Karlsruhe Institute of Technology

Computational Geometry · Lecture

Introduction & Convex Hulls

INSTITUT FÜR THEORETISCHE INFORMATIK · FAKULTÄT FÜR INFORMATIK

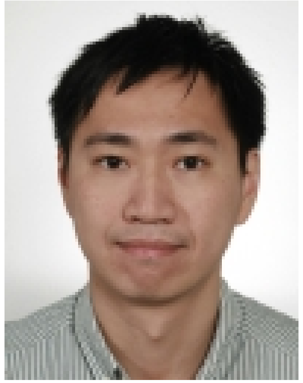
Chih-Hung Liu · **Tamara Mchedlidze**

18.4.2018

AlgoGeom-Team

Lecturers

- Tamara Mchedlidze
- mched@iti.uka.de
- Room 307
- Office hours: by appointment

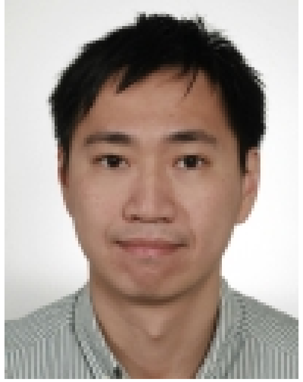


- Chih-Hung Liu
- chih-hung.liu@inf.ethz.ch
- ETH Zürich
- Office hours: by appointment

AlgoGeom-Team

Lecturers

- Tamara Mchedlidze
- mched@iti.uka.de
- Room 307
- Office hours: by appointment



- Chih-Hung Liu
- chih-hung.liu@inf.ethz.ch
- ETH Zürich
- Office hours: by appointment

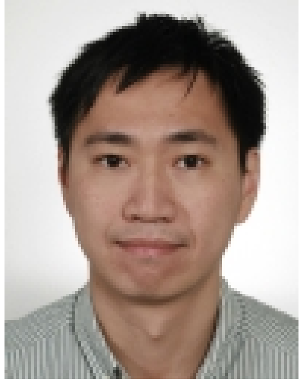
Exercise Leader

- Guido Brückner
- brueckner@kit.edu
- Room 317
- Office hours: by appointment

AlgoGeom-Team

Lecturers

- Tamara Mchedlidze
- `mched@iti.uka.de`
- Room 307
- Office hours: by appointment



- Chih-Hung Liu
- `chih-hung.liu@inf.ethz.ch`
- ETH Zürich
- Office hours: by appointment

Exercise Leader

- Guido Brückner
- `brueckner@kit.edu`
- Room 317
- Office hours: by appointment

Schedule

- Lecture: Wed. 14:00 – 15:30 SR 301
- Exercises: Mon. 15:45 – 17:15, SR 236 (starting ?)

Organization

Website

`http://i11www.iti.kit.edu/teaching/sommer2018/compgeom/`

- Course Information
- Lecture Slides
- Exercises
- Additional Material

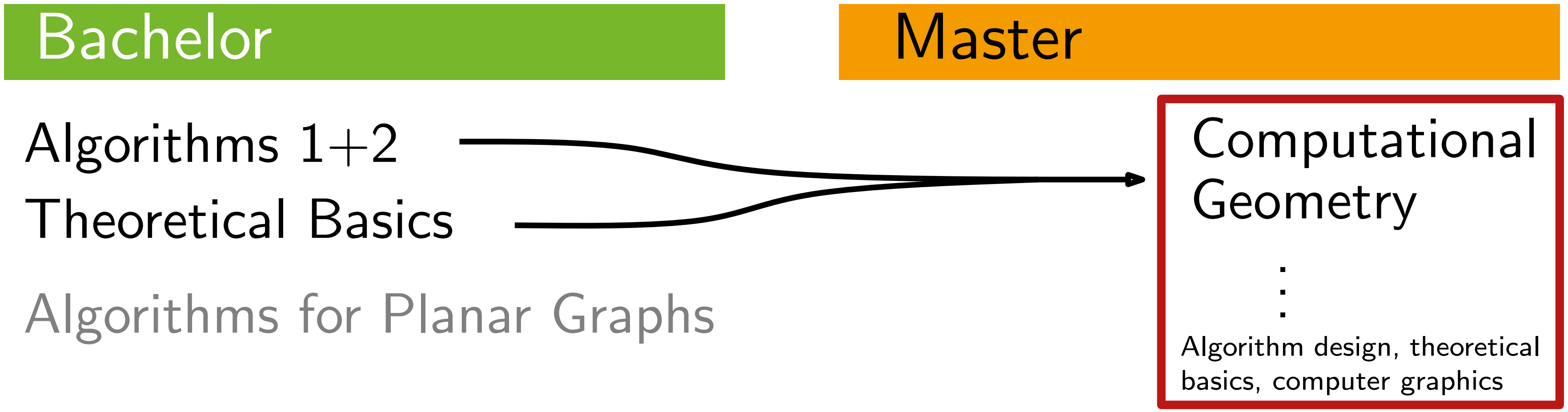
Organization

Website

`http://i11www.iti.kit.edu/teaching/sommer2018/compgeom/`

- Course Information
- Lecture Slides
- Exercises
- Additional Material

Computational Geometry in Computer Science Master's Studies



Bachelor

Master

Algorithms 1+2

Theoretical Basics

Algorithms for Planar Graphs

Computational Geometry

⋮

Algorithm design, theoretical basics, computer graphics

Prior Knowledge: Algorithms and Elementary Geometry

Organization

Exercises

- Every second Monday starting 27.04
- Exercise problems posted at least one week before an exercise session.
- Reinforce lecture material, help prepare for exam.

What will the exercises involve?

- Independent or group preparation
- Weekly meetings in class
- Active participation in class is expected - we expect that you present at least one solution on the board
- Can hand in exercises for feedback

Computational Geometry

Objectives: At the end of the course you will be able to...

- explain concepts, structures, and problem definitions
- understand the discussed algorithms, and explain and analyze them
- select and adapt appropriate algorithms and data structures
- analyze new geometric problems and develop efficient solutions

Computational Geometry

Objectives: At the end of the course you will be able to...

- explain concepts, structures, and problem definitions
- understand the discussed algorithms, and explain and analyze them
- select and adapt appropriate algorithms and data structures
- analyze new geometric problems and develop efficient solutions

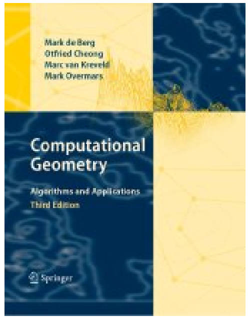
Computational Geometry

Objectives: At the end of the course you will be able to...

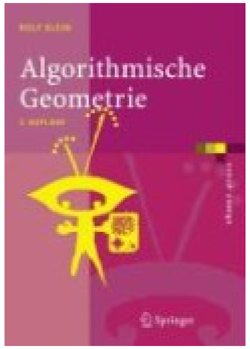
- explain concepts, structures, and problem definitions
- understand the discussed algorithms, and explain and analyze them
- select and adapt appropriate algorithms and data structures
- analyze new geometric problems and develop efficient solutions

Course Time Breakdown:	5LP = 150h
Time in lectures and exercise sessions	ca. 35h
Preparation and review	ca. 45h
Working on exercises	ca. 30h
Exam preparation	ca. 30h

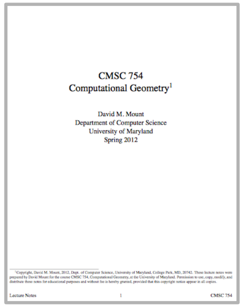
Class Materials



M. de Berg, O. Cheong, M. van Kreveld, M. Overmars:
Computational Geometry: Algorithms and Applications
Springer, 3rd Edition, 2008



Rolf Klein:
Algorithmische Geometrie
Springer, 2nd Edition, 2005



CMSC 754
Computational Geometry[1]

David M. Mount
Department of Computer Science
University of Maryland
Spring 2012

[1] Copyright, David M. Mount, 2012, Dept. of Computer Science, University of Maryland, College Park, MD, 20742. These lecture notes were prepared by David Mount for the course CMSC 754, Computational Geometry, at the University of Maryland. Permission to use, copy, modify, and distribute these notes for educational purposes and without fee is hereby granted, provided that this copyright notice appear in all copies.

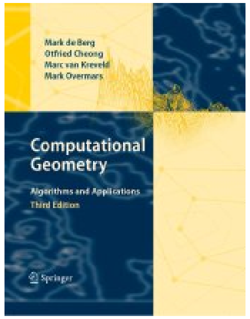
Lecture Notes 1 CMSC 754

David Mount:
Computational Geometry
Lecture Notes CMSC 754, U. Maryland, 2012

http://www.cs.umd.edu/class/spring2012/cmsc754/Lects/cmsc754-lects.pdf

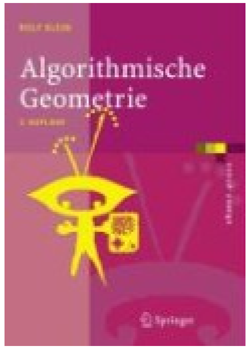
Both books are available in the library!

Class Materials

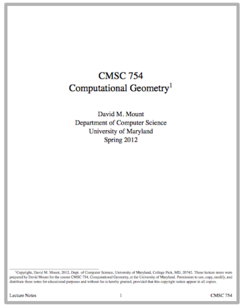


M. de Berg, O. Cheong, M. van Kreveld, M. Overmars:
Computational Geometry: Algorithms and Applications
Springer, 3rd Edition, 2008

PDF available on springerlink.com!



Rolf Klein:
Algorithmische Geometrie
Springer, 2nd Edition, 2005



CMSC 754
Computational Geometry[1]

David M. Mount
Department of Computer Science
University of Maryland
Spring 2012

[1] Copyright, David M. Mount, 2012, Dept. of Computer Science, University of Maryland, College Park, MD, 20742. These lecture notes were prepared by David Mount for the course CMSC 754, Computational Geometry, at the University of Maryland. Permission to use, copy, modify, and distribute these notes for educational purposes and without fee is hereby granted, provided that this copyright notice appear in all copies.

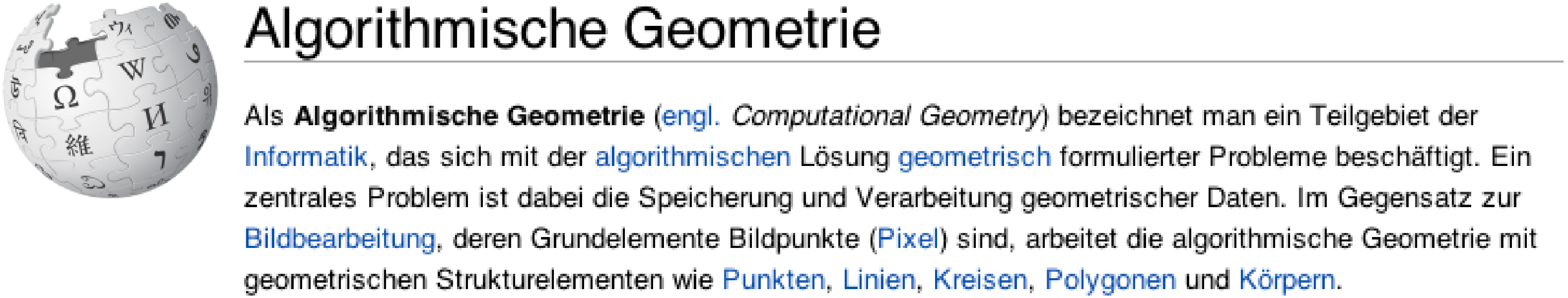
Lecture Notes 1 CMSC 754

David Mount:
Computational Geometry
Lecture Notes CMSC 754, U. Maryland, 2012

`http://www.cs.umd.edu/class/spring2012/cmsc754/Lects/cmsc754-lects.pdf`

Both books are available in the library!

What is Computational Geometry?



Algorithmische Geometrie

Als **Algorithmische Geometrie** (engl. *Computational Geometry*) bezeichnet man ein Teilgebiet der Informatik, das sich mit der algorithmischen Lösung geometrisch formulierter Probleme beschäftigt. Ein zentrales Problem ist dabei die Speicherung und Verarbeitung geometrischer Daten. Im Gegensatz zur Bildbearbeitung, deren Grundelemente Bildpunkte (Pixel) sind, arbeitet die algorithmische Geometrie mit geometrischen Strukturelementen wie Punkten, Linien, Kreisen, Polygonen und Körpern.

What is Computational Geometry?

Algorithmische Geometrie

Computational geometry is a branch of computer science that deals with algorithmic solutions to geometric problems. A central problem is the storage and processing of geometric data...such as points, lines, circles, polygons...

What is Computational Geometry?

Algorithmische Geometrie

Computational geometry is a branch of computer science that deals with algorithmic solutions to geometric problems. A central problem is the storage and processing of geometric data...such as points, lines, circles, polygons...

Where is Computational Geometry Used?

- Computer Graphics and Image Processing
- Visualization
- Geographic Information Systems (GIS)
- Robotics
- . . .

What is Computational Geometry?

Algorithmische Geometrie

Computational geometry is a branch of computer science that deals with algorithmic solutions to geometric problems. A central problem is the storage and processing of geometric data...such as points, lines, circles, polygons...

Where is Computational Geometry Used?

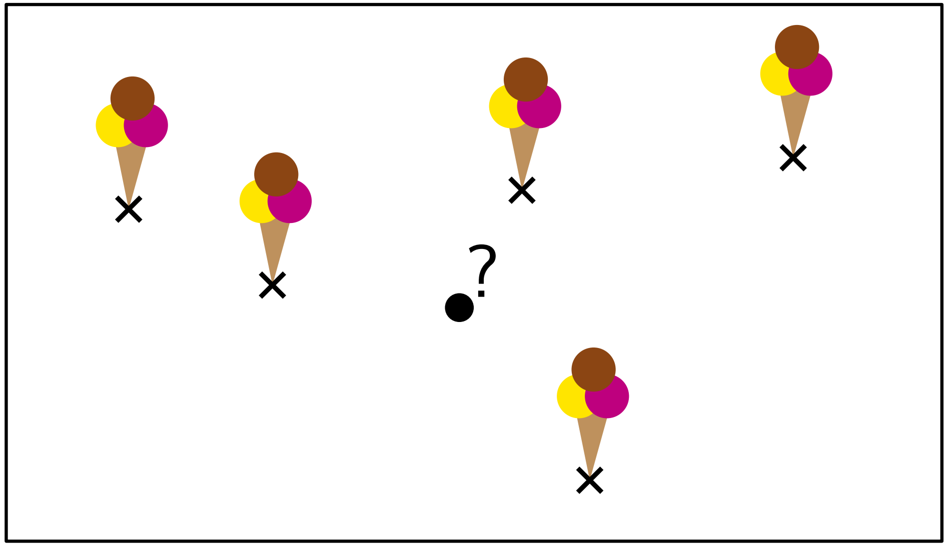
- Computer Graphics and Image Processing
- Visualization
- Geographic Information Systems (GIS)
- Robotics
- ...

Central Themes

- Geometric algorithms and data structures
- Discrete and combinatorial geometric problems

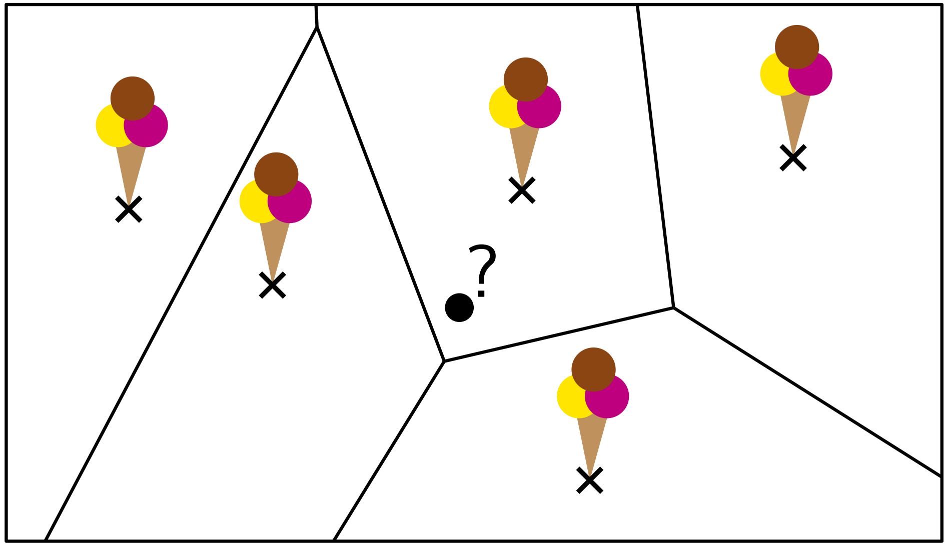
Example 1

It's a hot 42°C summer day in Karlsruhe. Suppose you know the location of every ice cream shop in the city. How can you determine the closest ice cream shop for any location on a map?



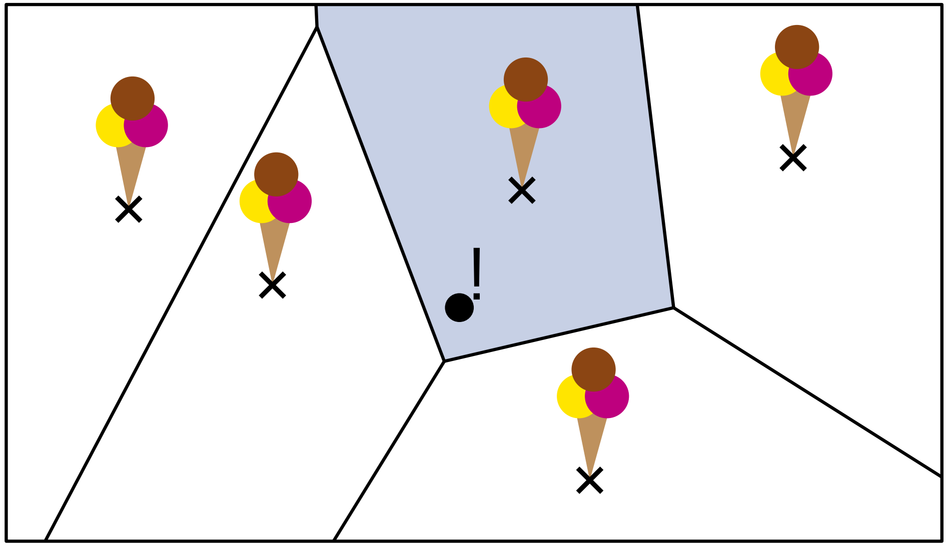
Example 1

It's a hot 42°C summer day in Karlsruhe. Suppose you know the location of every ice cream shop in the city. How can you determine the closest ice cream shop for any location on a map?



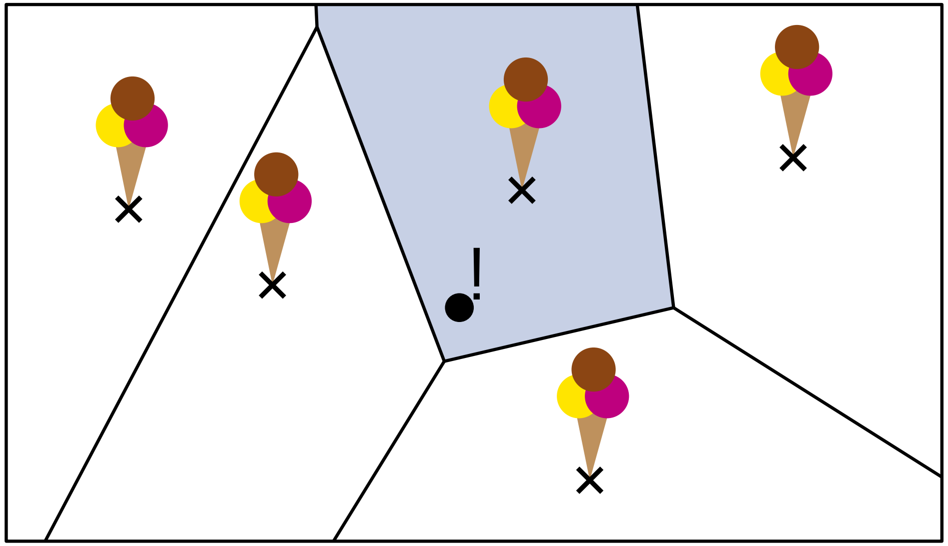
Example 1

It's a hot 42°C summer day in Karlsruhe. Suppose you know the location of every ice cream shop in the city. How can you determine the closest ice cream shop for any location on a map?



Example 1

It's a hot 42° C summer day in Karlsruhe. Suppose you know the location of every ice cream shop in the city. How can you determine the closest ice cream shop for any location on a map?

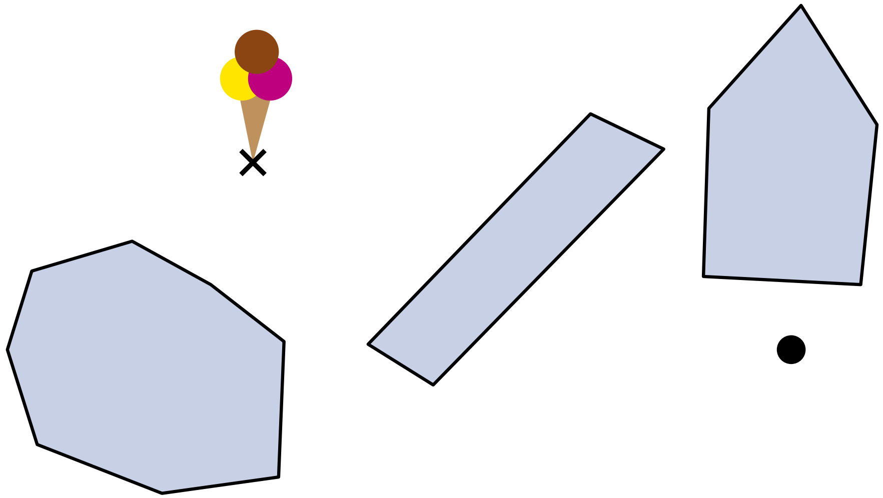


The solution is a *division* of $\mathbb{R}^2$, called a **Voronoi Diagram**.

Many applications in Natural sciences, Geometry, Informatics, Health, Engeneering,...

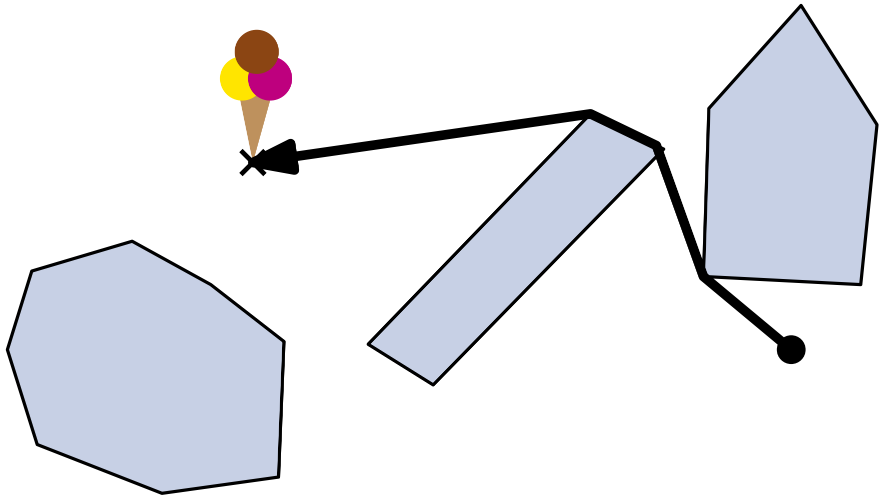
Example 2

Now it is 50°C in Karlsruhe. We want to send a robot to buy an ice cream cone. How can the robot reach the destination without passing through houses, park benches, and trees?



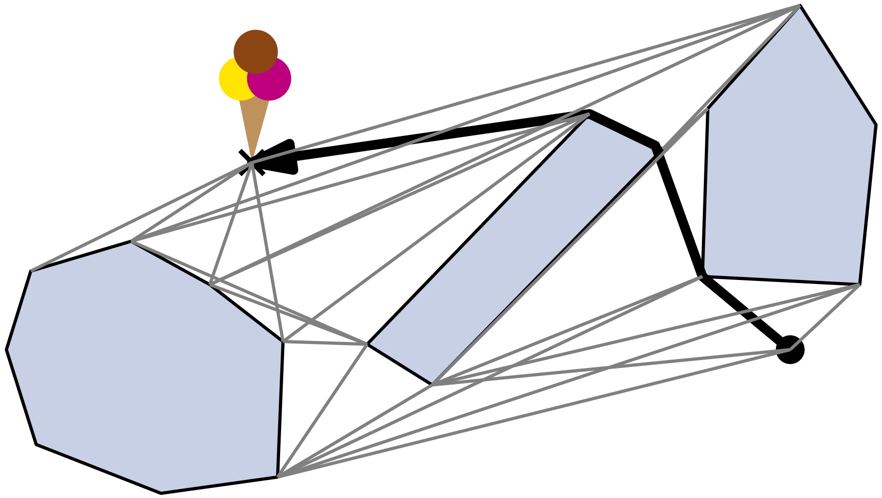
Example 2

Now it is 50°C in Karlsruhe. We want to send a robot to buy an ice cream cone. How can the robot reach the destination without passing through houses, park benches, and trees?



Example 2

Now it is 50°C in Karlsruhe. We want to send a robot to buy an ice cream cone. How can the robot reach the destination without passing through houses, park benches, and trees?

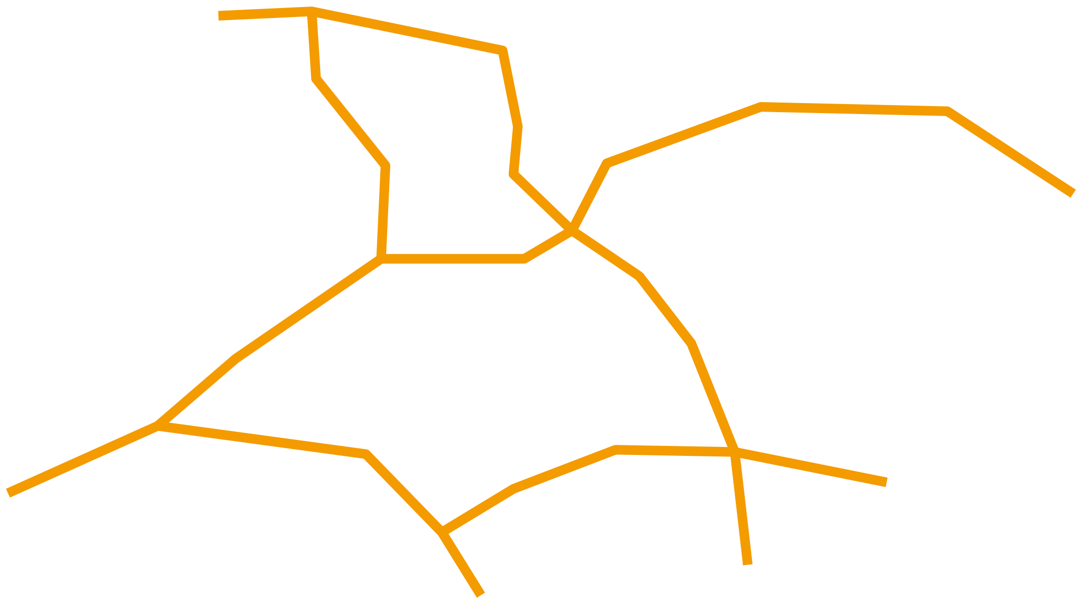


Motion planning problem in robotics:
Given a set of obstacles with a start and destination point, find a collision-free shortest route (e.g., using the **visibility graph).**

Example 3

Maps in geographic information systems consist of several levels (e.g., roads, water, borders, etc.). When superimposing several layers, what are the intersection points?

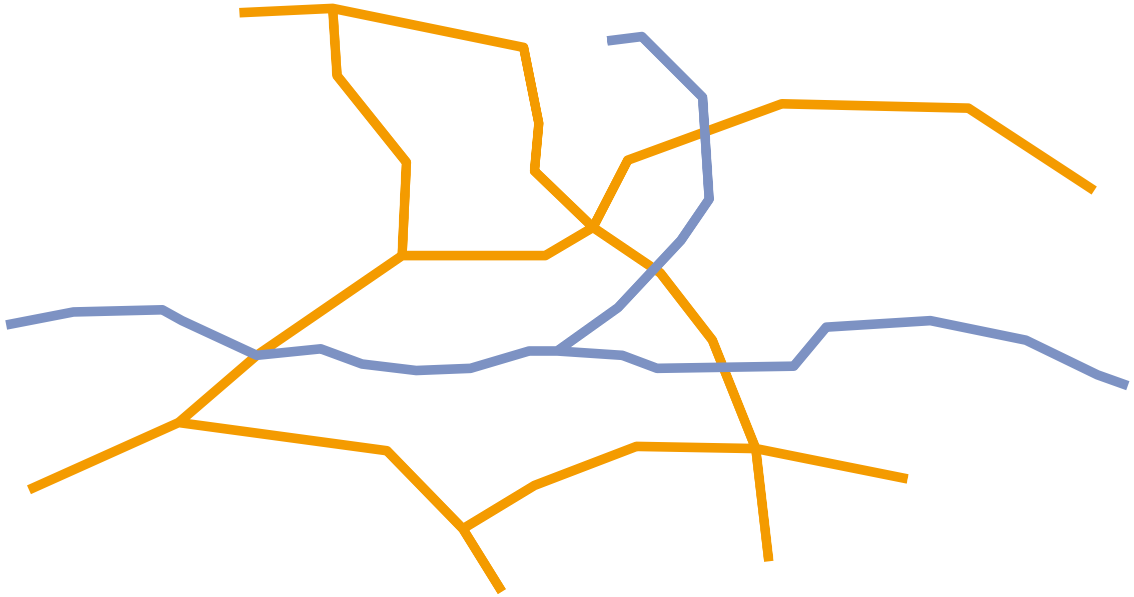
One example is to view all roads and rivers as a set of links and ask for the bridges. For these, you have to find all intersections between the two layers.



Example 3

Maps in geographic information systems consist of several levels (e.g., roads, water, borders, etc.). When superimposing several layers, what are the intersection points?

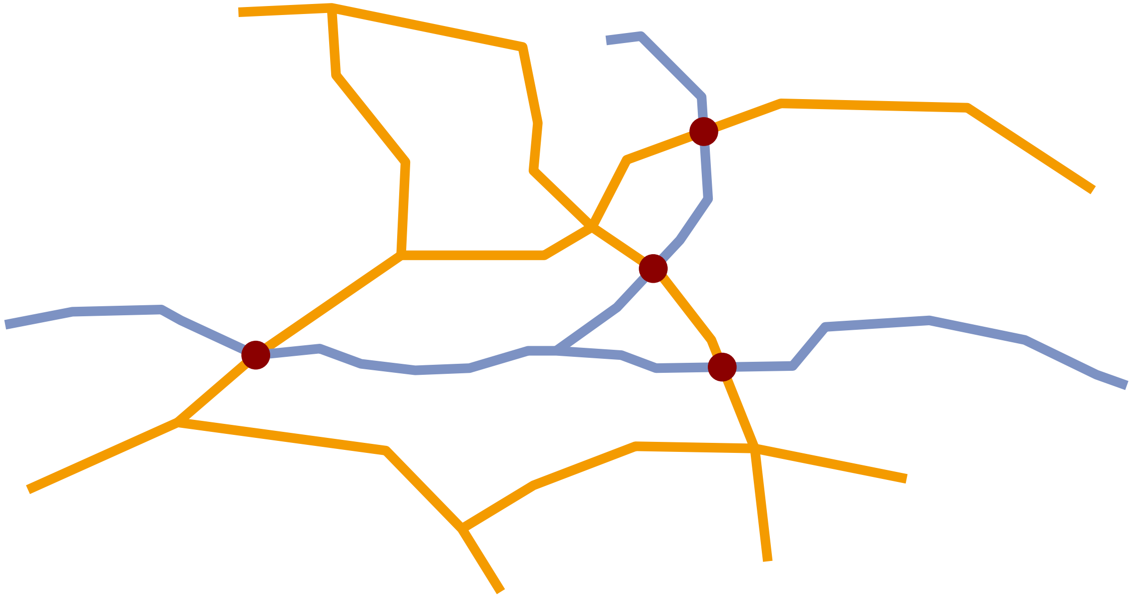
One example is to view all roads and rivers as a set of links and ask for the bridges. For these, you have to find all intersections between the two layers.



Example 3

Maps in geographic information systems consist of several levels (e.g., roads, water, borders, etc.). When superimposing several layers, what are the intersection points?

