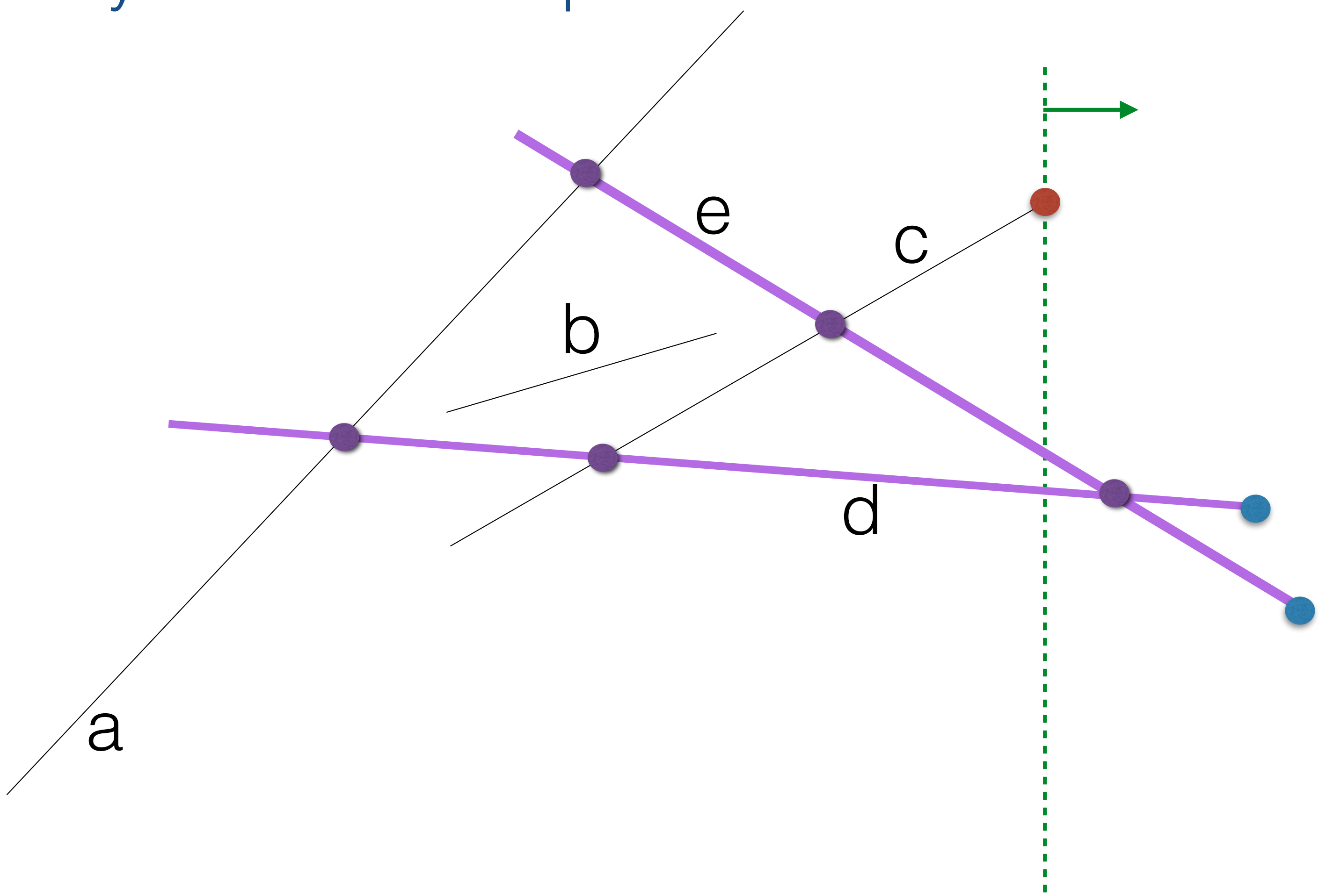
- next event is the intersection of (c,e):
  - flip c,e in AS: d < **e < c**
  - check new neighbors (d,e) for intersection to the right of the sweep line; this detects the intersection of (d,e); report it and insert it as future event

