Advanced Algorithms

Geometric Algorithm

Determining whether any pair of segments intersects

Our algorithm for determining whether any two of *n* line segments intersect makes two simplifying assumptions.

First, we assume that no input segment is vertical.

Second, we assume that no three input segments intersect at a single point.

This Shamos-Hoey Algorithm is developed in 1976

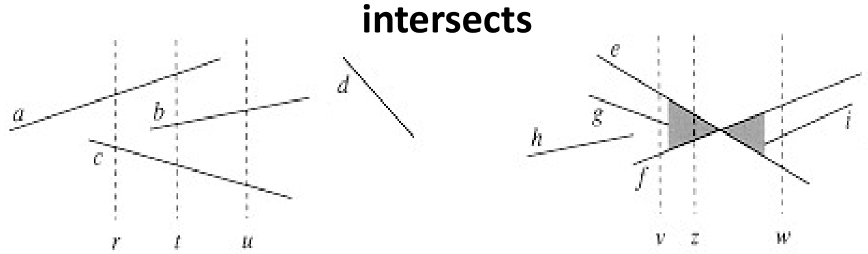
Determining whether any pair of segments intersects

The algorithm uses a technique known as "sweeping," which is common to many computational-geometry algorithms.

In ***sweeping***, an imaginary vertical ***sweep line*** passes through the given set of geometric objects, usually from left to right. The spatial dimension that the sweep line moves across, in this case the *x*-dimension, is treated as a dimension of time.

The line-segment-intersection algorithm considers all the line-segment endpoints in left-to-right order and checks for an intersection each time it encounters an endpoint.

Determining whether any pair of segments intersects



(a) (b)

a >r c

a >t b

a >t c

b >t c

b >u c

e >v f

f >w e

Determining whether any pair of segments intersects

Since we assume that there are no vertical segments, any input segment that intersects a given vertical sweep line intersects it at a single point.

We can thus order the segments that intersect a vertical sweep line according to the *y*-coordinates of the points of intersection.

Consider two nonintersecting segments s_1 and s_2 We say that these segments are ***comparable*** at x if the vertical sweep line with x-coordinate x intersects both of them.

We say that s_l is **above** s_2 at x, written s_1 >x s_2, if s_1 and s_2 are comparable at x and the intersection of s_1 with the sweep line at x is higher than the intersection of s_2 with the same sweep line.

Determining whether any pair of segments intersects

Sweeping algorithms typically manage two sets of data:

1. The ***sweep-line status*** gives the relationships among the objects intersected by the sweep line.

2. The ***event-point schedule*** is a sequence of *x*-coordinates, ordered from left to right, that defines the halting positions of the sweep line. We call each such halting position an ***event point***. Changes to the sweep-line status occur only at event points.

Determining whether any pair of segments intersects

The sweep-line status is a total order *T*, for which we require the following operations:

`INSERT`(*T, s*): insert segment *s* into *T.*

`DELETE`(*T*, *s*): delete segment *s* from *T.*

`ABOVE`(*T*, *s*): return the segment immediately above segment *s* in *T.*

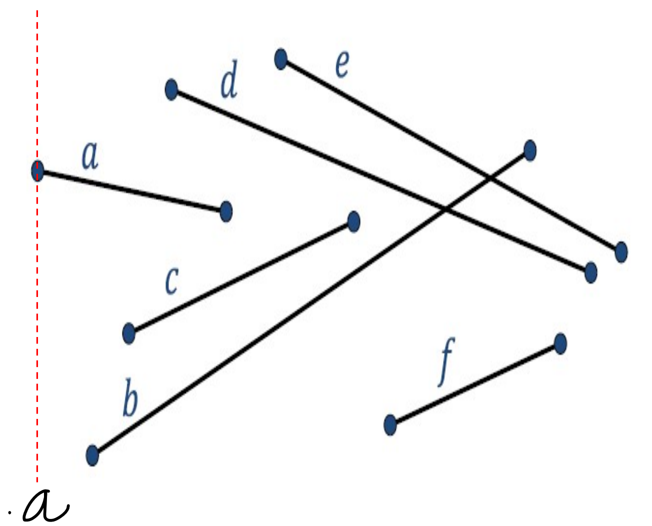
`BELOW`(*T*, *s*): return the segment immediately below segment *s* in *T.*

If there are *n* segments in the input, we can perform each of the above operations in *O(*lg *n)* time using red-black trees **OR** AVL Tree.

