Computational Geometry
Chapter 2: Convex Hull

Prof. Dr. Sándor Fekete

Algorithms Division
Department of Computer Science
TU Braunschweig

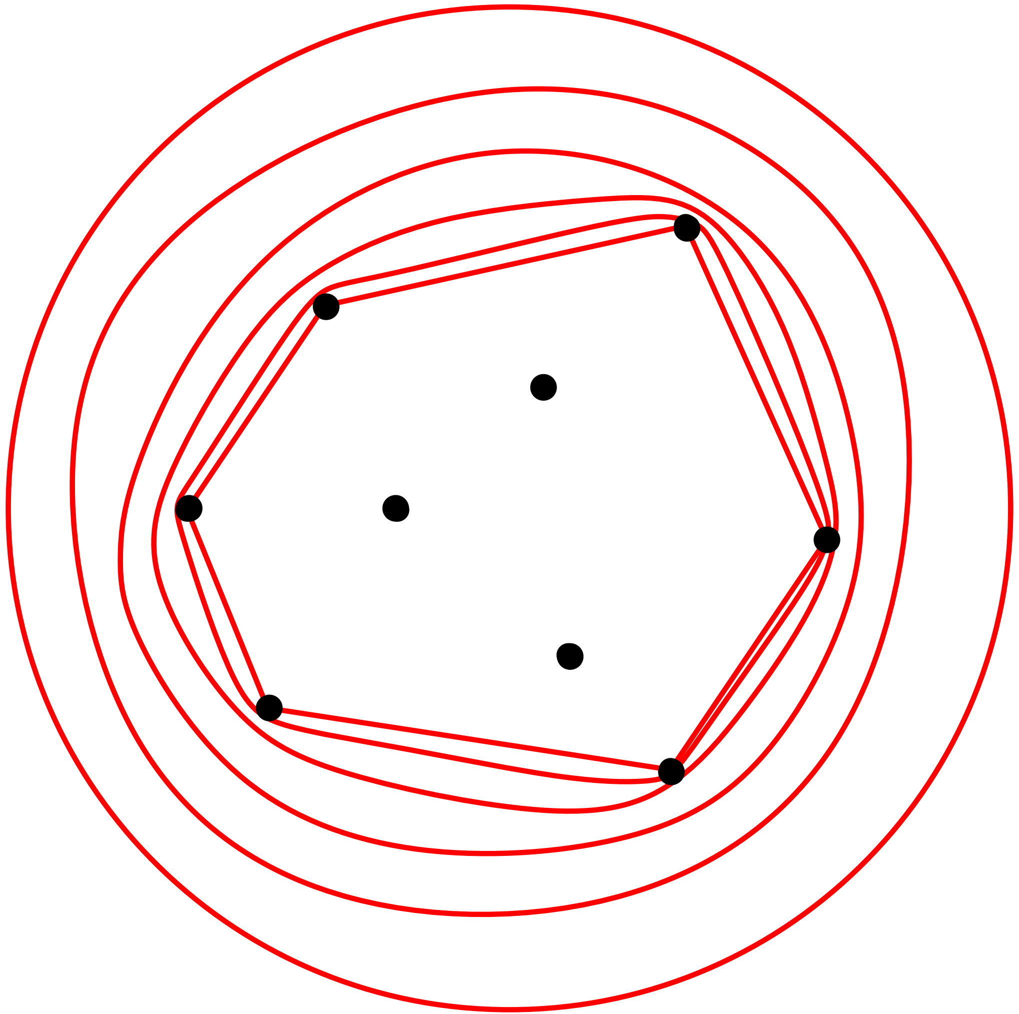
Technische
Universität
Braunschweig

Overview

1. Introduction and Definitions
2. Interlude: Algorithmic Paradigms
3. Jarvis' March
4. Quickhull
5. Divide-and-conquer and incremental construction
6. Graham's Scan
7. Optimal output-sensitive construction

Task:

- Given: Set of n points in $\mathbb{R}^d$
- Wanted: Smallest enclosing convex object



Intuition in $\mathbb{R}^2$:

- Draw points on a wooden board.
- Put in nails at points.
- Let a rubber band snap to the nails.

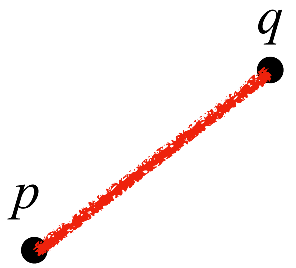
Theorem 2.18: Computing the convex hull takes $\Omega(n \log n)$.

Theorem 2.35: Computing the h vertices of the convex hull can be done in $O(n \log h)$.

Superhero!

