Algorithm Theory

01 - Introduction

Dr. Alexander Souza

Winter term 11/12

Organization

Lectures:	Tue	14-16	101-00-026
	Wed	16-17	101-00-026
Exercises:	Thu	8-10	101-01-018
	Thu	8-10	051-00-034
	Fri	12-14	101-00-018
	Fri	12-14	078-00-014

3 out of these 4 groups will take place biweekly
Registration during this class, teamwork up to 3 students

Sheet 1 will be out on Wed.,26.10.

Hand-in biweekly during Wed.-class

First hand-in Wed.,2.11., first tutorials Thu.,10.11./Fri.,11.11.

Web page: Contains slides, recording, schedule, sheets, grouping etc.

http://lak.informatik.uni-freiburg.de/
→ Teaching → Winter Term 2011/12 → Algorithm Theory

Organization

Final exam: Date and Time: t.b.a.

Admission

1 exercise presented during the tutorials

50% of total exercise points

More Details: Kursvorlesung, 3+1 SWS

6 ECTS Credits

Lectures in English

Tutorials supervised by Thomas Janson

English: Mahdi

German: Geißer, Jarecki

Camtasia recording available

Literature

Th. Ottmann, P. Widmayer:
Algorithmen und Datenstrukturen
4th Edition, Spektrum Akademischer Verlag,
Heidelberg, 2002

Th. Cormen, C. Leiserson, R. Rivest, C. Stein:
Introduction to Algorithms, Second Edition
MIT Press, 2001

Original literature

Algorithms and data structures

Design and analysis techniques for algorithms

- Divide and conquer
- Greedy approaches
- Dynamic programming
- Randomization
- Amortized analysis

Algorithms and data structures

Problems and application areas

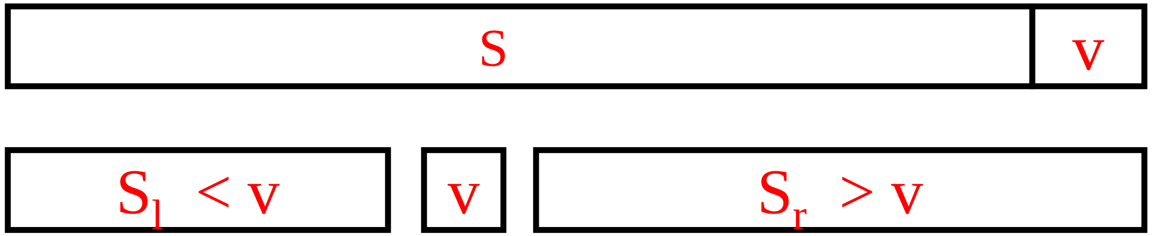
- Geometric algorithms
- Algebraic algorithms
- Graph algorithms
- Data structures
- Internet algorithms
- Optimization methods
- Algorithms on strings

Divide and Conquer

The divide-and-conquer paradigm

- Quicksort
- Formulation and analysis of the paradigm
- Geometric divide-and-conquer
  - Closest pair
  - Line segment intersection
  - Voronoi diagrams

Quicksort: Sorting by partitioning



function Quick (S: sequence): sequence;

{returns the sorted sequence S}

begin

  if #S <= 1 then Quick:=S

  else { choose pivot element v in S;

    partition S into S_l with elements < v,

    and S_r with elements > v

    Quick:= $\boxed{\text{Quick}(S_l) \mid v \mid \text{Quick}(S_r)}$ }

end;

Formulation of the D&C paradigm

Divide-and-conquer method for solving a problem instance of size n:

1. Divide

$n \leq c$: Solve the problem directly.

$n > c$: Divide the problem into k subproblems of sizes $n_1,\dots,n_k < n$ ($k \geq 2$).

2. Conquer

Solve the k subproblems in the same way (recursively).

3. Merge

Combine the partial solutions to generate a solution for the original instance.

Analysis

T(n) : **maximum number of steps necessary for solving an instance of size n**

$$T(n) = \begin{cases} a & n \leq c \\ T(n_1) + \ldots + T(n_k) & n > c \\ + \text{ cost for divide and merge} & \end{cases}$$

