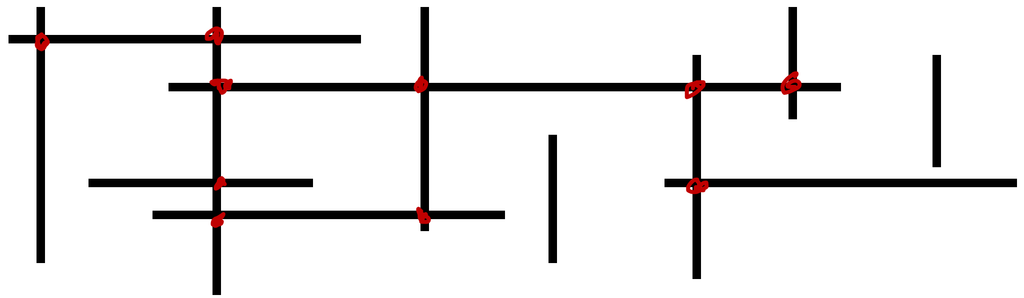
Line segment intersection

Find all pairs of intersecting line segments.



Naive approach $O(n^2)$

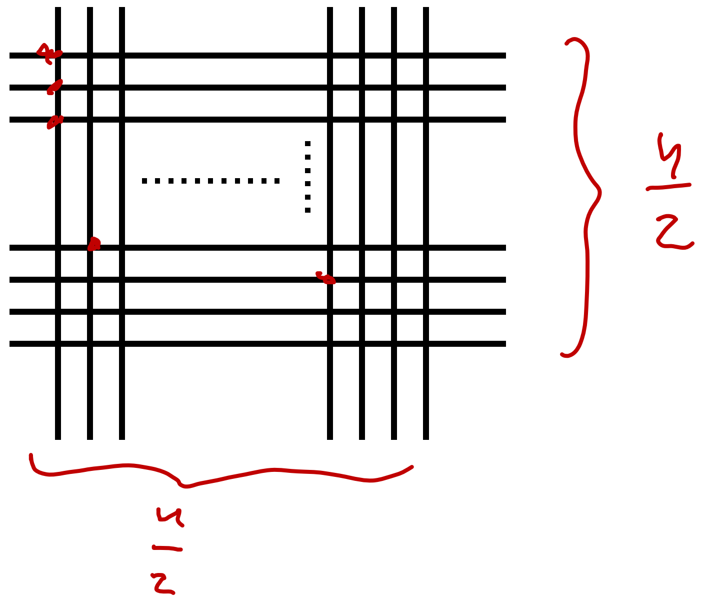
D&C

$O(n \cdot \log n)$? No!