CONVEX HULK

Definition - I



$p + \lambda(q-p), 0 \leq \lambda \leq 1$
$= (1-\lambda)p + \lambda q$

Definition 2.1

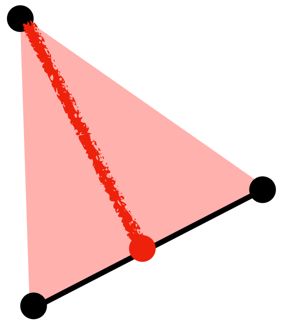
For $p, q \in \mathbb{R}^d : \overline{pq} := \{x \in \mathbb{R}^d \mid \exists \alpha, \beta \in \mathbb{R}, \alpha, \beta \geq 0, \alpha + \beta = 1, x = \alpha p + \beta q\}$

Definition 2.2

For $\{p_0, \ldots, p_{n-1}\} \subset \mathbb{R}^d$ point $x \in \mathbb{R}^d$ is a convex combination of $\{p_0, \ldots, p_{n-1}\}$, if

$\exists \alpha_0, \ldots, \alpha_{n-1} \in [0, 1]$ with 1. $\sum_{i=0}^{n} \alpha_i p_i = x$

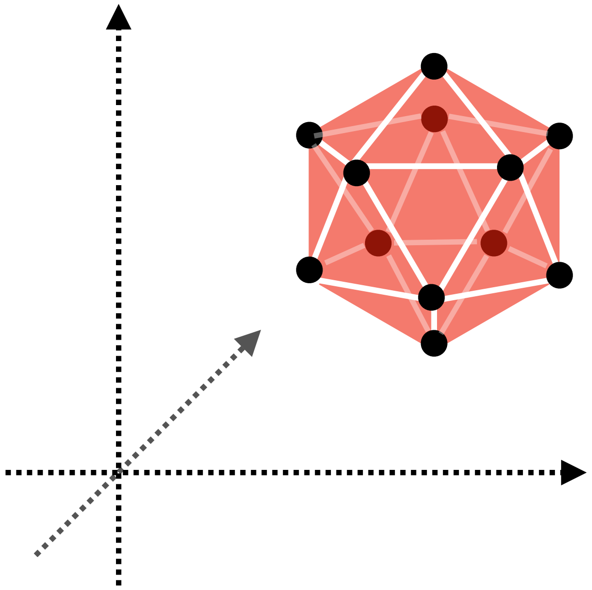
2. $\sum_{i=0}^{n} \alpha_i = 1$



- $\overline{pq} = \{x \mid x$ convex combination of $\{p, q\}\}$
- $\triangle(p, q, r) = \{x \mid x$ convex combination of $\{p, q, r\}\}$

Technische Universität Braunschweig

Definition - II



Definition 2.3

Convex hull $conv(\mathcal{P})$ of $\mathcal{P} := \{p_0, \ldots, p_{n-1}\}$:

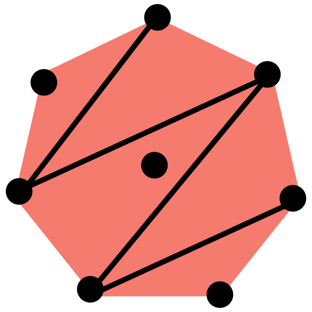
$$conv(\mathcal{P}) := \{x \in \mathbb{R}^d \mid x \text{ convex combination of } \mathcal{P}\}$$

Theorem 2.4 (Carathéodory)

$conv(\mathcal{P}) =$ Union of all convex combinations with at most $(d+1)$ points in $\mathcal{P}$

Corollary 2.5

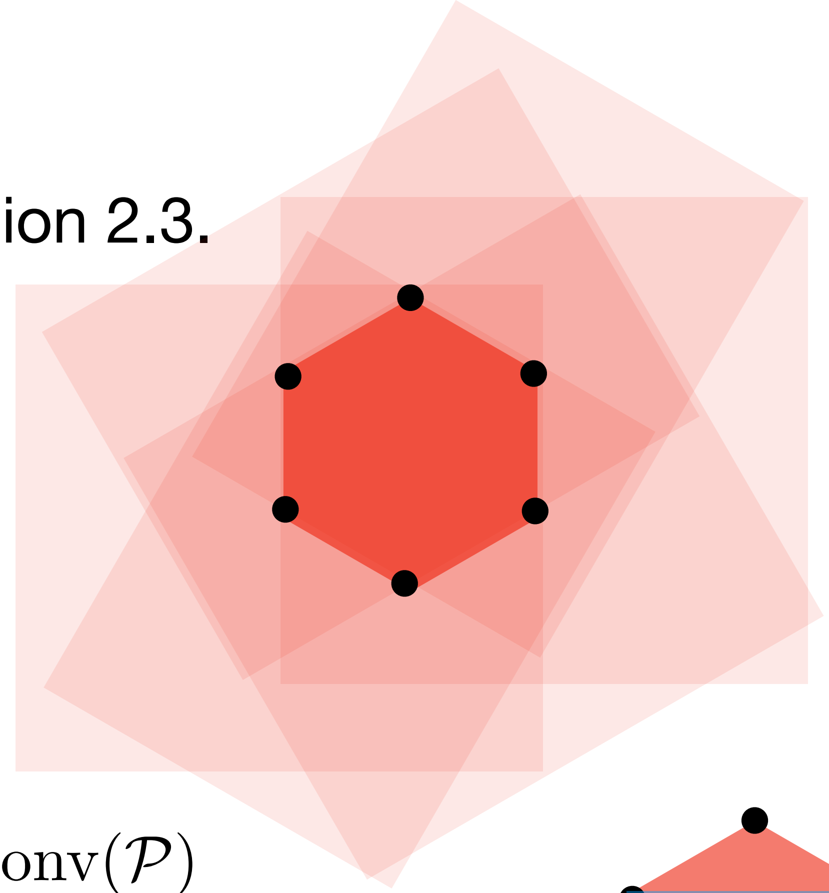
$\mathcal{P} \subset \mathbb{R}^2 \Rightarrow conv(\mathcal{P})$ union of all $\triangle(p, q, r)$ with $p, q, r \in \mathcal{P}$.