Bentley-Ottman sweep



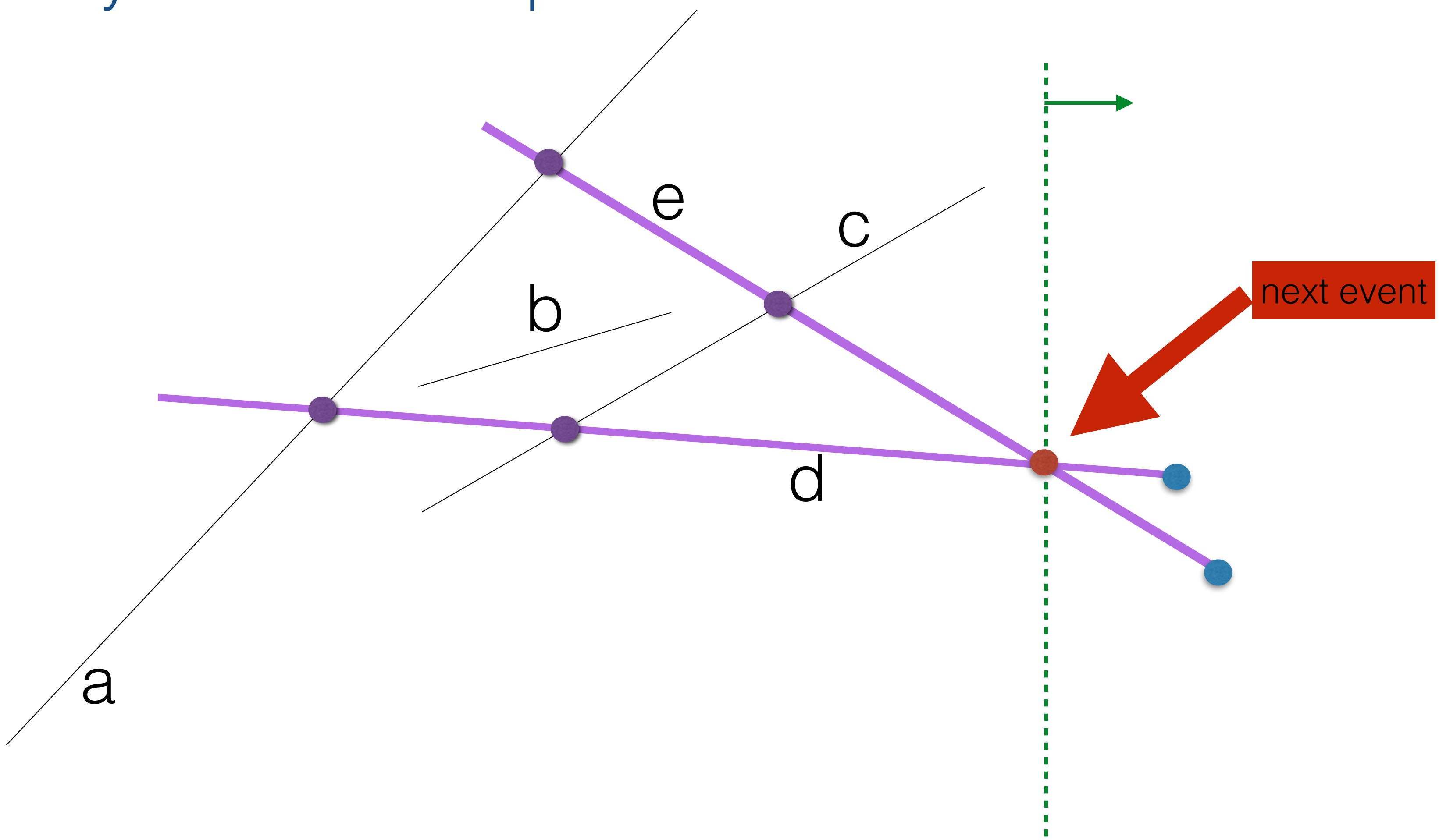
- c.end:
  - delete c in AS: d < e < c
  - no new neighbors