Special case: $k = 2, n_1 = n_2 = n/2$

cost for divide and merge: $DM(n)$

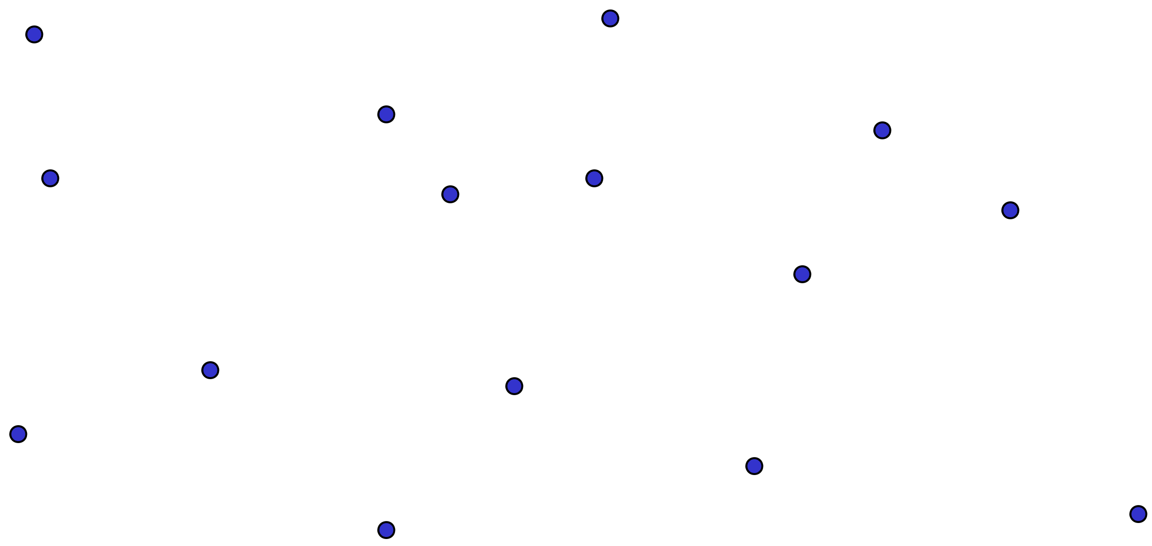
$T(1) = a$

$T(n) = 2T(n/2) + DM(n)$

Geometric divide-and-conquer

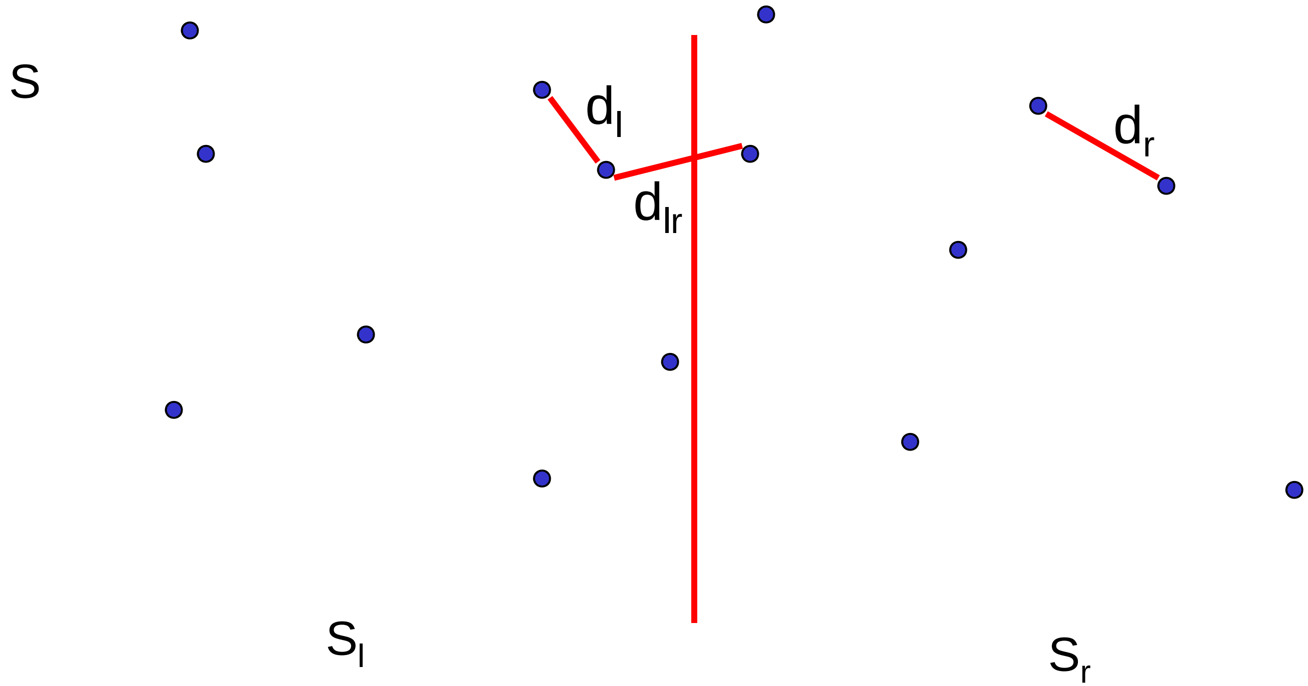
Closest Pair Problem:

Given a set *S* of *n* points, find a pair of points with the smallest distance.



Divide-and-conquer method

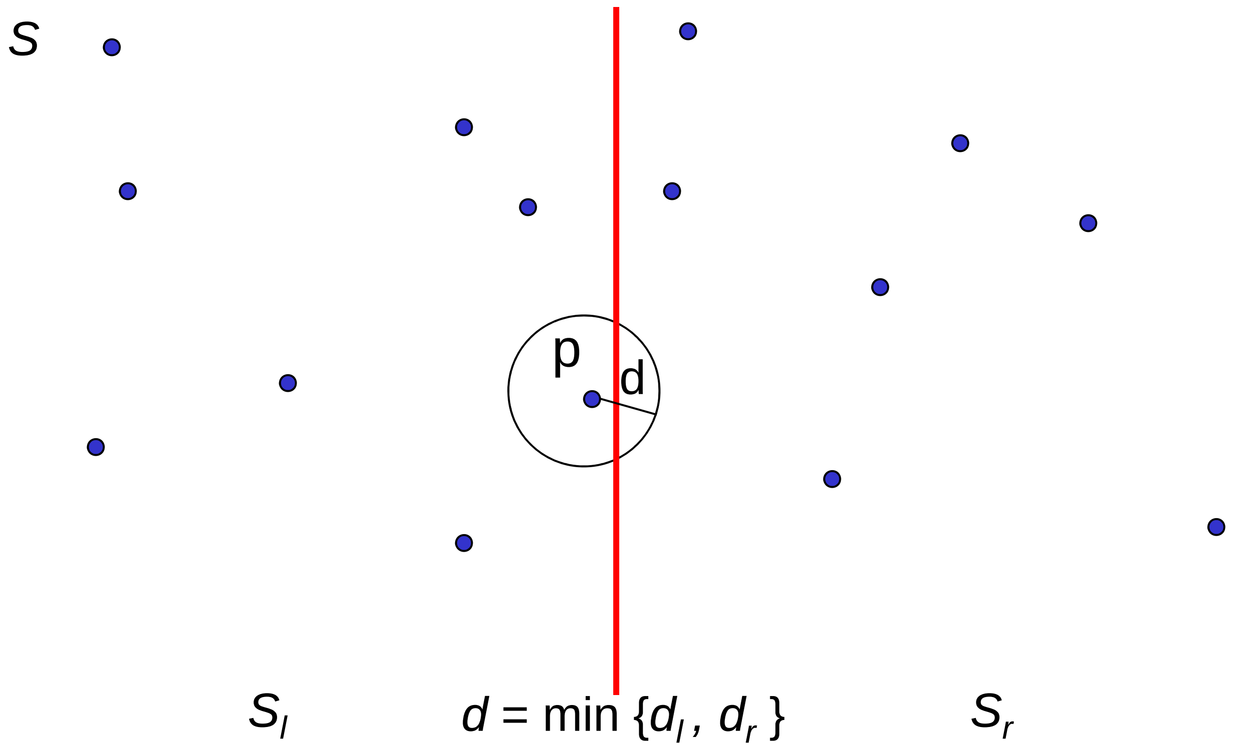
1. **Divide:** Divide S into two equal sized sets S_l und S_r .
2. **Conquer:** $d_l = mindist(S_l)$ $d_r = mindist(S_r)$
3. **Merge:** $d_{lr} = \min\{ d(p_l, p_r) \mid p_l \in S_l, p_r \in S_r \}$
   return $\min\{d_l, d_r, d_{lr}\}$