Output sensitive

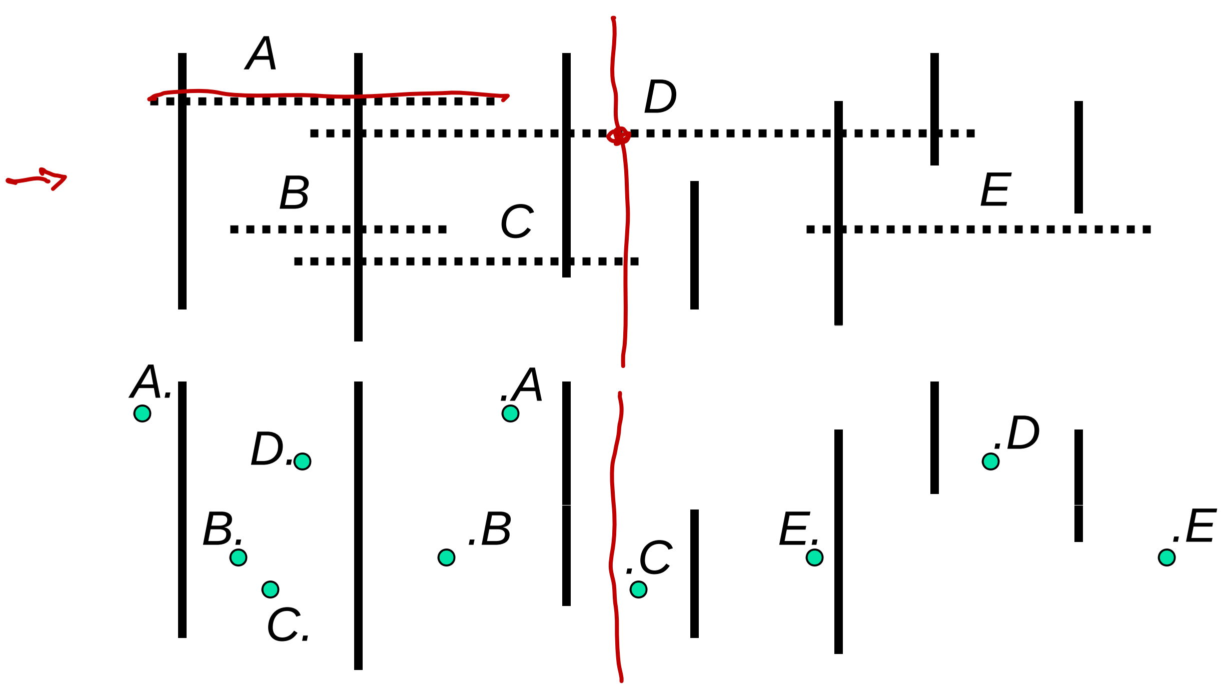
$O(n \log n + k)$

k: # intersections



Line segment intersection

Find all pairs of intersecting line segments.



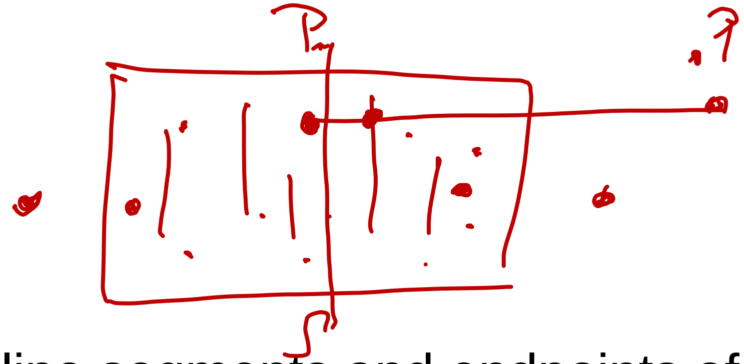
$A = ([x_1, x_2], y)$

$A. = (x_1, y)$

$.A = (x_2, y)$

The representation of the horizontal line segments by their endpoints allows for a vertical partitioning of all objects.

ReportCuts



Input: Set S of vertical line segments and endpoints of horizontal line segments.

Output: All intersections of vertical line segments with horizontal line segments, for which at least one endpoint is in S.

1. Divide

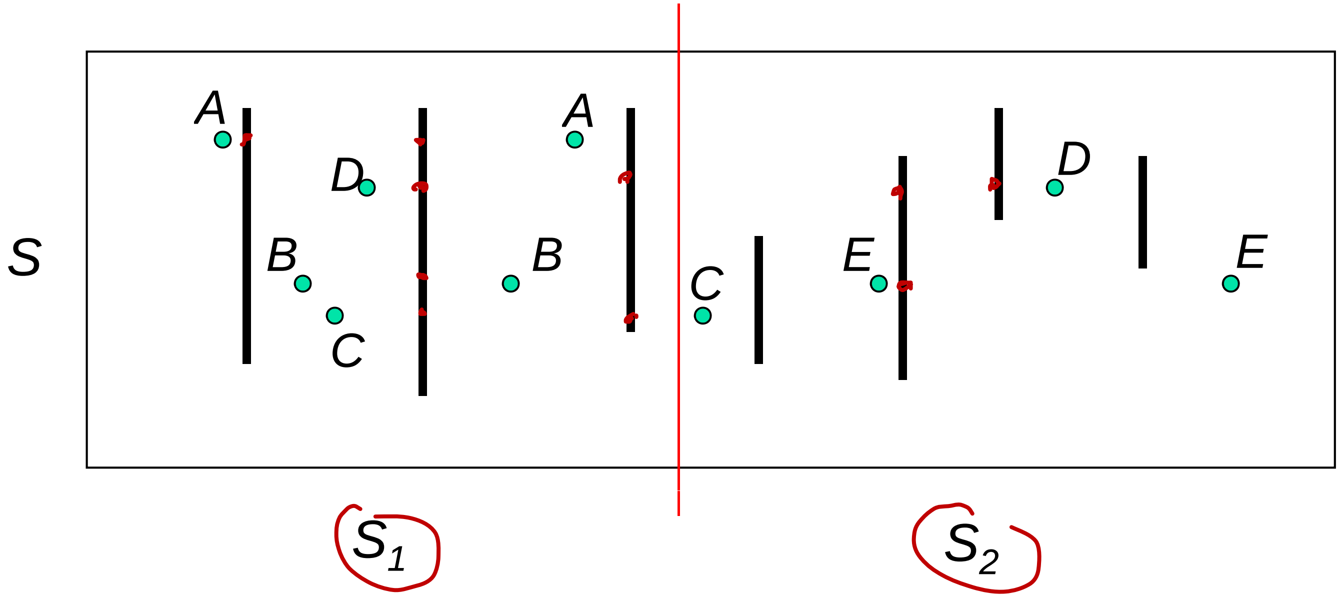
if $|S| > 1$

then using vertical bisection line L, divide S into equal size sets S_1 (to the left of L) and S_2 (to the right of L)