Bentley-Ottman sweep



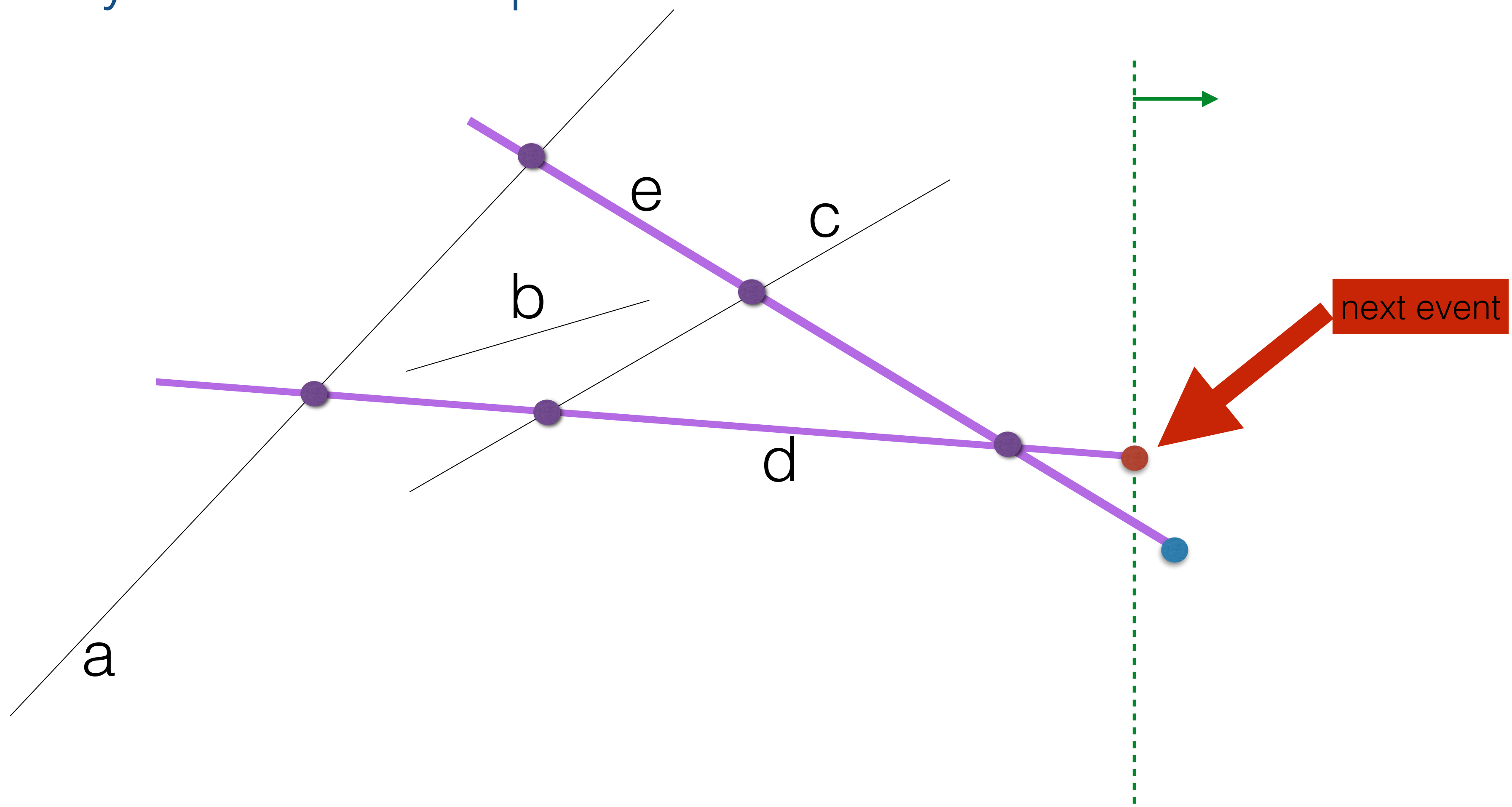
- c.end:
    - delete c in AS: d < e
    - no new neighbors