Divide-and-conquer method

1. **Divide:** Divide S into two equal sets S_l und S_r .
2. **Conquer:** $d_l = mindist(S_l)$ $d_r = mindist(S_r)$
3. **Merge:** $d_{lr} = \min\{ d(p_l, p_r) \mid p_l \in S_l, p_r \in S_r \}$
   return $\min\{d_l, d_r, d_{lr}\}$

Computation of d_{lr} :

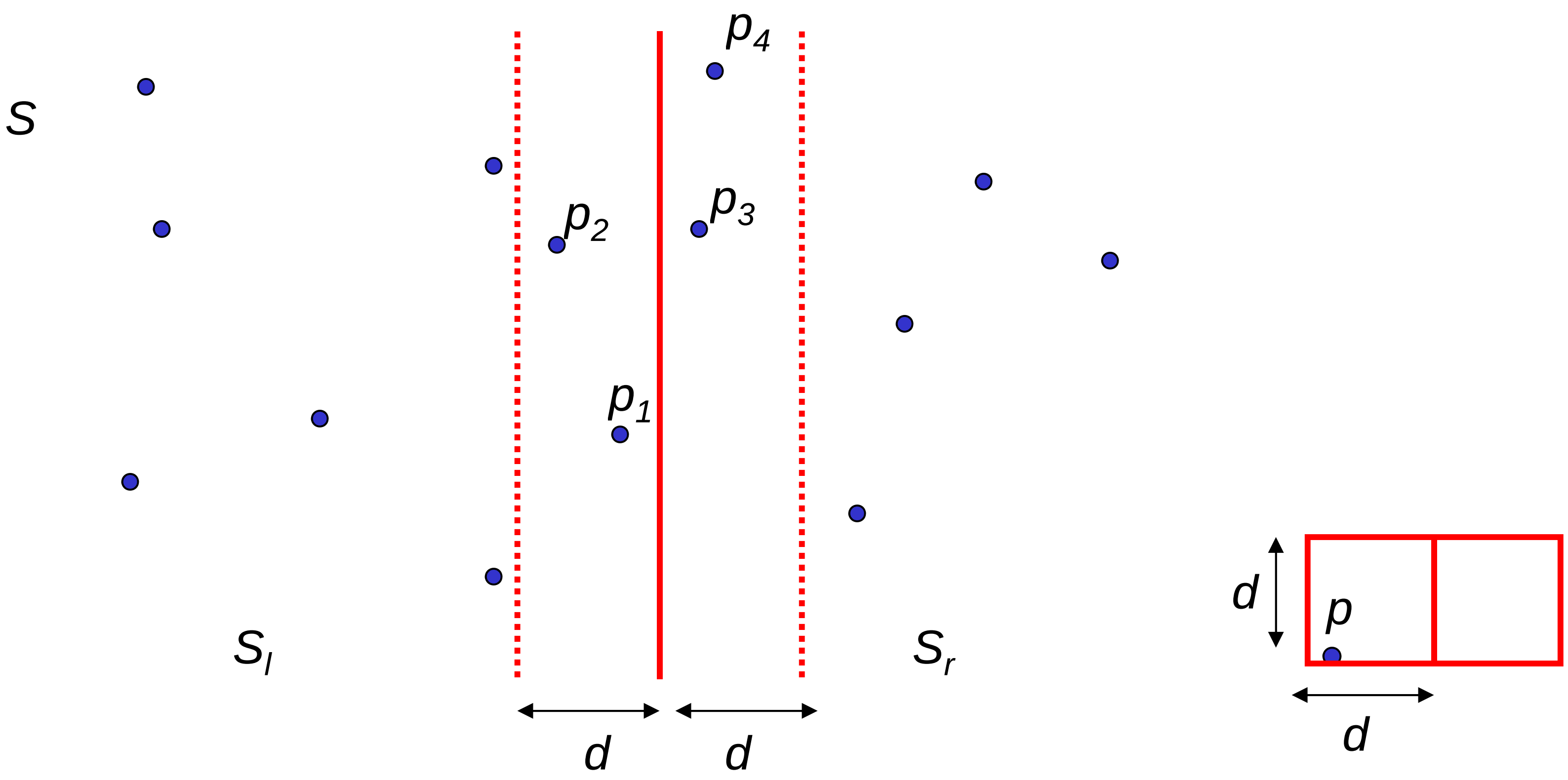


Merge step

1. Consider only points within distance *d* of the bisection line, in the order of increasing y-coordinates.

2. For each point *p* consider all points *q* within y-distance at most *d*; there are at most 7 such points.

Merge step



$d = \min \{ d_l, d_r \}$

Implementation

- Initially sort the points in S in order of increasing x-coordinates *O(n log n).*

- Once the subproblems S_l , S_r are solved, generate a list of the points in S in order of increasing y-coordinates (merge sort).