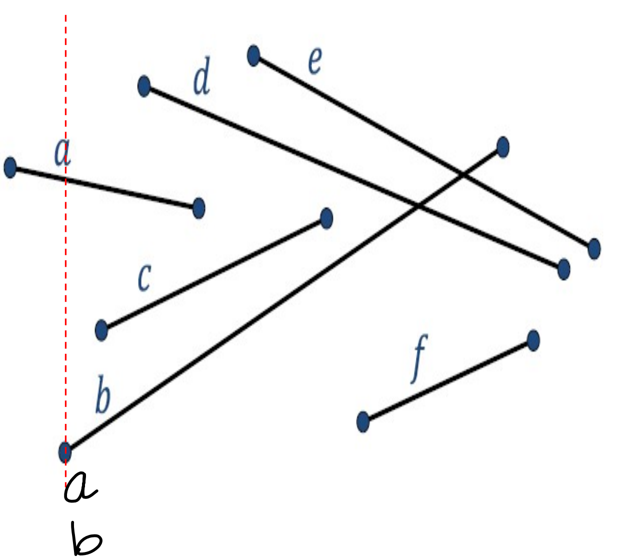
Determining whether any pair of segments intersects

```
ANY-SEGMENTS-INTERSECT(S)
T ← Ø
sort the endpoints of the segments in S from left to right,
          breaking ties by putting left endpoints before right endpoints
          and breaking further ties by putting points with lower
          y-coordinates first
for each point p in the sorted list of endpoints
     do if p is the left endpoint of a segment s
           then INSERT(T, s)
                if (ABOVE(T, s) exists and intersects s)
                       or (BELOW(T, s) exists and intersects s)
                   then return TRUE
        if p is the right endpoint of a segment s
           then if both ABOVE(T, s) and BELOW(T, s) exist
                        and ABOVE(T, s) intersects BELOW(T, s)
                   then return TRUE
                DELETE(T, s)
return FALSE
```

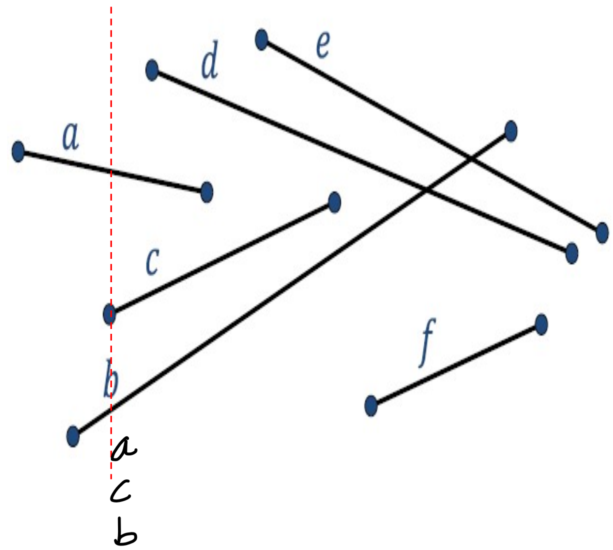
Determining whether any pair of segments intersects



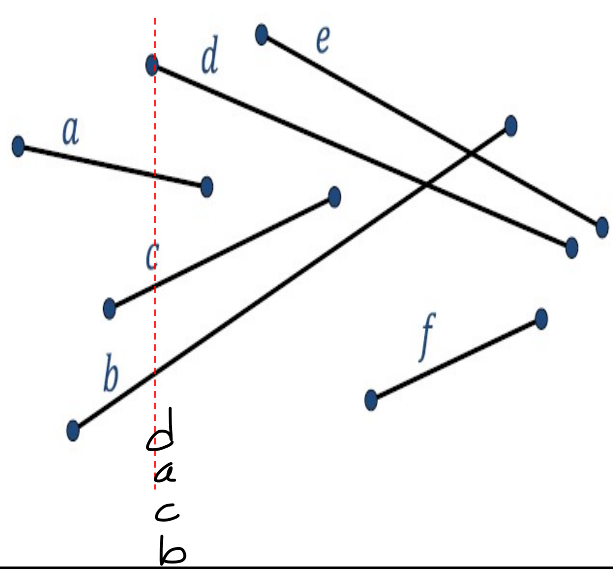
Determining whether any pair of segments intersects