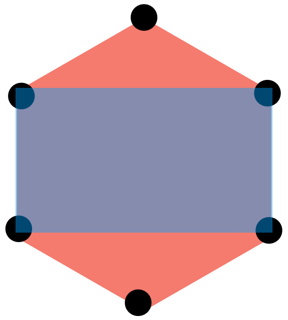
Alternative Definitions

Lemma 2.6

The following definitions are equivalent to Definition 2.3.

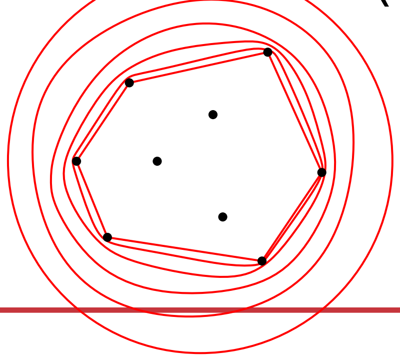


1. $conv(\mathcal{P}) := \bigcap_{P \supset \mathcal{P}, P \text{ convex}} P$



2. For $d = 2$: $\nexists$ convex polygon $P : \mathcal{P} \subseteq P \subsetneq \text{conv}(\mathcal{P})$

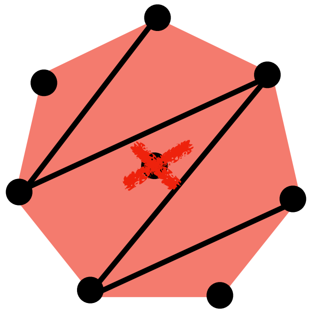
3. For $d = 2$: $conv(\mathcal{P}) :=$ conv. polygon P with minimal circumference (area) with $\mathcal{P} \subset P$



Computing the Convex Hull - I

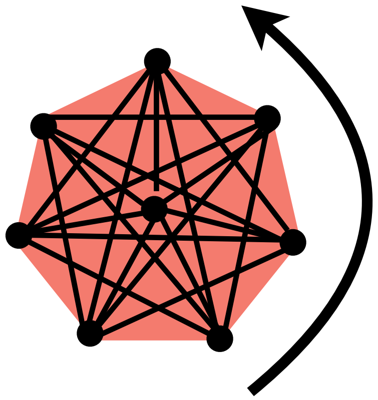
- From now on $\mathcal{P} \subset \mathbb{R}^2$
- First Approach: Find vertices of $conv(\mathcal{P})$ by elimination
- Negation of Corollary 2.5:

$$x \text{ not vertex of } conv(\mathcal{P})$$
$$\Updownarrow$$
$$\exists p_i, p_j, p_k \in \mathcal{P} : x = \text{non-trivial convex combination of } p_i, p_j, p_k.$$