Running time (divide-and-conquer)

$$T(n) = \begin{cases} 2T(n/2) + an & n > 3 \\ a & n \leq 3 \end{cases}$$

- Guess the solution by repeated substitution.
- Verify by induction.

Solution: *O(n log n)*

Guess by repeated substitution

$$T(n) = \begin{cases} 2T(n/2) + an & n > 3 \\ a & n \leq 3 \end{cases}$$

$$T(n) =$$

Verify by induction

$$T(n) \le an \log n$$

$$T(n) = \begin{cases} 2T(n/2) + an & n > 3 \\ a & n \le 3 \end{cases}$$

$n = 2^i$

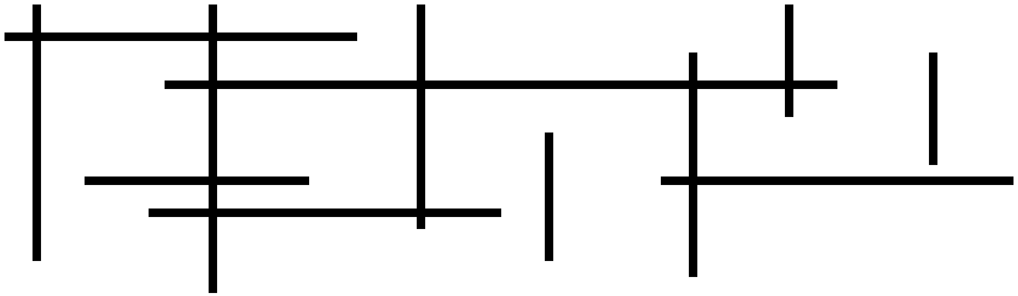
$i = 1$: ok

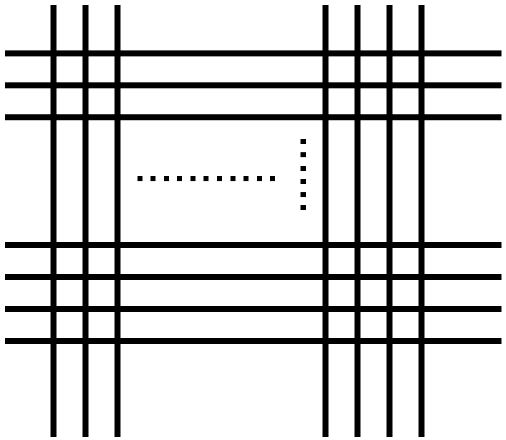
$i > 1$

$T(2^i) \;=$

Line segment intersection

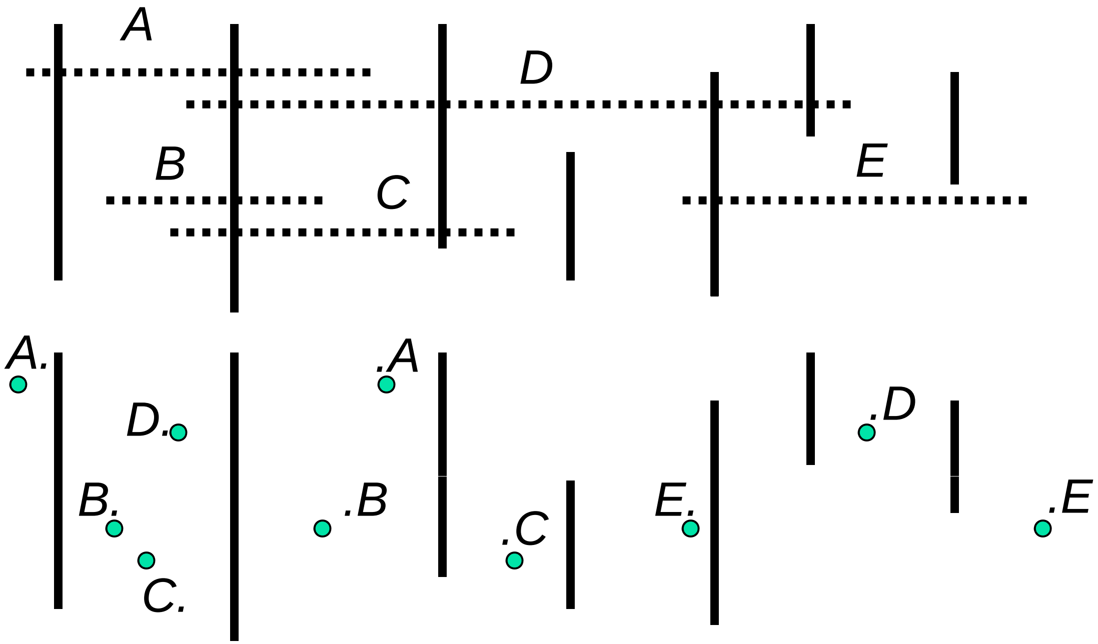
Find all pairs of intersecting line segments.





Line segment intersection

Find all pairs of intersecting line segments.