else S contains no intersections

ReportCuts

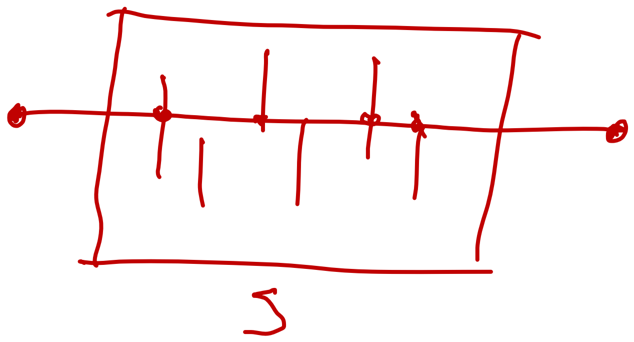
1. Divide:



2. Conquer:

ReportCuts(S_1); ReportCuts(S_2)

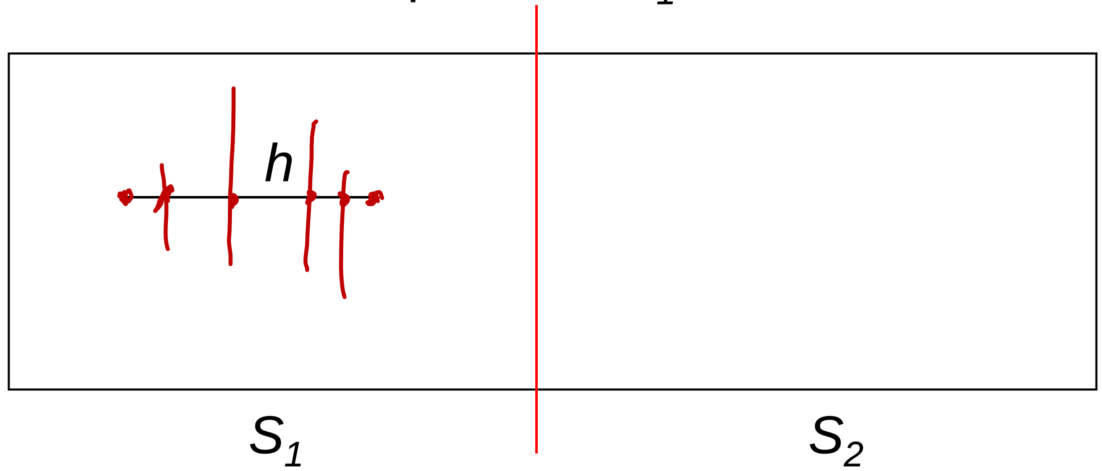
ReportCuts



3. Merge: ???

Possible intersections of a horizontal line-segment h in S_1

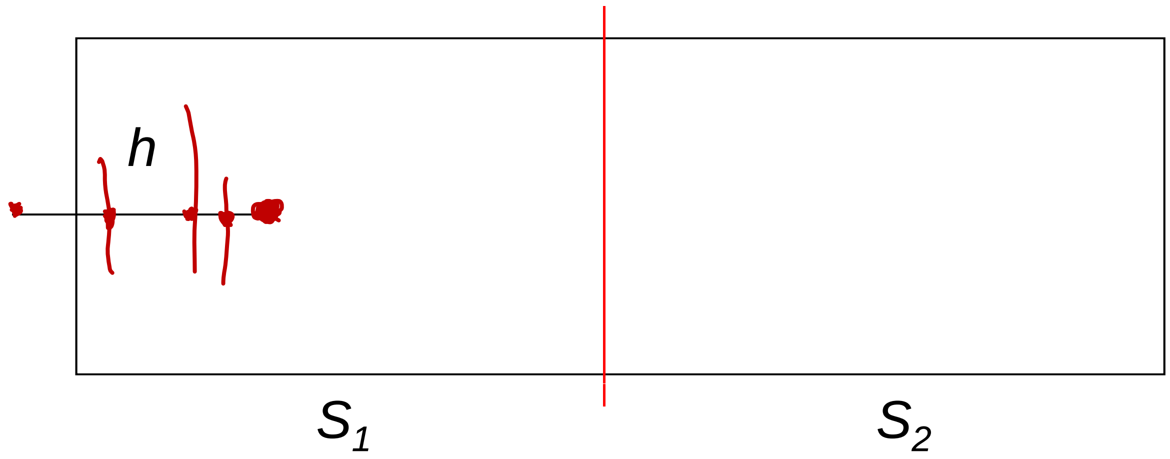
Case 1: both endpoints in S_1



ReportCuts

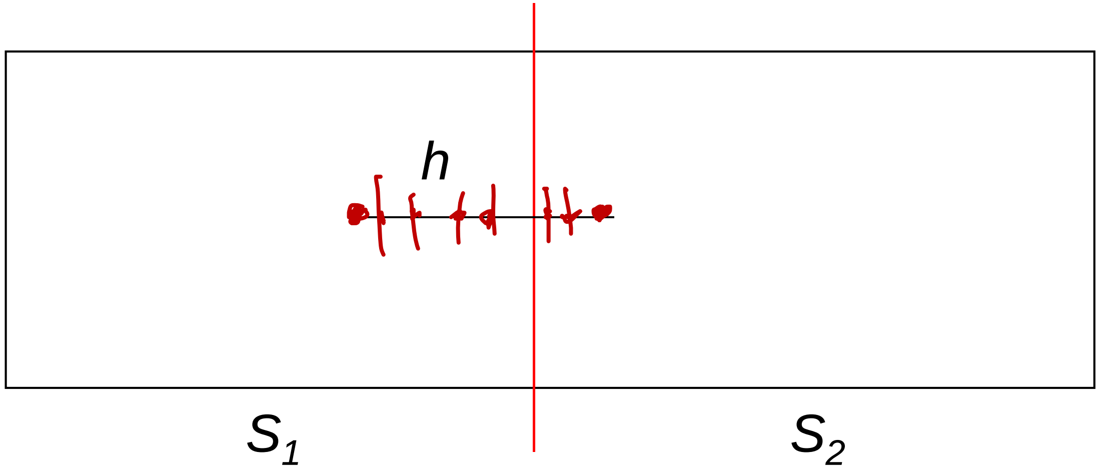
Case 2: only one endpoint of *h* in S_1

2 a) right endpoint in S_1

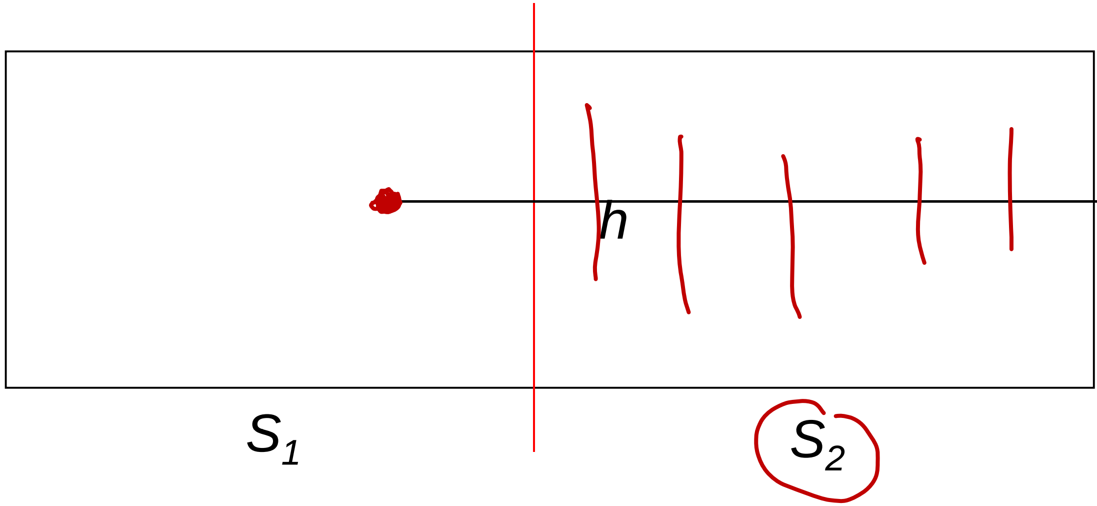


ReportCuts

2 b) left endpoint of h in S_1



right endpoint in S_2



right endpoint not in S_2

No evidence of h.l.s in S_2