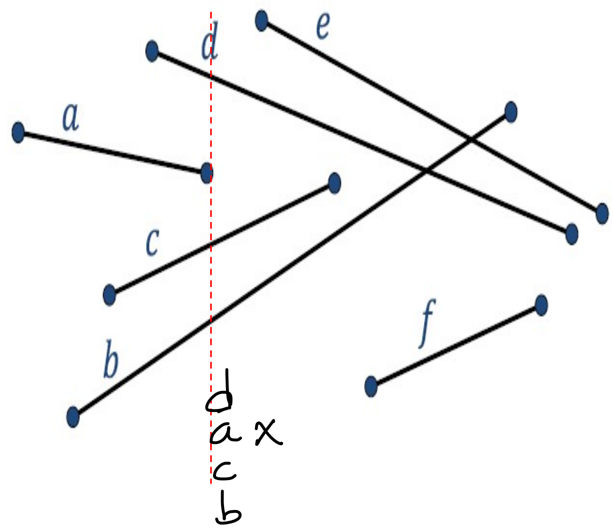
Determining whether any pair of segments intersects



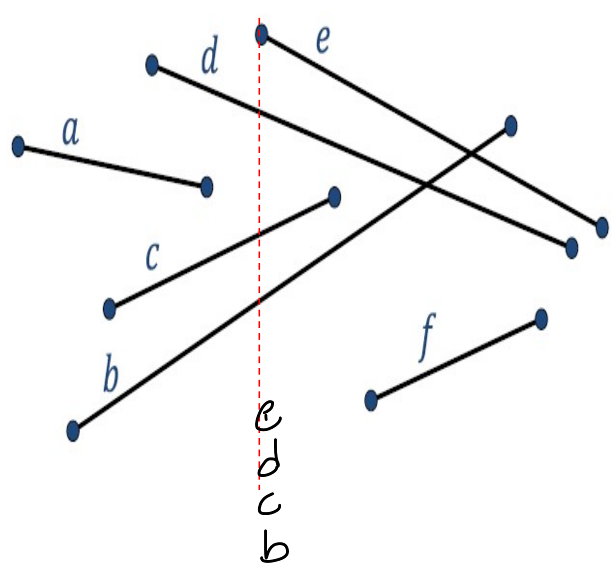
Determining whether any pair of segments intersects



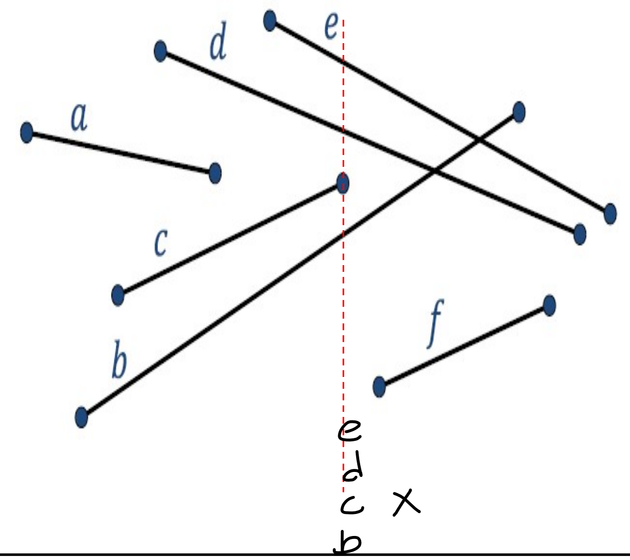
Determining whether any pair of segments intersects