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

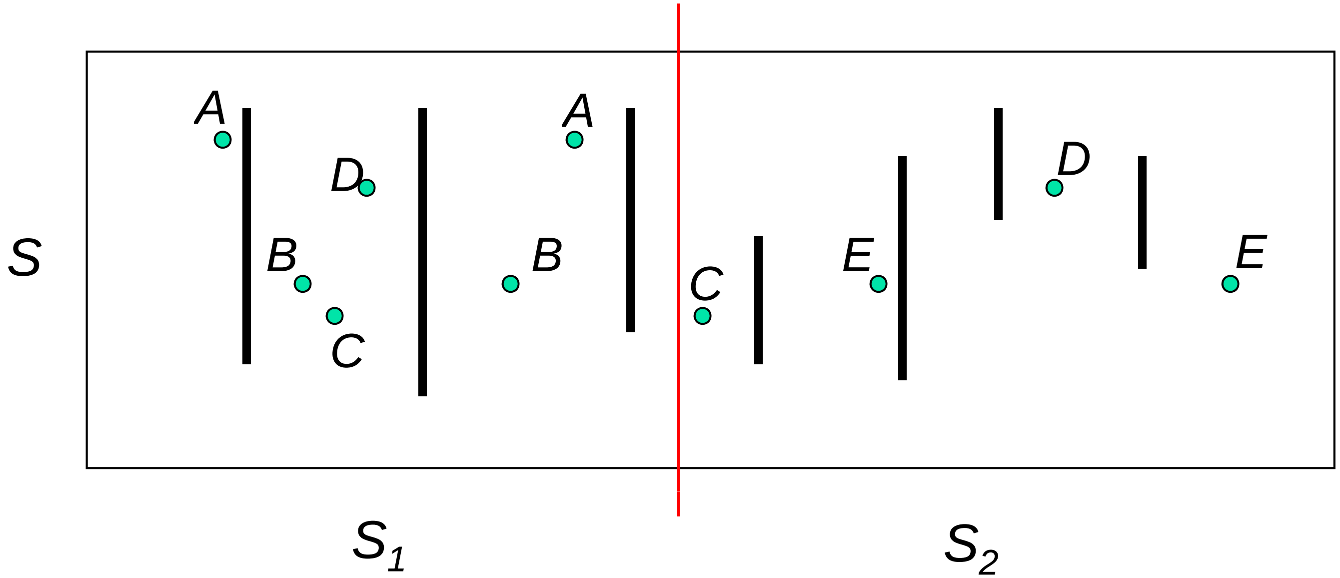
if $|S| > 1$

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:

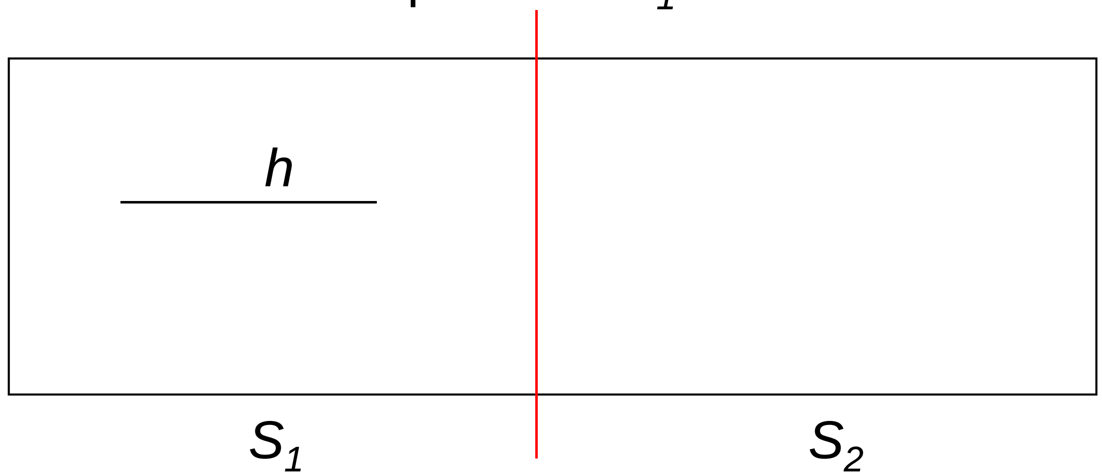
ReportCuts(S_1); ReportCuts(S_2)