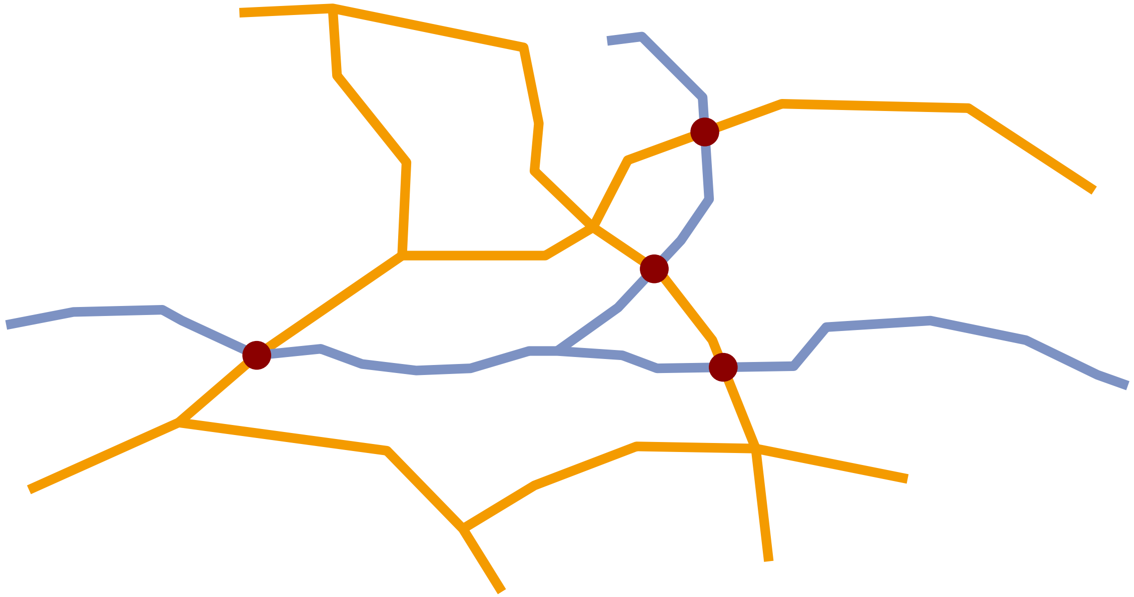
One example is to view all roads and rivers as a set of links and ask for the bridges. For these, you have to find all intersections between the two layers.



Example 3

Maps in geographic information systems consist of several levels (e.g., roads, water, borders, etc.). When superimposing several layers, what are the intersection points?

One example is to view all roads and rivers as a set of links and ask for the bridges. For these, you have to find all intersections between the two layers.

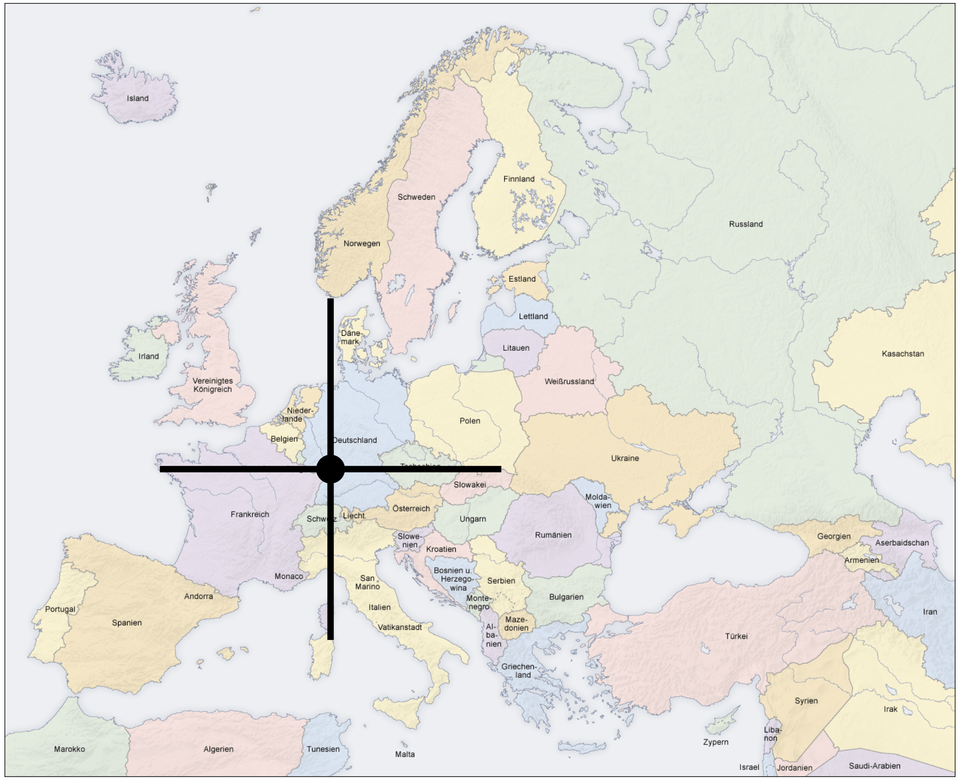


Testing all edge pairs is slow. How can you quickly find all intersections?

(**Line Segment Intersections**)

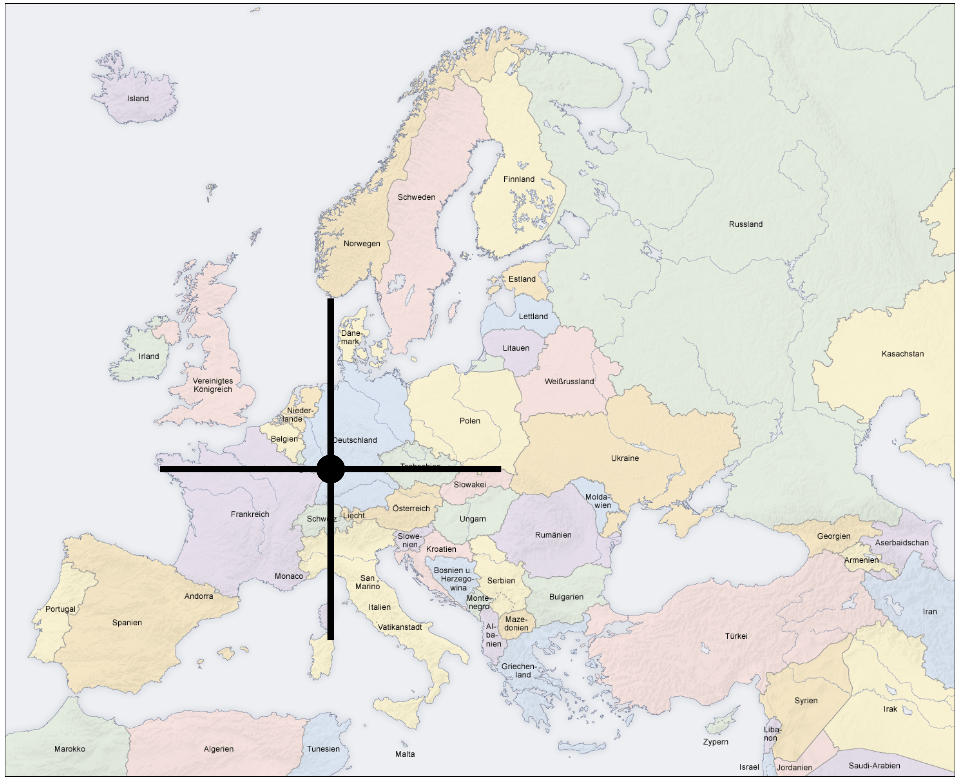
Example 4

Given a map and a query point q (e.g., a mouse click), determine the country containing q.



Example 4

Given a map and a query point q (e.g., a mouse click), determine the country containing q.



We want a fast data structure for answering point queries.
(**Point Location**)

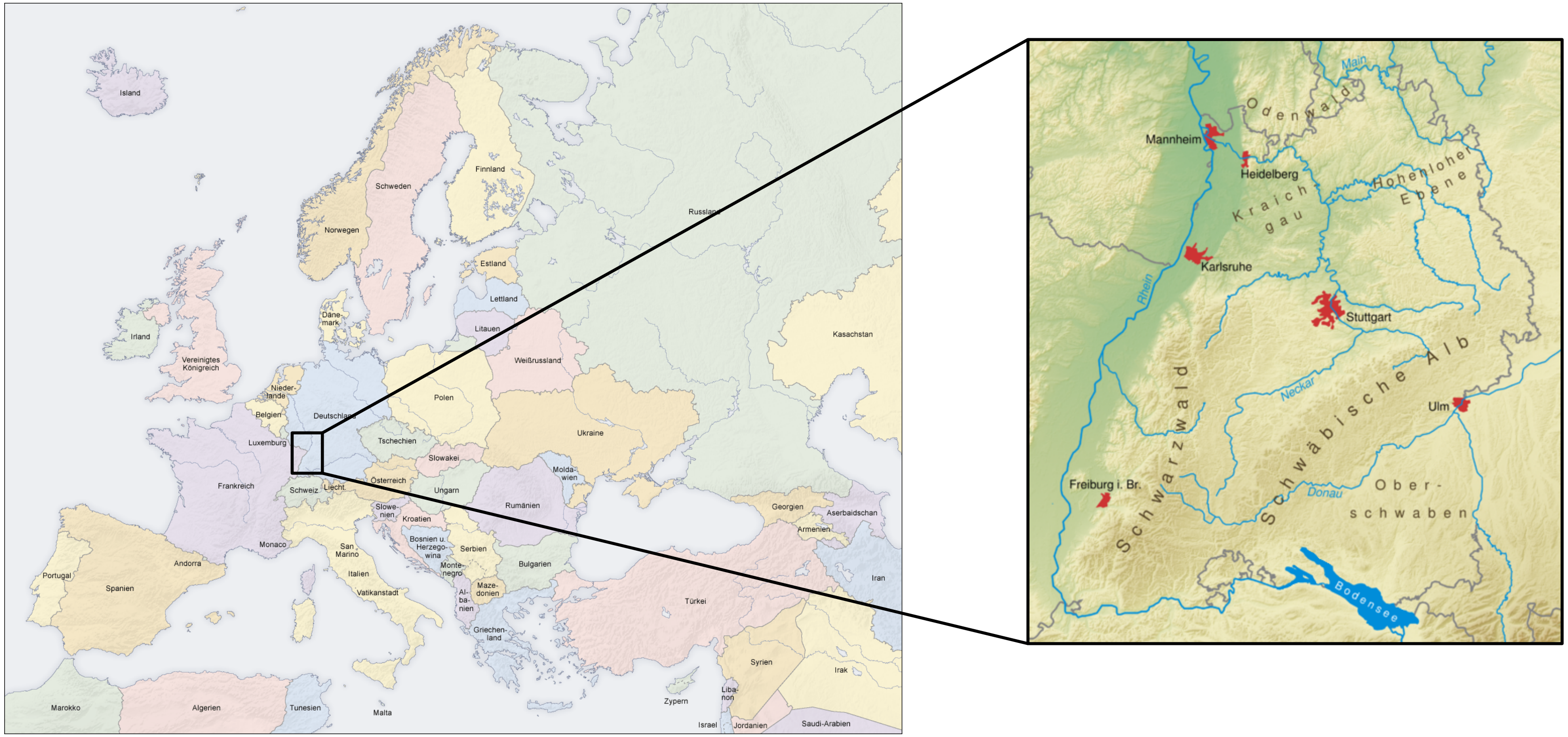
Example 5

A navigation system should display a current map. How can we effectively choose the data to display?

Island
Finnland
Schweden
Norwegen
Russland
Estland
Lettland
Däne-
mark
Litauen
Irland
Vereinigtes
Königreich
Weißrussland
Kasachstan
Nieder-
lande
Polen
Belgien
Deutschland
Ukraine
Luxemburg
Tschechien
Slowakei
Molda-
wien
Frankreich
Schweiz
Liecht
Österreich
Ungarn
Rumänien
Georgien
Aserbaidschan
Slowe-
nien
Kroatien
Armenien
Monaco
Bosnien u.
Herzego-
wina
Serbien
San
Marino
Andorra
Bulgarien
Monte-
negro
Iran
Portugal
Italien
Maze-
donien
Spanien
Vatikanstadt
Al-
ba-
nien
Türkei
Griechen-
land
Syrien
Irak
Liba-
non
Zypern
Marokko
Algerien
Tunesien
Malta
Israel
Jordanien
Saudi-Arabien

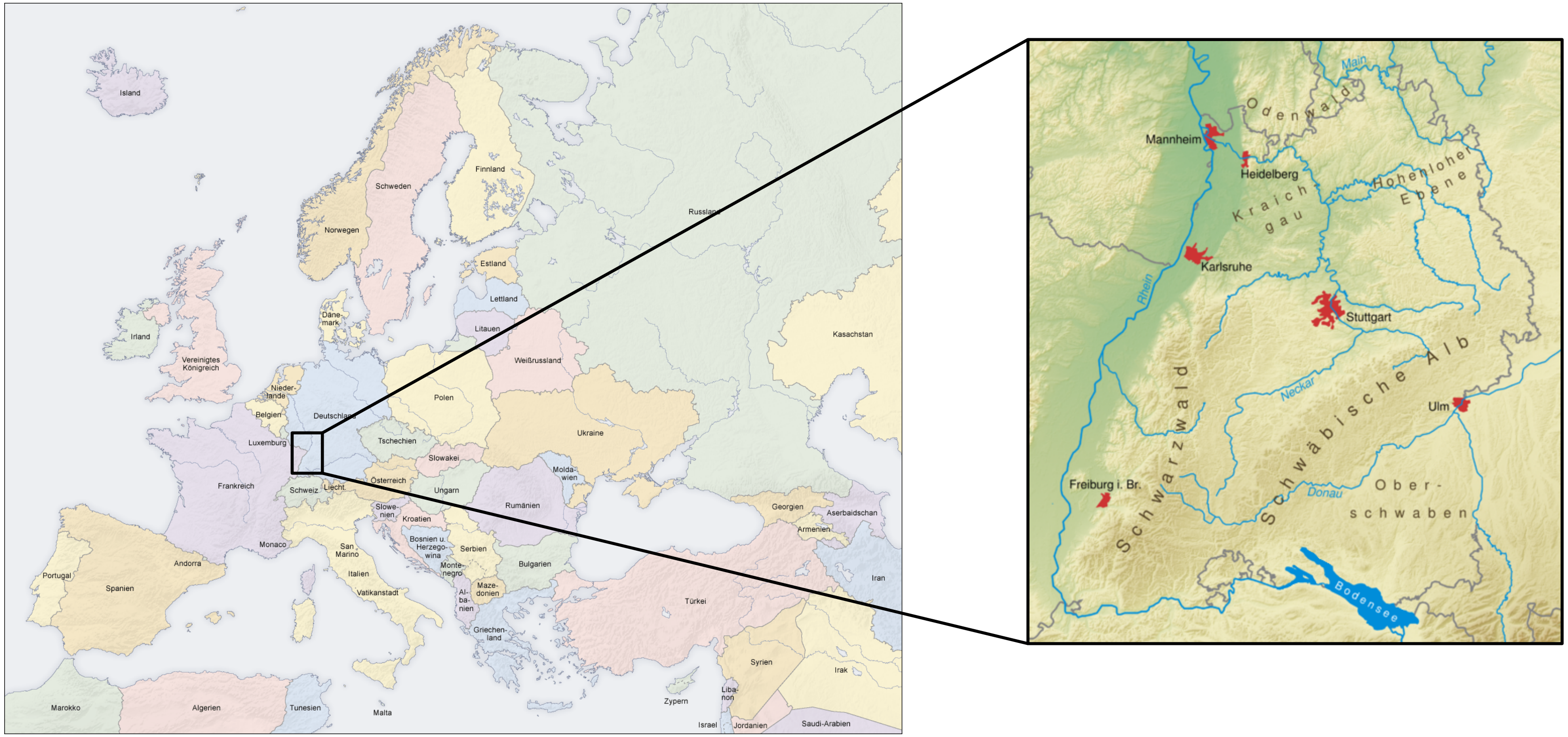
Example 5

A navigation system should display a current map. How can we effectively choose the data to display?



Example 5

A navigation system should display a current map. How can we effectively choose the data to display?



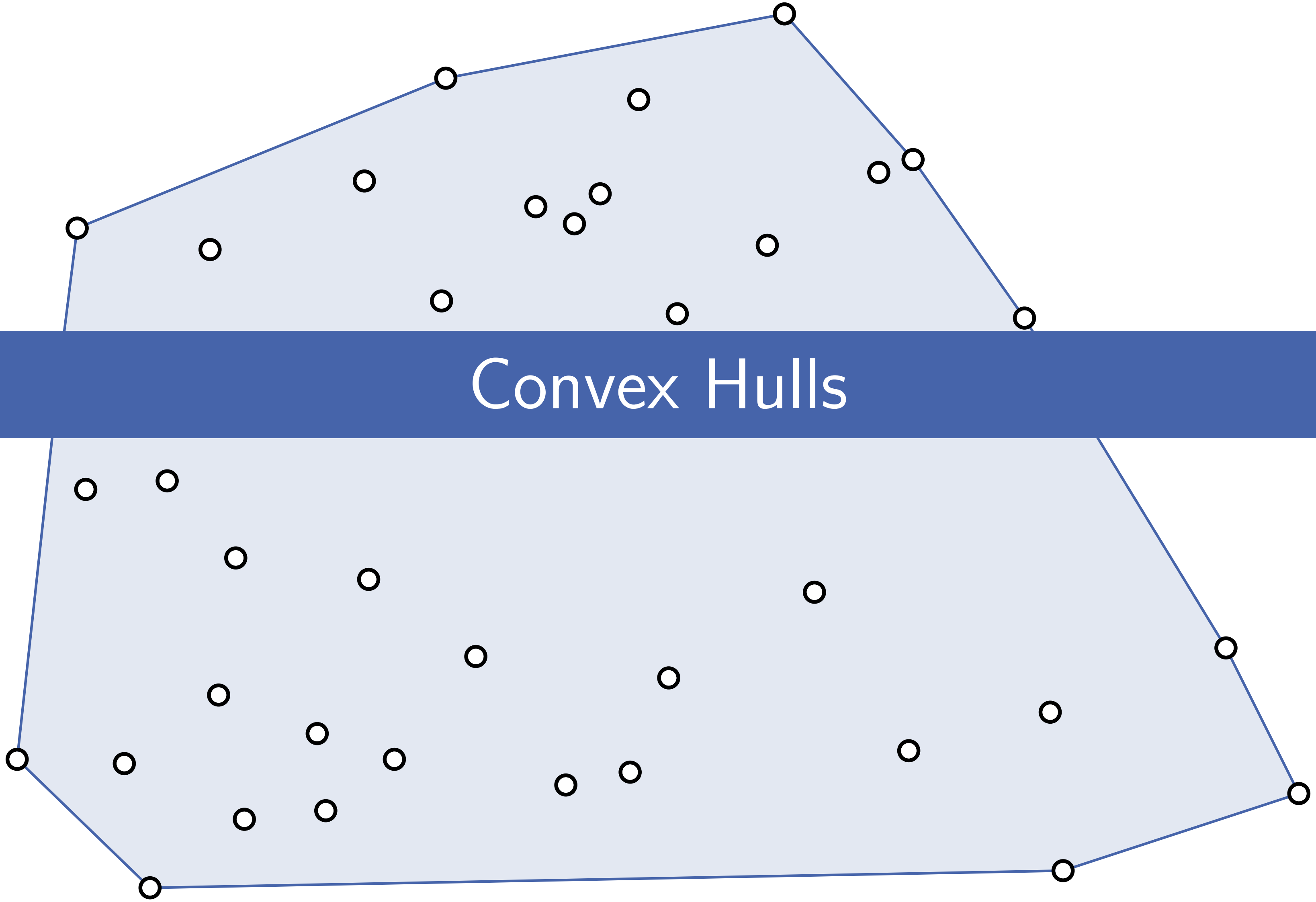
Evaluating each map feature is unrealistic.

We want a fast data structure for answering **range queries**

Topics

We will cover the following topics:

- Convex Hulls
- Line Segment Intersection
- Polygon Triangulation
- Geometric Linear Programming
- Data Structures for Range Queries
- Data Structure for Point Location Queries
- Voronoi Diagrams and Delaunay Triangulation
- Duality of Points and Lines
- Quadtrees
- Well-Separated Pair Decompositions
- Visibility Graphs



Convex Hulls

Mixing Ratios

Given...

Mixture	fraction A	fraction B
s_1	$10\,\%$	$35\,\%$
s_2	$20\,\%$	$5\,\%$

Mixing Ratios

Given...

Mixture	fraction A	fraction B
s_1	$10\,\%$	$35\,\%$
s_2	$20\,\%$	$5\,\%$

can we mix

q_1	$15\,\%$	$20\,\%$
q_2	$25\,\%$	$28\,\%$

using s_1, s_2?

Mixing Ratios

Given...

Mixture	fraction A	fraction B
s_1	$10\,\%$	$35\,\%$
s_2	$20\,\%$	$5\,\%$

can we mix

q_1	$15\,\%$	$20\,\%$
q_2	$25\,\%$	$28\,\%$

using s_1, s_2?

q_1: Yes! Ratio 1:1

q_2: No!

Mixing Ratios

Given...

Mixture	fraction A	fraction B
s_1	$10\,\%$	$35\,\%$
s_2	$20\,\%$	$5\,\%$
s_3	$40\,\%$	$25\,\%$

can we mix

q_1	$15\,\%$	$20\,\%$
q_2	$25\,\%$	$28\,\%$

using s_1, s_2, s_3?

Mixing Ratios

Given...

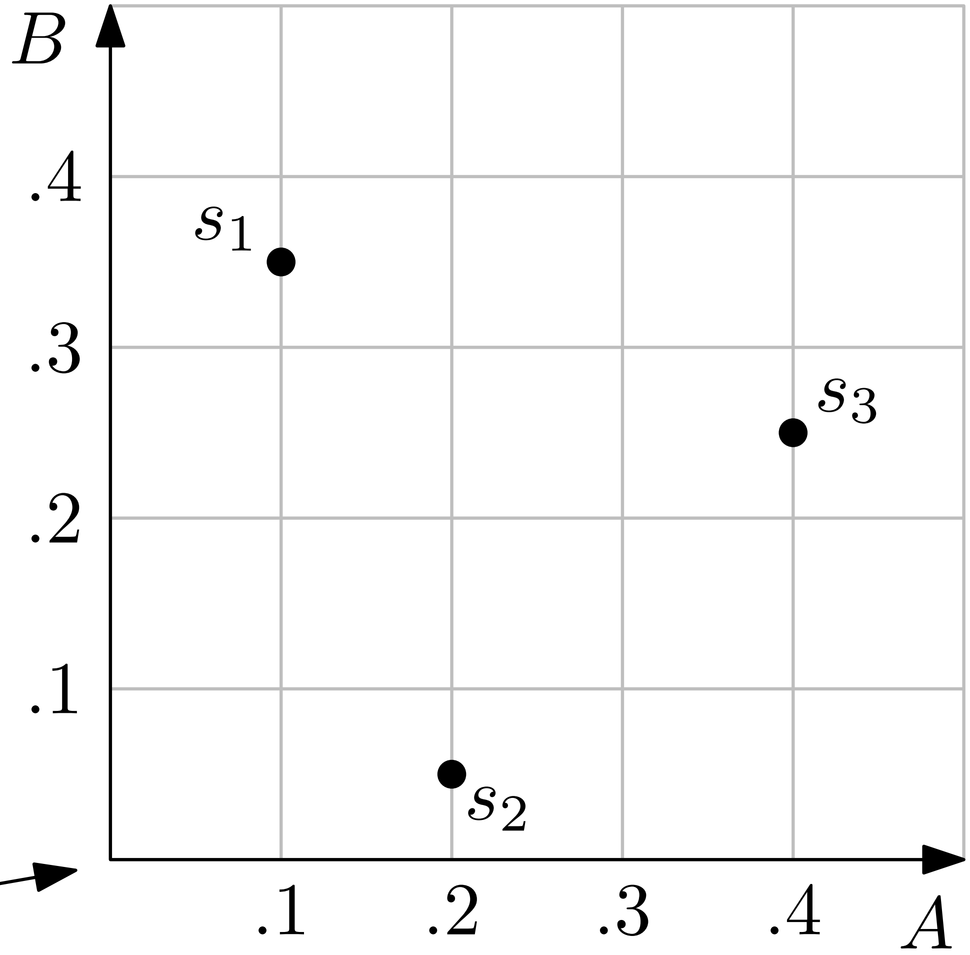
Mixture	fraction A	fraction B
s_1	$10\,\%$	$35\,\%$
s_2	$20\,\%$	$5\,\%$
s_3	$40\,\%$	$25\,\%$

can we mix

q_1	$15\,\%$	$20\,\%$
q_2	$25\,\%$	$28\,\%$

using s_1, s_2, s_3?

geom. interpretation



Mixing Ratios

Given...

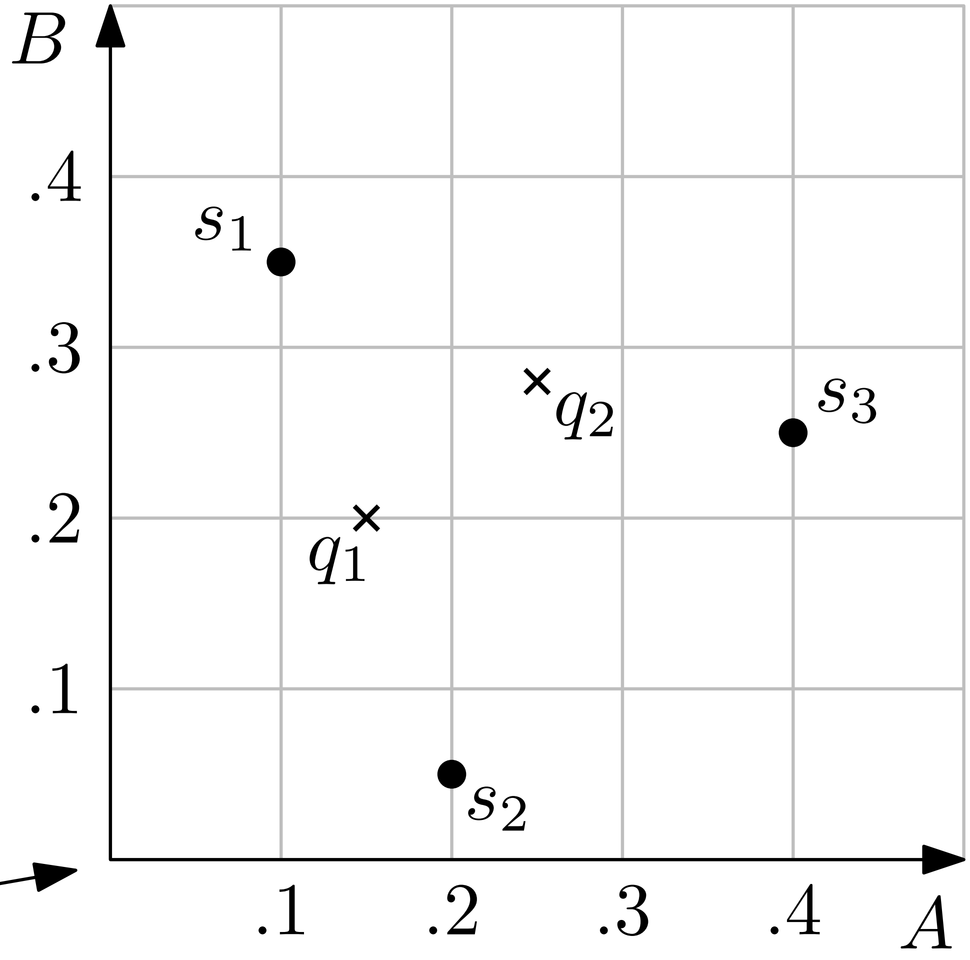
Mixture	fraction A	fraction B
s_1	$10\,\%$	$35\,\%$
s_2	$20\,\%$	$5\,\%$
s_3	$40\,\%$	$25\,\%$

can we mix

q_1	$15\,\%$	$20\,\%$
q_2	$25\,\%$	$28\,\%$

using s_1, s_2, s_3?

geom. interpretation



Mixing Ratios

Given...

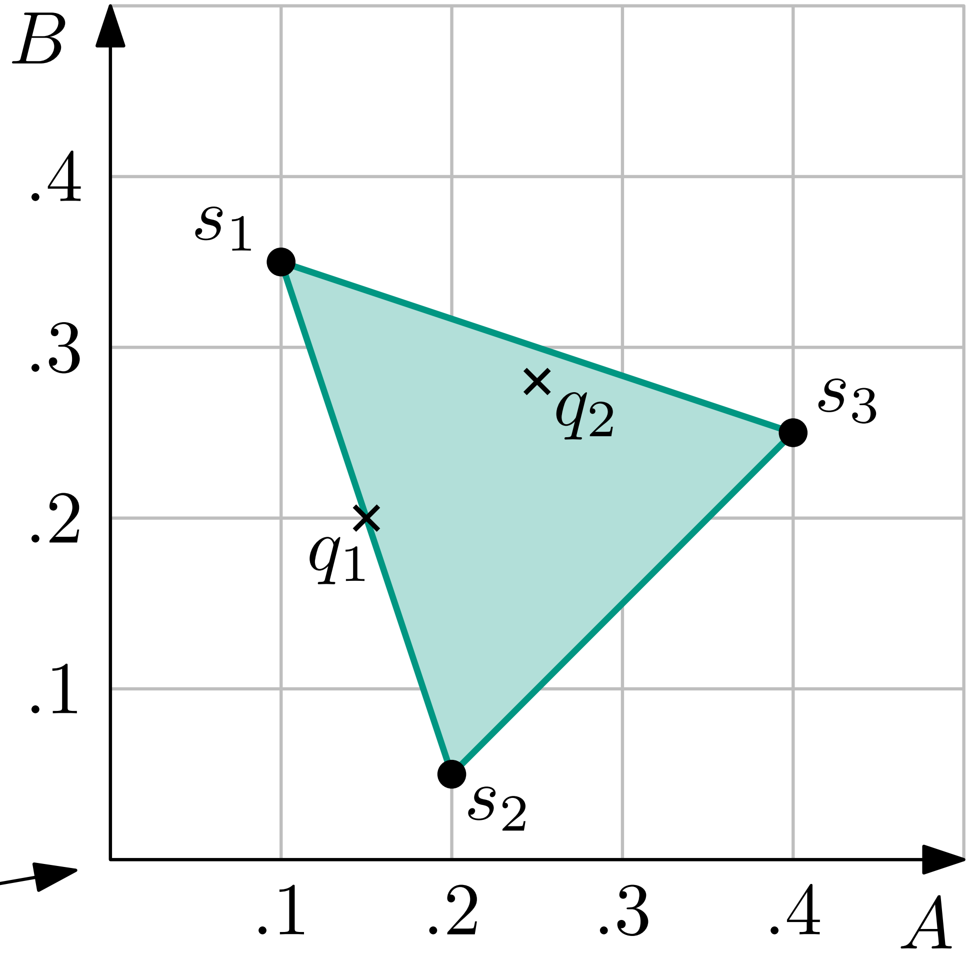
Mixture	fraction A	fraction B
s_1	10 %	35 %
s_2	20 %	5 %
s_3	40 %	25 %

can we mix

q_1	15 %	20 %
q_2	25 %	28 %

using s_1, s_2, s_3?

geom. interpretation



Mixing Ratios

Given...

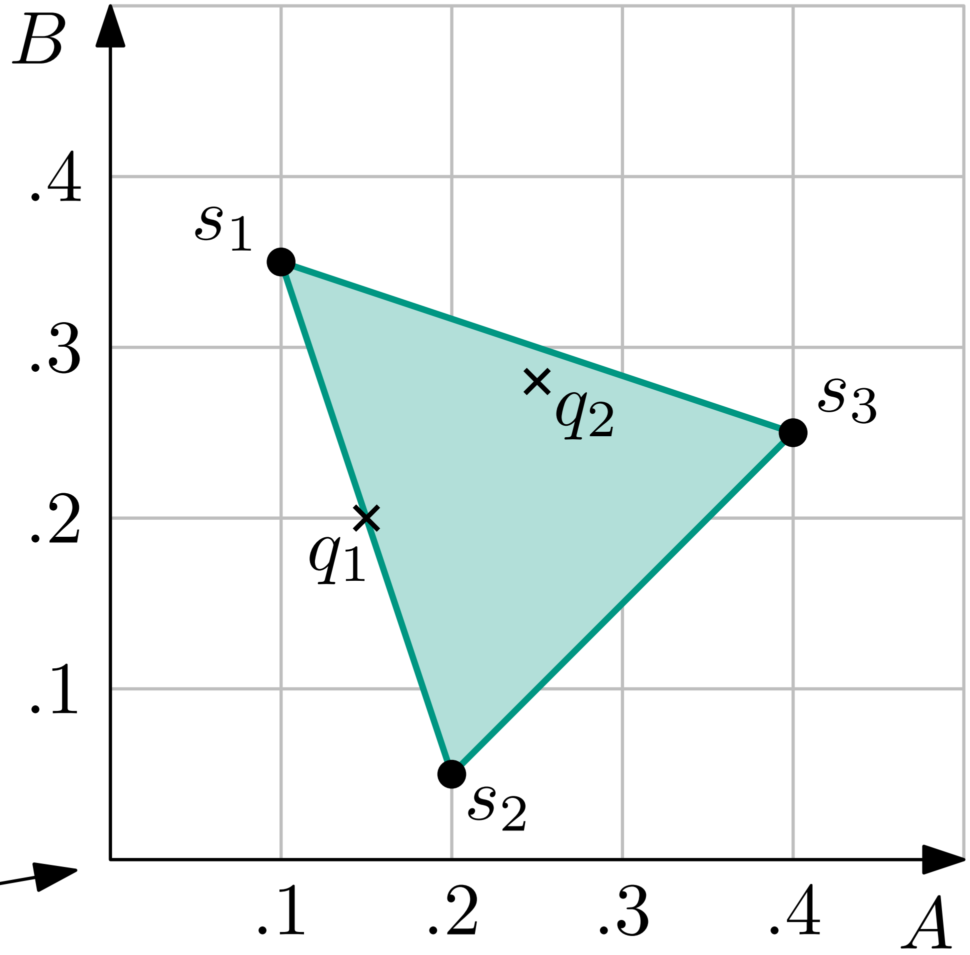
Mixture	fraction A	fraction B
s_1	$10\,\%$	$35\,\%$
s_2	$20\,\%$	$5\,\%$
s_3	$40\,\%$	$25\,\%$

can we mix

q_1	$15\,\%$	$20\,\%$
q_2	$25\,\%$	$28\,\%$

using s_1, s_2, s_3?

geom. interpretation



Obs: Given a set $S \subset \mathbb{R}^2$ of mixtures, we can make another mixture $q \in \mathbb{R}^2$ out of S $\Leftrightarrow$

Mixing Ratios

Given...