Bentley-Ottman sweep



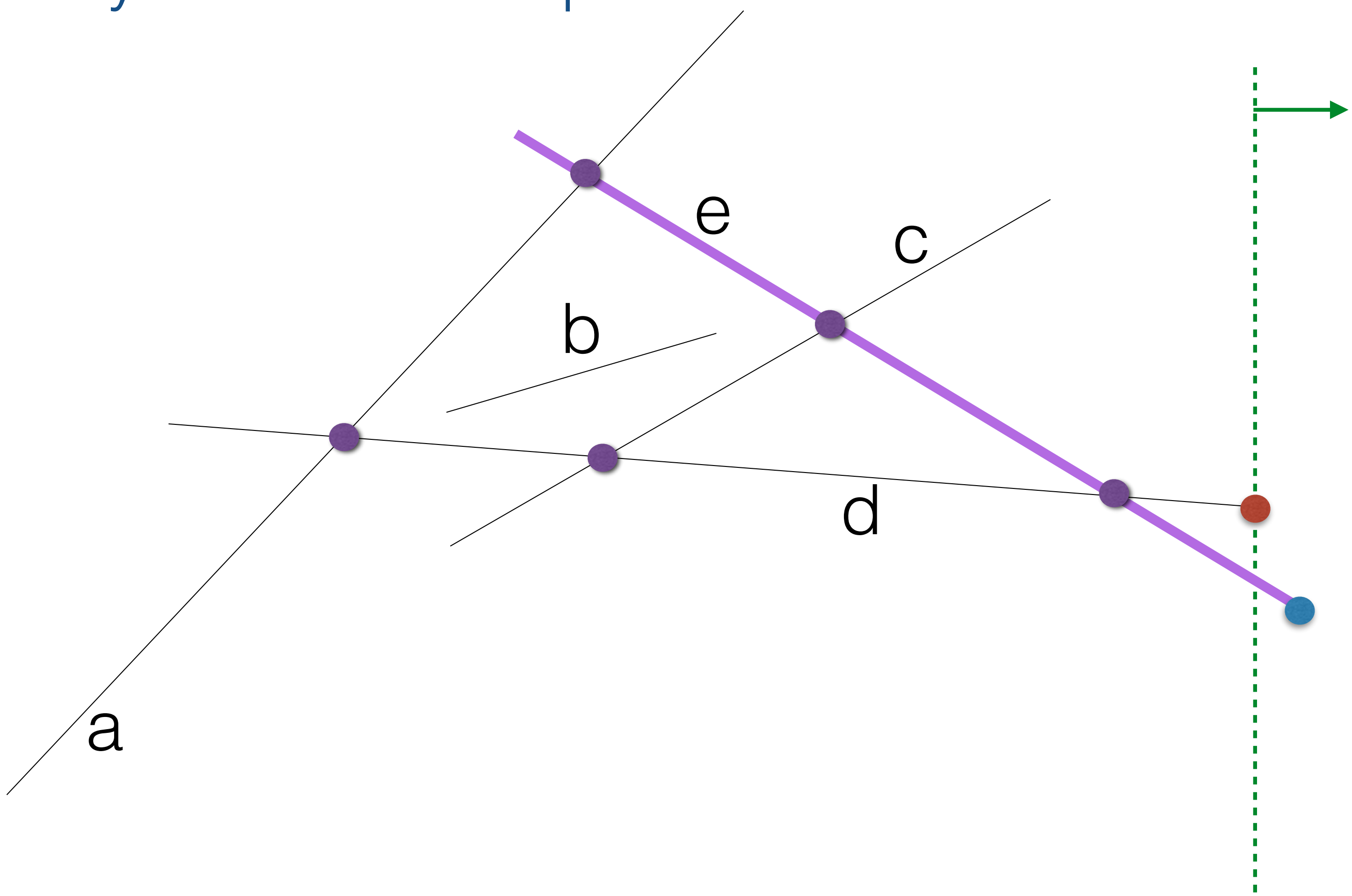
- next event is the intersection of (d,e):
  - flip d,e in AS: **e < d**
  - no new neighbors