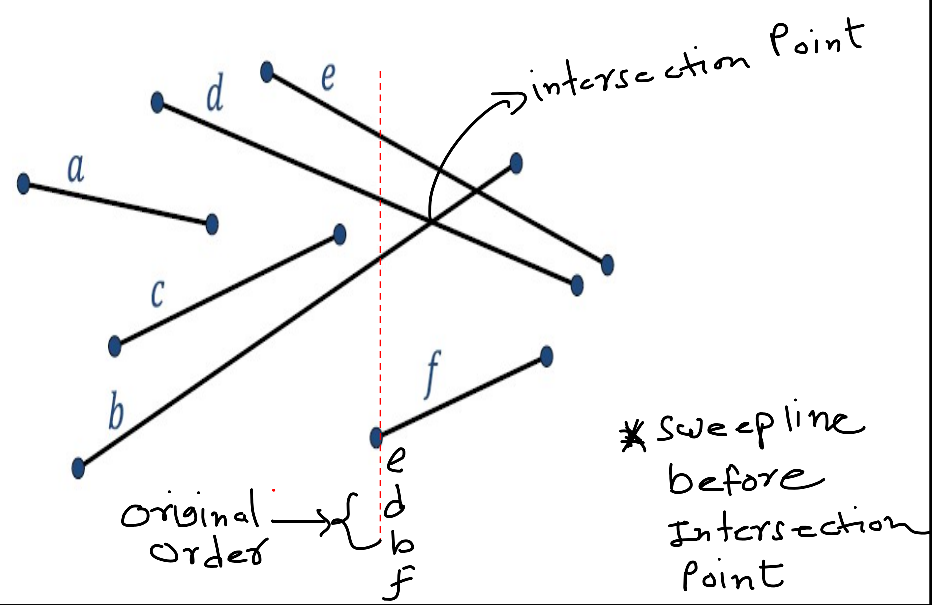
Determining whether any pair of segments intersects



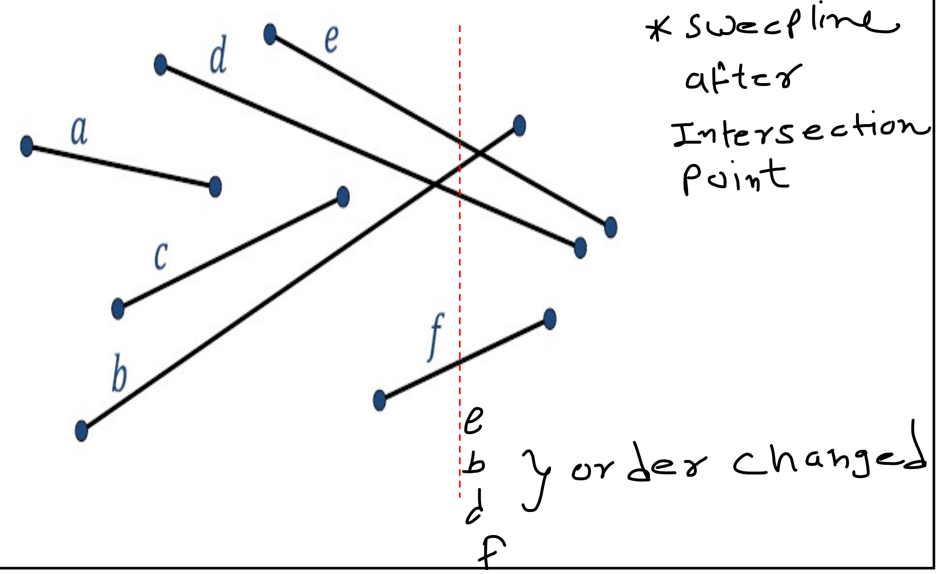
Determining whether any pair of segments intersects