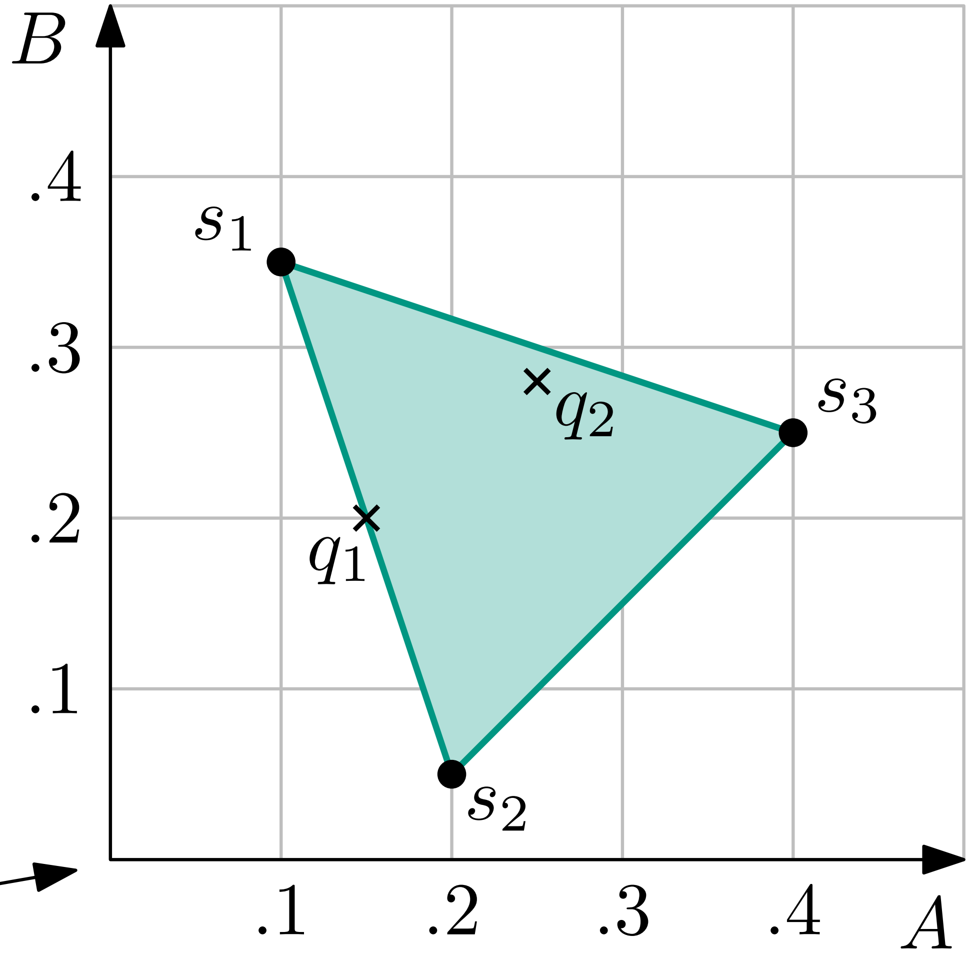
Mixture	fraction A	fraction B
s_1	10 %	35 %
s_2	20 %	5 %
s_3	40 %	25 %

can we mix

q_1	15 %	20 %
q_2	25 %	28 %

using s_1, s_2, s_3?

geom. interpretation



Obs: Given a set $S \subset \mathbb{R}^2$ of mixtures, we can make another mixture $q \in \mathbb{R}^2$ out of S $\Leftrightarrow$ $q \in$ convex hull $CH(S)$.

$$q = \sum_i \lambda_i s_i \text{ with } \sum_i \lambda_i = 1.$$

Mixing Ratios

Given...

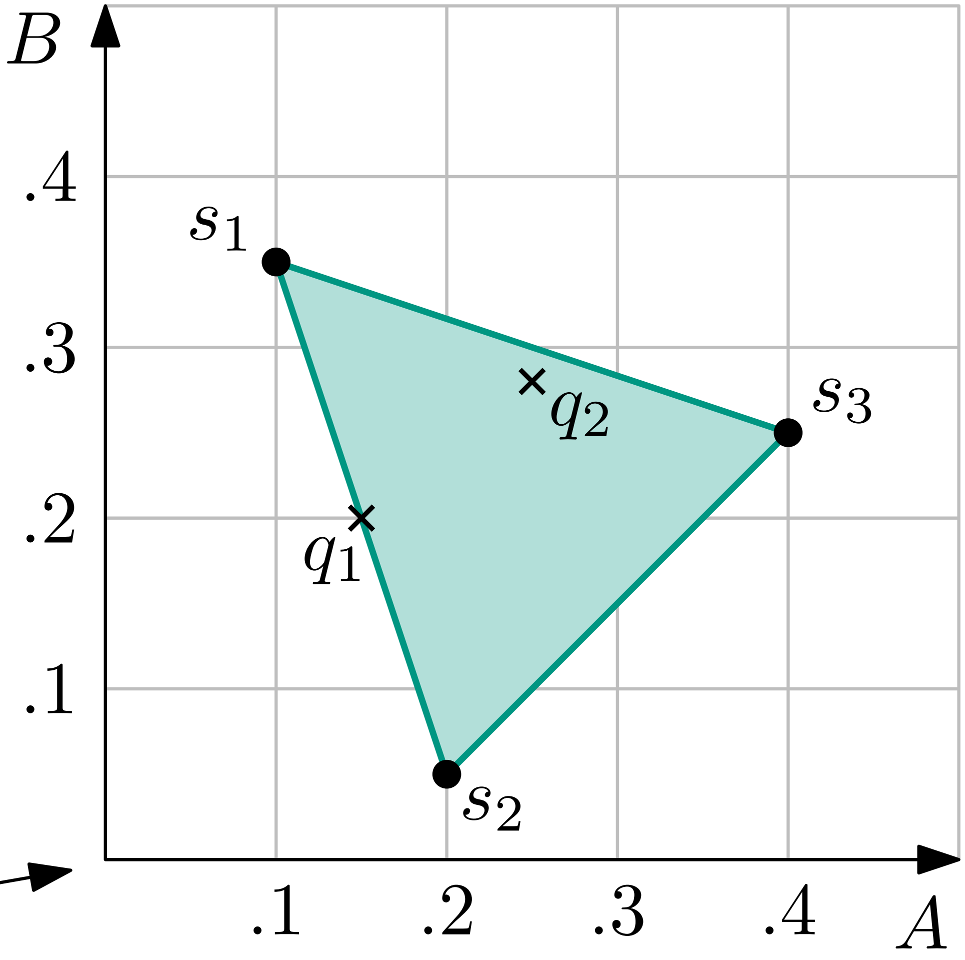
Mixture	fraction A	fraction B
s_1	10 %	35 %
s_2	20 %	5 %
s_3	40 %	25 %

can we mix

q_1	15 %	20 %
q_2	25 %	28 %

using s_1, s_2, s_3?

geom. interpretation



Obs: Given a set $S \subset \mathbb{R}^{\cancel{2}d}$ of mixtures, we can make another mixture $q \in \mathbb{R}^{\cancel{2}d}$ out of S $\Leftrightarrow$ $q \in$ convex hull $CH(S)$.

$$q = \sum_i \lambda_i s_i \text{ with } \sum_i \lambda_i = 1.$$

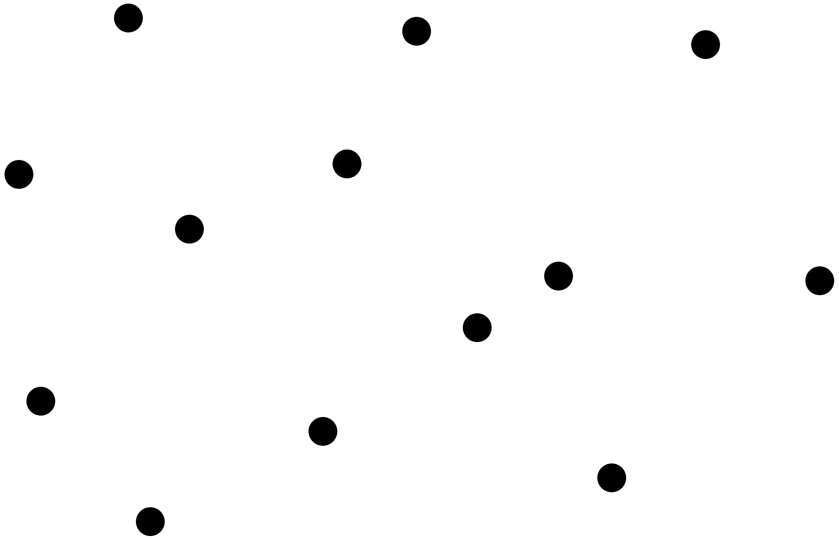
Definition of Convex Hull

Def: A region $S \subseteq \mathbb{R}^2$ is called **convex**, when for two points $p, q \in S$, line $\overline{pq} \in S$.

The **convex hull** $CH(S)$ of S is the smallest convex region containing S.

Definition of Convex Hull

Def: A region $S \subseteq \mathbb{R}^2$ is called **convex**, when for two points $p, q \in S$, line $\overline{pq} \in S$.
The **convex hull** $CH(S)$ of S is the smallest convex region containing S.

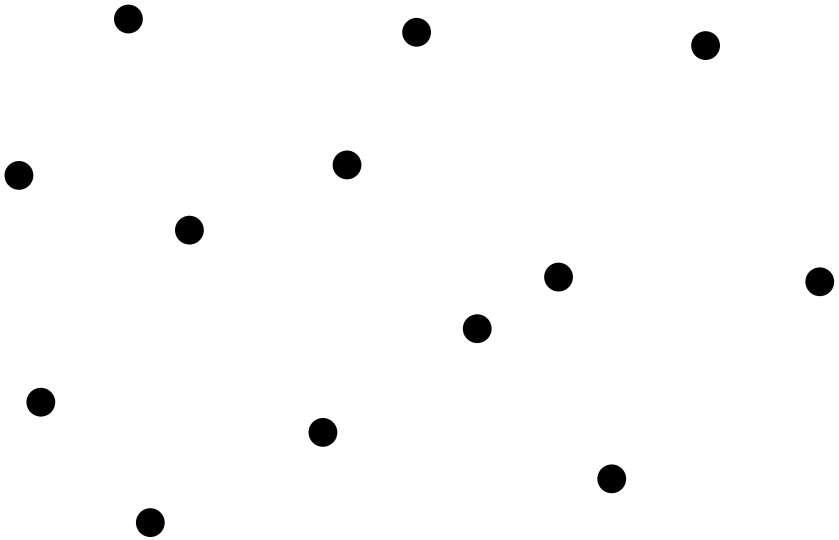


Definition of Convex Hull

Def: A region $S \subseteq \mathbb{R}^2$ is called **convex**, when for two points $p, q \in S$, line $\overline{pq} \in S$.
The **convex hull** $CH(S)$ of S is the smallest convex region containing S.

In mathematics:

- define $CH(S) = \bigcap_{C \supseteq S:\ C \text{ convex}} C$

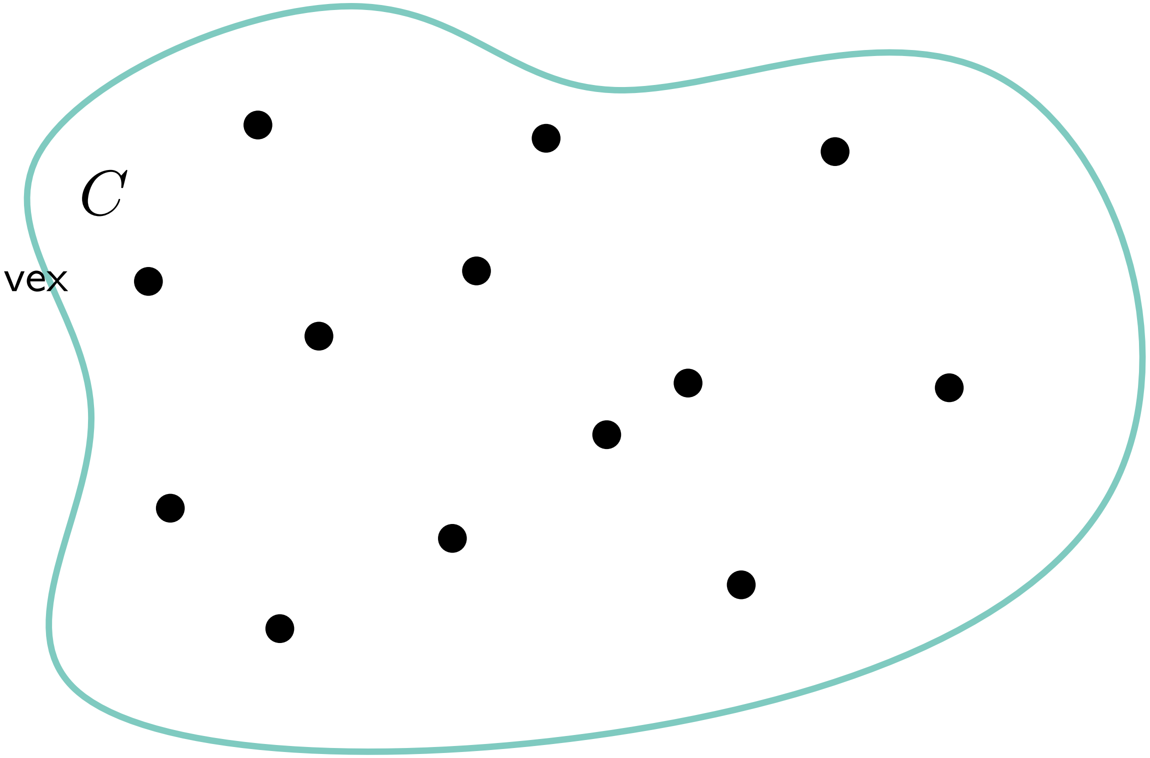


Definition of Convex Hull

Def: A region $S \subseteq \mathbb{R}^2$ is called **convex**, when for two points $p, q \in S$, line $\overline{pq} \in S$.
The **convex hull** $CH(S)$ of S is the smallest convex region containing S.

In mathematics:

- define $CH(S) = \bigcap\limits_{C \supseteq S:\ C \text{ convex}} C$



In physics:

- put a large rubber band around all points

Definition of Convex Hull

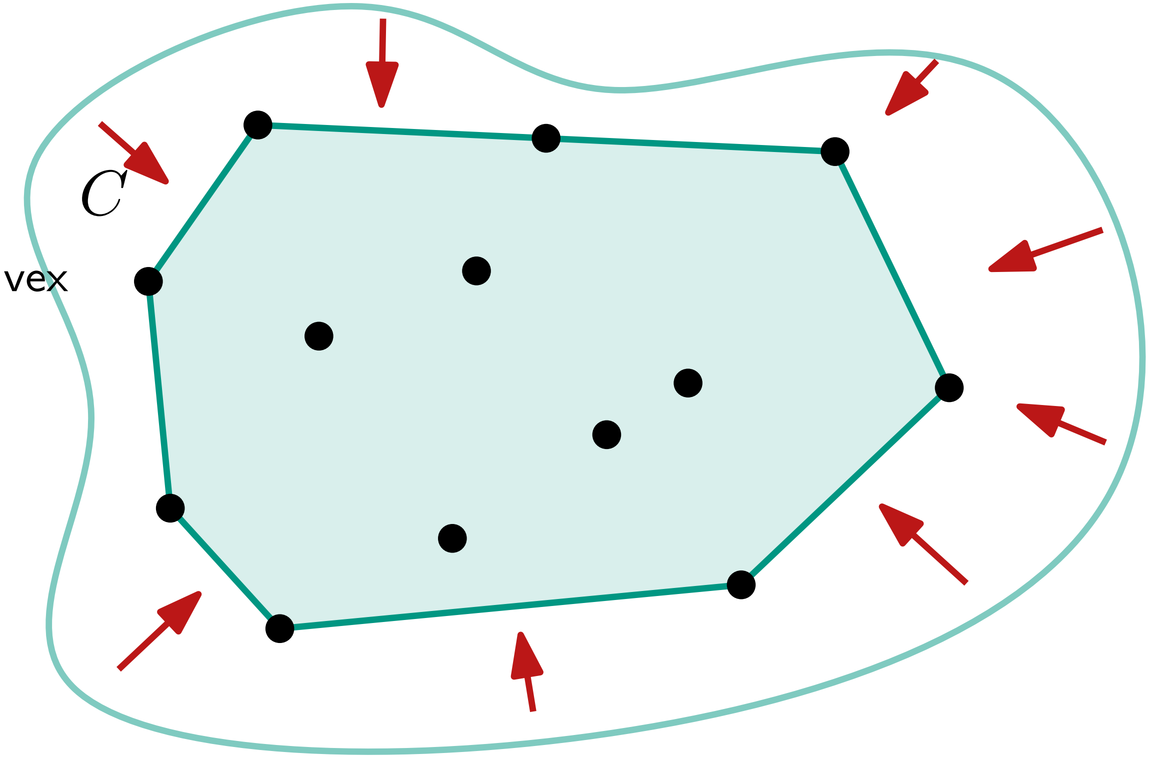
Def: A region $S \subseteq \mathbb{R}^2$ is called **convex**, when for two points $p, q \in S$, line $\overline{pq} \in S$.
The **convex hull** $CH(S)$ of S is the smallest convex region containing S.

In mathematics:

- define $CH(S) = \displaystyle\bigcap_{C \supseteq S:\ C \text{ convex}} C$

In physics:

- put a large rubber band around all points
- and let it go!

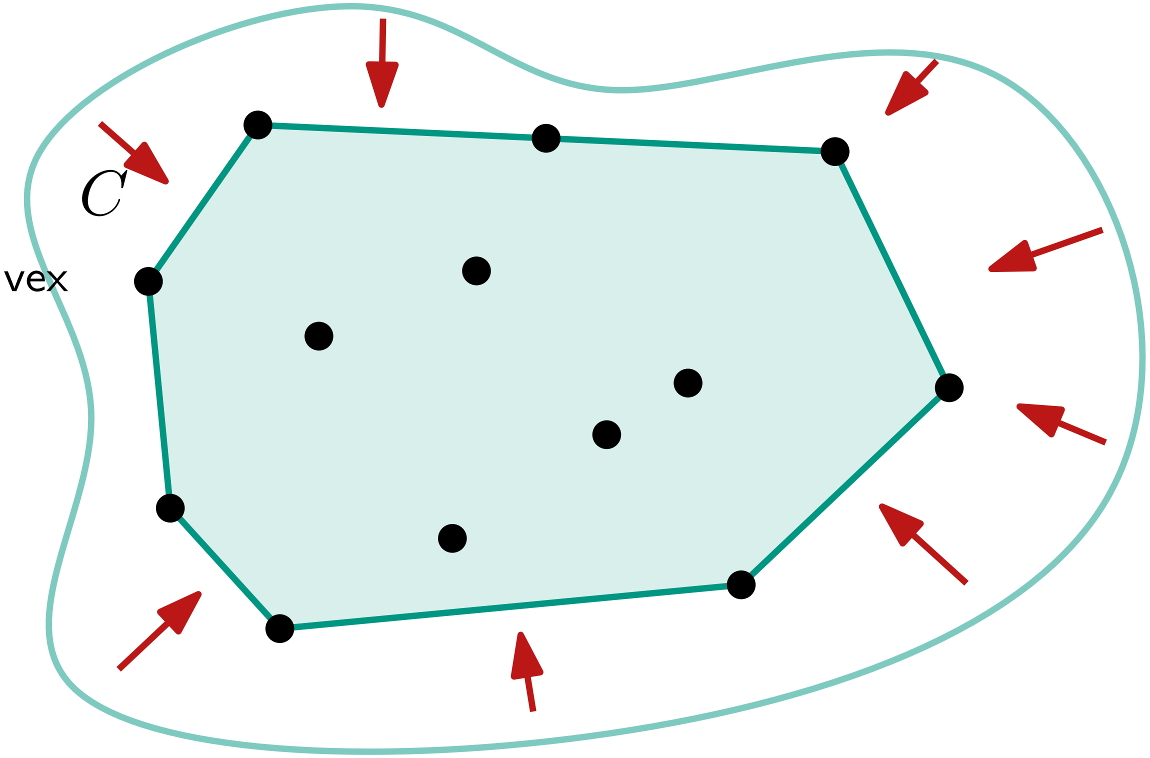


Definition of Convex Hull

Def: A region $S \subseteq \mathbb{R}^2$ is called **convex**, when for two points $p, q \in S$, line $\overline{pq} \in S$.
The **convex hull** $CH(S)$ of S is the smallest convex region containing S.

In mathematics:

- define $CH(S) = \bigcap_{C \supseteq S:\ C \text{ convex}} C$



In physics:

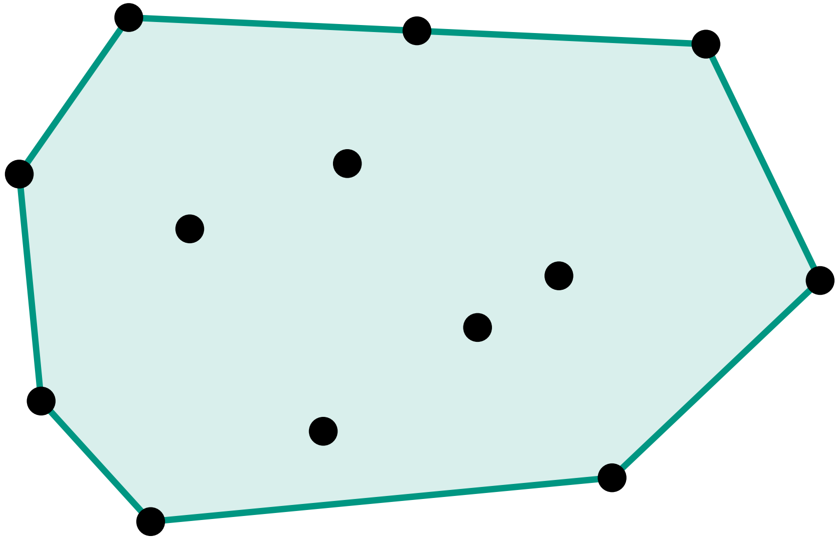
- put a large rubber band around all points
- and let it go!

unfortunately none help algorithmically

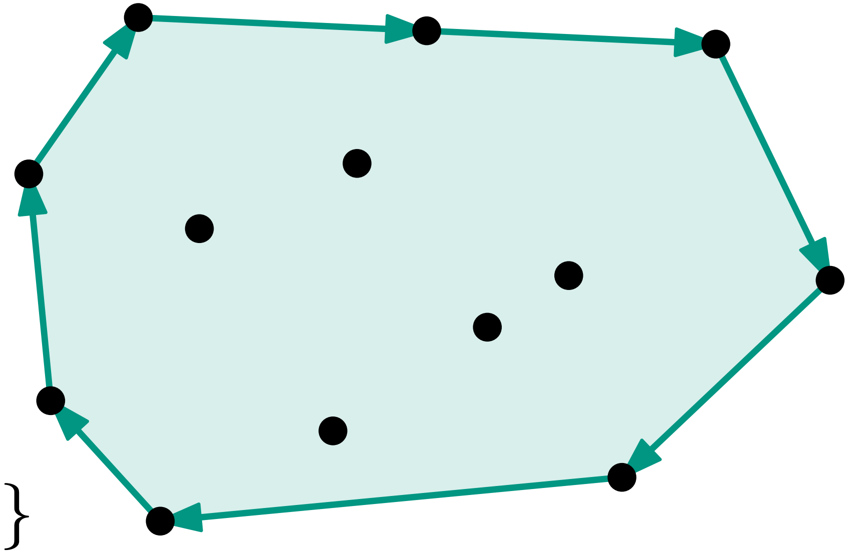
Algorithmic Approach

Lemma:

For a set of points $P \subseteq \mathbb{R}^2$, $CH(P)$ is a convex polygon that contains P and whose vertices are in P.



Algorithmic Approach



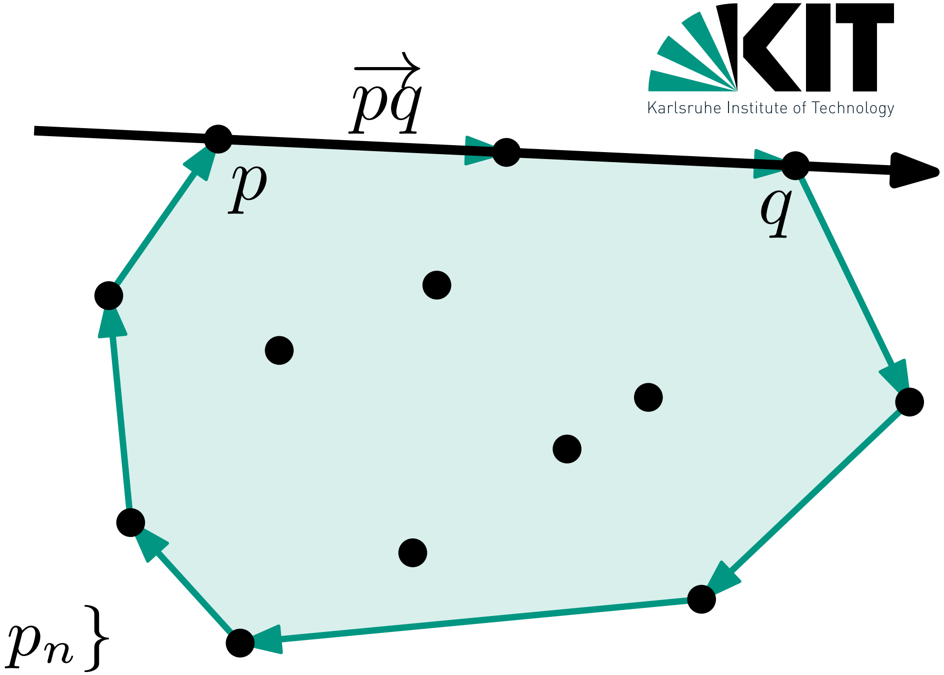
Lemma:

For a set of points $P \subseteq \mathbb{R}^2$, $CH(P)$ is a convex polygon that contains P and whose vertices are in P.

Input: A set of points $P = \{p_1, \ldots, p_n\}$

Output: List of vertices of $CH(P)$ in clockwise order

Algorithmic Approach



Lemma:

For a set of points $P \subseteq \mathbb{R}^2$, $CH(P)$ is a convex polygon that contains P and whose vertices are in P.

Input: A set of points $P = \{p_1, \ldots, p_n\}$

Output: List of vertices of $CH(P)$ in clockwise order

Observation:

(p, q) is an edge of $CH(P) \Leftrightarrow$ each point $r \in P \setminus \{p, q\}$

- strictly right of the oriented line $\overrightarrow{pq}$ or
- on the line segment $\overline{pq}$

A First Algorithm

```
FirstConvexHull($P$)
    $E \leftarrow \emptyset$
    foreach $(p, q) \in P \times P$ with $p \neq q$ do
        valid $\leftarrow$ true
        foreach $r \in P$ do
            if not ($r$ strictly right of $\overrightarrow{pq}$ or $r \in \overline{pq}$) then
                valid $\leftarrow$ false
        if valid then
            $E \leftarrow E \cup \{(p, q)\}$
    construct sorted node list $L$ of $CH(P)$ from $E$
    return $L$
```

A First Algorithm

```
FirstConvexHull($P$)
    $E \leftarrow \emptyset$
    foreach $(p, q) \in P \times P$ with $p \neq q$ do
        valid $\leftarrow$ true
        foreach $r \in P$ do
            if not ($r$ strictly right of $\overrightarrow{pq}$ or $r \in \overline{pq}$) then
                valid $\leftarrow$ false
        if valid then
            $E \leftarrow E \cup \{(p, q)\}$
    construct sorted node list $L$ of $CH(P)$ from $E$
    return $L$
```

Check all possible edges (p, q)

A First Algorithm

```
FirstConvexHull($P$)
    $E \leftarrow \emptyset$
    foreach $(p, q) \in P \times P$ with $p \neq q$ do
        valid ← true
        foreach $r \in P$ do
            if not ($r$ strictly right of $\overrightarrow{pq}$ or $r \in \overline{pq}$) then
                valid ← false
        if valid then
            $E \leftarrow E \cup \{(p, q)\}$
    construct sorted node list $L$ of $CH(P)$ from $E$
    return $L$
```

Check all possible edges (p, q)

Test in $O(1)$ time with

$$\begin{vmatrix} x_r & y_r & 1 \\ x_p & y_p & 1 \\ x_q & y_q & 1 \end{vmatrix} < 0$$

Running Time Analysis

```
FirstConvexHull($P$)
  $E \leftarrow \emptyset$
  foreach $(p, q) \in P \times P$ with $p \neq q$ do
      valid ← true
      foreach $r \in P$ do
          if not ($r$ strictly right of $\overrightarrow{pq}$ or $r \in \overline{pq}$) then
              valid ← false
      if valid then
          $E \leftarrow E \cup \{(p, q)\}$
  construct sorted node list $L$ of $CH(P)$ from $E$
  return $L$
```

Running Time Analysis

FirstConvexHull(P)

```
$E \leftarrow \emptyset$
**foreach** $(p, q) \in P \times P$ with $p \neq q$ **do**
    *valid* $\leftarrow$ *true*
    **foreach** $r \in P$ **do**
        **if not** ($r$ strictly right of $\overrightarrow{pq}$ **or** $r \in \overline{pq}$) **then**    $\Theta(1)$
            *valid* $\leftarrow$ *false*
    **if** *valid* **then**
        $E \leftarrow E \cup \{(p, q)\}$
construct sorted node list $L$ of $CH(P)$ from $E$
**return** $L$
```

Running Time Analysis

FirstConvexHull(P)

```
$E \leftarrow \emptyset$
foreach $(p, q) \in P \times P$ with $p \neq q$ do
    valid $\leftarrow$ true
    foreach $r \in P$ do                                                    $\Theta(n)$
        if not ($r$ strictly right of $\overrightarrow{pq}$ or $r \in \overline{pq}$) then      $\Theta(1)$
            valid $\leftarrow$ false
    if valid then
        $E \leftarrow E \cup \{(p, q)\}$
construct sorted node list $L$ of $CH(P)$ from $E$
return $L$
```

Running Time Analysis

FirstConvexHull(P)

```
$E \leftarrow \emptyset$
foreach $(p, q) \in P \times P$ with $p \neq q$ do                                    $(n^2 - n)\cdot$
    valid $\leftarrow$ true
    foreach $r \in P$ do
        if not ($r$ strictly right of $\overrightarrow{pq}$ or $r \in \overline{pq}$) then     $\Theta(1)$   $\Theta(n)$
            valid $\leftarrow$ false
    if valid then
        $E \leftarrow E \cup \{(p, q)\}$
construct sorted node list $L$ of $CH(P)$ from $E$
return $L$
```

Running Time Analysis

```
FirstConvexHull($P$)
    $E \leftarrow \emptyset$
    foreach $(p, q) \in P \times P$ with $p \neq q$ do                         $(n^2 - n)\cdot$
        valid ← true
        foreach $r \in P$ do
            if not ($r$ strictly right of $\overrightarrow{pq}$ or $r \in \overline{pq}$) then      $\Theta(1)$   $\Theta(n)$   $\Theta(n^3)$
                valid ← false
        if valid then
            $E \leftarrow E \cup \{(p, q)\}$
    construct sorted node list $L$ of $CH(P)$ from $E$
    return $L$
```

Running Time Analysis

FirstConvexHull(P)

```
$E \leftarrow \emptyset$
foreach $(p,q) \in P \times P$ with $p \neq q$ do                     $(n^2 - n)\cdot$
    valid $\leftarrow$ true
    foreach $r \in P$ do
        if not ($r$ strictly right of $\overrightarrow{pq}$ or $r \in \overline{pq}$) then     $\Theta(1)$   $\Theta(n)$   $\Theta(n^3)$
            valid $\leftarrow$ false
    if valid then
        $E \leftarrow E \cup \{(p,q)\}$
construct sorted node list $L$ of $CH(P)$ from $E$
return $L$
```

Question: How do we implement this?