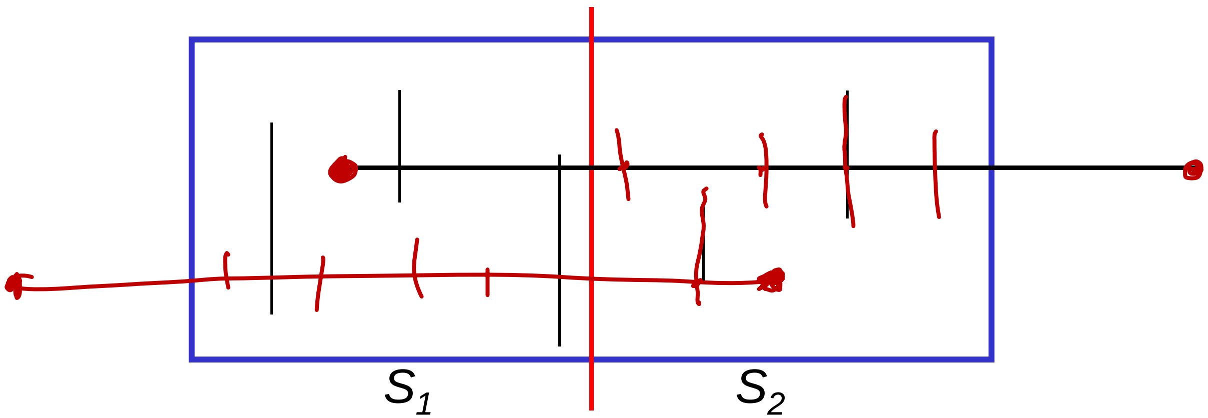
ReportCuts(S_2) does not compute intersections

Must be done by hand.

Procedure: ReportCuts(S)

3. Merge:

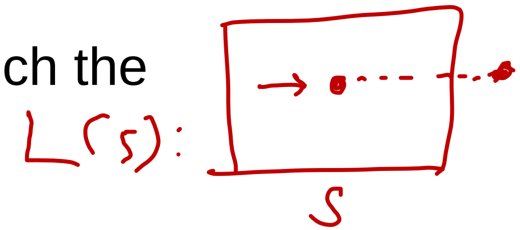
Return the intersections of vertical line segments in S_2 with horizontal line segments in S_1, for which the left endpoint is in S_1 and the right endpoint is neither in S_1 nor in S_2 .
Proceed analogously for S_1 .



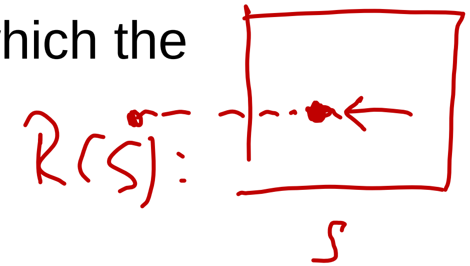
Implementation

Set S   Bottom up, linked lists

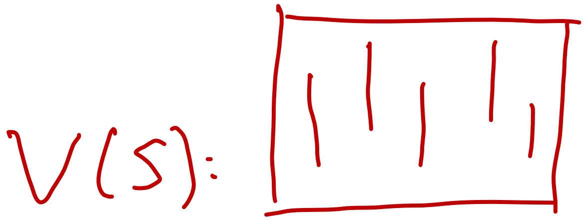
L(S): y-coordinates of all left endpoints in *S*, for which the corresponding right endpoint is not in *S.*



R(S): y-coordinates of all right endpoints in *S*, for which the corresponding left endpoint is not in *S.*



V(S): y-intervals of all vertical line-segments in *S.*



Base cases

S contains only one element s.

Case 1: $s = (x,y)$ is a left endpoint

$L(S) = \{y\}$ $R(S) = \emptyset$ $V(S) = \emptyset$

Case 2: $s = (x,y)$ is a right endpoint

$L(S) = \emptyset$ $R(S) = \{y\}$ $V(S) = \emptyset$

Case 3: $s = (x, y_1, y_2)$ is a vertical line-segment

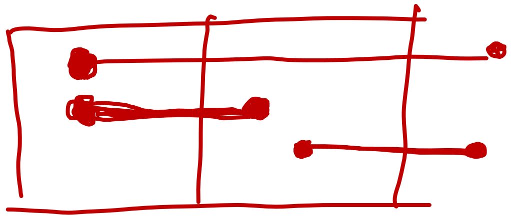
$L(S) = \emptyset$ $R(S) = \emptyset$ $V(S) = \{ [y_1, y_2] \}$

Merge step

Assume that $L(S_i)$, $R(S_i)$, $V(S_i)$ are known for $i = 1,2$.