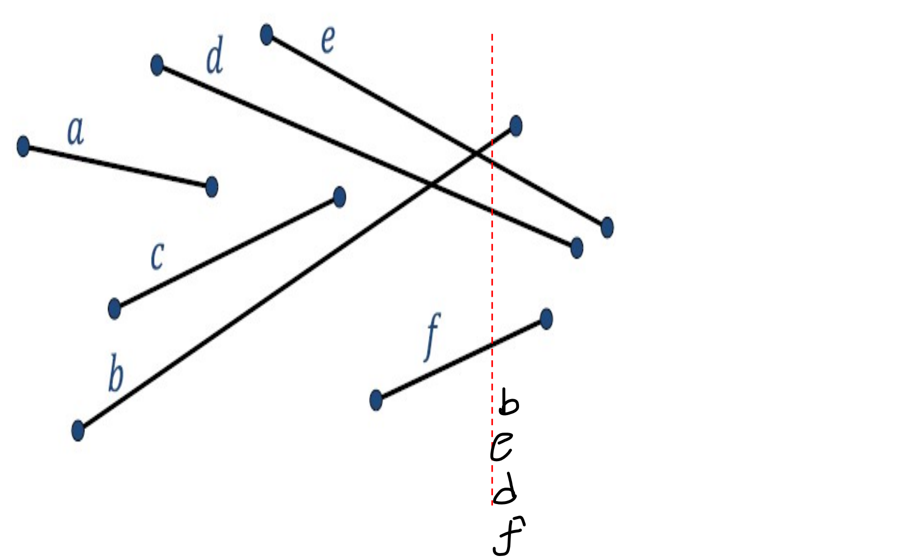
Determining whether any pair of segments intersects



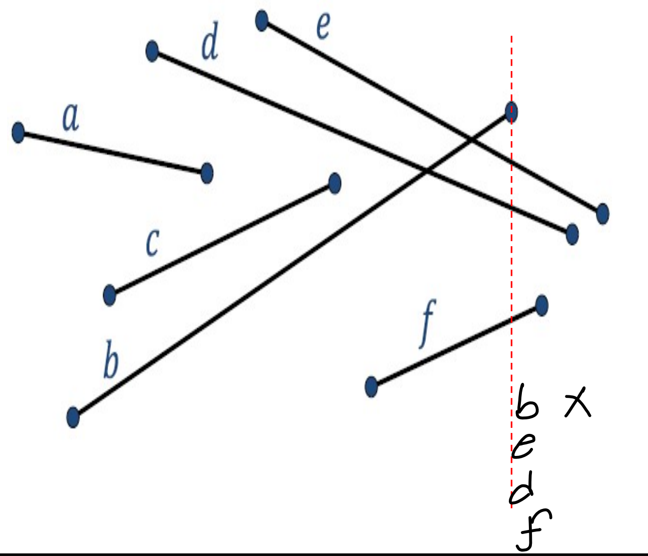
Determining whether any pair of segments intersects



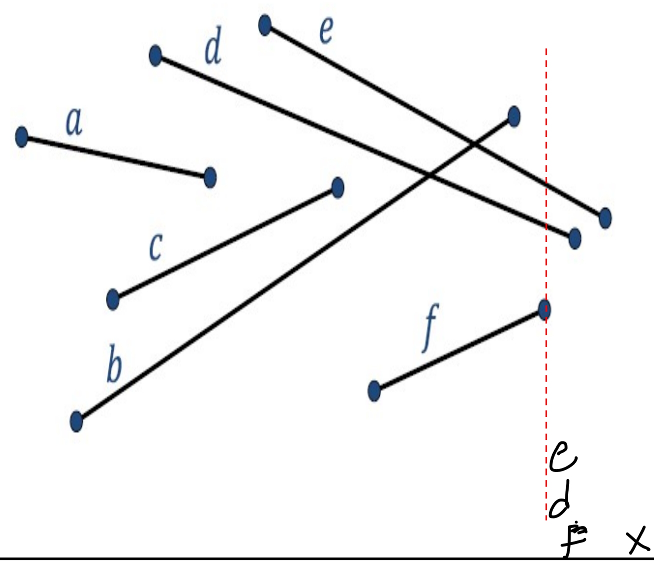
Determining whether any pair of segments intersects



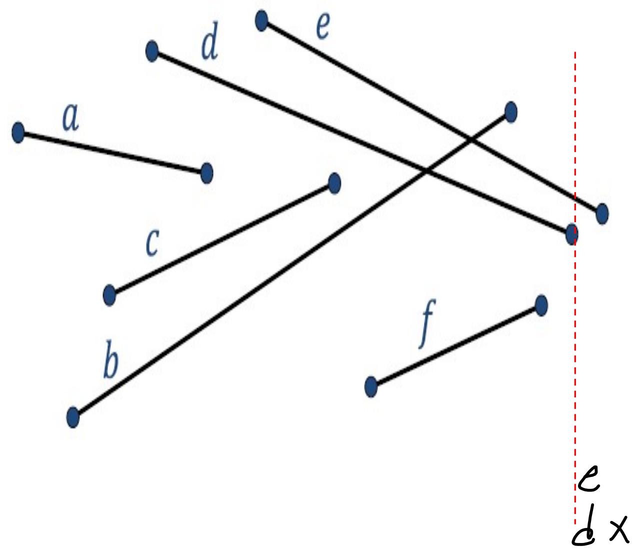
Determining whether any pair of segments intersects