Running Time Analysis

FirstConvexHull(P)

```
$E \leftarrow \emptyset$
foreach $(p,q) \in P \times P$ with $p \neq q$ do          $(n^2 - n)\cdot$
    valid ← true
    foreach $r \in P$ do
        if not ($r$ strictly right of $\overrightarrow{pq}$ or $r \in \overline{pq}$) then
            valid ← false
    if valid then
        $E \leftarrow E \cup \{(p,q)\}$
construct sorted node list $L$ of $CH(P)$ from $E$
return $L$
```

$\Theta(1)$

$\Theta(n)$

$\Theta(n^3)$

$O(n^2)$

Running Time Analysis

```
FirstConvexHull($P$)
  $E \leftarrow \emptyset$
  foreach $(p, q) \in P \times P$ with $p \neq q$ do                      $(n^2 - n)\cdot$
      valid $\leftarrow$ true
      foreach $r \in P$ do
          if not ($r$ strictly right of $\overrightarrow{pq}$ or $r \in \overline{pq}$) then     $\Theta(1)$   $\Theta(n)$   $\Theta(n^3)$
              valid $\leftarrow$ false
      if valid then
          $E \leftarrow E \cup \{(p, q)\}$
  construct sorted node list $L$ of $CH(P)$ from $E$                        $O(n^2)$
  return $L$
```

Lemma: The convex hull of n points in the plane can be computed in $O(n^3)$ time.

Running Time Analysis

```
FirstConvexHull($P$)
    $E \leftarrow \emptyset$
    foreach $(p, q) \in P \times P$ with $p \neq q$ do                    $(n^2 - n)\cdot$    $\Theta(n^3)$
        valid $\leftarrow$ true
        foreach $r \in P$ do                                                 $\Theta(n)$
            if not ($r$ strictly right of $\overrightarrow{pq}$ or $r \in \overline{pq}$) then    $\Theta(1)$
                valid $\leftarrow$ false
        if valid then
            $E \leftarrow E \cup \{(p, q)\}$
    construct sorted node list $L$ of $CH(P)$ from $E$                     $O(n^2)$
    return $L$
```

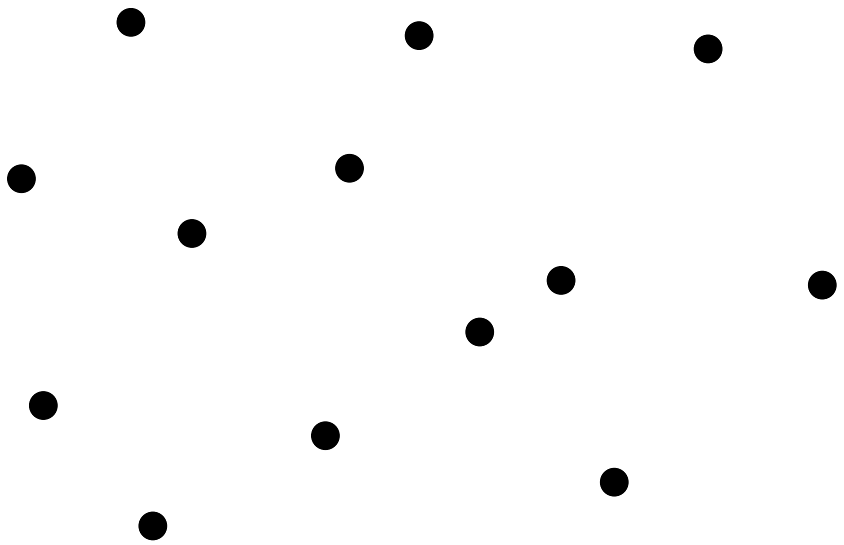
Can we do better?

Lemma: The convex hull of n points in the plane can be computed in $O(n^3)$ time.

Incremental Approach

Idea: For $i = 1, \ldots, n$ compute $CH(P_i)$ where $P_i = \{p_1, \ldots, p_i\}$

Question: Which ordering of the points is useful?

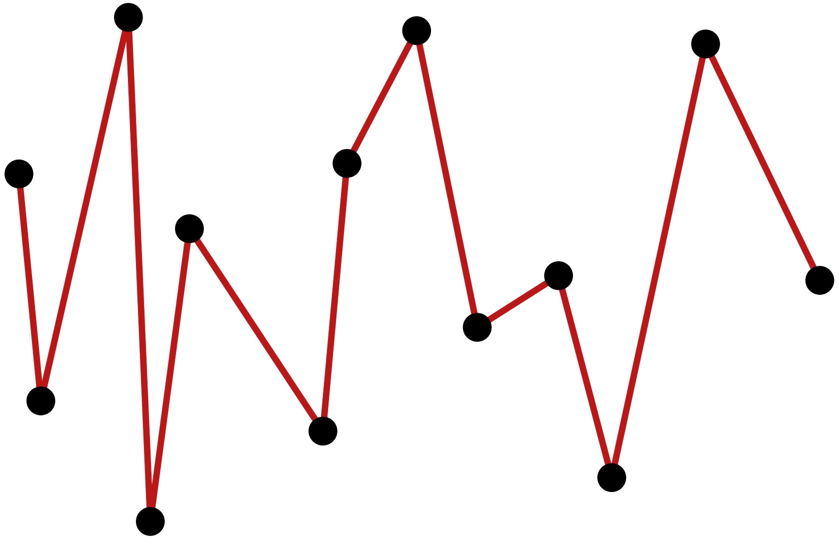


Incremental Approach

Idea: For $i = 1, \ldots, n$ compute $CH(P_i)$ where $P_i = \{p_1, \ldots, p_i\}$

Question: Which ordering of the points is useful?

Answer: From left to right!



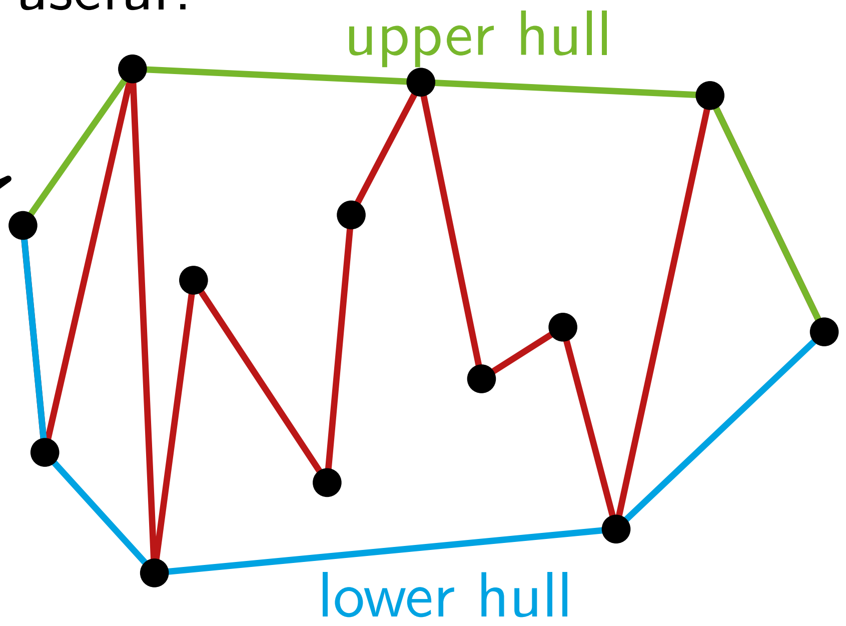
Incremental Approach

Idea: For $i = 1, \ldots, n$ compute $CH(P_i)$ where $P_i = \{p_1, \ldots, p_i\}$

Question: Which ordering of the points is useful?

Answer: From left to right!

Consider the upper and lower hull separately



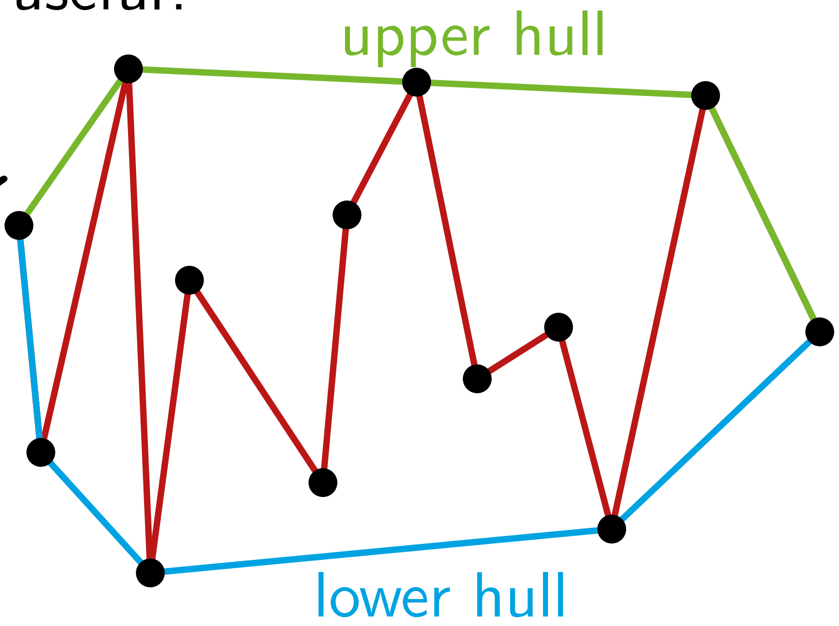
Incremental Approach

Idea: For $i = 1, \ldots, n$ compute $CH(P_i)$ where $P_i = \{p_1, \ldots, p_i\}$

Question: Which ordering of the points is useful?

Answer: From left to right!

Consider the upper and lower hull separately



```
UpperConvexHull($P$)
    $\langle p_1, p_2, \ldots, p_n \rangle \leftarrow$ sort $P$ from left to right
    $L \leftarrow \langle p_1, p_2 \rangle$
    for $i \leftarrow 3$ to $n$ do
        $L$.append($p_i$)
        ?
    return $L$
```

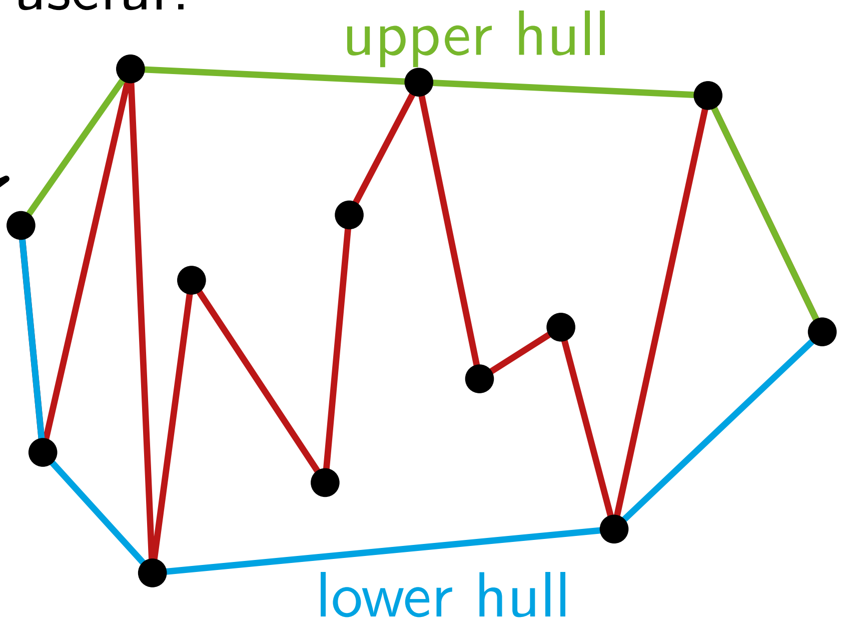
Incremental Approach

Idea: For $i = 1, \ldots, n$ compute $CH(P_i)$ where $P_i = \{p_1, \ldots, p_i\}$

Question: Which ordering of the points is useful?

Answer: From left to right!

Consider the upper and lower hull separately



```
UpperConvexHull($P$)
  $\langle p_1, p_2, \ldots, p_n \rangle \leftarrow$ sort $P$ from left to right
  $L \leftarrow \langle p_1, p_2 \rangle$
  for $i \leftarrow 3$ to $n$ do
      $L$.append($p_i$)
      while $|L| > 2$ and the last 3 points in $L$ do not form right turn do
          remove the second-to-last point in $L$
  return $L$
```

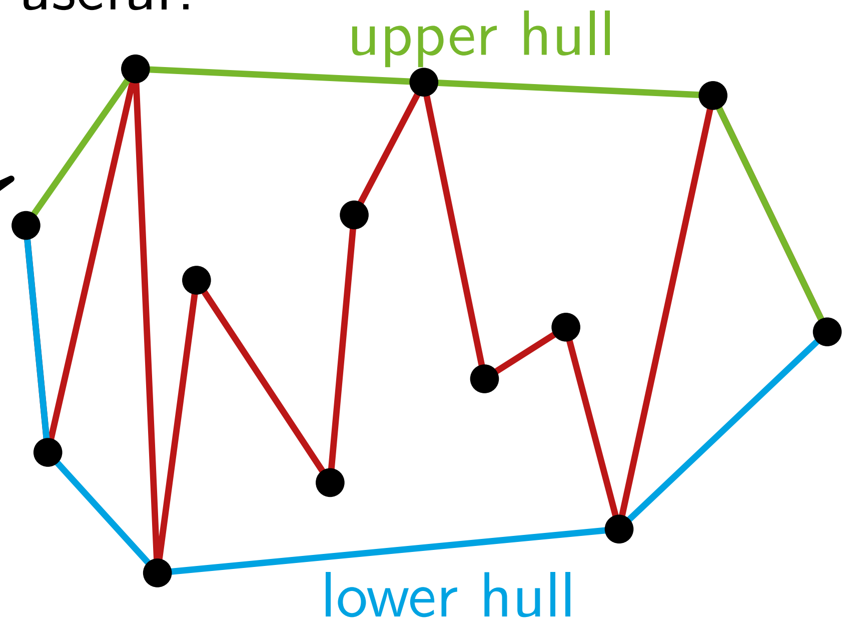
Incremental Approach

Idea: For $i = 1, \dots, n$ compute $CH(P_i)$ where $P_i = \{p_1, \dots, p_i\}$

Question: Which ordering of the points is useful?

Answer: From left to right!

Consider the upper and lower hull separately



```
UpperConvexHull($P$)
    $\langle p_1, p_2, \dots, p_n \rangle \leftarrow$ sort $P$ from left to right
    $L \leftarrow \langle p_1, p_2 \rangle$
    for $i \leftarrow 3$ to $n$ do
        $L$.append($p_i$)
        while $|L| > 2$ and the last 3 points in $L$ do not form right turn do
            remove the second-to-last point in $L$
    return $L$
```

lower hull is handled similarly!

Running Time Analysis

```
UpperConvexHull($P$)
    $\langle p_1, p_2, \ldots, p_n \rangle \leftarrow$ sort $P$ from right to left
    $L \leftarrow \langle p_1, p_2 \rangle$
    for $i \leftarrow 3$ to $n$ do
        $L$.append($p_i$)
        while $|L| > 2$ and last 3 points in $L$ do not form right turn do
            remove the second-to-last point from $L$
    return $L$
```

Running Time Analysis

```
UpperConvexHull($P$)
    $\langle p_1, p_2, \ldots, p_n \rangle \leftarrow$ sort $P$ from right to left        $O(n \log n)$
    $L \leftarrow \langle p_1, p_2 \rangle$
    for $i \leftarrow 3$ to $n$ do
        $L$.append($p_i$)
        while $|L| > 2$ and last 3 points in $L$ do not form right turn do
            remove the second-to-last point from $L$
    return $L$
```

Running Time Analysis

```
UpperConvexHull($P$)
    $\langle p_1, p_2, \ldots, p_n \rangle \leftarrow$ sort $P$ from right to left        $O(n \log n)$
    $L \leftarrow \langle p_1, p_2 \rangle$
    for $i \leftarrow 3$ to $n$ do                                                        $(n-2)\cdot$
        $L$.append($p_i$)
        while $|L| > 2$ and last 3 points in $L$ do not form right turn do
            remove the second-to-last point from $L$                                      ?
    return $L$
```

Running Time Analysis

```
UpperConvexHull($P$)
    $\langle p_1, p_2, \ldots, p_n \rangle \leftarrow$ sort $P$ from right to left          $O(n \log n)$
    $L \leftarrow \langle p_1, p_2 \rangle$
    for $i \leftarrow 3$ to $n$ do                                          $(n-2)\cdot$
        $L$.append($p_i$)
        while $|L| > 2$ and last 3 points in $L$ do not form right turn do
            remove the second-to-last point from $L$                     ?
    return $L$
```

Amortized Analysis

- Each point is inserted into L exactly once
- A point in L is removed at most once from L
- $\Rightarrow$ Running time of the **for** loop including the **while** loop is $O(n)$

Running Time Analysis

```
UpperConvexHull($P$)
  $\langle p_1, p_2, \dots, p_n \rangle \leftarrow$ sort $P$ from right to left      $O(n \log n)$
  $L \leftarrow \langle p_1, p_2 \rangle$
  for $i \leftarrow 3$ to $n$ do                                                     ~~$(n-2)\cdot$~~
      $L$.append($p_i$)                                                                   $O(n)$
      while $|L| > 2$ and last 3 points in $L$ do not form right turn do
          remove the second-to-last point from $L$                                ~~?~~
  return $L$
```

Amortized Analysis

- Each point is inserted into L exactly once
- A point in L is removed at most once from L
- $\Rightarrow$ Running time of the **for** loop including the **while** loop is $O(n)$

Running Time Analysis

```
UpperConvexHull($P$)
    $\langle p_1, p_2, \ldots, p_n \rangle \leftarrow$ sort $P$ from right to left        $O(n \log n)$
    $L \leftarrow \langle p_1, p_2 \rangle$
    for $i \leftarrow 3$ to $n$ do                                                    ~~$(n-2)\cdot$~~
        $L$.append($p_i$)                                                            $O(n)$
        while $|L| > 2$ and last 3 points in $L$ do not form right turn do
            remove the second-to-last point from $L$                                  ~~?~~
    return $L$
```

Amortized Analysis

- Each point is inserted into L exactly once
- A point in L is removed at most once from L
- $\Rightarrow$ Running time of the **for** loop including the **while** loop is $O(n)$

Theorem 1: The convex hull of n points in the plane can be computed in $O(n \log n)$ time. → *Graham's Scan*.

Alternative Approach: Gift Wrapping

Idea: Begin with a point p_1 of $CH(P)$, then find the next edge of $CH(P)$ in clockwise order.

Alternative Approach: Gift Wrapping

Idea: Begin with a point p_1 of $CH(P)$, then find the next edge of $CH(P)$ in clockwise order.

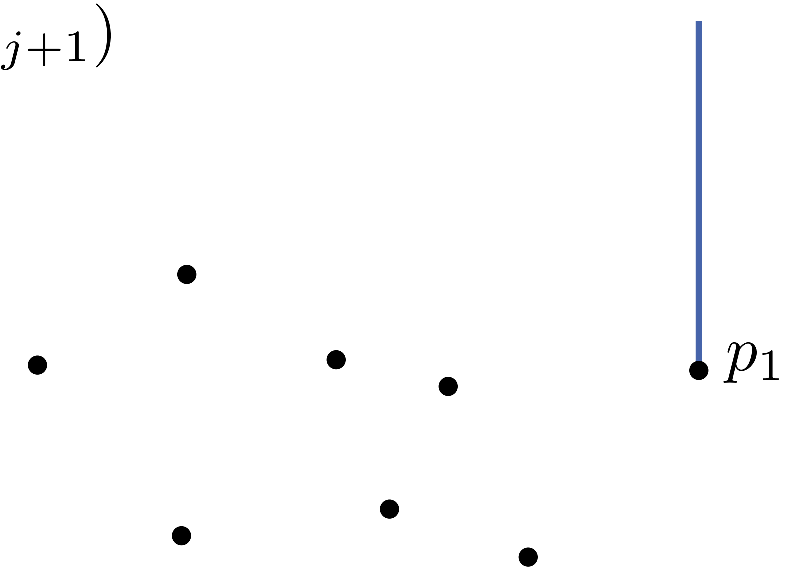
GiftWrapping(P)

```
$p_1 = (x_1, y_1) \leftarrow$ rightmost point in $P$; $p_0 \leftarrow (x_1, \infty)$; $j \leftarrow 1$
while true do
    $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1}, p_j, q \mid q \in P \setminus \{p_{j-1}, p_j\}\}$
    if $p_{j+1} = p_1$ then break else $j \leftarrow j + 1$
return $(p_1, \ldots, p_{j+1})$
```

Alternative Approach: Gift Wrapping

Idea: Begin with a point p_1 of $CH(P)$, then find the next edge of $CH(P)$ in clockwise order.

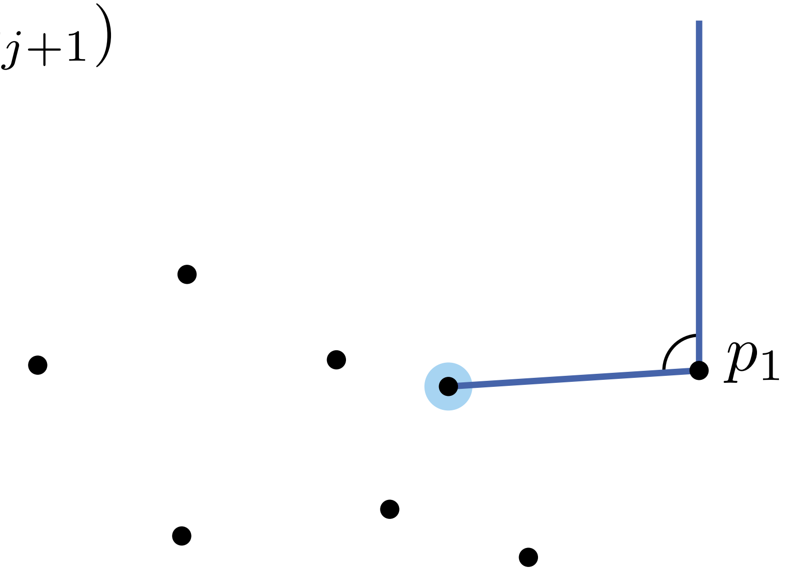
```
GiftWrapping($P$)
    $p_1 = (x_1, y_1) \leftarrow$ rightmost point in $P$; $p_0 \leftarrow (x_1, \infty)$; $j \leftarrow 1$
    while true do
        $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1}, p_j, q \mid q \in P \setminus \{p_{j-1}, p_j\}\}$
        if $p_{j+1} = p_1$ then break else $j \leftarrow j + 1$
    return $(p_1, \ldots, p_{j+1})$
```



Alternative Approach: Gift Wrapping

Idea: Begin with a point p_1 of $CH(P)$, then find the next edge of $CH(P)$ in clockwise order.

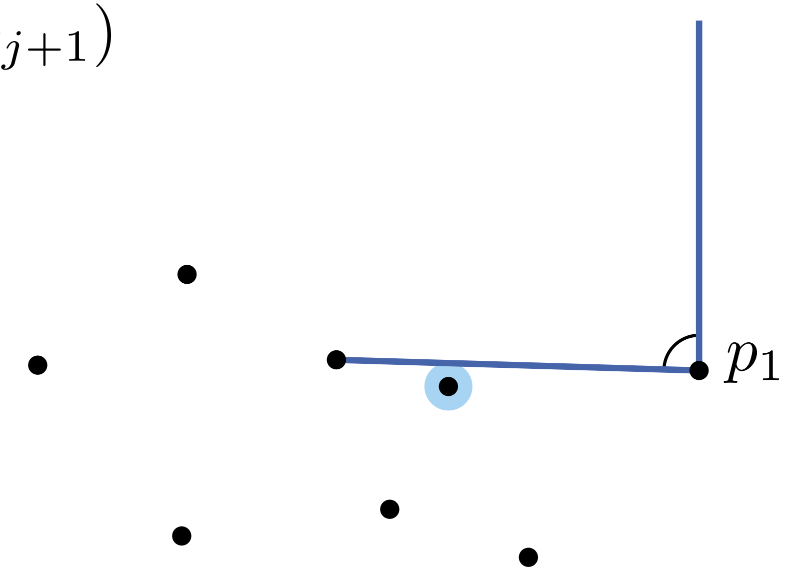
```
GiftWrapping($P$)
  $p_1 = (x_1, y_1) \leftarrow$ rightmost point in $P$; $p_0 \leftarrow (x_1, \infty)$; $j \leftarrow 1$
  while true do
      $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1}, p_j, q \mid q \in P \setminus \{p_{j-1}, p_j\}\}$
      if $p_{j+1} = p_1$ then break else $j \leftarrow j + 1$
  return $(p_1, \dots, p_{j+1})$
```



Alternative Approach: Gift Wrapping

Idea: Begin with a point p_1 of $CH(P)$, then find the next edge of $CH(P)$ in clockwise order.

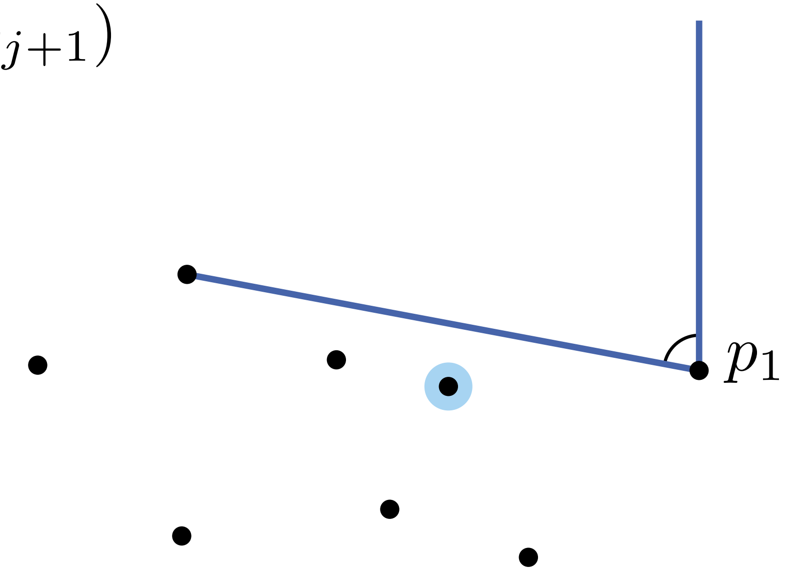
```
GiftWrapping($P$)
  $p_1 = (x_1, y_1) \leftarrow$ rightmost point in $P$; $p_0 \leftarrow (x_1, \infty)$; $j \leftarrow 1$
  while true do
      $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1}, p_j, q \mid q \in P \setminus \{p_{j-1}, p_j\}\}$
      if $p_{j+1} = p_1$ then break else $j \leftarrow j + 1$
  return $(p_1, \ldots, p_{j+1})$
```



Alternative Approach: Gift Wrapping

Idea: Begin with a point p_1 of $CH(P)$, then find the next edge of $CH(P)$ in clockwise order.

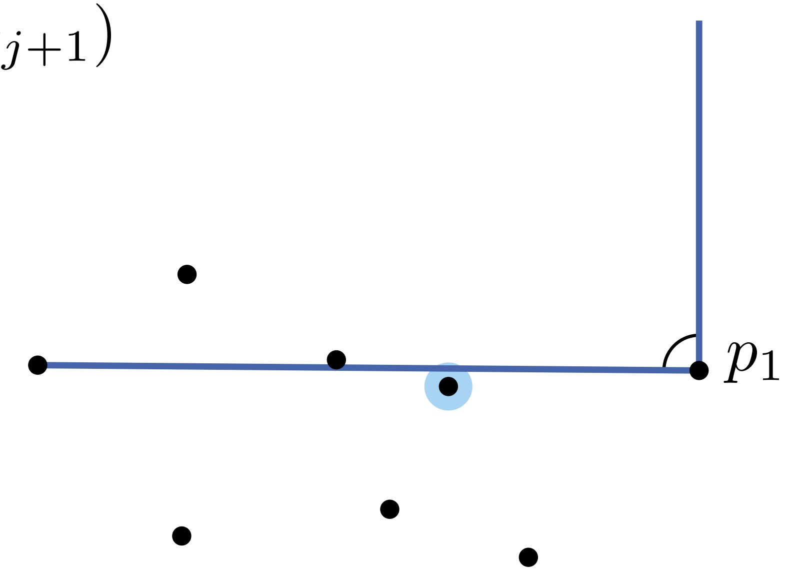
```
GiftWrapping($P$)
  $p_1 = (x_1, y_1) \leftarrow$ rightmost point in $P$; $p_0 \leftarrow (x_1, \infty)$; $j \leftarrow 1$
  while true do
      $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1}, p_j, q \mid q \in P \setminus \{p_{j-1}, p_j\}\}$
      if $p_{j+1} = p_1$ then break else $j \leftarrow j + 1$
  return $(p_1, \ldots, p_{j+1})$
```



Alternative Approach: Gift Wrapping

Idea: Begin with a point p_1 of $CH(P)$, then find the next edge of $CH(P)$ in clockwise order.

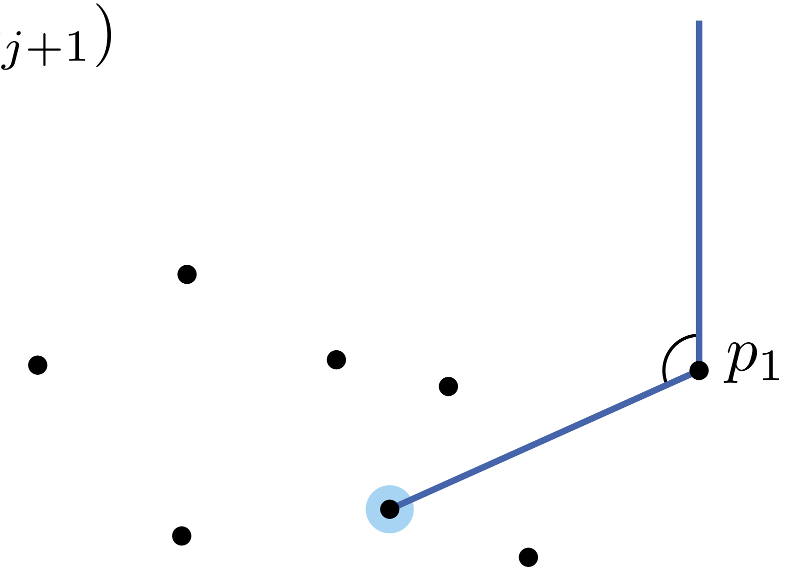
```
GiftWrapping($P$)
  $p_1 = (x_1, y_1) \leftarrow$ rightmost point in $P$; $p_0 \leftarrow (x_1, \infty)$; $j \leftarrow 1$
  while true do
      $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1}, p_j, q \mid q \in P \setminus \{p_{j-1}, p_j\}\}$
      if $p_{j+1} = p_1$ then break else $j \leftarrow j + 1$
  return $(p_1, \dots, p_{j+1})$
```



Alternative Approach: Gift Wrapping

Idea: Begin with a point p_1 of $CH(P)$, then find the next edge of $CH(P)$ in clockwise order.

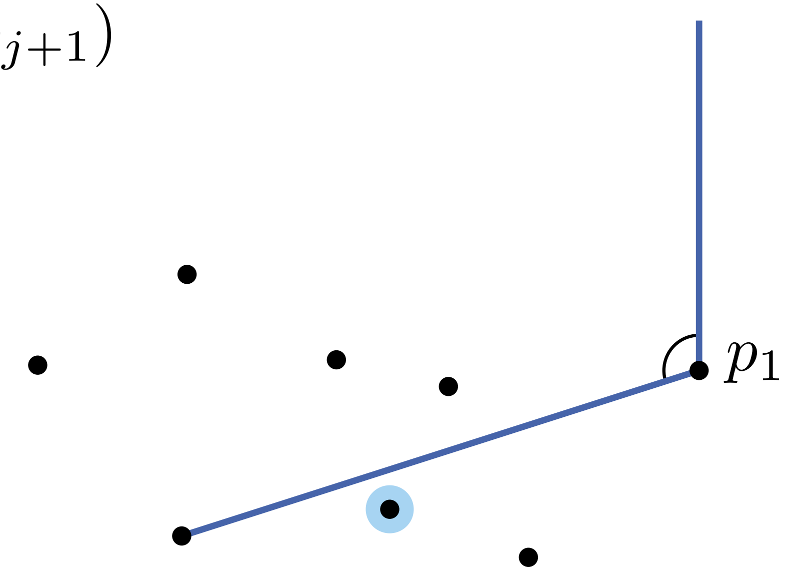
```
GiftWrapping($P$)
  $p_1 = (x_1, y_1) \leftarrow$ rightmost point in $P$; $p_0 \leftarrow (x_1, \infty)$; $j \leftarrow 1$
  while true do
      $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1}, p_j, q \mid q \in P \setminus \{p_{j-1}, p_j\}\}$
      if $p_{j+1} = p_1$ then break else $j \leftarrow j + 1$
  return $(p_1, \dots, p_{j+1})$
```



Alternative Approach: Gift Wrapping

Idea: Begin with a point p_1 of $CH(P)$, then find the next edge of $CH(P)$ in clockwise order.