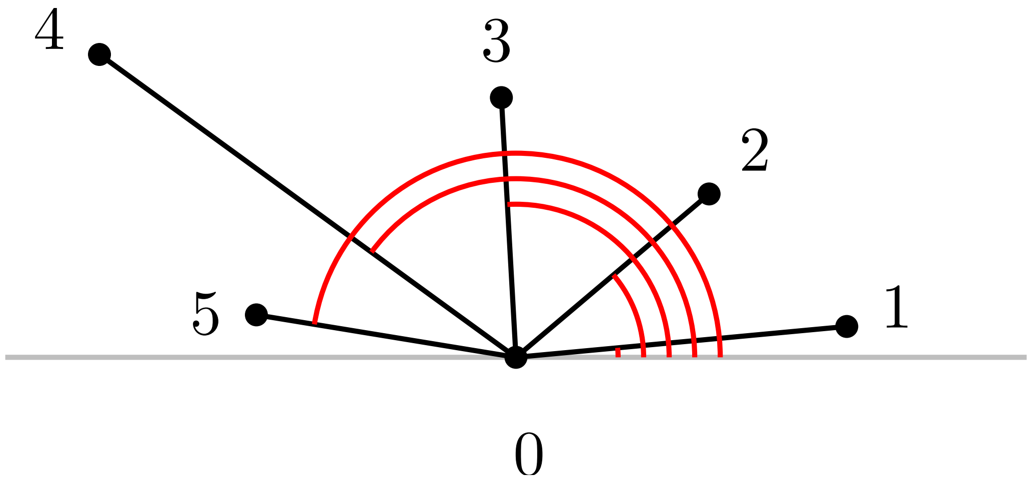
Algorithm 2.7

```
for ( all triples $(\boldsymbol{p}_i, \boldsymbol{p}_j, \boldsymbol{p}_k)$ of points in $\mathcal{P}$) do
    for ( all points in $\in \mathcal{P}$) do
        if ($\boldsymbol{p}$ lies in the inside of $\Delta(\boldsymbol{p}_i, \boldsymbol{p}_j, \boldsymbol{p}_k)$ or on a boundary edge of $\Delta(\boldsymbol{p}_i, \boldsymbol{p}_j, \boldsymbol{p}_k)$) then
            mark $\boldsymbol{p}$ as an interior point.
$\mathcal{P}' := \{\boldsymbol{p} \in \mathcal{P} \mid$ is unmarked $\}$;
```



Analysis of Algorithm 2.7:

- $\binom{n}{3} \in \Theta(n^3)$ triangles
- Per triangle: $\Theta(n)$ further points
- Sort $\mathcal{O}(n)$ vertices
- Total runtime: $\mathcal{O}(n^4 + n \log n) = \mathcal{O}(n^4)$



Sorting criterion:

- CCW on $conv(\mathcal{P})$.
- Polar angle wrt y-minimal point in $\mathcal{P}$.
- Issue: Do we need trigonometry?

Lexicographic Order

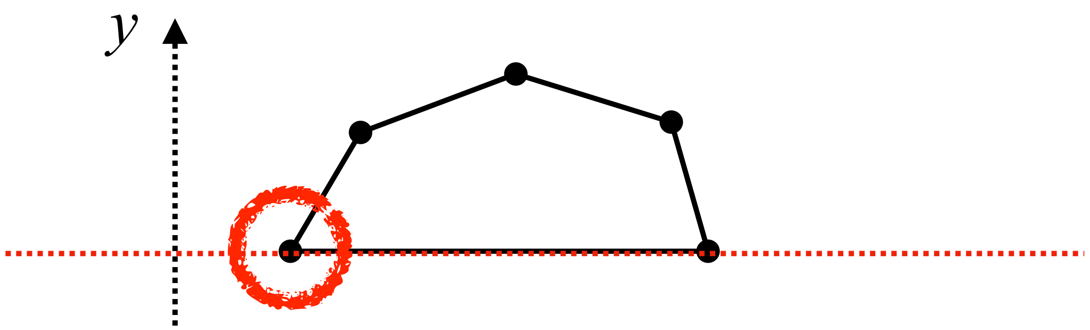
Lexicographic order

- Choose the „y-minimal“ point by lexicographic order:

$$(p.x, p.y) \leq_y (q.x, q.y) :\Leftrightarrow ((p.y < q.y) \vee ((p.y = q.y) \wedge (p.x \leq q.x)))$$

- Analogously: by x-coordinate

$$(p.x, p.y) \leq_x (q.x, q.y) :\Leftrightarrow ((p.x < q.x) \vee ((p.x = q.x) \wedge (p.y \leq q.y)))$$