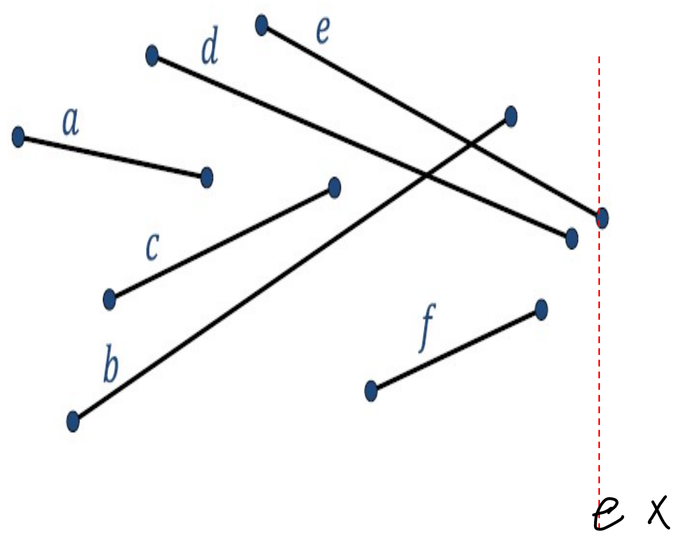
Determining whether any pair of segments intersects



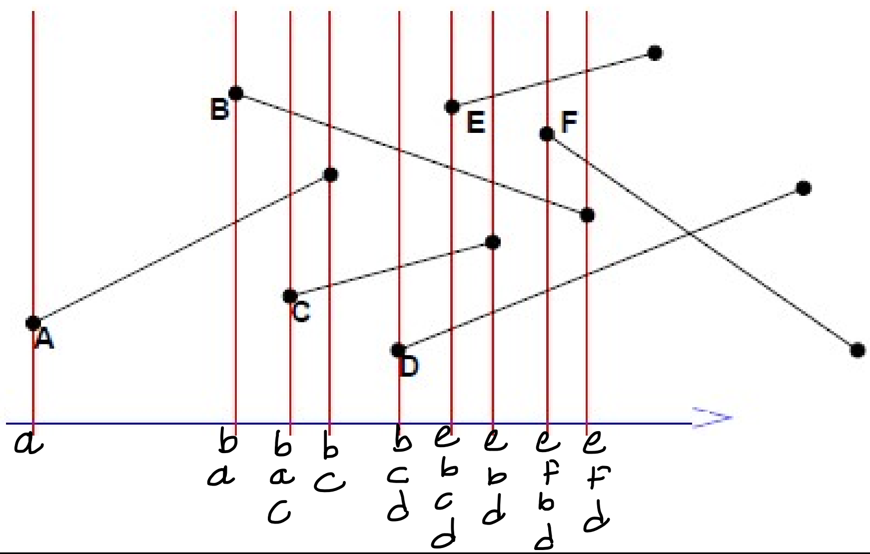
Determining whether any pair of segments intersects



Determining whether any pair of segments intersects



Example-2



Determining whether any pair of segments intersects

If there are *n* segments in set *S*, then ANY-SEGMENTS-INTERSECT runs in time *O(n* lg *n*).

Line 1 takes *O*(1) time.
Line 2 takes *O(n* lg *n*) time, using merge sort or heapsort.

Since there are 2*n* event points, the **for** loop of lines 3-11 iterates at most 2*n* times.

Each iteration takes *O(*lg *n*) time, since each red-black-tree operation takes *O*(lg *n*) time and, using INTERSECTION OF TWO LINE SEGMENTS METHOD, each intersection test takes *O*(1) time.

The total time is thus *O*(*n* lg *n*).