$S = S_1 \cup S_2$

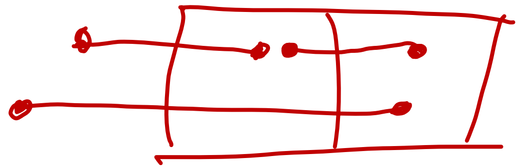
$L(S) = \left(L(S_1) \setminus R(S_2)\right) \cup L(S_2)$



$S_1 \cup S_2$

$R(S) = \left(R(S_2) \setminus L(S_1)\right) \cup R(S_1)$

$V(S) = V(S_1) \cup V(S_2)$



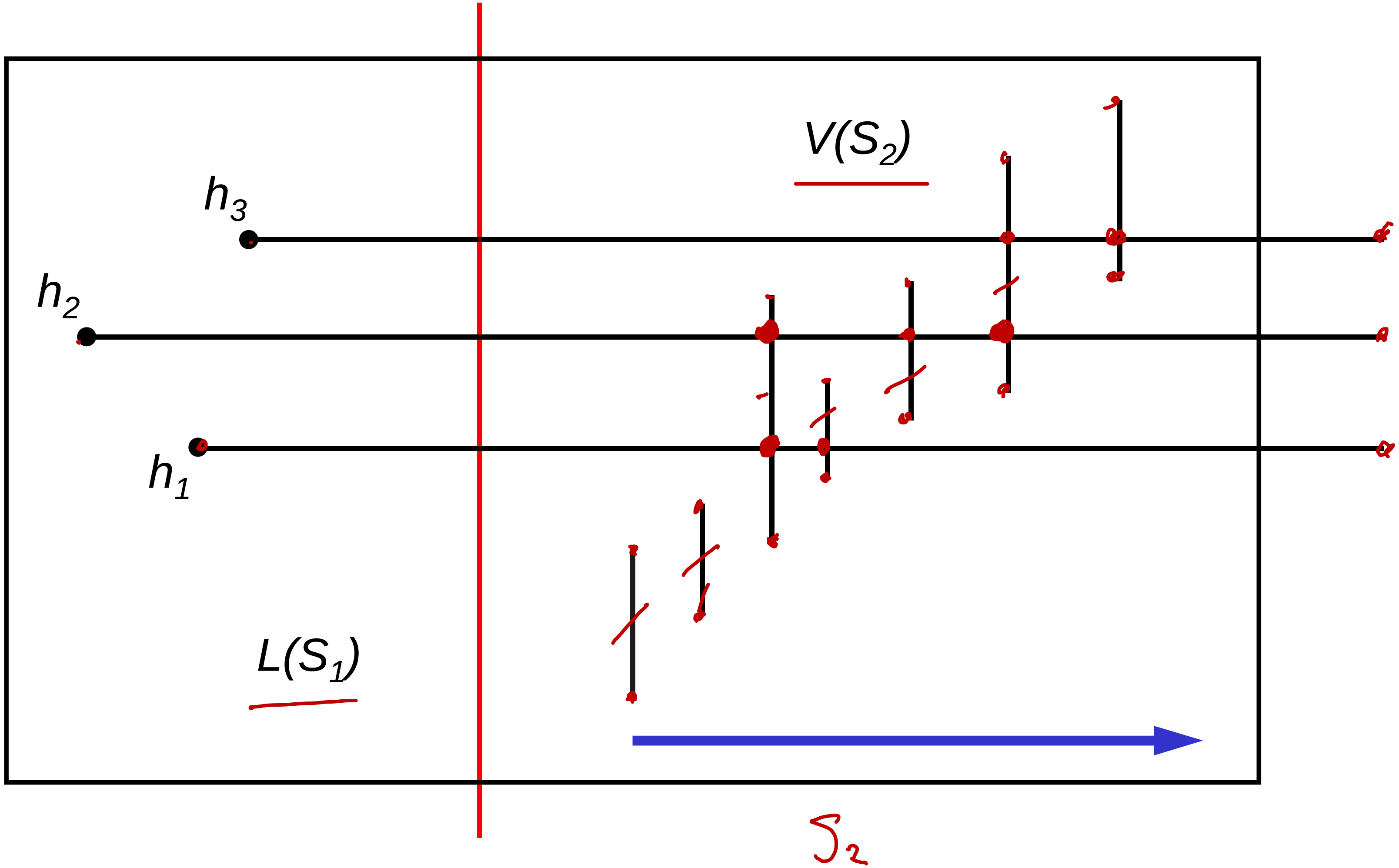
$S_1 \cup S_2$

L, R: ordered by increasing y-coordinates
linked lists — linear time

V: ordered by increasing lower endpoints
linked list — linear time

Output of the intersections

Running time: $|V(S_2)| + \frac{\#\text{ intersections in } S_1}{\text{with h.l.S. in } L(S_1)}$



Running time

Initially, the input (vertical line segments, left/right endpoints of horizontal line segments) has to be sorted and stored in an array.