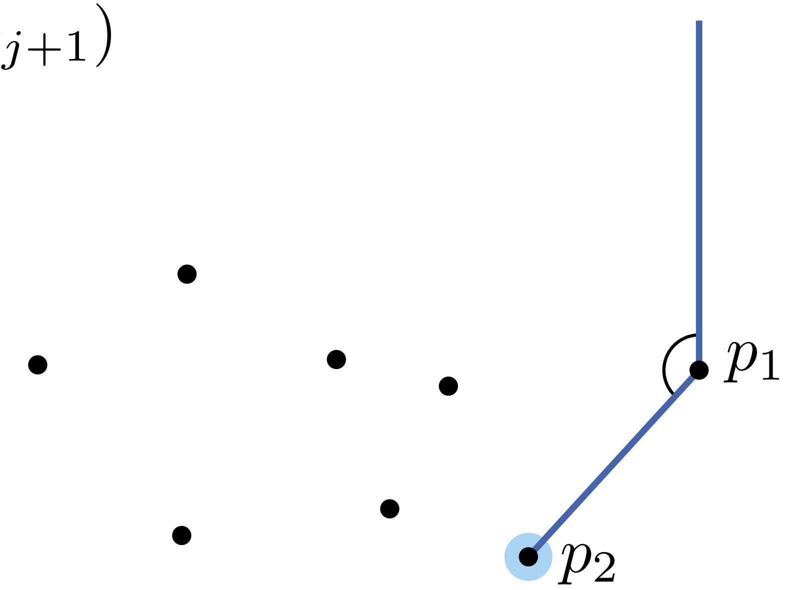
```
GiftWrapping($P$)
  $p_1 = (x_1, y_1) \leftarrow$ rightmost point in $P$; $p_0 \leftarrow (x_1, \infty)$; $j \leftarrow 1$
  while true do
      $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1}, p_j, q \mid q \in P \setminus \{p_{j-1}, p_j\}\}$
      if $p_{j+1} = p_1$ then break else $j \leftarrow j + 1$
  return $(p_1, \ldots, p_{j+1})$
```



Alternative Approach: Gift Wrapping

Idea: Begin with a point p_1 of $CH(P)$, then find the next edge of $CH(P)$ in clockwise order.

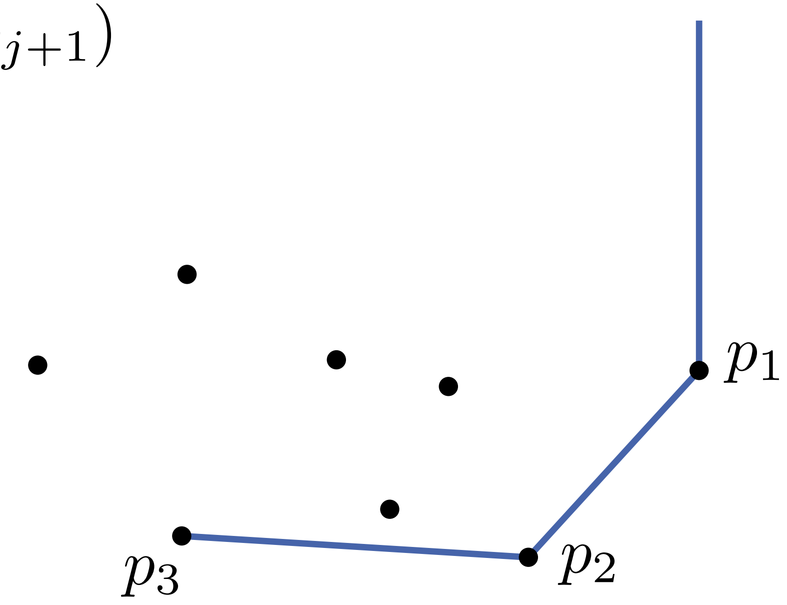
```
GiftWrapping($P$)
  $p_1 = (x_1, y_1) \leftarrow$ rightmost point in $P$; $p_0 \leftarrow (x_1, \infty)$; $j \leftarrow 1$
  while true do
      $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1}, p_j, q \mid q \in P \setminus \{p_{j-1}, p_j\}\}$
      if $p_{j+1} = p_1$ then break else $j \leftarrow j + 1$
  return $(p_1, \ldots, p_{j+1})$
```



Alternative Approach: Gift Wrapping

Idea: Begin with a point p_1 of $CH(P)$, then find the next edge of $CH(P)$ in clockwise order.

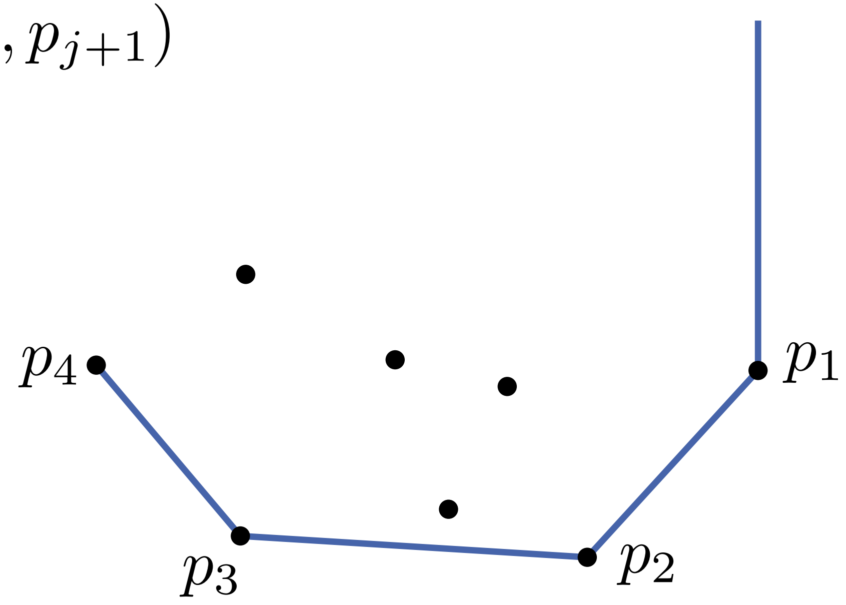
```
GiftWrapping($P$)
  $p_1 = (x_1, y_1) \leftarrow$ rightmost point in $P$; $p_0 \leftarrow (x_1, \infty)$; $j \leftarrow 1$
  while true do
      $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1}, p_j, q \mid q \in P \setminus \{p_{j-1}, p_j\}\}$
      if $p_{j+1} = p_1$ then break else $j \leftarrow j + 1$
  return $(p_1, \ldots, p_{j+1})$
```



Alternative Approach: Gift Wrapping

Idea: Begin with a point p_1 of $CH(P)$, then find the next edge of $CH(P)$ in clockwise order.

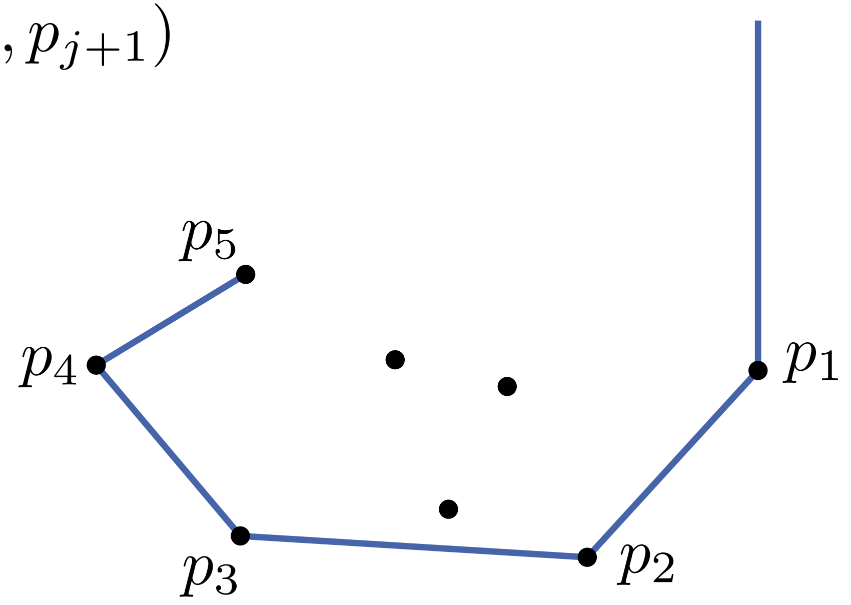
```
GiftWrapping($P$)
  $p_1 = (x_1, y_1) \leftarrow$ rightmost point in $P$; $p_0 \leftarrow (x_1, \infty)$; $j \leftarrow 1$
  while true do
      $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1}, p_j, q \mid q \in P \setminus \{p_{j-1}, p_j\}\}$
      if $p_{j+1} = p_1$ then break else $j \leftarrow j + 1$
  return $(p_1, \ldots, p_{j+1})$
```



Alternative Approach: Gift Wrapping

Idea: Begin with a point p_1 of $CH(P)$, then find the next edge of $CH(P)$ in clockwise order.

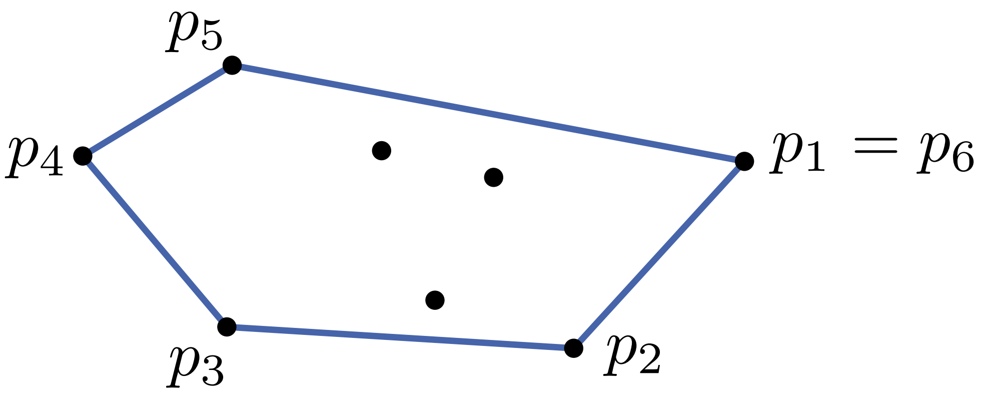
```
GiftWrapping($P$)
  $p_1 = (x_1, y_1) \leftarrow$ rightmost point in $P$; $p_0 \leftarrow (x_1, \infty)$; $j \leftarrow 1$
  while true do
      $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1}, p_j, q \mid q \in P \setminus \{p_{j-1}, p_j\}\}$
      if $p_{j+1} = p_1$ then break else $j \leftarrow j + 1$
  return $(p_1, \ldots, p_{j+1})$
```



Alternative Approach: Gift Wrapping

Idea: Begin with a point p_1 of $CH(P)$, then find the next edge of $CH(P)$ in clockwise order.

```
GiftWrapping($P$)
  $p_1 = (x_1, y_1) \leftarrow$ rightmost point in $P$; $p_0 \leftarrow (x_1, \infty)$; $j \leftarrow 1$
  while true do
      $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1}, p_j, q \mid q \in P \setminus \{p_{j-1}, p_j\}\}$
      if $p_{j+1} = p_1$ then break else $j \leftarrow j + 1$
  return $(p_1, \ldots, p_{j+1})$
```

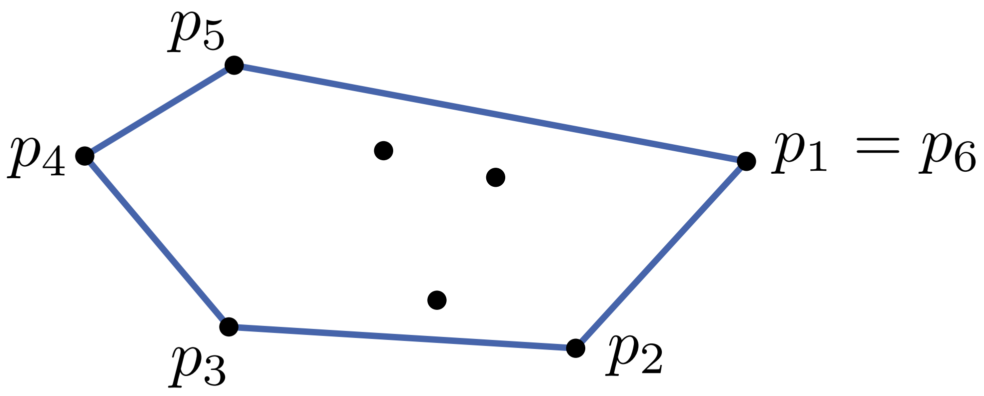


Alternative Approach: Gift Wrapping

Idea: Begin with a point p_1 of $CH(P)$, then find the next edge of $CH(P)$ in clockwise order.

GiftWrapping(P)

```
$p_1 = (x_1, y_1) \leftarrow$ rightmost point in $P$; $p_0 \leftarrow (x_1, \infty)$; $j \leftarrow 1$
while true do
    $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1}, p_j, q \mid q \in P \setminus \{p_{j-1}, p_j\}\}$
    if $p_{j+1} = p_1$ then break else $j \leftarrow j + 1$
return $(p_1, \ldots, p_{j+1})$
```

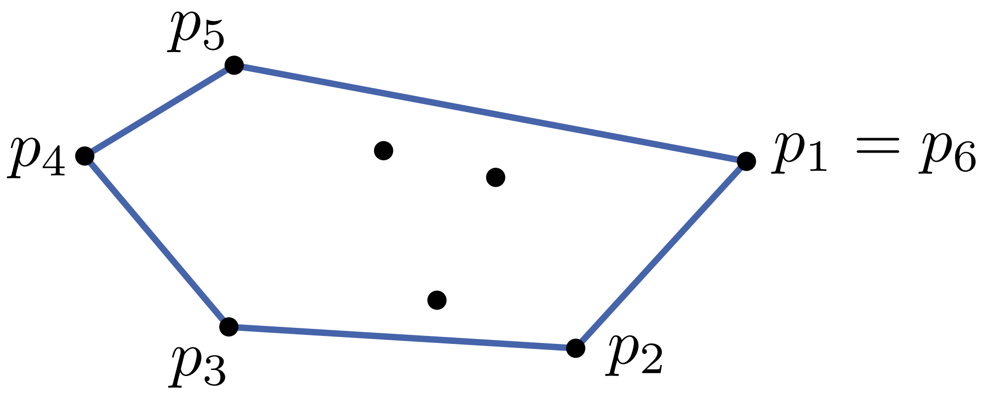


Alternative Approach: Gift Wrapping

Idea: Begin with a point p_1 of $CH(P)$, then find the next edge of $CH(P)$ in clockwise order.

GiftWrapping(P)

```
$p_1 = (x_1, y_1) \leftarrow$ rightmost point in $P$; $p_0 \leftarrow (x_1, \infty)$; $j \leftarrow 1$
while true do
    $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1}, p_j, q \mid q \in P \setminus \{p_{j-1}, p_j\}\}$    $O(n)$
    if $p_{j+1} = p_1$ then break else $j \leftarrow j + 1$
return $(p_1, \ldots, p_{j+1})$
```

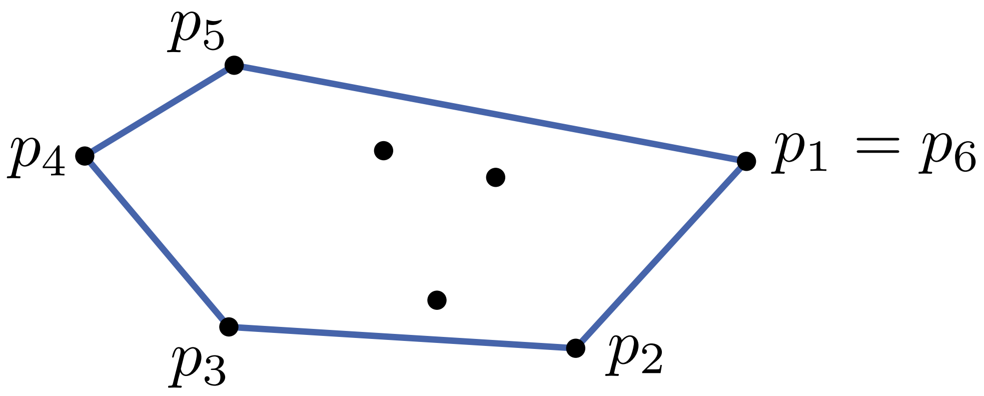


Alternative Approach: Gift Wrapping

Idea: Begin with a point p_1 of $CH(P)$, then find the next edge of $CH(P)$ in clockwise order.

GiftWrapping(P)

```
$p_1 = (x_1, y_1) \leftarrow$ rightmost point in $P$; $p_0 \leftarrow (x_1, \infty)$; $j \leftarrow 1$
while true do                                                              $O(h)$
    $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1}, p_j, q \mid q \in P \setminus \{p_{j-1}, p_j\}\}$    $O(n)$
    if $p_{j+1} = p_1$ then break else $j \leftarrow j + 1$
return $(p_1, \ldots, p_{j+1})$
```

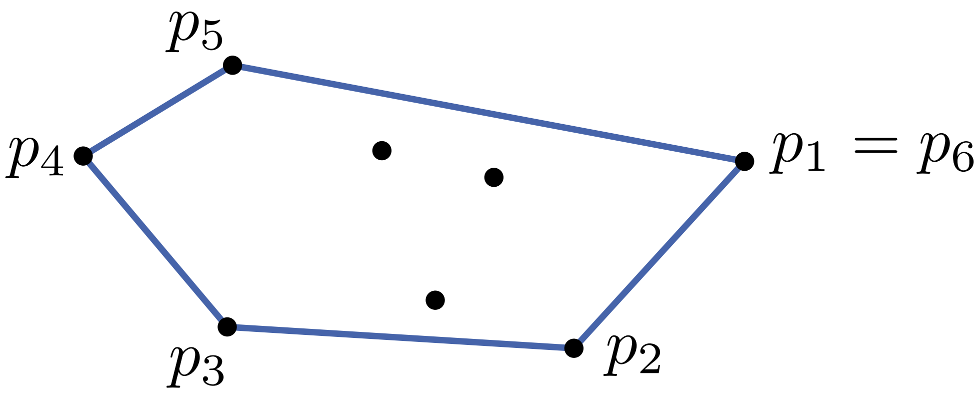


Alternative Approach: Gift Wrapping

Idea: Begin with a point p_1 of $CH(P)$, then find the next edge of $CH(P)$ in clockwise order.

GiftWrapping(P)

```
$p_1 = (x_1, y_1) \leftarrow$ rightmost point in $P$; $p_0 \leftarrow (x_1, \infty)$; $j \leftarrow 1$        $O(n)$
while true do                                                                                          $O(h)$
    $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1}, p_j, q \mid q \in P \setminus \{p_{j-1}, p_j\}\}$     $O(n)$    $O(n \cdot h)$
    if $p_{j+1} = p_1$ then break else $j \leftarrow j + 1$
return $(p_1, \ldots, p_{j+1})$
```



Alternative Approach: Gift Wrapping

Idea: Begin with a point p_1 of $CH(P)$, then find the next edge of $CH(P)$ in clockwise order.

```
GiftWrapping($P$)
$p_1 = (x_1, y_1) \leftarrow$ rightmost point in $P$; $p_0 \leftarrow (x_1, \infty)$; $j \leftarrow 1$          $O(n)$
while true do                                                                  $O(h)$
    $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1}, p_j, q \mid q \in P \setminus \{p_{j-1}, p_j\}\}$      $O(n)$        $O(n \cdot h)$
    if $p_{j+1} = p_1$ then break else $j \leftarrow j + 1$
return $(p_1, \ldots, p_{j+1})$
```

Theorem 2: The convex hull $CH(P)$ of n points P in $\mathbb{R}^2$ can be computed in $O(n \cdot h)$ time using *Gift Wrapping* (also called *Jarvis' March*), where $h = |CH(P)|$.

Alternative Approach: Gift Wrapping

Idea: Begin with a point p_1 of $CH(P)$, then find the next edge of $CH(P)$ in clockwise order.

GiftWrapping(P)

```
$p_1 = (x_1, y_1) \leftarrow$ rightmost point in $P$; $p_0 \leftarrow (x_1, \infty)$; $j \leftarrow 1$        $O(n)$
while true do                                                                                                $O(h)$
    $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1}, p_j, q \mid q \in P \setminus \{p_{j-1}, p_j\}\}$          $O(n)$      $O(n \cdot h)$
    if $p_{j+1} = p_1$ then break else $j \leftarrow j + 1$
return $(p_1, \dots, p_{j+1})$
```

Theorem 2: The convex hull $CH(P)$ of n points P in $\mathbb{R}^2$ can be computed in $O(n \cdot h)$ time using *Gift Wrapping* (also called *Jarvis' March*), where $h = |CH(P)|$.

→ more on that in the exercises!

Comparison

Which algorithm is better?

- Graham's Scan: $O(n \log n)$ time
- Jarvis' March: $O(n \cdot h)$ time

Comparison

Which algorithm is better?

- Graham's Scan: $O(n \log n)$ time
- Jarvis' March: $O(n \cdot h)$ time

It depends on how large $CH(P)$ is!

Comparison

Which algorithm is better?

- Graham's Scan: $O(n \log n)$ time
- Jarvis' March: $O(n \cdot h)$ time

It depends on how large $CH(P)$ is!

Idea: Combine the two approaches into an optimal algorithm!

Chan's Algorithm

Suppose we know h:

```
ChanHull($P$,$h$)
    Divide $P$ into sets $P_i$ with $\leq h$ nodes
    for $i$ from 1 to $\lceil n/h \rceil$ do
        Compute with GrahamScan $CH(P_i)$
    $p_1 = (x_1, y_1) \leftarrow$ rightmost point in $P$
    $p_0 \leftarrow (x_1, \infty)$
    for $j = 1$ to $h$ do
        for $i = 1$ to $\lceil n/h \rceil$ do
            $q_i \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in P_i \setminus \{p_{j-1}, p_j\}\}$
        $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in \{q_1, \dots, q_{\lceil n/h \rceil}\}\}$
    return $(p_1, \dots, p_h)$
```

Chan's Algorithm

Suppose we know h:

ChanHull(P,h)

```
Divide $P$ into sets $P_i$ with $\leq h$ nodes
for $i$ from 1 to $\lceil n/h \rceil$ do
    Compute with GrahamScan $CH(P_i)$
$p_1 = (x_1, y_1) \leftarrow$ rightmost point in $P$
$p_0 \leftarrow (x_1, \infty)$
for $j = 1$ to $h$ do
    for $i = 1$ to $\lceil n/h \rceil$ do
        $q_i \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in P_i \setminus \{p_{j-1}, p_j\}\}$
    $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in \{q_1, \ldots, q_{\lceil n/h \rceil}\}\}$
return $(p_1, \ldots, p_h)$
```

Chan's Algorithm

Suppose we know h:

ChanHull(P,h)

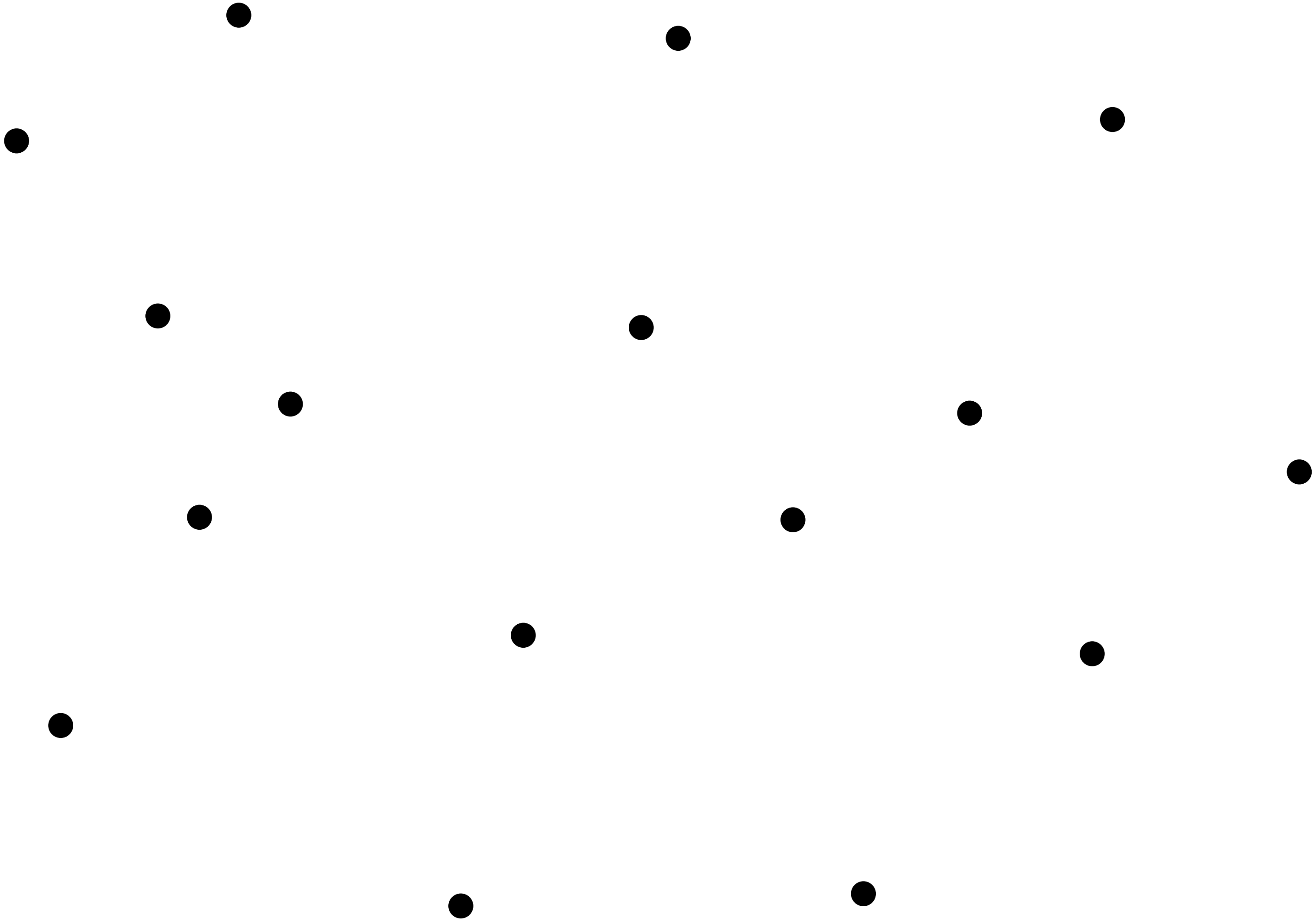
```
Divide $P$ into sets $P_i$ with $\leq h$ nodes
for $i$ from 1 to $\lceil n/h \rceil$ do
    Compute with GrahamScan $CH(P_i)$
$p_1 = (x_1, y_1) \leftarrow$ rightmost point in $P$
$p_0 \leftarrow (x_1, \infty)$
for $j = 1$ to $h$ do
    for $i = 1$ to $\lceil n/h \rceil$ do
        $q_i \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in P_i \setminus \{p_{j-1}, p_j\}\}$
    $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in \{q_1, \ldots, q_{\lceil n/h \rceil}\}\}$
return $(p_1, \ldots, p_h)$
```