ReportCuts

3. Merge: ???

Possible intersections of a horizontal line-segment h in S_1

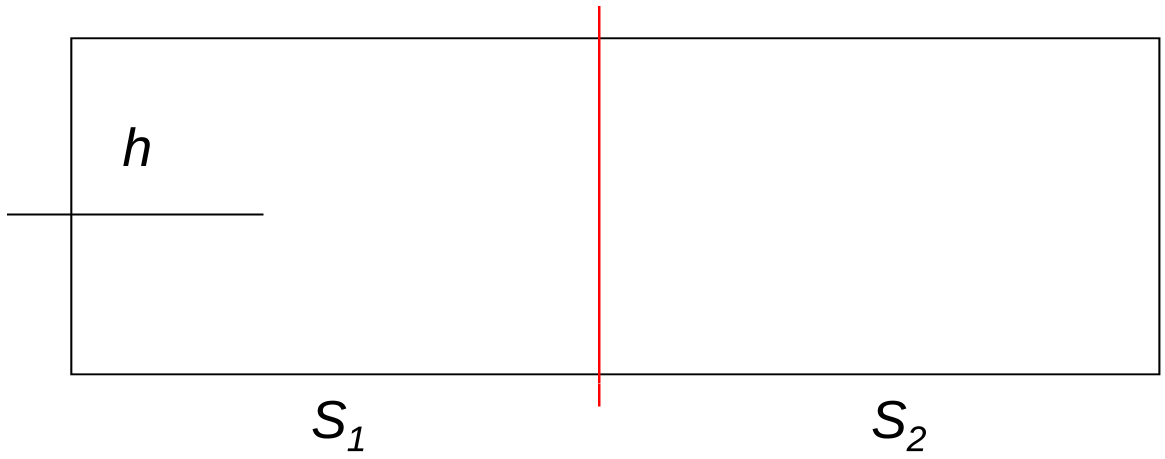
Case 1: both endpoints in S_1



ReportCuts

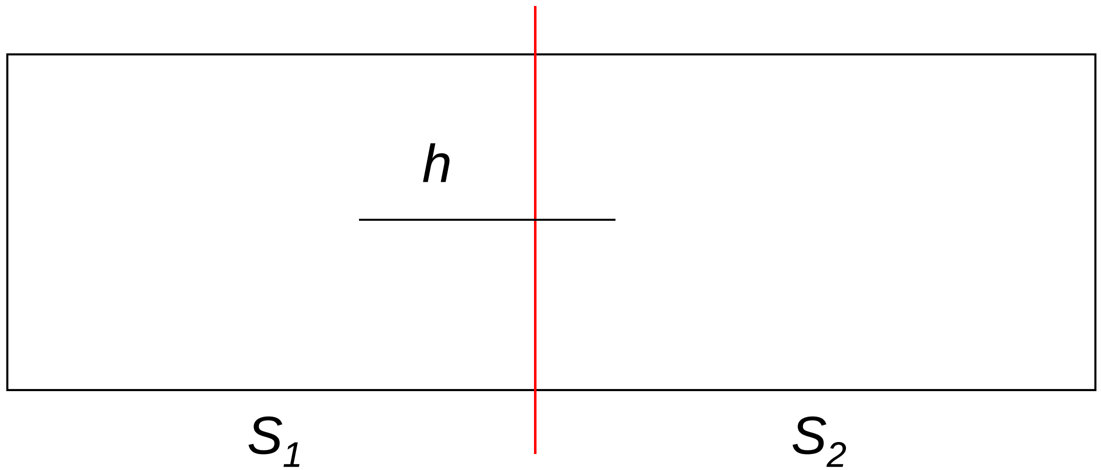
Case 2: only one endpoint of h in S_1

2 a) right endpoint in S_1

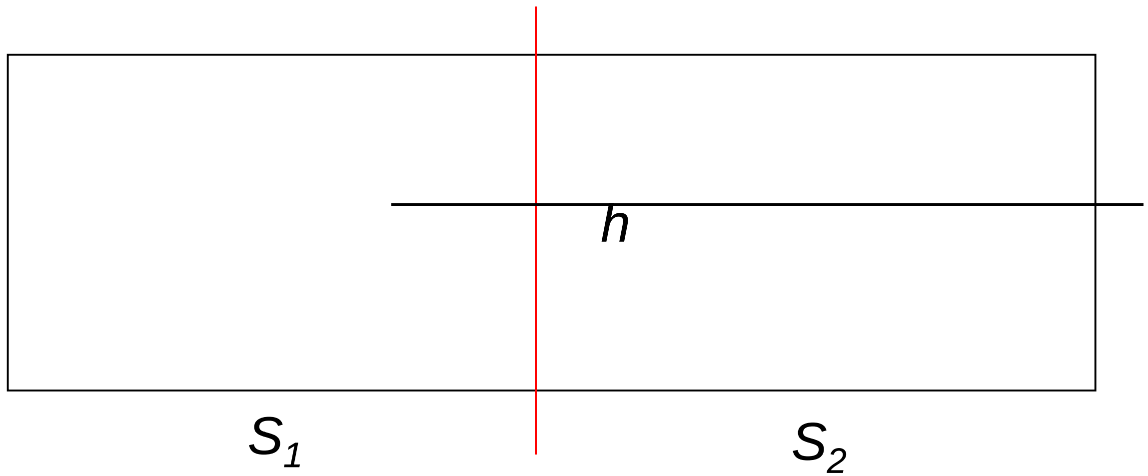


ReportCuts

2 b) left endpoint of h in S_1



right endpoint in S_2

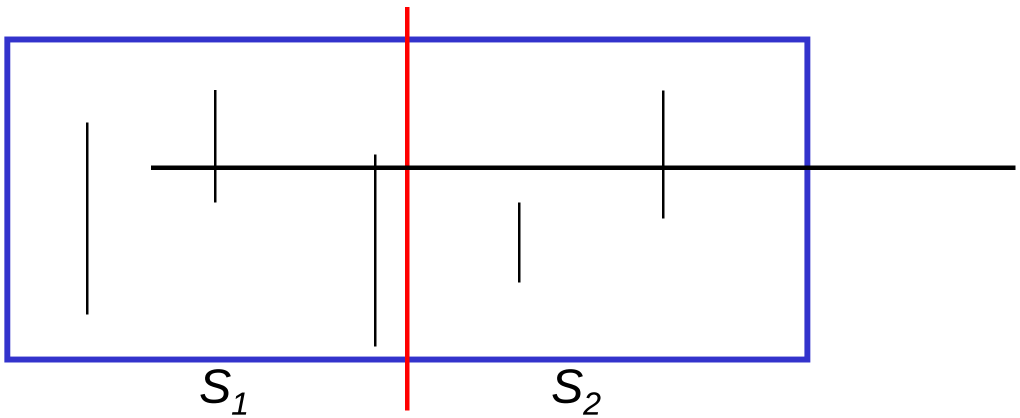


right endpoint not in S_2

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

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

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

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

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

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

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

Merge step

Assume that $L(S_i)$, $R(S_i)$, $V(S_i)$ are known for $i = 1,2$.
$S = S_1 \cup S_2$