Divide-and-conquer:

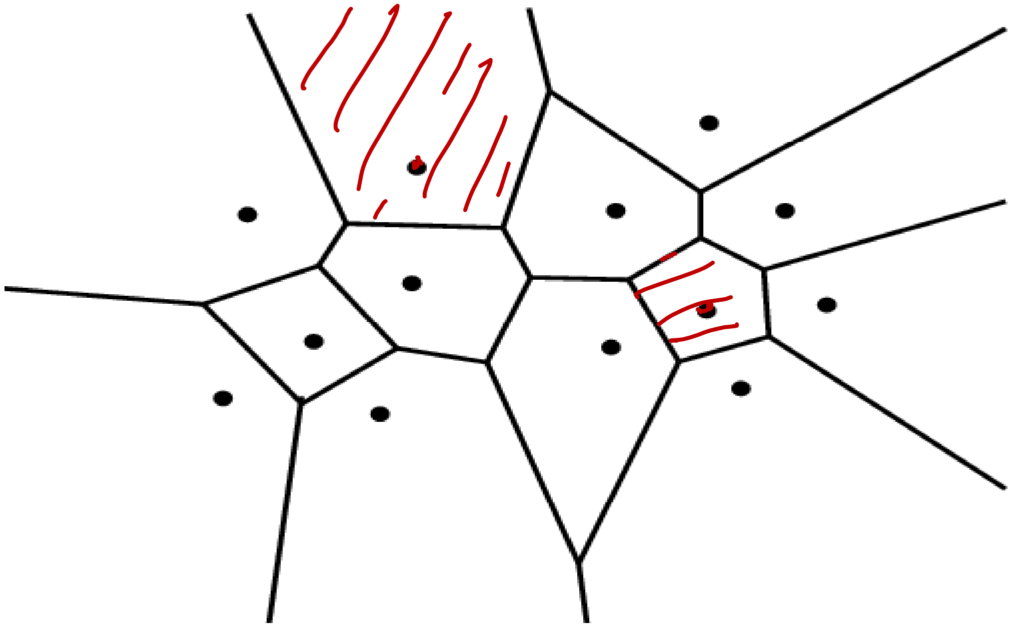
$$\boxed{T(n) = 2T(n/2) + an} + \text{size of output}$$
$$T(1) = O(1)$$

$O(n \log n + k)$ $\quad$ k = # intersections

Computation of a Voronoi diagram

Input: Set of sites.

Output: Partition of the plane into regions, each consisting of the points closer to one particular site than to any other site.



Definition of Voronoi diagrams

P : Set of sites

H(*p* | *p'*) = {*x* | *x* is closer to *p* than to *p'* }

Voronoi region of *p:*

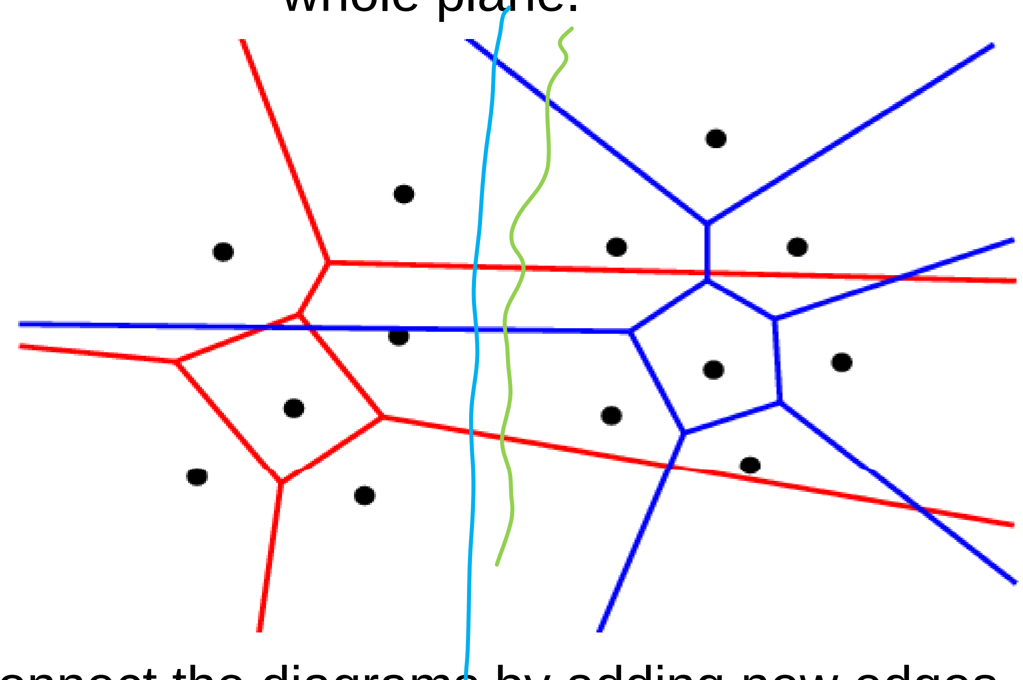
$$VR(p) = \bigcap_{p' \in P \setminus \{p\}} H(p \mid p')$$