GrahamScan

Gift Wrapping

Example

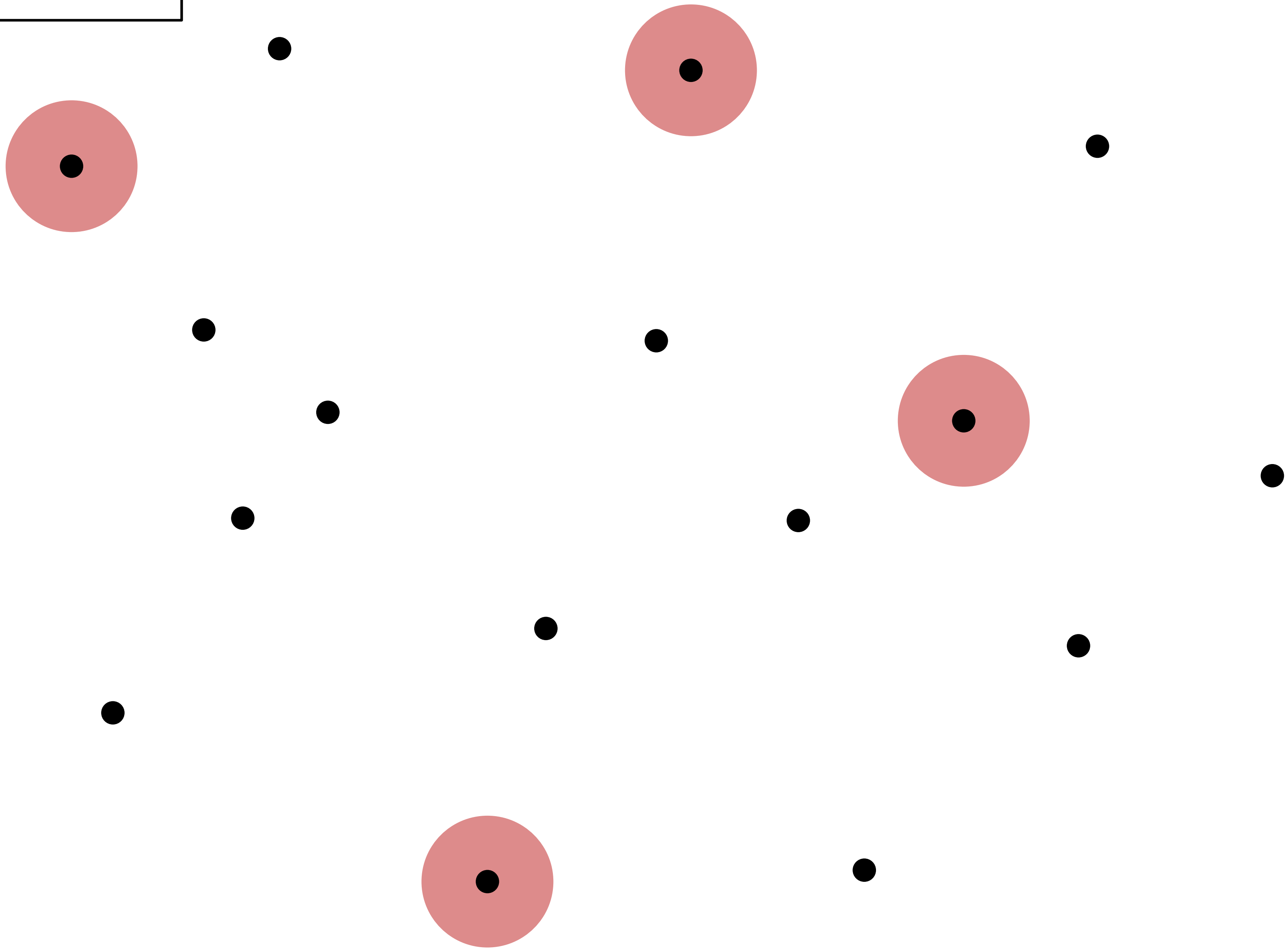
GrahamScan



n = 16

Example

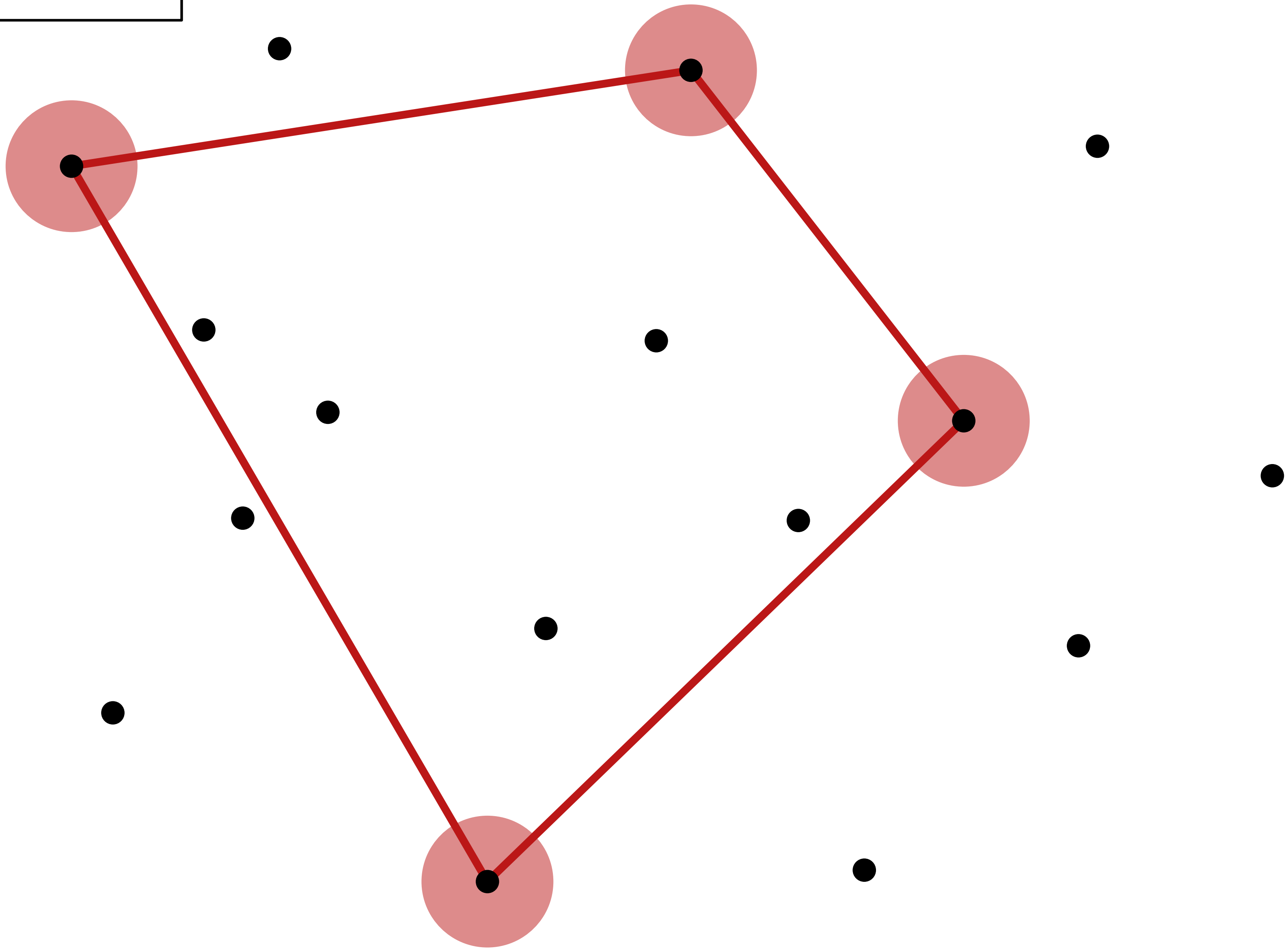
GrahamScan



$n = 16$

Example

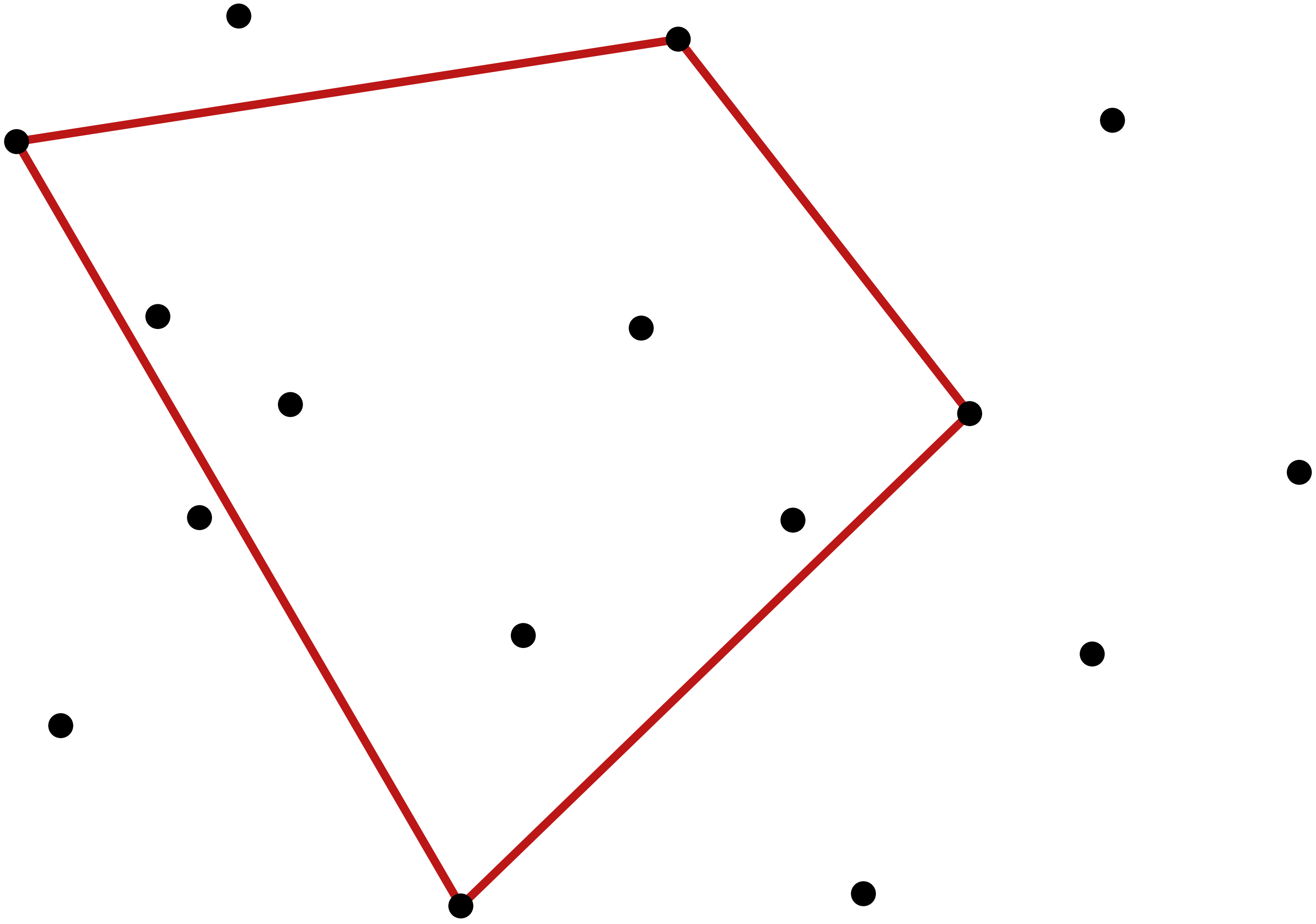
GrahamScan



n = 16

Example

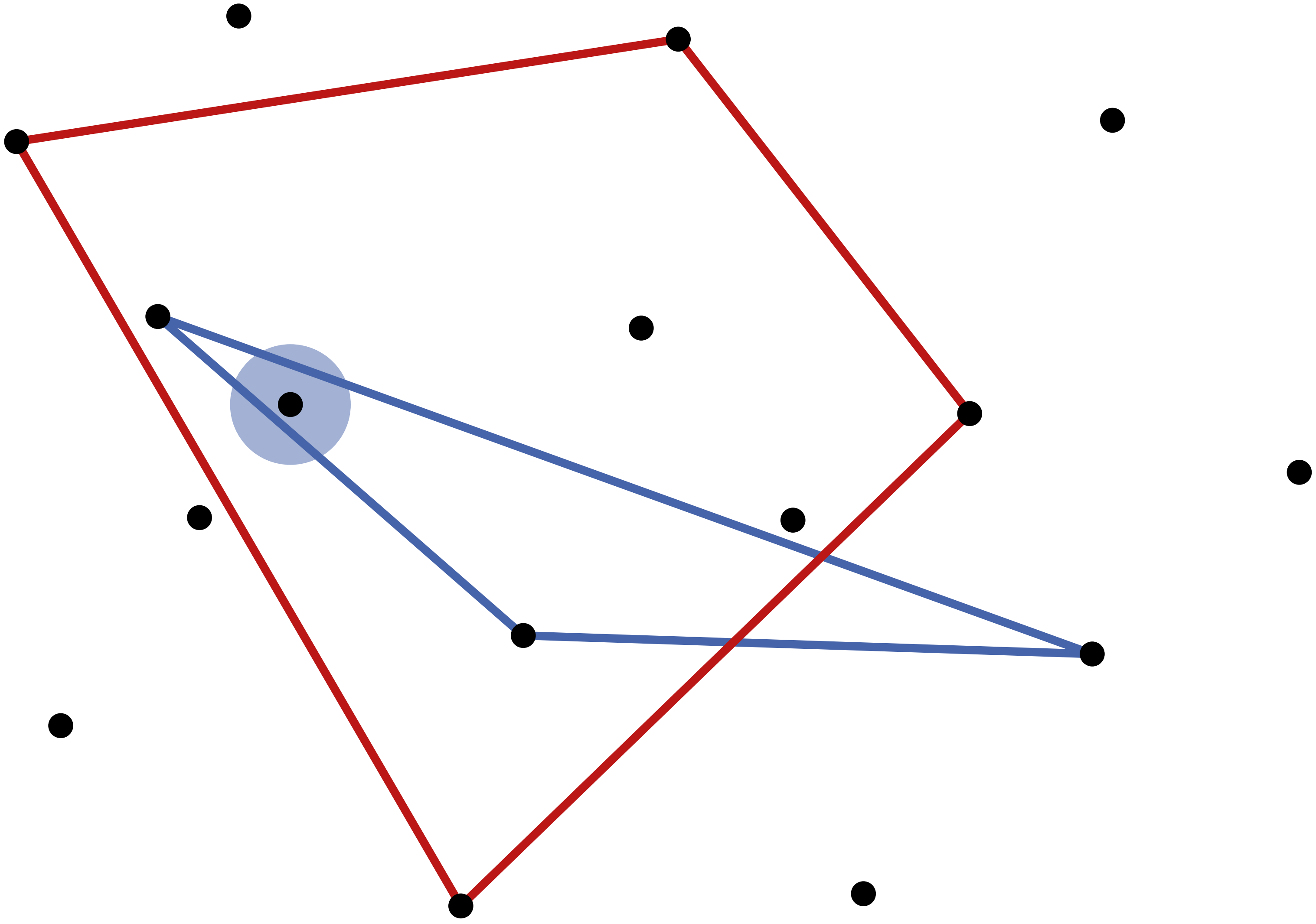
GrahamScan



n = 16

Example

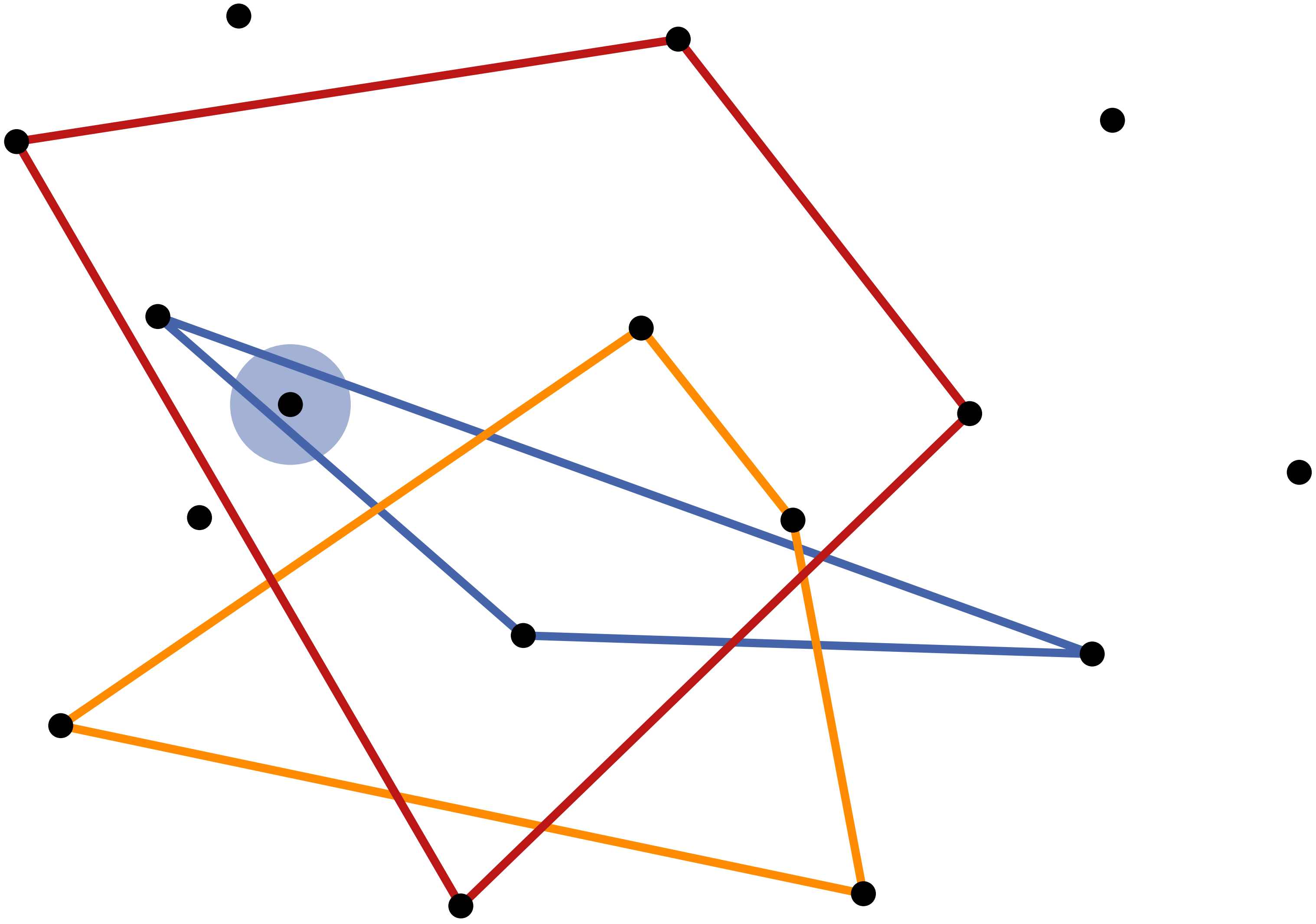
GrahamScan



$n = 16$

Example

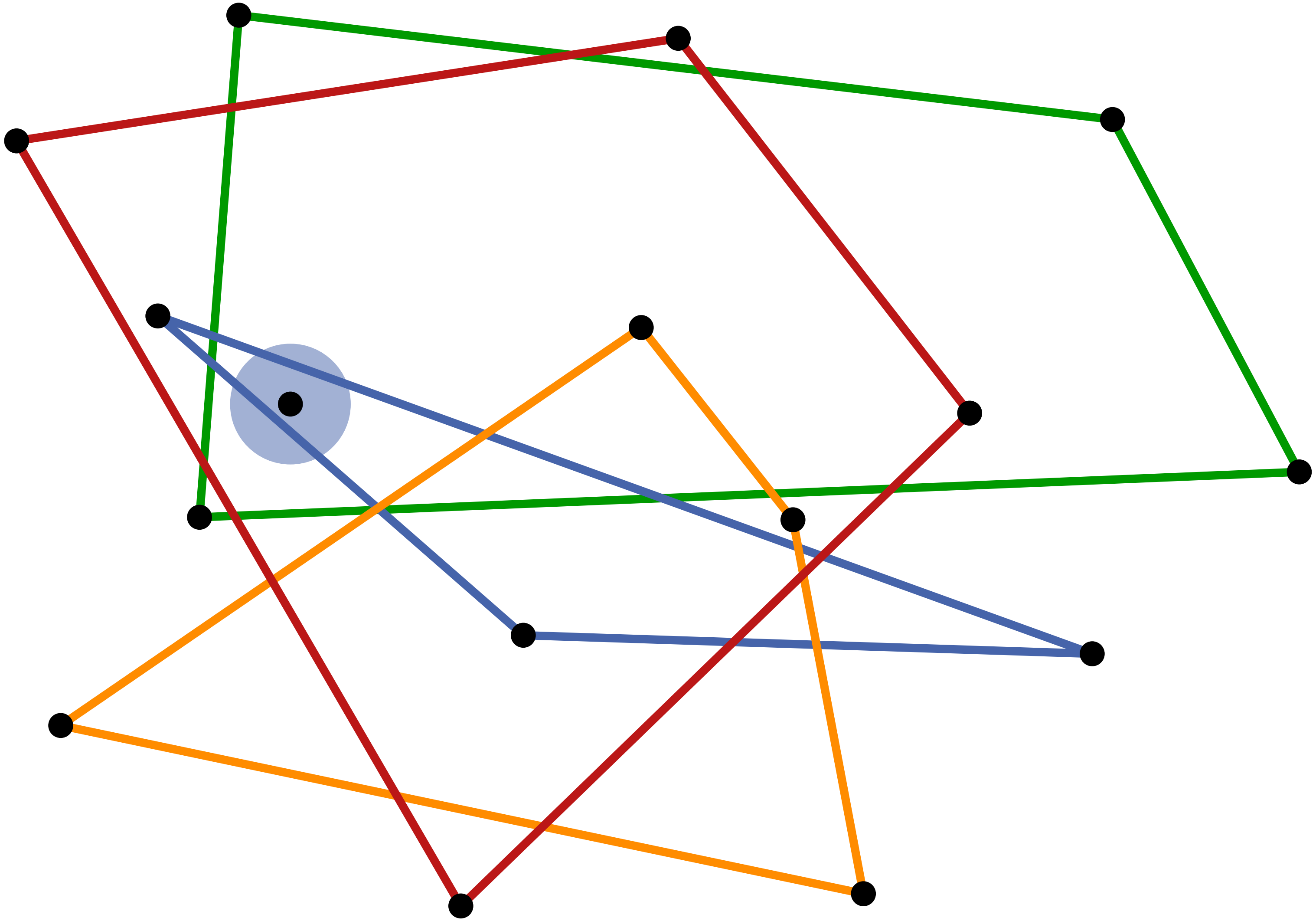
GrahamScan



n = 16

Example

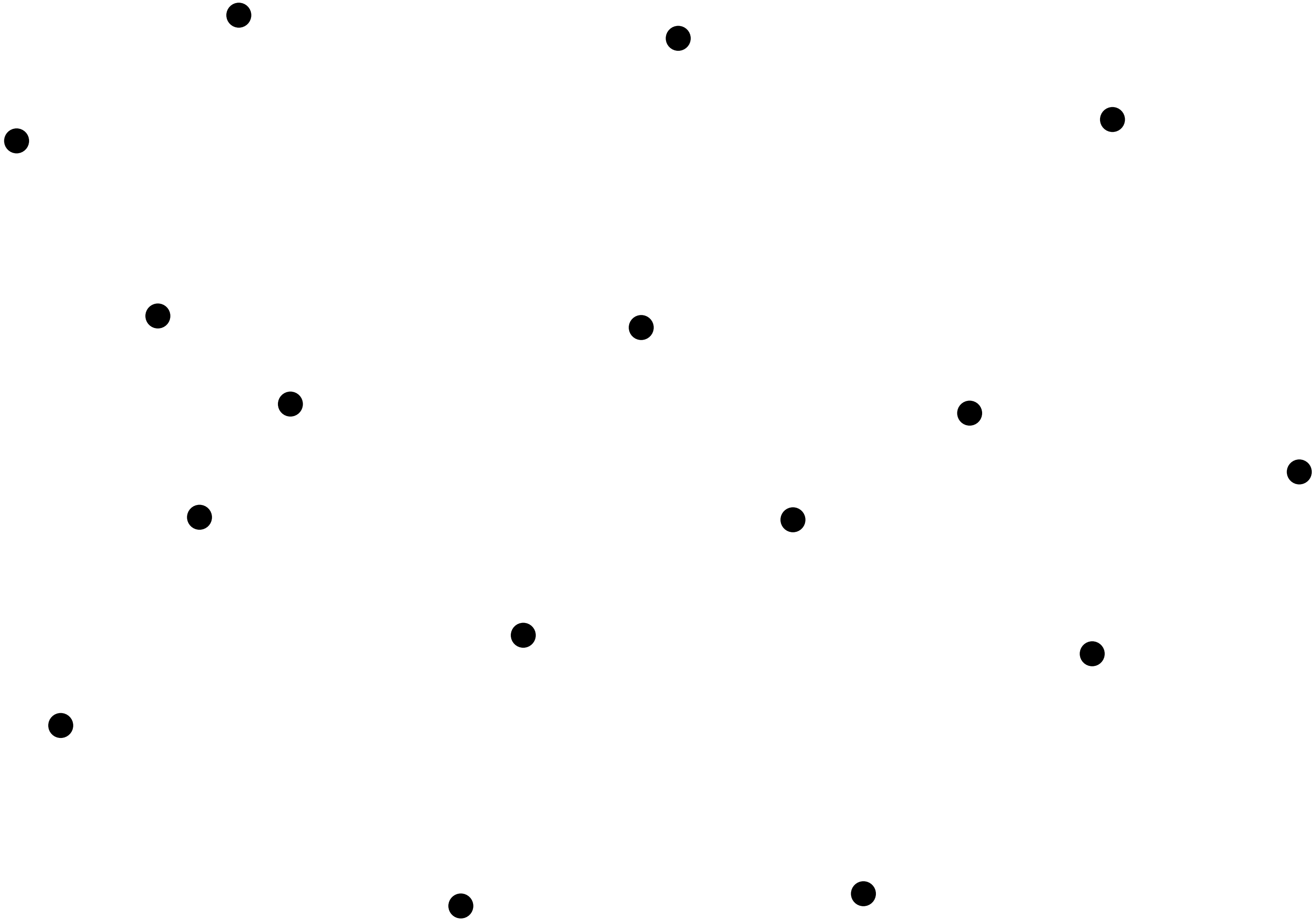
GrahamScan



n = 16

Example

GrahamScan

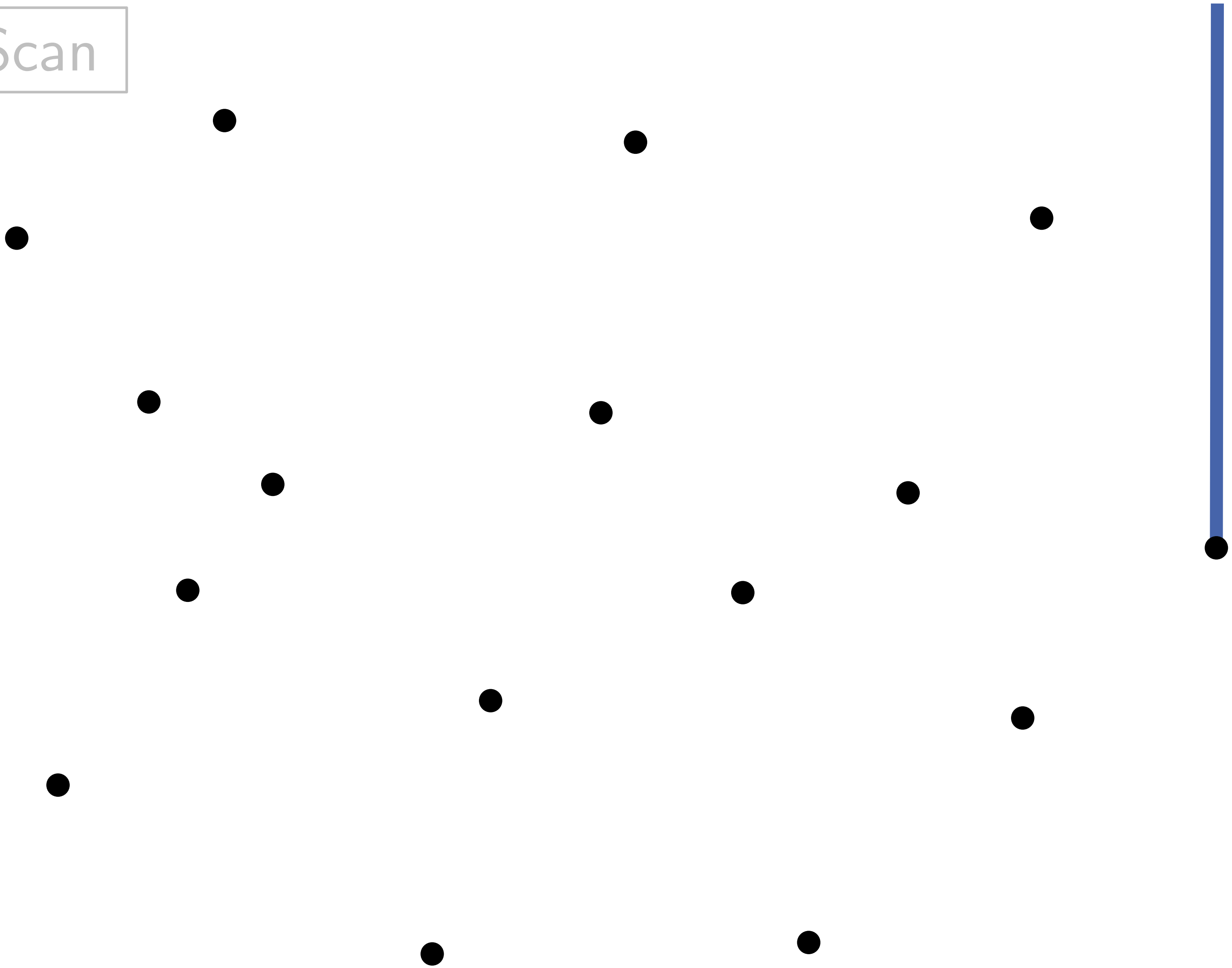


n = 16

Gift Wrapping

Example

GrahamScan

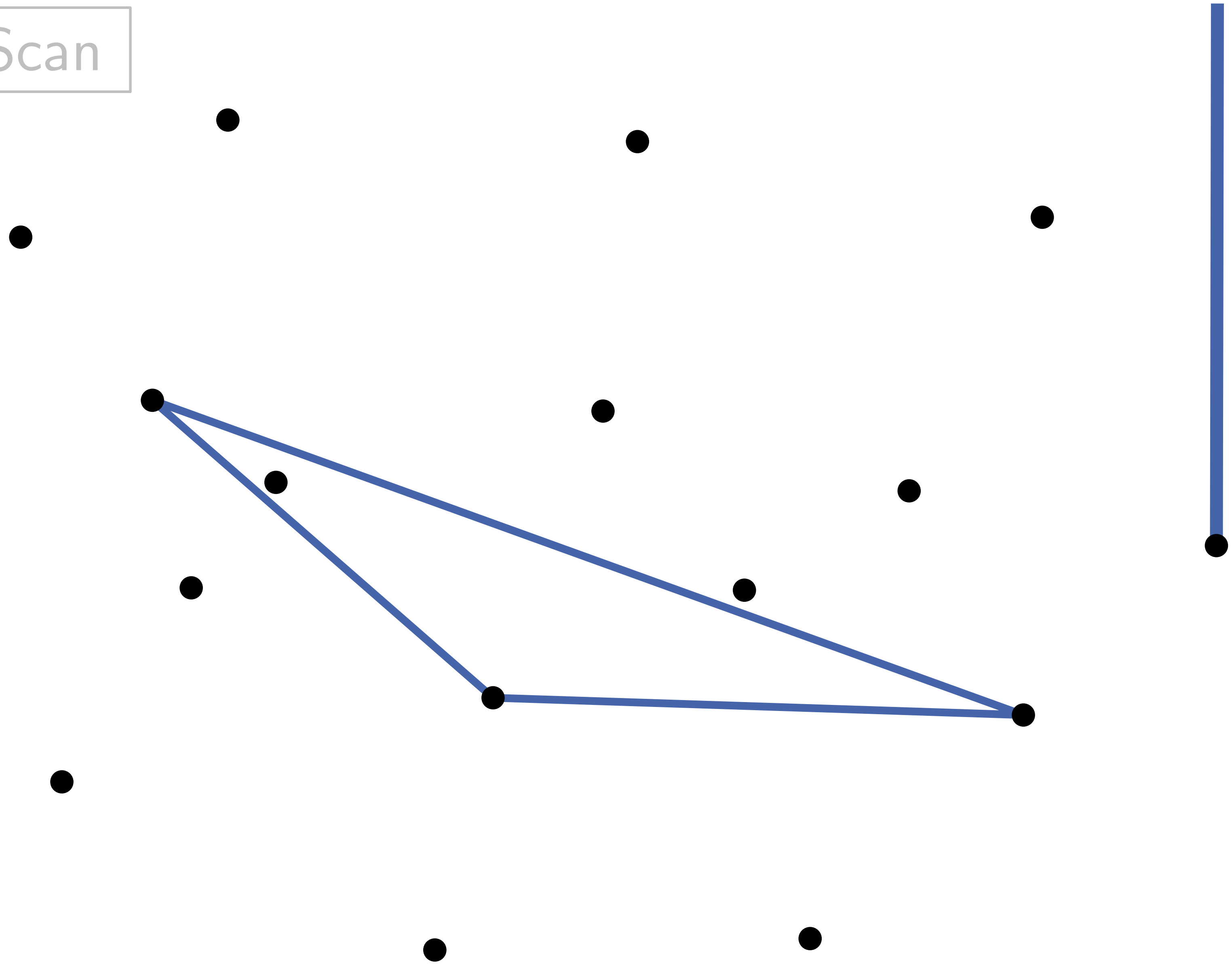


n = 16

Gift Wrapping

Example

GrahamScan

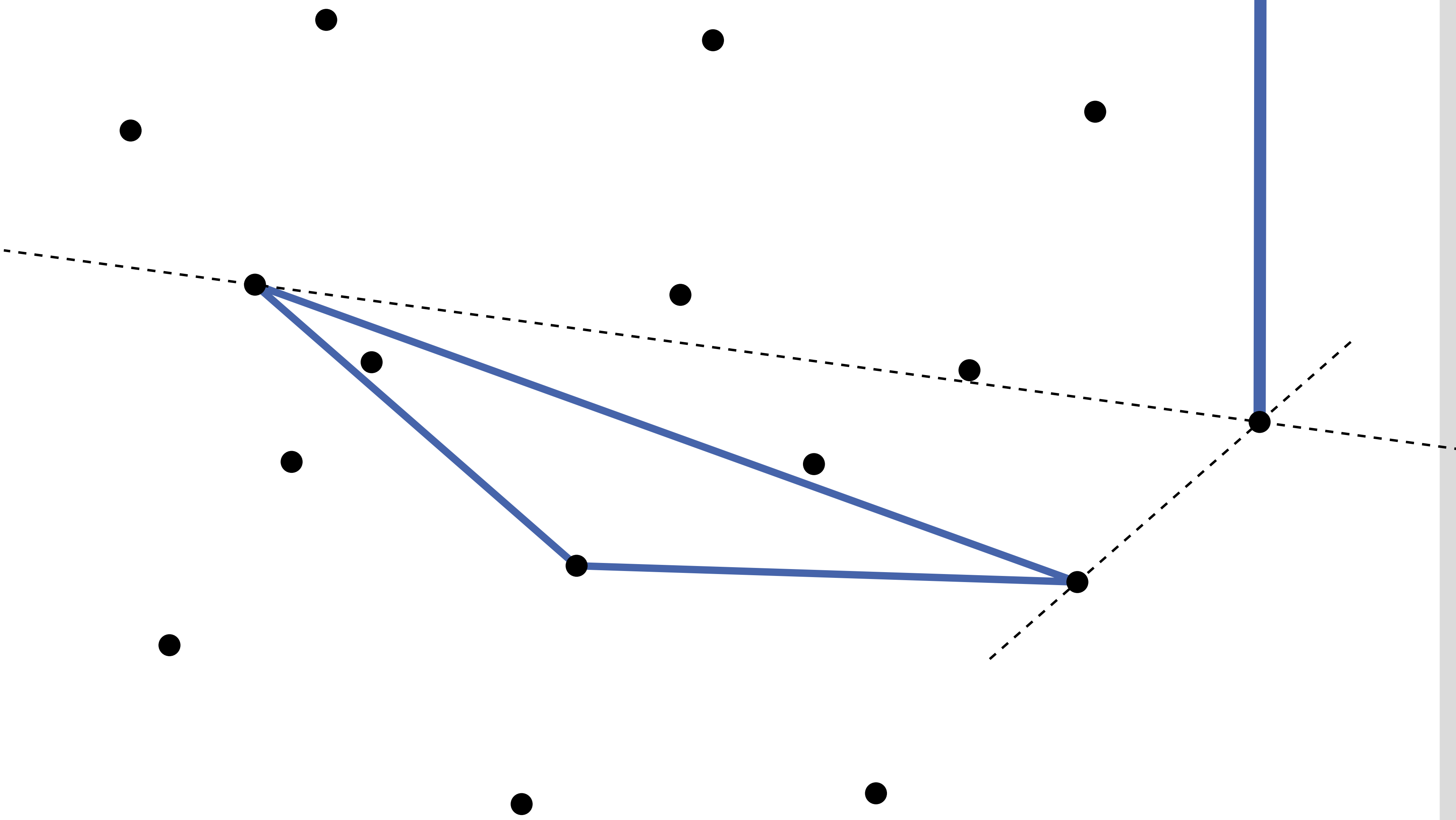


n = 16

Gift Wrapping

Example

GrahamScan

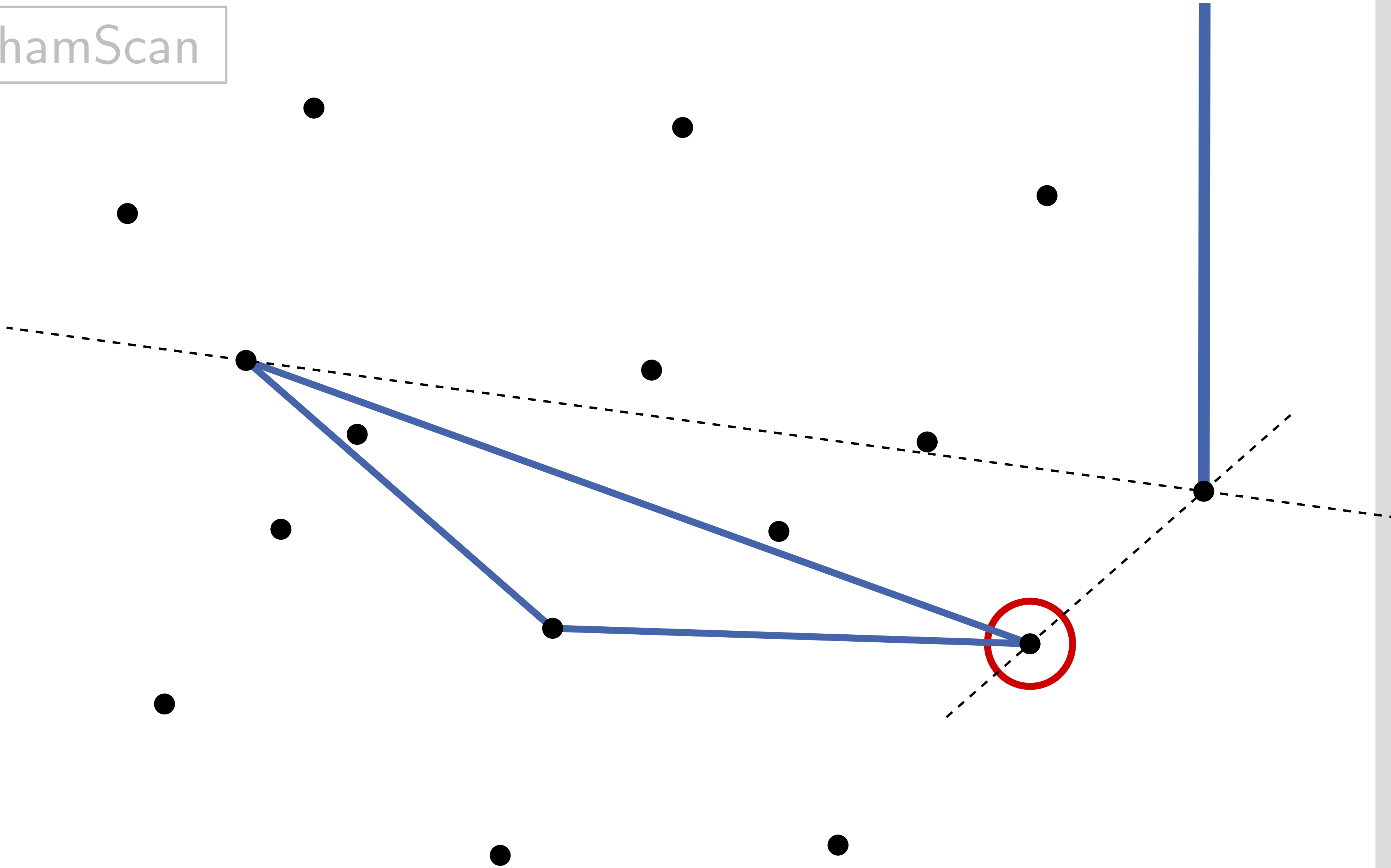


n = 16

Gift Wrapping

Example

GrahamScan

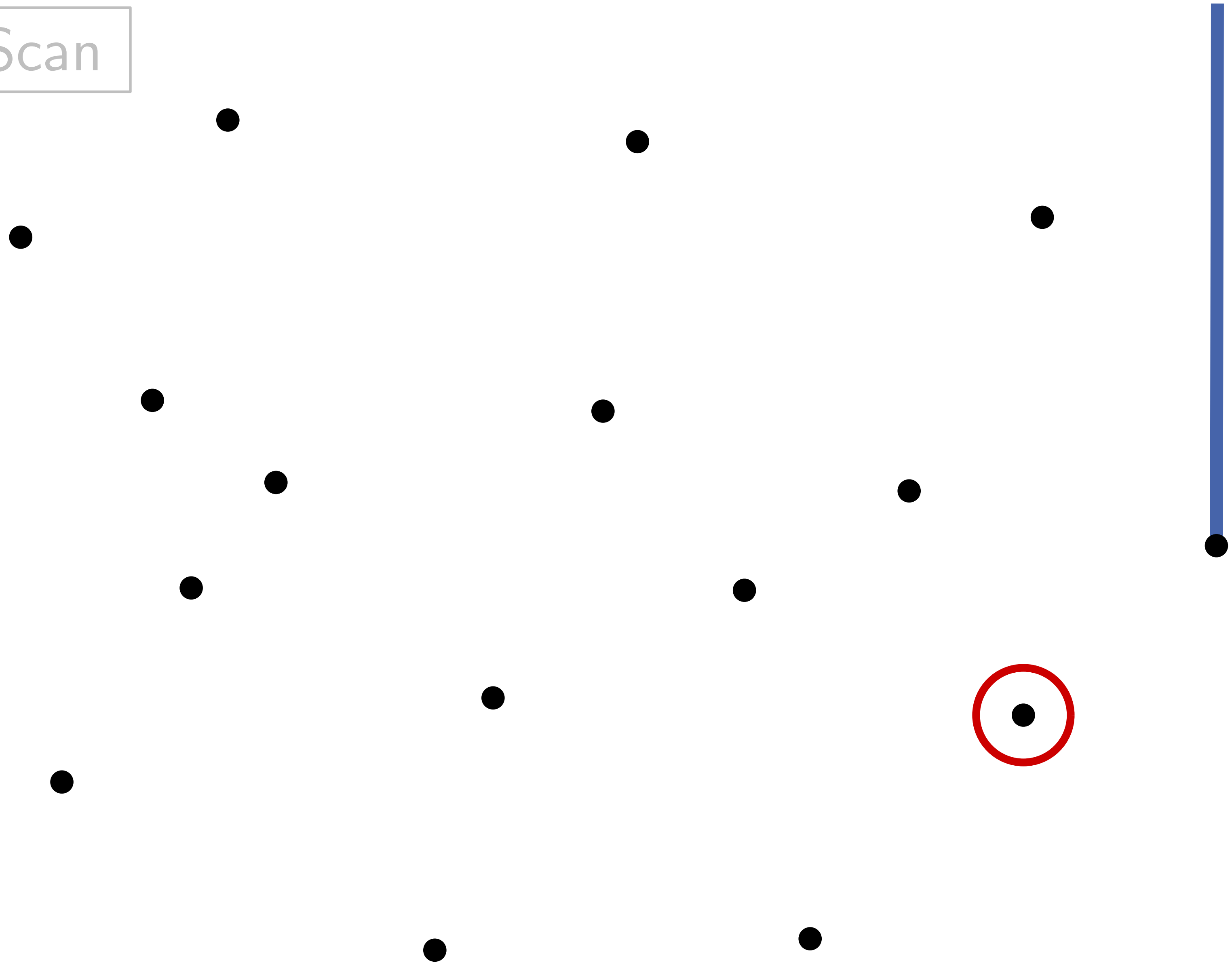


n = 16

Gift Wrapping

Example

GrahamScan

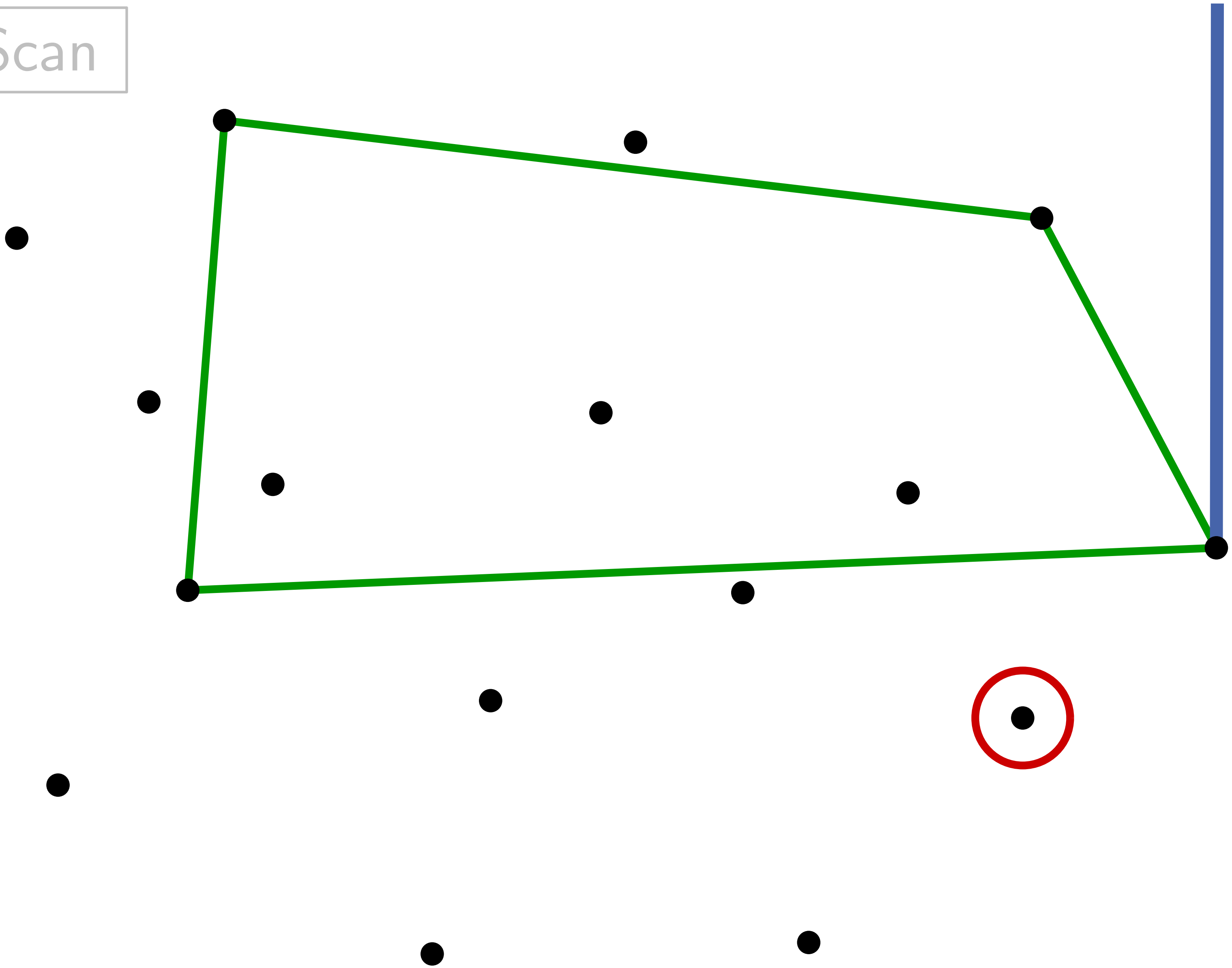


n = 16

Gift Wrapping

Example

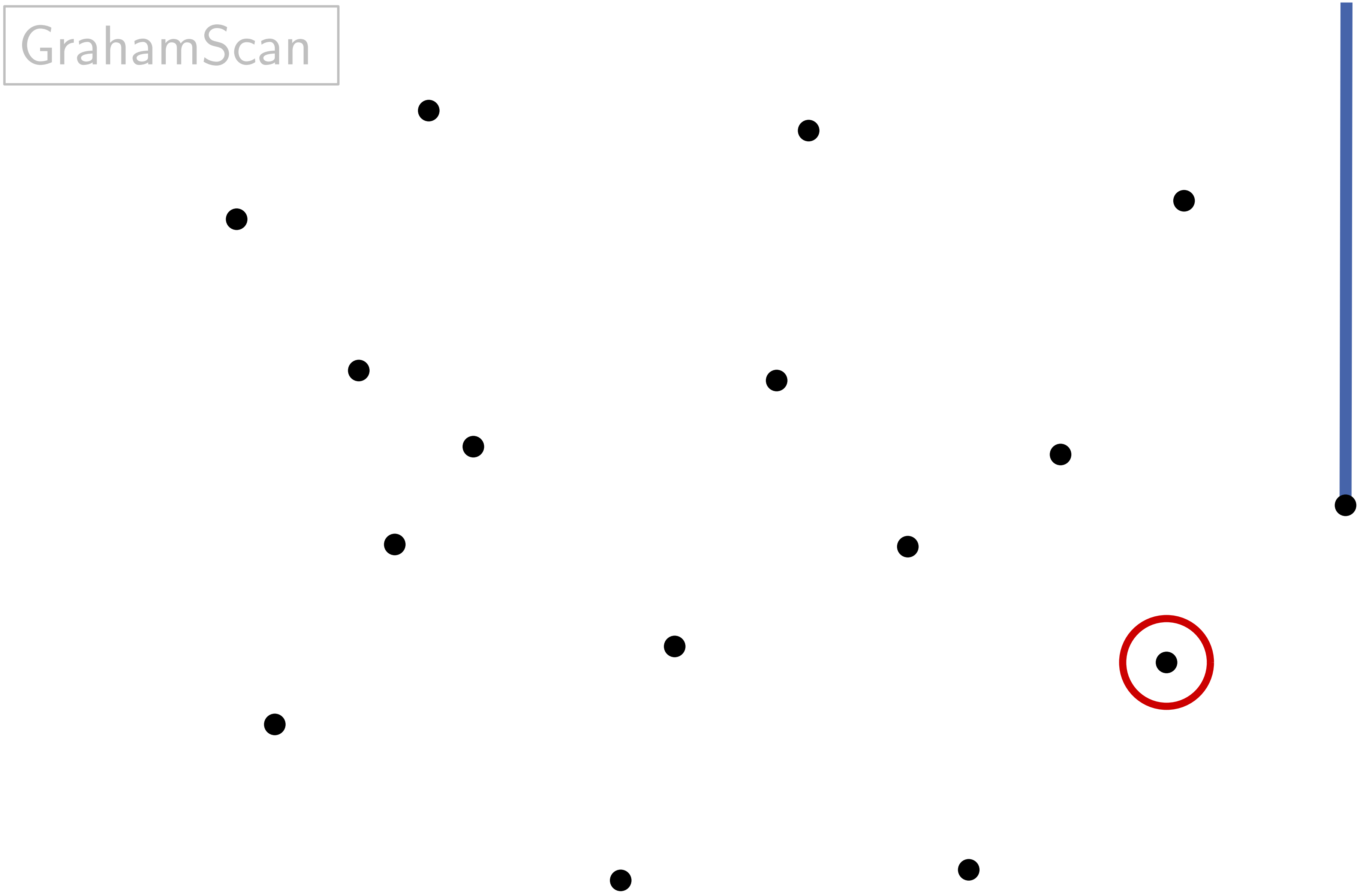
GrahamScan



n = 16

Gift Wrapping

Example

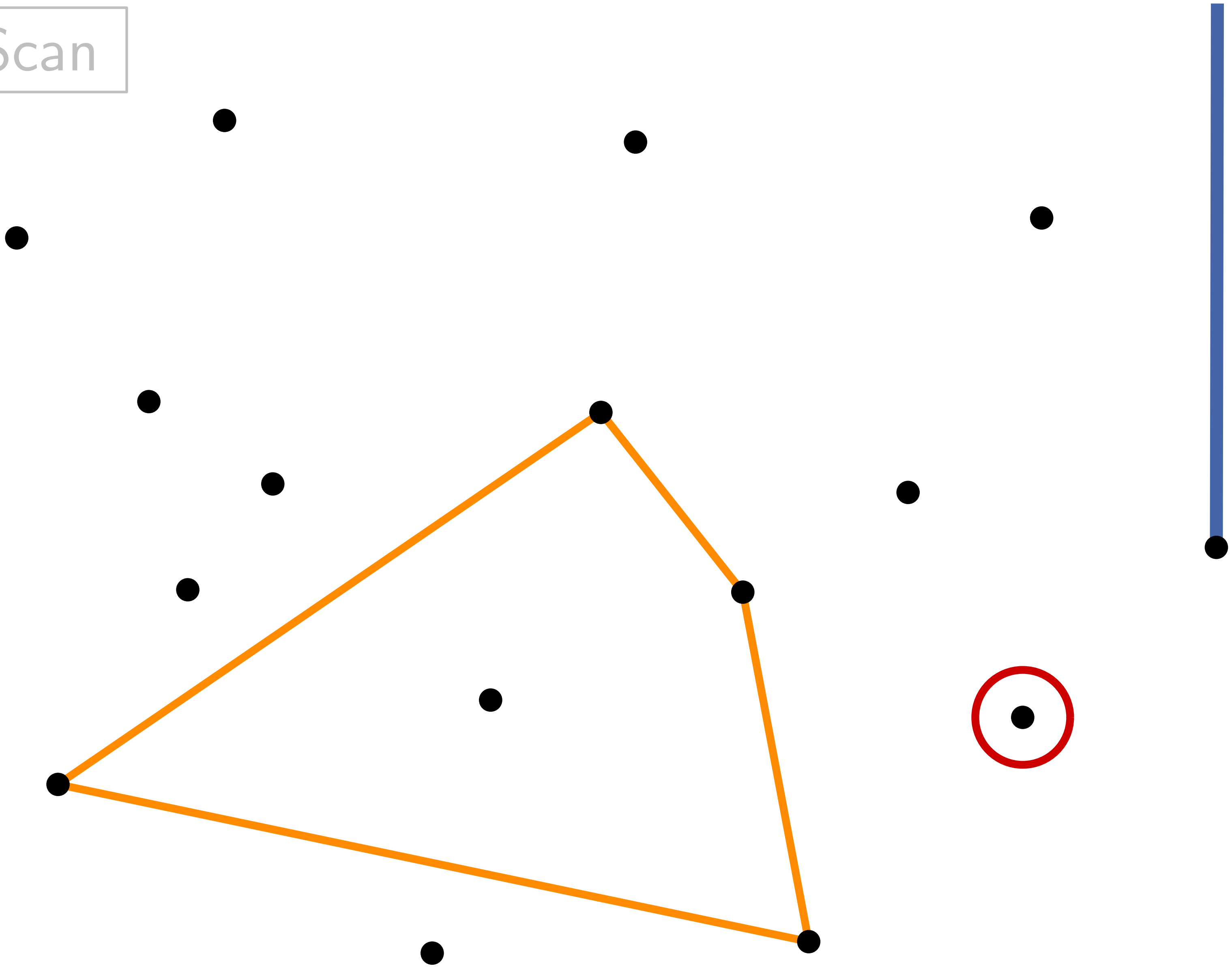


n = 16

Gift Wrapping

Example

GrahamScan

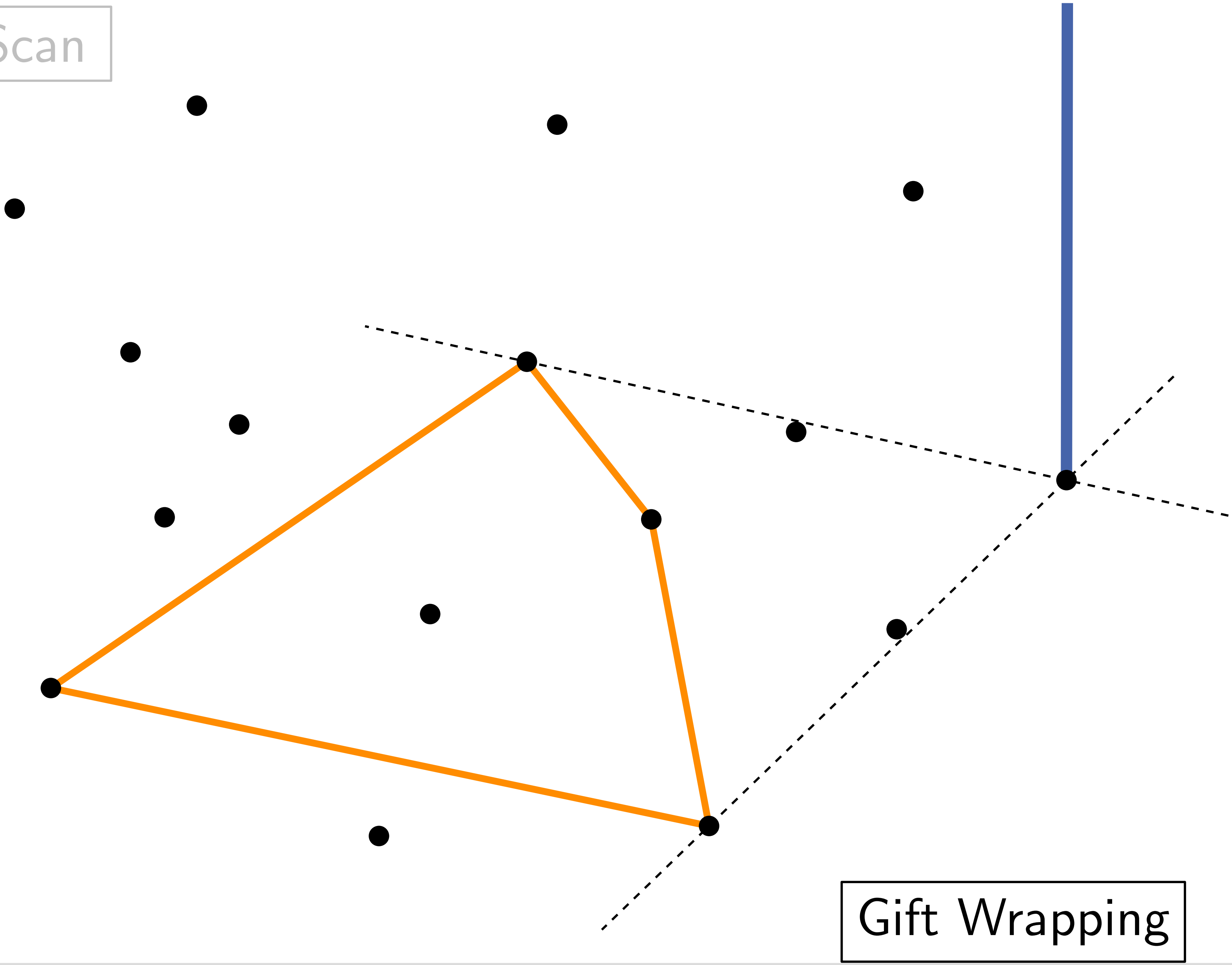


n = 16

Gift Wrapping

Example

GrahamScan

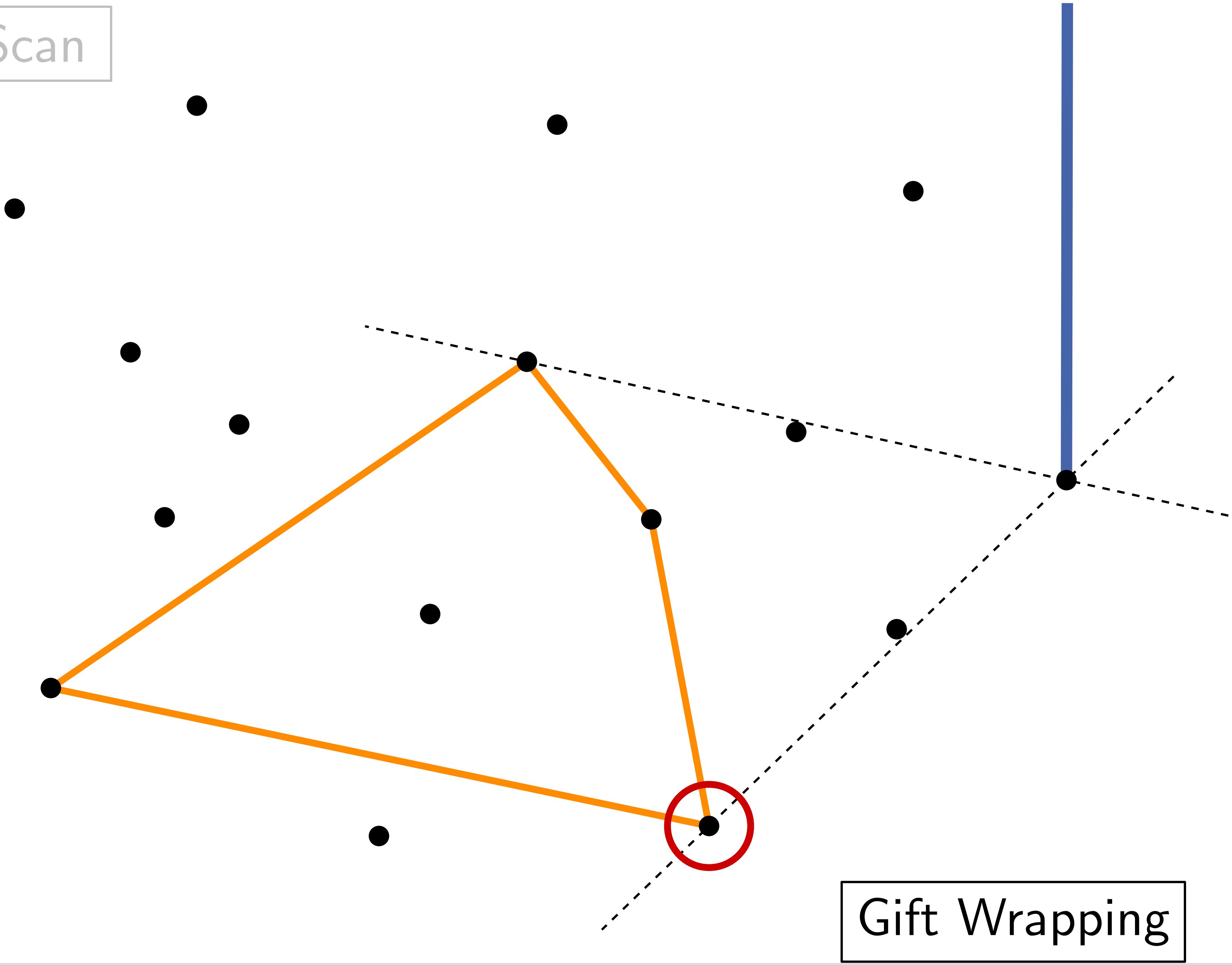


n = 16

Gift Wrapping

Example

GrahamScan

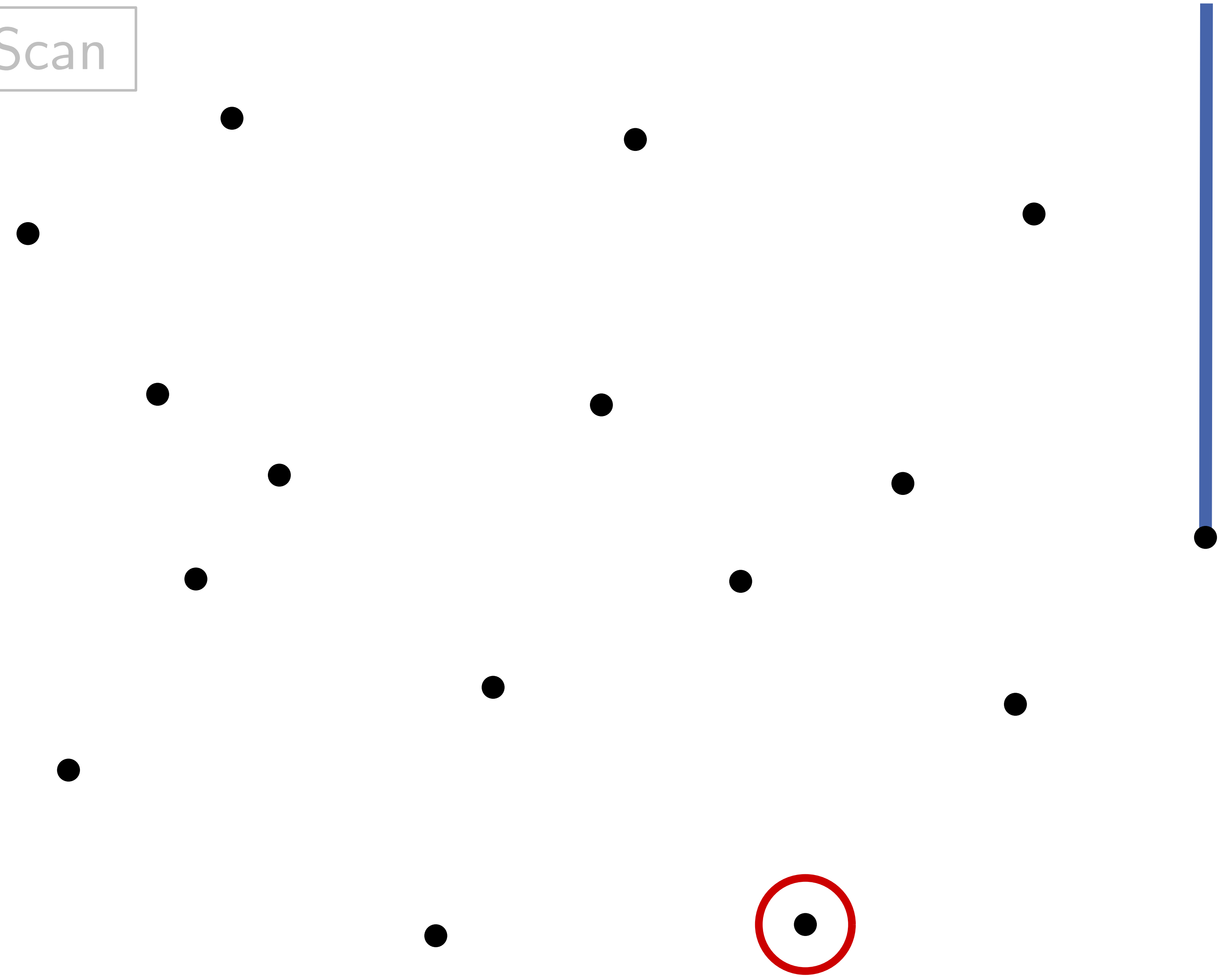


n = 16

Gift Wrapping

Example

GrahamScan

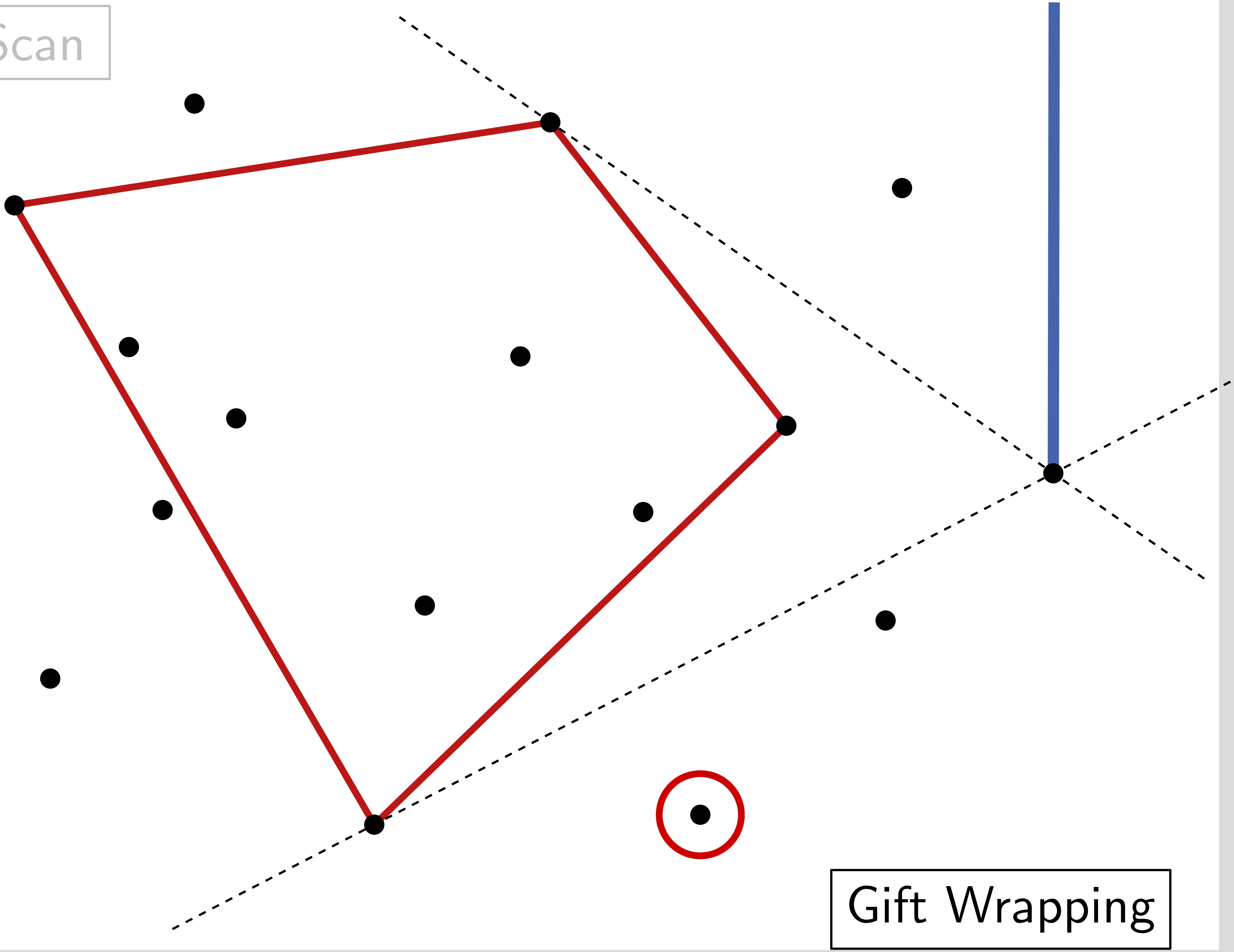


n = 16

Gift Wrapping

Example

GrahamScan

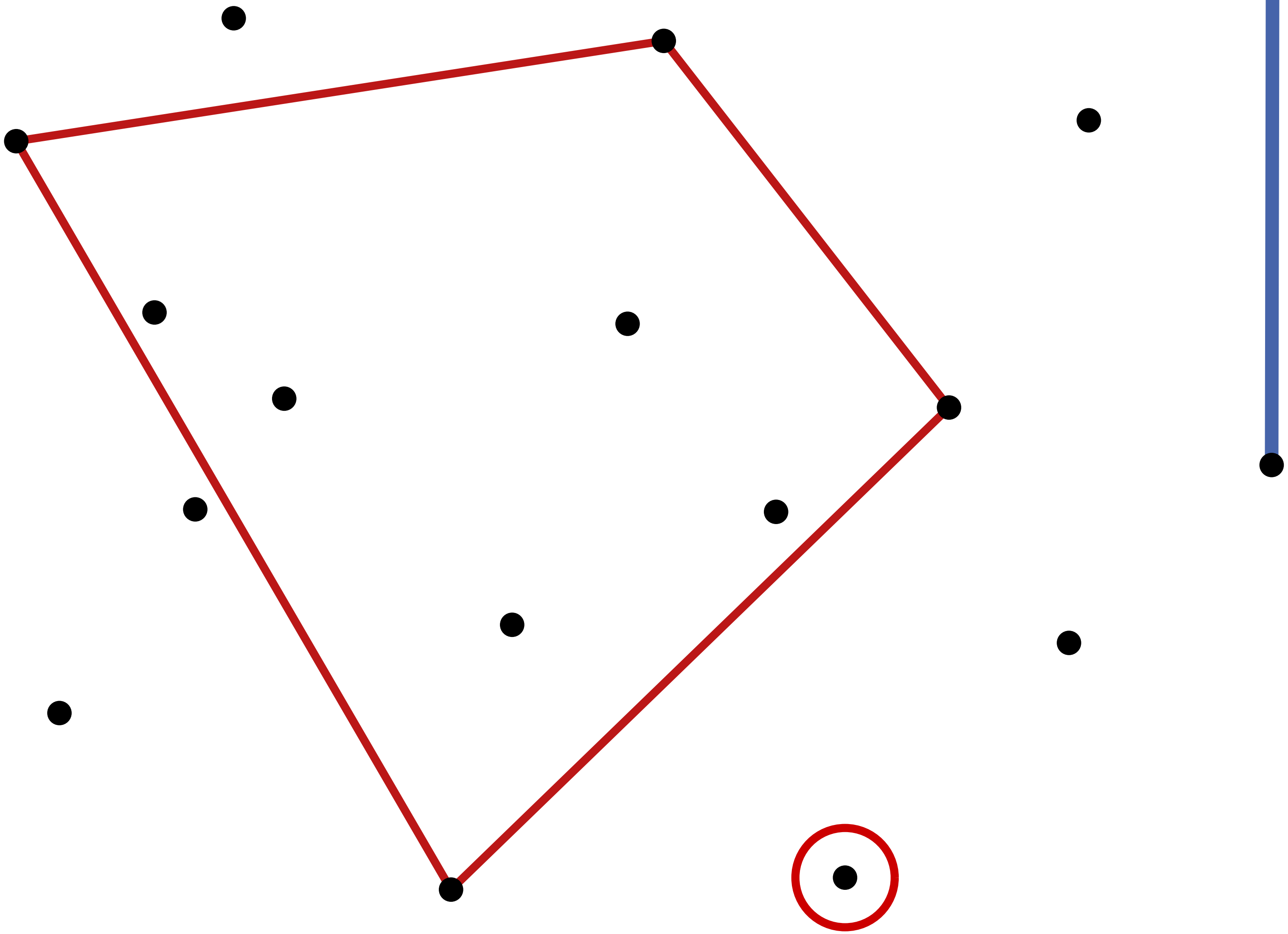


n = 16

Gift Wrapping

Example

GrahamScan

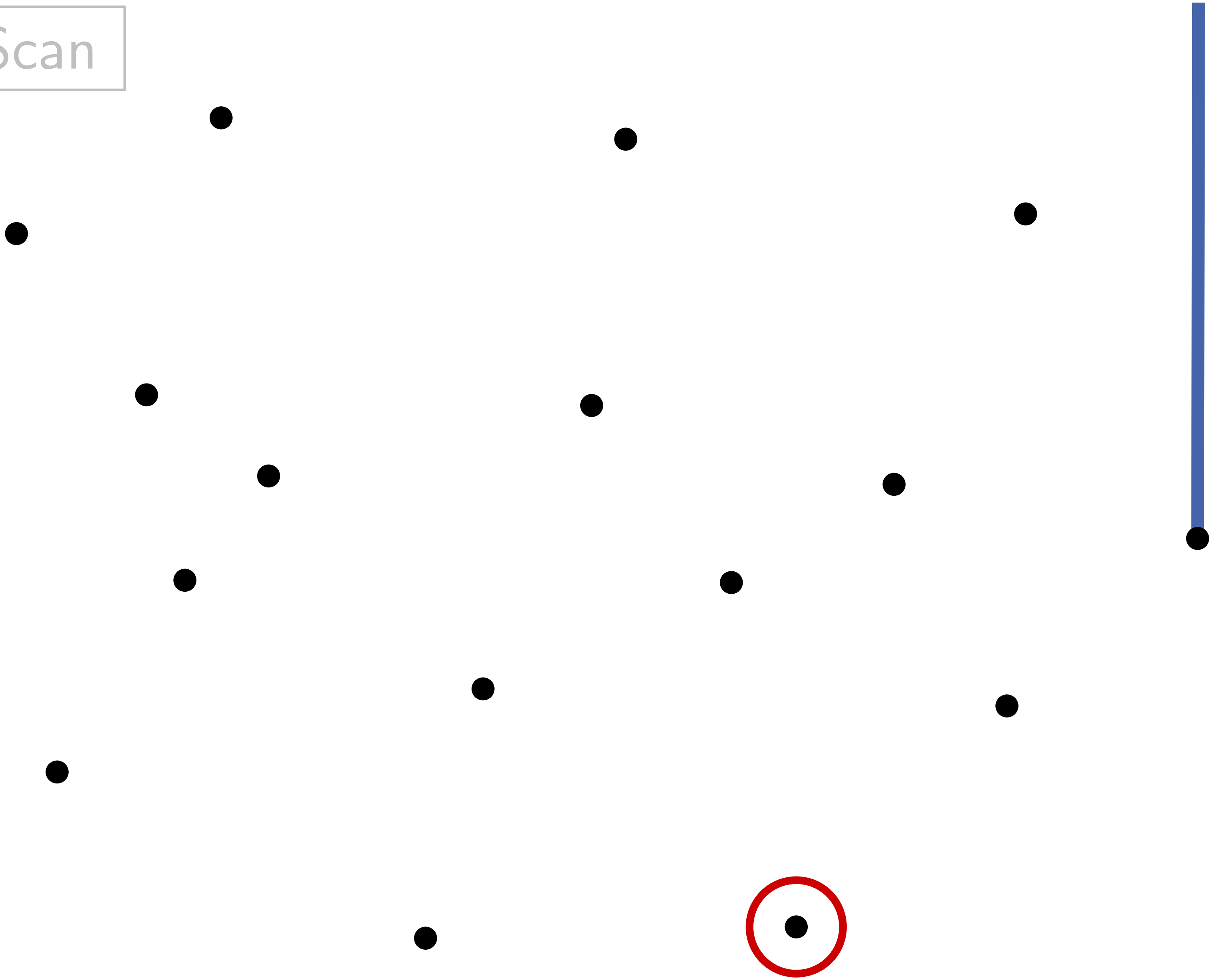


n = 16

Gift Wrapping

Example

GrahamScan

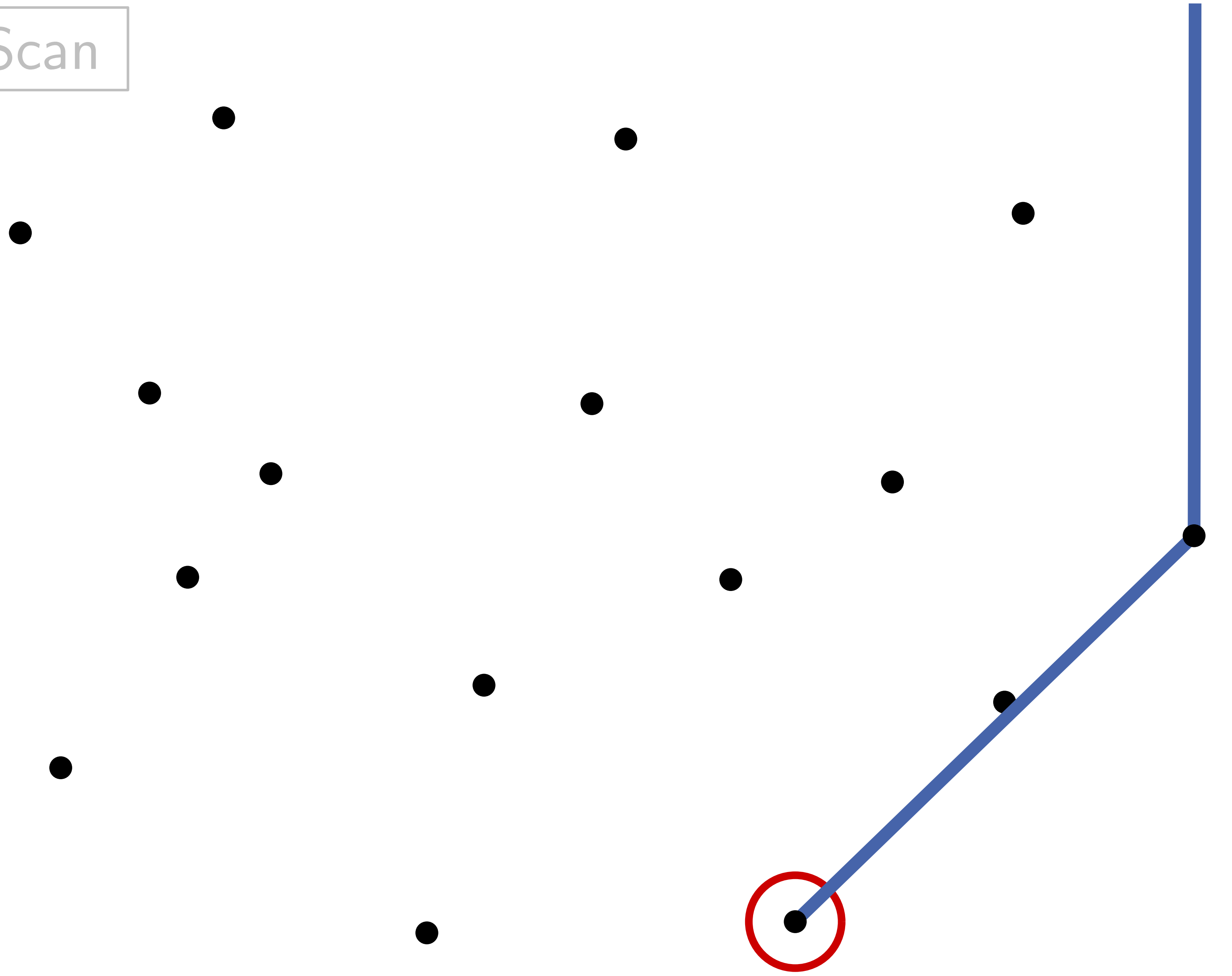


n = 16

Gift Wrapping

Example

GrahamScan



n = 16

Gift Wrapping

Chan's Algorithm

Suppose we know h:

ChanHull(P,h)

```
Divide $P$ into sets $P_i$ with $\leq h$ nodes
for $i$ from 1 to $\lceil n/h \rceil$ do
    Compute with GrahamScan $CH(P_i)$
$p_1 = (x_1, y_1) \leftarrow$ rightmost point in $P$
$p_0 \leftarrow (x_1, \infty)$
for $j = 1$ to $h$ do
    for $i = 1$ to $\lceil n/h \rceil$ do
        $q_i \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in P_i \setminus \{p_{j-1}, p_j\}\}$
    $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in \{q_1, \dots, q_{\lceil n/h \rceil}\}\}$
return $(p_1, \dots, p_h)$
```

GrahamScan

Gift Wrapping

Chan's Algorithm

Suppose we know h:

ChanHull(P,h)

```
Divide $P$ into sets $P_i$ with $\leq h$ nodes
for $i$ from 1 to $\lceil n/h \rceil$ do
    Compute with GrahamScan $CH(P_i)$
$p_1 = (x_1, y_1) \leftarrow$ rightmost point in $P$
$p_0 \leftarrow (x_1, \infty)$
for $j = 1$ to $h$ do
    for $i = 1$ to $\lceil n/h \rceil$ do
        $q_i \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in P_i \setminus \{p_{j-1}, p_j\}\}$
    $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in \{q_1, \ldots, q_{\lceil n/h \rceil}\}\}$
return $(p_1, \ldots, p_h)$
```

Chan's Algorithm

Suppose we know h:

ChanHull(P,h)

```
Divide $P$ into sets $P_i$ with $\leq h$ nodes
for $i$ from 1 to $\lceil n/h \rceil$ do
    Compute with GrahamScan $CH(P_i)$        $\mathcal{O}(h \log h)$
$p_1 = (x_1, y_1) \leftarrow$ rightmost point in $P$
$p_0 \leftarrow (x_1, \infty)$
for $j = 1$ to $h$ do
    for $i = 1$ to $\lceil n/h \rceil$ do
        $q_i \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in P_i \setminus \{p_{j-1}, p_j\}\}$
    $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in \{q_1, \dots, q_{\lceil n/h \rceil}\}\}$
return $(p_1, \dots, p_h)$
```

Chan's Algorithm

Suppose we know h:

ChanHull(P,h)

```
    Divide $P$ into sets $P_i$ with $\leq h$ nodes
    for $i$ from 1 to $\lceil n/h \rceil$ do                              $\mathcal{O}(n/h)$
        Compute with GrahamScan $CH(P_i)$                            $\mathcal{O}(h \log h)$
    $p_1 = (x_1, y_1) \leftarrow$ rightmost point in $P$
    $p_0 \leftarrow (x_1, \infty)$
    for $j = 1$ to $h$ do
        for $i = 1$ to $\lceil n/h \rceil$ do
            $q_i \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in P_i \setminus \{p_{j-1}, p_j\}\}$
        $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in \{q_1, \ldots, q_{\lceil n/h \rceil}\}\}$
    return $(p_1, \ldots, p_h)$
```

Chan's Algorithm

Suppose we know h:

ChanHull(P,h)

```
Divide $P$ into sets $P_i$ with $\leq h$ nodes
for $i$ from 1 to $\lceil n/h \rceil$ do                                      $\mathcal{O}(n/h)$
    Compute with GrahamScan $CH(P_i)$                                  $\mathcal{O}(h \log h)$
$p_1 = (x_1, y_1) \leftarrow$ rightmost point in $P$
$p_0 \leftarrow (x_1, \infty)$
for $j = 1$ to $h$ do
    for $i = 1$ to $\lceil n/h \rceil$ do
        $q_i \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in P_i \setminus \{p_{j-1}, p_j\}\}$
    $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in \{q_1, \ldots, q_{\lceil n/h \rceil}\}\}$
return $(p_1, \ldots, p_h)$
```

$\mathcal{O}(n \log h)$

Chan's Algorithm

Suppose we know h:

ChanHull(P,h)

```
Divide $P$ into sets $P_i$ with $\leq h$ nodes
for $i$ from 1 to $\lceil n/h \rceil$ do                              $\mathcal{O}(n/h)$
    Compute with GrahamScan $CH(P_i)$                                 $\mathcal{O}(h \log h)$
$p_1 = (x_1, y_1) \leftarrow$ rightmost point in $P$
$p_0 \leftarrow (x_1, \infty)$
for $j = 1$ to $h$ do
    for $i = 1$ to $\lceil n/h \rceil$ do                             $\mathcal{O}(\log h) \rightarrow$ Exercise!
        $q_i \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in P_i \setminus \{p_{j-1}, p_j\}\}$
    $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in \{q_1, \ldots, q_{\lceil n/h \rceil}\}\}$
return $(p_1, \ldots, p_h)$
```

$\mathcal{O}(n \log h)$

Chan's Algorithm

Suppose we know h:

ChanHull(P,h)

```
Divide $P$ into sets $P_i$ with $\leq h$ nodes
for $i$ from 1 to $\lceil n/h \rceil$ do                                   $\mathcal{O}(n/h)$
    Compute with GrahamScan $CH(P_i)$                                       $\mathcal{O}(h \log h)$
$p_1 = (x_1, y_1) \leftarrow$ rightmost point in $P$
$p_0 \leftarrow (x_1, \infty)$
for $j = 1$ to $h$ do          $\mathcal{O}(h) \cdot \mathcal{O}(n/h) = \mathcal{O}(n)$
    for $i = 1$ to $\lceil n/h \rceil$ do               $\mathcal{O}(\log h) \rightarrow$ Exercise!
        $q_i \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in P_i \setminus \{p_{j-1}, p_j\}\}$