$L(S) =$

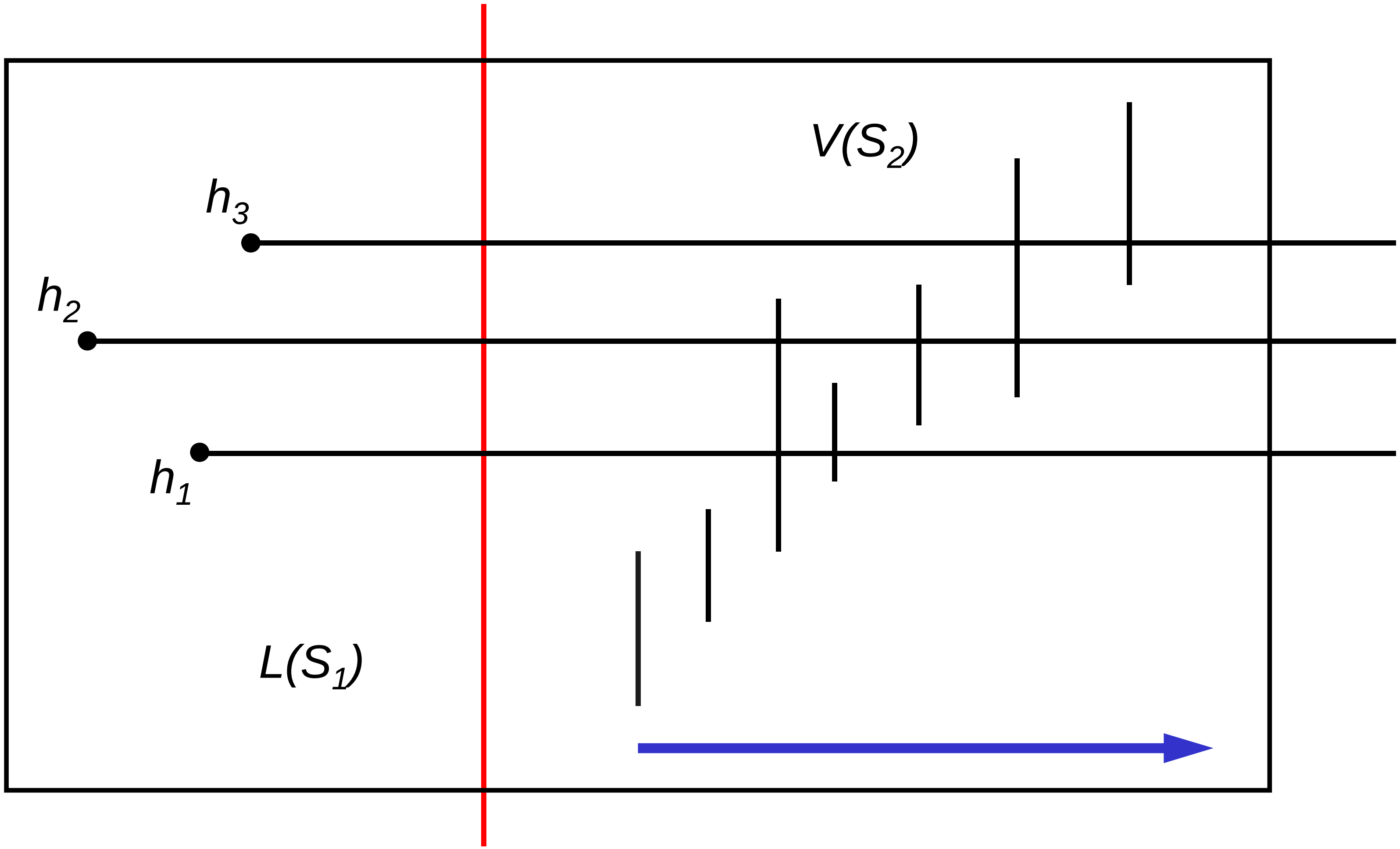
$R(S) =$

$V(S) =$

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

V: ordered by increasing lower endpoints
linked list

Output of the intersections



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:

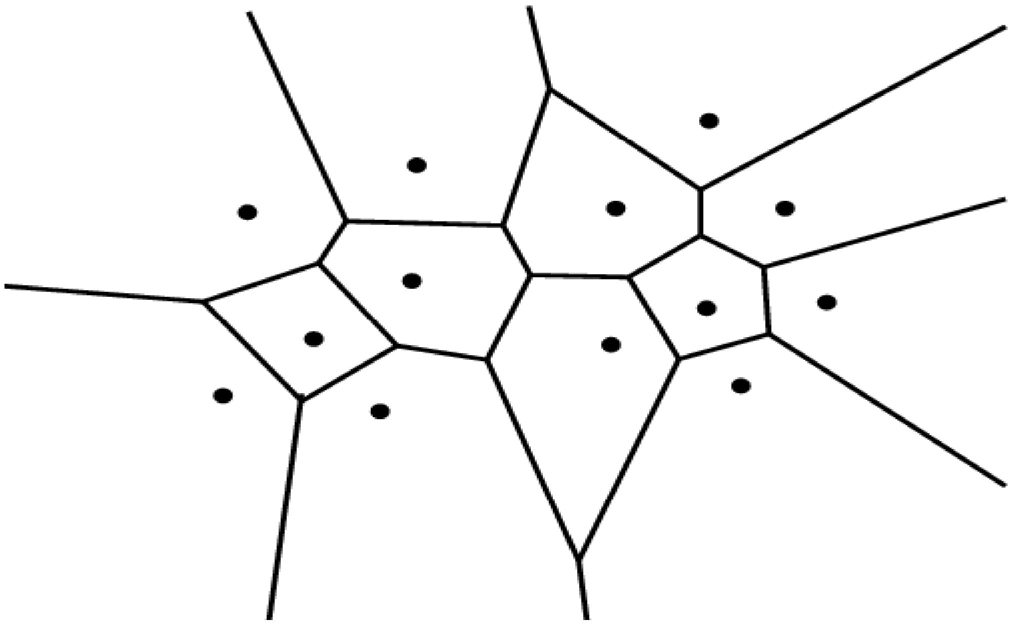
$T(n) = 2T(n/2) + an$ + size of output
$T(1) = O(1)$