Bentley-Ottman sweep



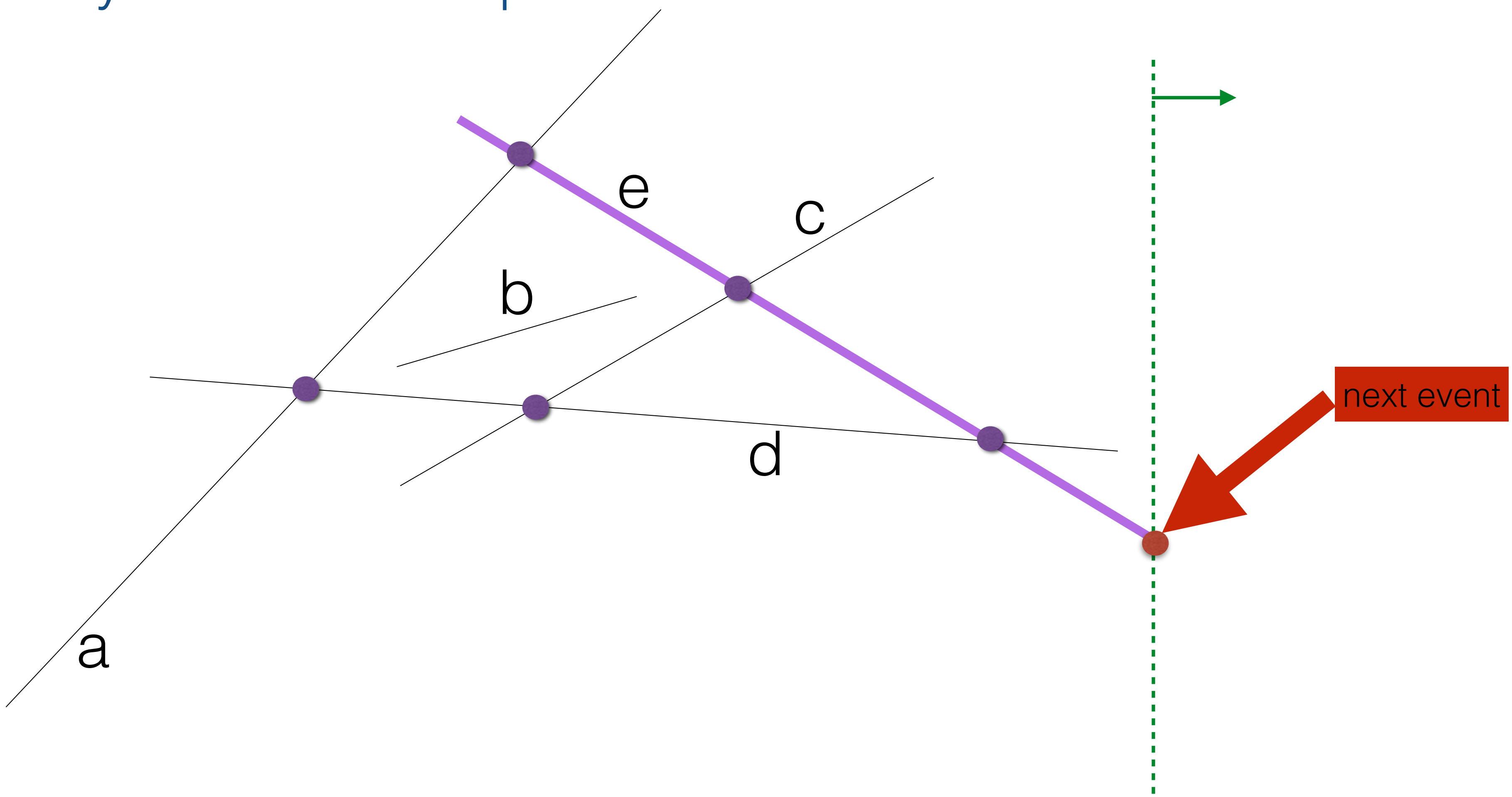
- d.end:
    - delete d in AS: e
    - no new neighbors