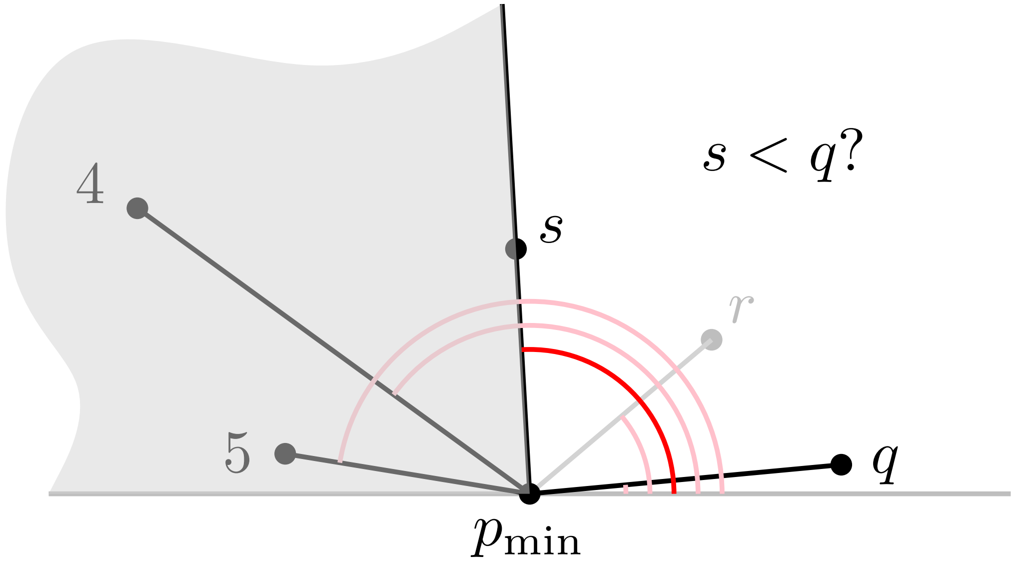


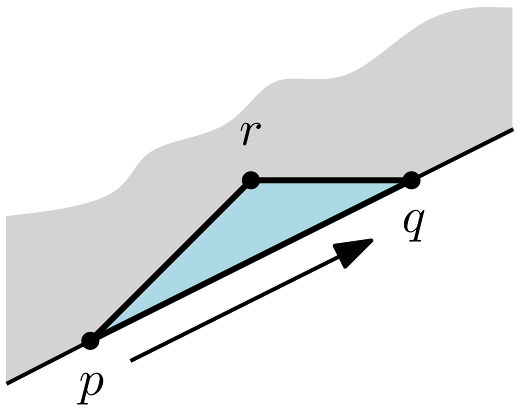
Technische Universität Braunschweig

Sorting by Polar Coordinates

Observation:

- Goal: CCW order
- Sort by polar angle.
- Sufficient: pairwise comparion of points s, q
- Check relative position of q wrt $\overline{p_{\min} s}$





Consequence:

- Predicate: $a \leq b :\Leftrightarrow ((a = p_{\min}) \vee (a = b) \vee (\textsc{LeftTurn}(p_{\min}, a, b) = \textsc{true}))$

Overview

1. Introduction and Definitions
2. Interlude: Algorithmic Paradigms
3. Jarvis' March
4. Quickhull
5. Divide-and-conquer and incremental construction
6. Graham's Scan
7. Optimal output-sensitive construction

Review: Sorting

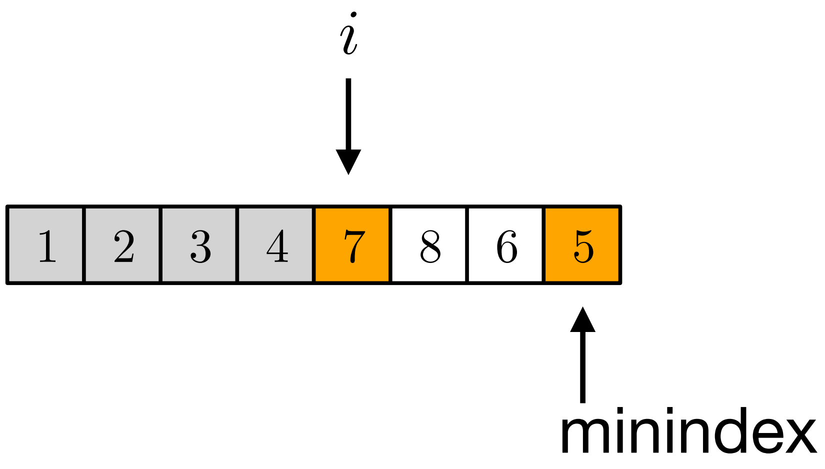
- Algorithms and Data Structures 1
- Various algorithmic paradigms

Sorting algorithms:

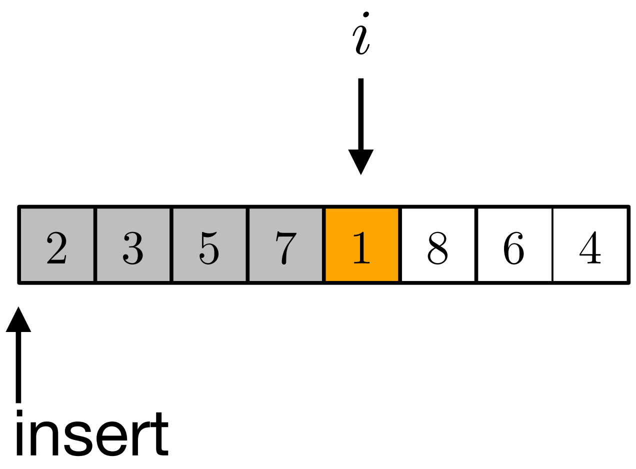
- Incremental methods:
  - Bubble Sort, Selection Sort, Insertion Sort
- Divide-and-conquer methods:
  - Quicksort, Mergesort
- Methods based on data structures:
  - Heapsort, sorting by AVL tree
- Other methods:
  - Bucket Sort, Shellsort, …