    $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in \{q_1, \dots, q_{\lceil n/h \rceil}\}\}$
return $(p_1, \dots, p_h)$
```

$\mathcal{O}(n \log h)$

Chan's Algorithm

Suppose we know h:

ChanHull(P,h)

```
Divide $P$ into sets $P_i$ with $\leq h$ nodes
for $i$ from 1 to $\lceil n/h \rceil$ do                                  $\mathcal{O}(n/h)$
    Compute with GrahamScan $CH(P_i)$                                     $\mathcal{O}(h \log h)$
$p_1 = (x_1, y_1) \leftarrow$ rightmost point in $P$
$p_0 \leftarrow (x_1, \infty)$
for $j = 1$ to $h$ do              $\mathcal{O}(h) \cdot \mathcal{O}(n/h) = \mathcal{O}(n)$
    for $i = 1$ to $\lceil n/h \rceil$ do                      $\mathcal{O}(\log h) \rightarrow$ Exercise!
        $q_i \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in P_i \setminus \{p_{j-1}, p_j\}\}$
    $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in \{q_1, \ldots, q_{\lceil n/h \rceil}\}\}$
return $(p_1, \ldots, p_h)$
```

$\mathcal{O}(n \log h)$

$\mathcal{O}(n \log h)$

Chan's Algorithm

Suppose we know h:

ChanHull(P,h)

```
Divide $P$ into sets $P_i$ with $\leq h$ nodes
for $i$ from 1 to $\lceil n/h \rceil$ do                          $\mathcal{O}(n/h)$
    Compute with GrahamScan $CH(P_i)$                   $\mathcal{O}(h \log h)$
$p_1 = (x_1, y_1) \leftarrow$ rightmost point in $P$
$p_0 \leftarrow (x_1, \infty)$
for $j = 1$ to $h$ do      $\mathcal{O}(h) \cdot \mathcal{O}(n/h) = \mathcal{O}(n)$
    for $i = 1$ to $\lceil n/h \rceil$ do          $\mathcal{O}(\log h) \rightarrow$ Exercise!
        $q_i \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in P_i \setminus \{p_{j-1}, p_j\}\}$
    $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in \{q_1, \ldots, q_{\lceil n/h \rceil}\}\}$
return $(p_1, \ldots, p_h)$
```

$\mathcal{O}(n \log h)$

$\mathcal{O}(n \log h)$

Total: $\mathcal{O}(n \log h)$

Chan's Algorithm

Karlsruhe Institute of Technology

But in general h is unknown!

```
ChanHull($P$,$h$)
    Divide $P$ into sets $P_i$ with $\leq h$ nodes
    for $i$ from 1 to $\lceil n/h \rceil$ do                          $\mathcal{O}(n/h)$
        Compute with GrahamScan $CH(P_i)$                        $\mathcal{O}(h \log h)$
    $p_1 = (x_1, y_1) \leftarrow$ rightmost point in $P$
    $p_0 \leftarrow (x_1, \infty)$
    for $j = 1$ to $h$ do     $\mathcal{O}(h) \cdot \mathcal{O}(n/h) = \mathcal{O}(n)$
        for $i = 1$ to $\lceil n/h \rceil$ do            $\mathcal{O}(\log h) \rightarrow$ Exercise!
            $q_i \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in P_i \setminus \{p_{j-1}, p_j\}\}$
        $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in \{q_1, \ldots, q_{\lceil n/h \rceil}\}\}$
    return $(p_1, \ldots, p_h)$
```

$\mathcal{O}(n \log h)$

$\mathcal{O}(n \log h)$

Chan's Algorithm

But in general h is unknown!

```
ChanHull($P, \cancel{h}$) ← $m$
    Divide $P$ into sets $P_i$ with $\leq h$ nodes
    for $i$ from 1 to $\lceil n/h \rceil$ do                                  $\mathcal{O}(n/h)$
        Compute with GrahamScan $CH(P_i)$                                   $\mathcal{O}(h \log h)$
    $p_1 = (x_1, y_1) \leftarrow$ rightmost point in $P$
    $p_0 \leftarrow (x_1, \infty)$
    for $j = 1$ to $h$ do        $\mathcal{O}(h) \cdot \mathcal{O}(n/h) = \mathcal{O}(n)$
        for $i = 1$ to $\lceil n/h \rceil$ do                 $\mathcal{O}(\log h) \rightarrow$ Exercise!
            $q_i \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in P_i \setminus \{p_{j-1}, p_j\}\}$
        $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in \{q_1, \ldots, q_{\lceil n/h \rceil}\}\}$
    return $(p_1, \ldots, p_h)$
```

$\mathcal{O}(n \log h)$

$\mathcal{O}(n \log h)$

Chan's Algorithm

But in general h is unknown!

```
ChanHull($P$,$m$)
    Divide $P$ into sets $P_i$ with $\leq m$ nodes
    for $i$ from 1 to $\lceil n/m \rceil$ do                                  $\mathcal{O}(n/m)$
        Compute with GrahamScan $CH(P_i)$                                    $\mathcal{O}(m \log m)$
    $p_1 = (x_1, y_1) \leftarrow$ rightmost point in $P$
    $p_0 \leftarrow (x_1, \infty)$
    for $j = 1$ to $m$ do        $\mathcal{O}(m) \cdot \mathcal{O}(n/m) = \mathcal{O}(n)$
        for $i = 1$ to $\lceil n/m \rceil$ do                               $\mathcal{O}(\log m)$
            $q_i \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in P_i \setminus \{p_{j-1}, p_j\}\}$
        $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in \{q_1, \ldots, q_{\lceil n/m \rceil}\}\}$
    return $(p_1, \ldots, p_m)$
```

$\mathcal{O}(n \log m)$

$\mathcal{O}(n \log m)$

Total: $\mathcal{O}(n \log h)$

Chan's Algorithm

But in general h is unknown!

```
ChanHull($P$,$m$)
    Divide $P$ into sets $P_i$ with $\leq m$ nodes
    for $i$ from 1 to $\lceil n/m \rceil$ do                         $\mathcal{O}(n/m)$
        Compute with GrahamScan $CH(P_i)$                          $\mathcal{O}(m \log m)$
    $p_1 = (x_1, y_1) \leftarrow$ rightmost point in $P$
    $p_0 \leftarrow (x_1, \infty)$
    for $j = 1$ to $m$ do          $\mathcal{O}(m) \cdot \mathcal{O}(n/m) = \mathcal{O}(n)$
        for $i = 1$ to $\lceil n/m \rceil$ do                         $\mathcal{O}(\log m)$
            $q_i \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in P_i \setminus \{p_{j-1}, p_j\}\}$
        $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in \{q_1, \ldots, q_{\lceil n/m \rceil}\}\}$
    return $(p_1, \ldots, p_m)$
```

$\mathcal{O}(n \log m)$

$\mathcal{O}(n \log m)$

Total: $\mathcal{O}(n \log h)$

Chan's Algorithm

But in general h is unknown!

```
ChanHull($P$,$m$)
    Divide $P$ into sets $P_i$ with $\leq m$ nodes
    **for** $i$ from 1 to $\lceil n/m \rceil$ **do**                                $\mathcal{O}(n/m)$
        Compute with GrahamScan $CH(P_i)$                       $\mathcal{O}(m \log m)$
    $p_1 = (x_1, y_1) \leftarrow$ rightmost point in $P$
    $p_0 \leftarrow (x_1, \infty)$
    **for** $j = 1$ **to** $m$ **do**          $\mathcal{O}(m) \cdot \mathcal{O}(n/m) = \mathcal{O}(n)$
        **for** $i = 1$ **to** $\lceil n/m \rceil$ **do**                                $\mathcal{O}(\log m)$
            $q_i \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in P_i \setminus \{p_{j-1}, p_j\}\}$
        $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in \{q_1, \ldots, q_{\lceil n/m \rceil}\}\}$
        **if** $p_{j+1} = p_1$ **then return** $(p_1, \ldots, p_{j+1})$
    **return** failure
```

$\mathcal{O}(n \log m)$

$\mathcal{O}(n \log m)$