Bentley-Ottman sweep



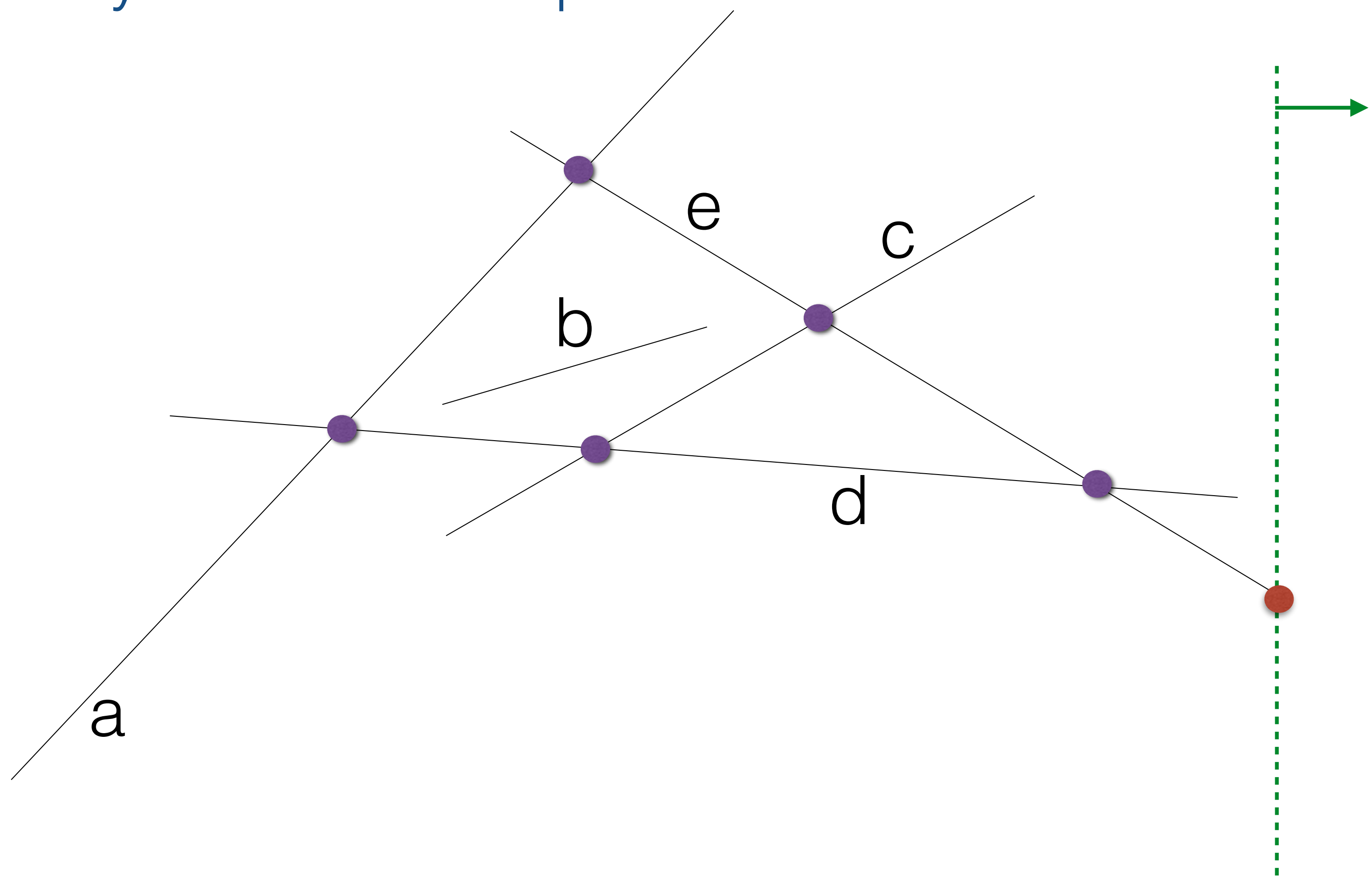
- this event is the end of d:
  - delete d in AS: e
  - no new neighbors

Bentley-Ottman sweep



- e.end:
  - delete e in AS:
  - no new neighbors

Bentley-Ottman sweep



- e.end:
  - delete e in AS:
  - no new neighbors

Bentley-Ottman sweep

SL: sweep line

To implement these ideas, we'll maintain two data structures:

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

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

Algorithm Bentley-Ottman (S)

- 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 EventList

  //sweep line moves to SL.x=e.x

  - if e is left endpoint of a segment l

    // segment l becomes active
    - insert l in AS
    - 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 l
    - delete l from AS
    - …

  - if e is the intersection of two segments
    - search for the two segments in AS and flip their order…
    - ….

Bentley-Ottman sweep

- For simplicity, we made some 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

The details

- Active structure
  - What data structure should we use for AS?
    - What operations do we do on AS?

- EventList
  - Note that we know a priori the 2n events corresponding to start and end-points of segments, but the events corresponding to intersection points are generated on the fly
  - What data structure should we use for EL?
    - What operations do we do on EL?

Analysis

Running time

- AS
  - Size: $O(n)$
  - How many operations? $O(n+k)$
  - Overall time? $O((n+k) \cdot \lg n)$

- EventList
  - Size: $O(n+k)$
  - How many operations? $O(n+k)$
  - Overall time? $O((n+k) \cdot \lg n)$

Result: The intersections of a set of n segments in the plane can be found with the Bentley-Ottman sweep algorithm in $O((n+k) \cdot \lg n)$ time.