Computation of a Voronoi Diagram

Divide: Partition the set of sites into two equal sized sets.

Conquer: Recursive computation of the two smaller Voronoi diagrams.

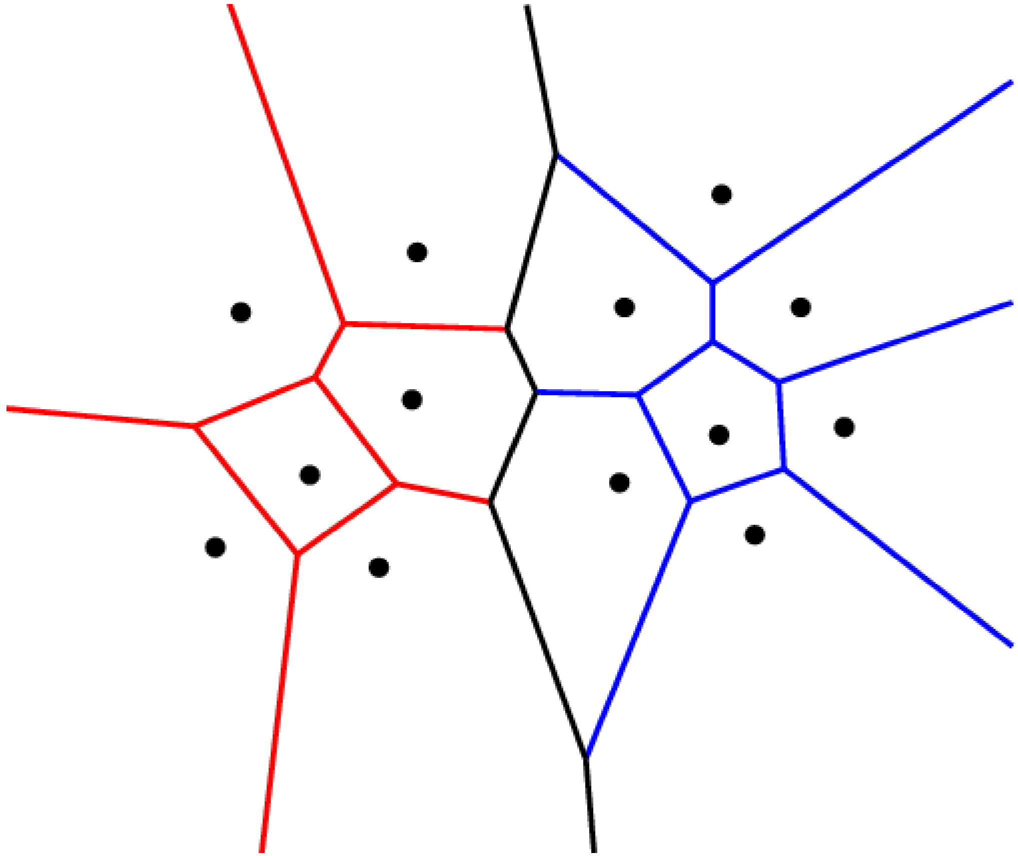
Stopping condition: The Voronoi diagram of a single site is the whole plane.



Merge: Connect the diagrams by adding new edges.

Computation of a Voronoi diagram

Output: The complete Voronoi diagram.



Running time: $O(n \log n)$, where n is the number of sites.