$O(n \log n + k)$ 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

$H(p \mid p') = \{x \mid x$ is closer to p than to $p'\}$

Voronoi region of *p:*

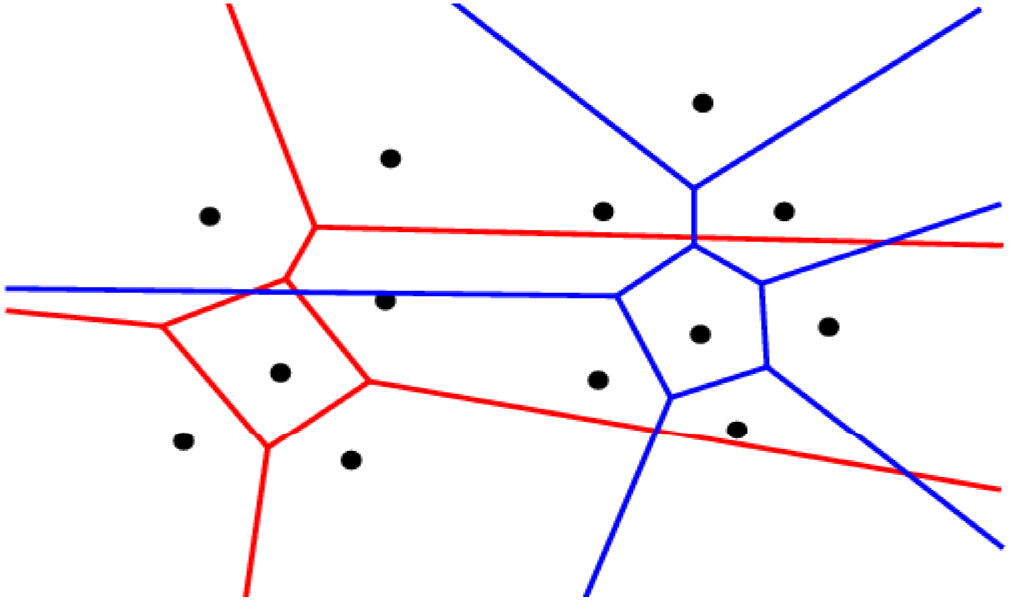
$$VR(p) = \bigcap_{p' \in P \setminus \{p\}} H(p \mid p')$$

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.

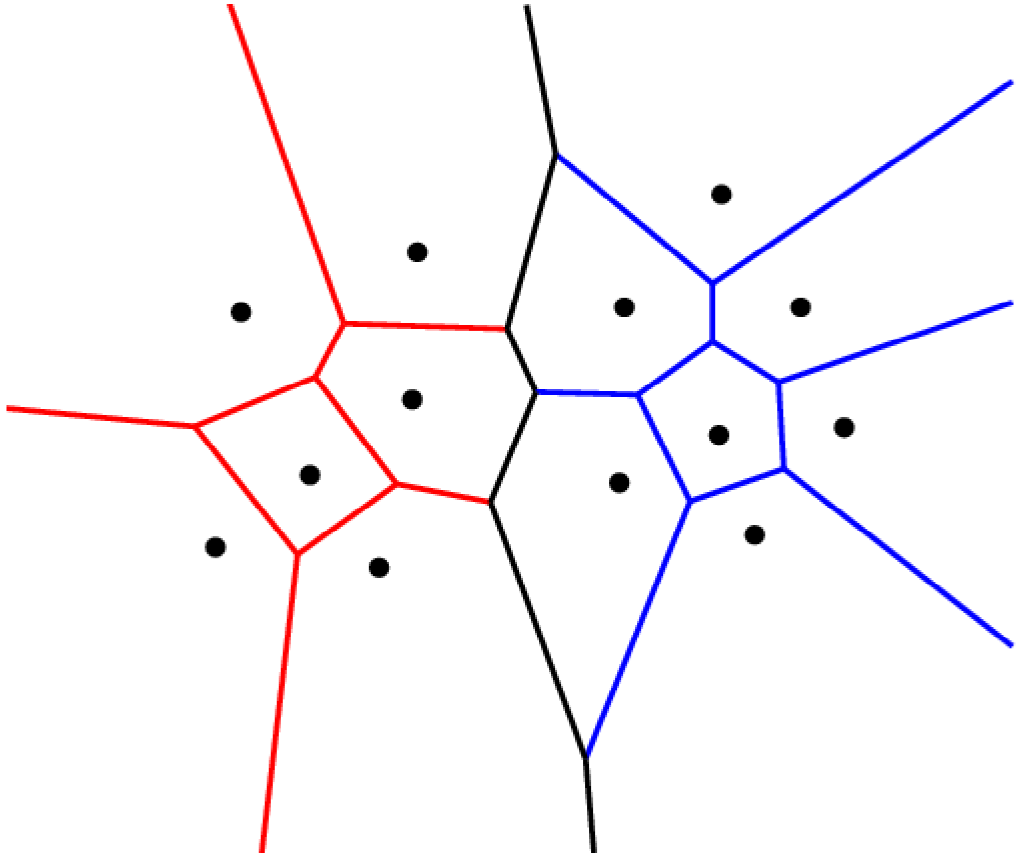
Stopping condition: The Voronoi diagram of a single site is the whole plane.



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.