Total: $\mathcal{O}(n \log m)$

What to do with m?

Suggestions?

What to do with m?

```
FullChanHull($P$)
    for $t = 0, 1, 2, \ldots$ do
        $m =\leftarrow \min\{n, 2^{2^t}\}$
        result ← ChanHull($P$, $m$)
        if result $\neq$ failure then break
    return result
```

What to do with m?

FullChanHull(P)

```
for $t = 0, 1, 2, \ldots$ do
    $m =\leftarrow \min\{n, 2^{2^t}\}$
    result $\leftarrow$ ChanHull($P$, $m$)          $\mathcal{O}(n \log m) =$
    if result $\neq$ failure then break          $\mathcal{O}(n \log 2^{2^t})$
return result
```

Running time:

What to do with m?

FullChanHull(P)

```
for $t = 0, 1, 2, \ldots$ do
    $m =\leftarrow \min\{n, 2^{2^t}\}$
    result ← ChanHull($P$, $m$)
    if result $\neq$ failure then break
return result
```

$\mathcal{O}(n \log m) = \mathcal{O}(n \log 2^{2^t})$

Running time:

$$\sum_{t=0}^{\lceil \log \log h \rceil} \mathcal{O}(n \log 2^{2^t})$$

What to do with m?

FullChanHull(P)

```
for $t = 0, 1, 2, \ldots$ do
    $m =\leftarrow \min\{n, 2^{2^t}\}$
    result ← ChanHull($P$, $m$)
    if result $\neq$ failure then break
return result
```

$\mathcal{O}(n \log m) = \mathcal{O}(n \log 2^{2^t})$

Running time:

$$\sum_{t=0}^{\lceil \log \log h \rceil} \mathcal{O}(n \log 2^{2^t}) = \mathcal{O}(n) \sum_{t=0}^{\lceil \log \log h \rceil} \mathcal{O}(2^t)$$

What to do with m?

FullChanHull(P)

```
for $t = 0, 1, 2, \ldots$ do
    $m = \leftarrow \min\{n, 2^{2^t}\}$
    result ← ChanHull($P$, $m$)
    if result $\neq$ failure then break
return result
```

$$\mathcal{O}(n \log m) = \mathcal{O}(n \log 2^{2^t})$$

Running time:

$$\sum_{t=0}^{\lceil \log \log h \rceil} \mathcal{O}(n \log 2^{2^t}) \qquad = \mathcal{O}(n) \sum_{t=0}^{\lceil \log \log h \rceil} \mathcal{O}(2^t)$$

$$\leq \mathcal{O}(n) \cdot \mathcal{O}(2^{\log \log h})$$

What to do with m?

FullChanHull(P)

```
for $t = 0, 1, 2, \ldots$ do
    $m =\leftarrow \min\{n, 2^{2^t}\}$
    result ← ChanHull($P$, $m$)
    if result $\neq$ failure then break
return result
```

$\mathcal{O}(n \log m) = \mathcal{O}(n \log 2^{2^t})$

Running time:

$$\sum_{t=0}^{\lceil \log \log h \rceil} \mathcal{O}(n \log 2^{2^t}) = \mathcal{O}(n) \sum_{t=0}^{\lceil \log \log h \rceil} \mathcal{O}(2^t)$$

$$\leq \mathcal{O}(n) \cdot \mathcal{O}(2^{\log \log h}) = \mathcal{O}(n) \cdot \mathcal{O}(\log h)$$

What to do with m?

FullChanHull(P)

```
for $t = 0, 1, 2, \ldots$ do
    $m = \leftarrow \min\{n, 2^{2^t}\}$
    result ← ChanHull($P$, $m$)          $\mathcal{O}(n \log m) =$
    if result $\neq$ failure then break  $\mathcal{O}(n \log 2^{2^t})$
return result
```

Running time:

$$\sum_{t=0}^{\lceil \log\log h \rceil} \mathcal{O}(n \log 2^{2^t}) \quad = \mathcal{O}(n) \sum_{t=0}^{\lceil \log\log h \rceil} \mathcal{O}(2^t)$$

$$\leq \mathcal{O}(n) \cdot \mathcal{O}(2^{\log\log h}) \quad = \mathcal{O}(n) \cdot \mathcal{O}(\log h)$$

$$= \mathcal{O}(n \log h)$$

What to do with m?

```
FullChanHull($P$)
    for $t = 0, 1, 2, \ldots$ do
        $m = \leftarrow \min\{n, 2^{2^t}\}$
        result ← ChanHull($P$, $m$)
        if result $\neq$ failure then break
    return result
```

Theorem 3: The convex hull $CH(P)$ of n points P in $\mathbb{R}^2$ can be computed in $O(n \log h)$ time with Chan's Algorithm, where $h = |CH(P)|$.

Discussion

Is it possible to compute faster than $O(n \log n)$ or $O(n \log h)$ time?

Discussion

Is it possible to compute faster than $O(n \log n)$ or $O(n \log h)$ time?

Generally not! An algorithm to compute the convex hull can also sort (exercise!) $\Rightarrow$ lower bound $\Omega(n \log n)$.

Discussion

Is it possible to compute faster than $O(n \log n)$ or $O(n \log h)$ time?

Generally not! An algorithm to compute the convex hull can also sort (exercise!) ⇒ lower bound $\Omega(n \log n)$.

What happens in Graham's Scan when sorting P is not unique?

Discussion

Is it possible to compute faster than $O(n \log n)$ or $O(n \log h)$ time?

Generally not! An algorithm to compute the convex hull can also sort (exercise!) ⇒ lower bound $\Omega(n \log n)$.

What happens in Graham's Scan when sorting P is not unique?

Use lexicographic order!

Discussion

Is it possible to compute faster than $O(n \log n)$ or $O(n \log h)$ time?

Generally not! An algorithm to compute the convex hull can also sort (exercise!) $\Rightarrow$ lower bound $\Omega(n \log n)$.

What happens in Graham's Scan when sorting P is not unique?

Use lexicographic order!

What happens with collinear points in $CH(P)$?

Discussion

Is it possible to compute faster than $O(n \log n)$ or $O(n \log h)$ time?

Generally not! An algorithm to compute the convex hull can also sort (exercise!) $\Rightarrow$ lower bound $\Omega(n \log n)$.

What happens in Graham's Scan when sorting P is not unique?

Use lexicographic order!

What happens with collinear points in $CH(P)$?

Graham: Forms no right turn, so an interior point is deleted.
Jarvis: Choose farthest point

Discussion

Is it possible to compute faster than $O(n \log n)$ or $O(n \log h)$ time?

Generally not! An algorithm to compute the convex hull can also sort (exercise!) $\Rightarrow$ lower bound $\Omega(n \log n)$.

What happens in Graham's Scan when sorting P is not unique?

Use lexicographic order!

What happens with collinear points in $CH(P)$?

Graham: Forms no right turn, so an interior point is deleted.
Jarvis: Choose farthest point

What about the robustness of the algorithms?

Discussion

Is it possible to compute faster than $O(n \log n)$ or $O(n \log h)$ time?

Generally not! An algorithm to compute the convex hull can also sort (exercise!) $\Rightarrow$ lower bound $\Omega(n \log n)$.

What happens in Graham's Scan when sorting P is not unique?

Use lexicographic order!

What happens with collinear points in $CH(P)$?

Graham: Forms no right turn, so an interior point is deleted.
Jarvis: Choose farthest point

What about the robustness of the algorithms?

- Regarding robustness: imprecision of floating-point arithmetic
- FirstConvexHull possibly produces a valid polygon
- Graham and Jarvis always provide a polygon, but it may have minor defects

Designing Geometric Algorithms–Guidelines

1.) Eliminate degenerate cases (→ *general position*)

- unique x-coordinates
- no three collinear points
- . . .

2.) Adjust degenerate inputs

- integrate into existing solutions (e.g., compute lexicographic order if x-coordinates are not unique)
- may require special treament

3.) Implementation

- primitive operations (available in libraries?)
- robustness