Difference: Selection Sort <-> Insertion Sort

- Selection Sort: Search in *unsorted* part of array
- Insertion Sort: Search in *sorted* part of array



Selection Sort

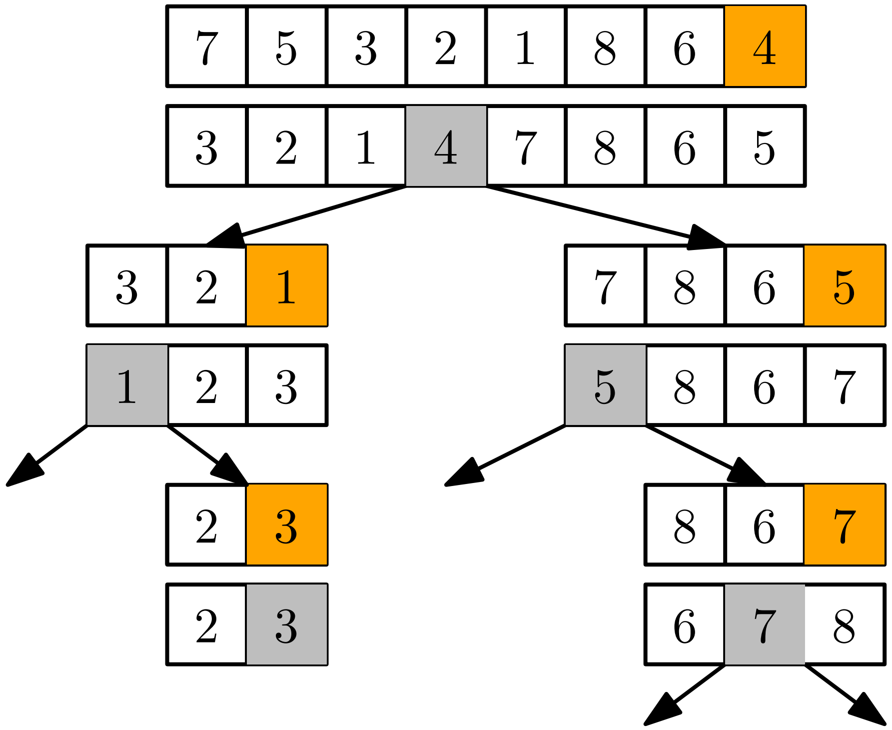


Insertion Sort

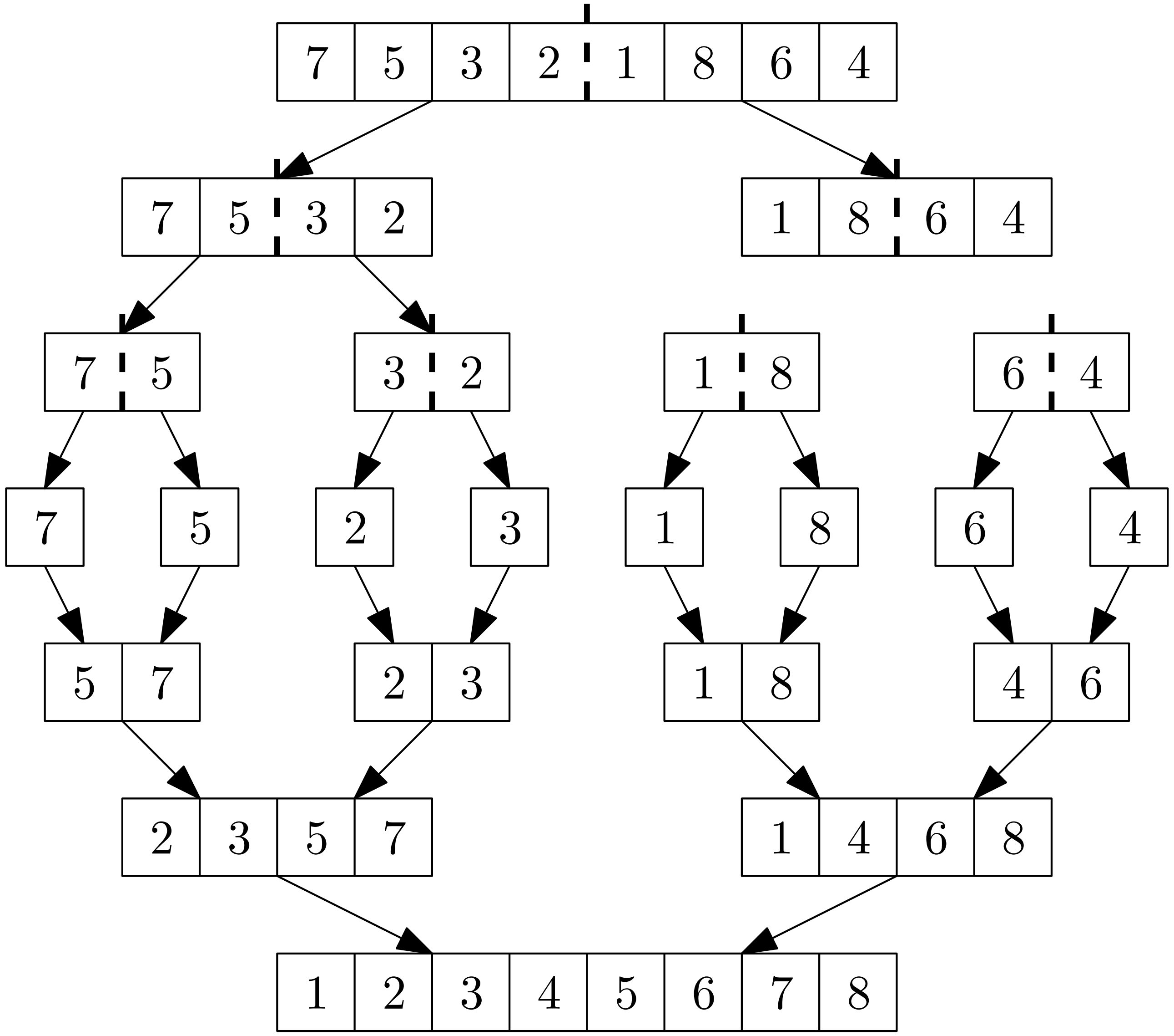
Approach:

- Selection Sort: Find next element for extending the order
- Insertion Sort: Insert next element, such that sequence remains sorted

- Split (Quicksort) or combine (Mergesort)

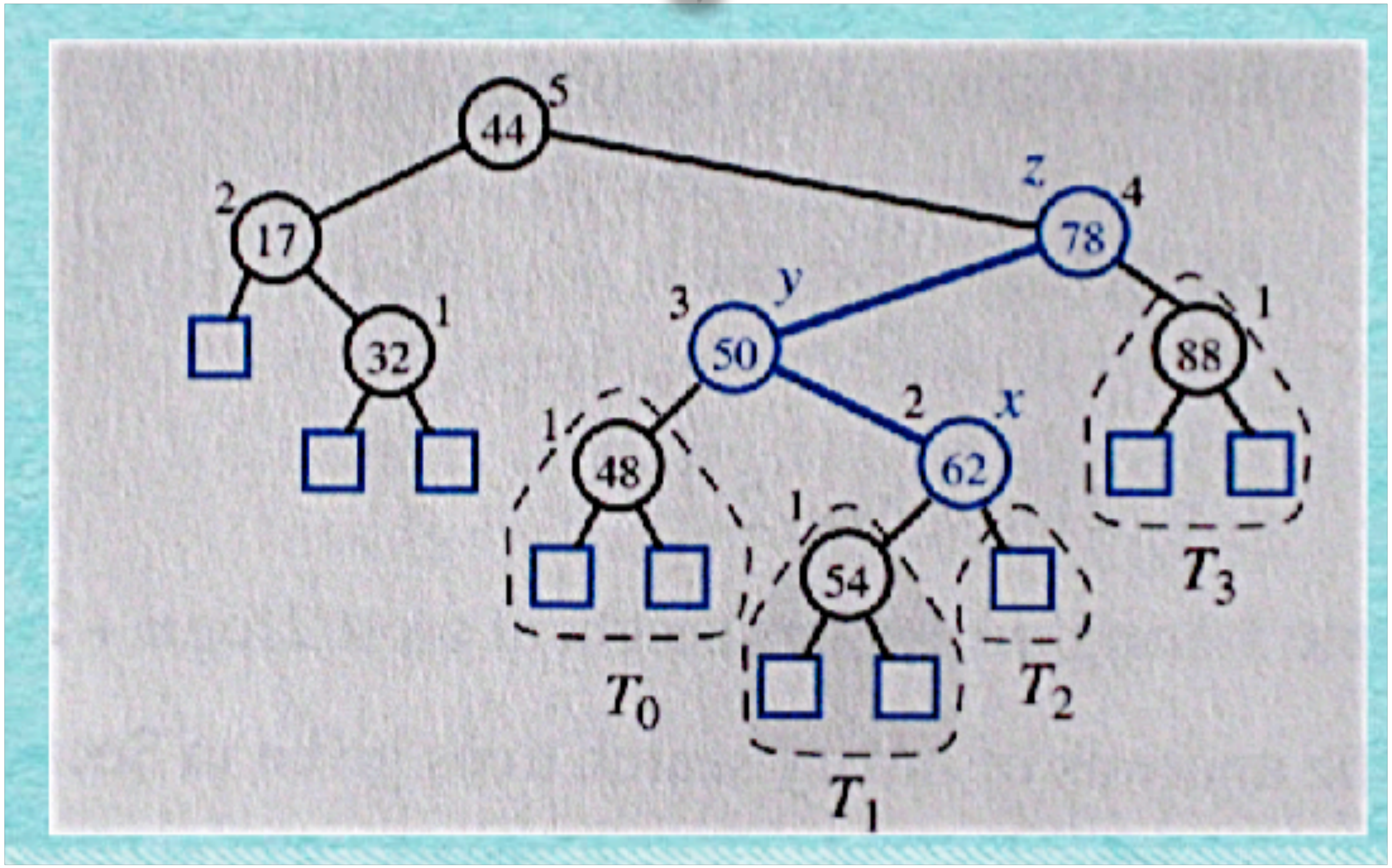


Quicksort



Mergesort

„Algorithms and Data Structures“: AVL-tree



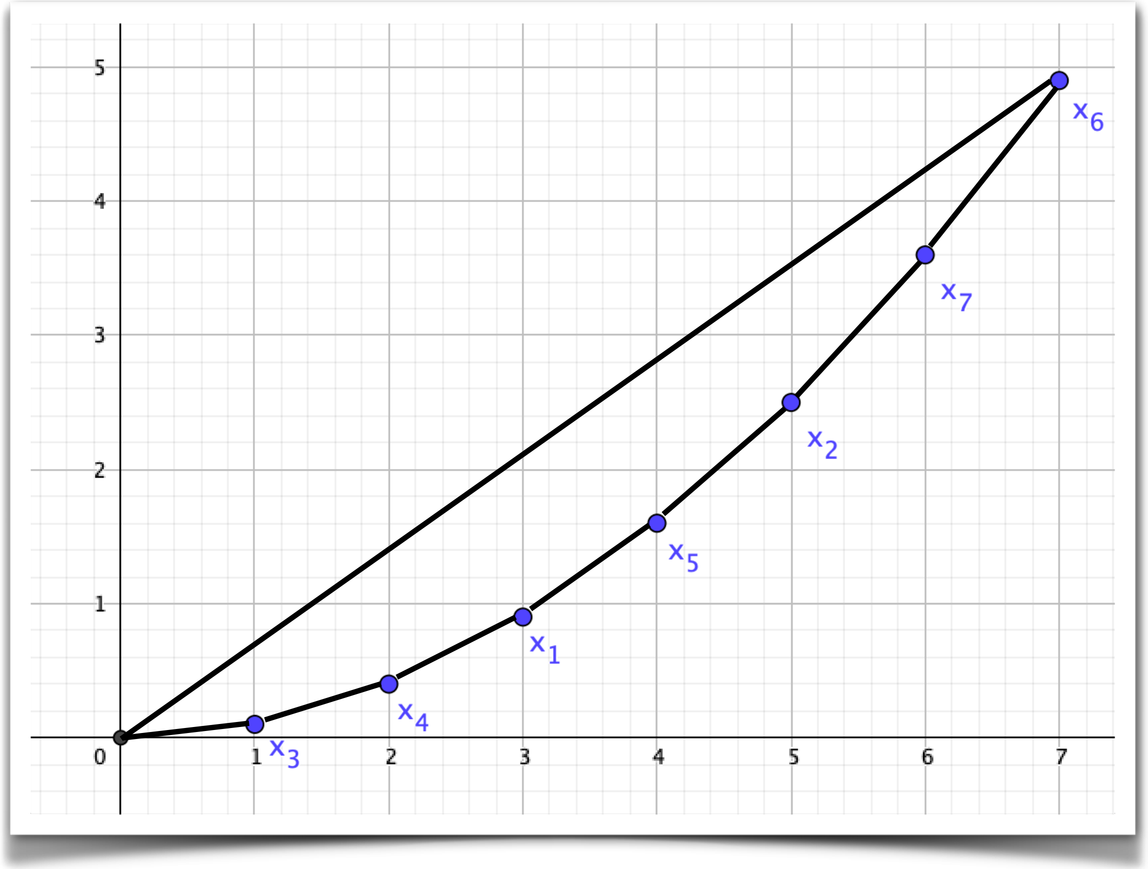
Sorting as a Lower Bound

Theorem 1.18: Computing the convex hull takes $\Omega(n \log n)$ in certain models of computation.

Proof: Recall that comparison-based sorting takes $\Omega(n \log n)$.

Consider a set of n numbers, $x_1, \ldots, x_n$.

Map them to the points $(x_1, x_1^2) \ldots, (x_n, x_n^2)$.



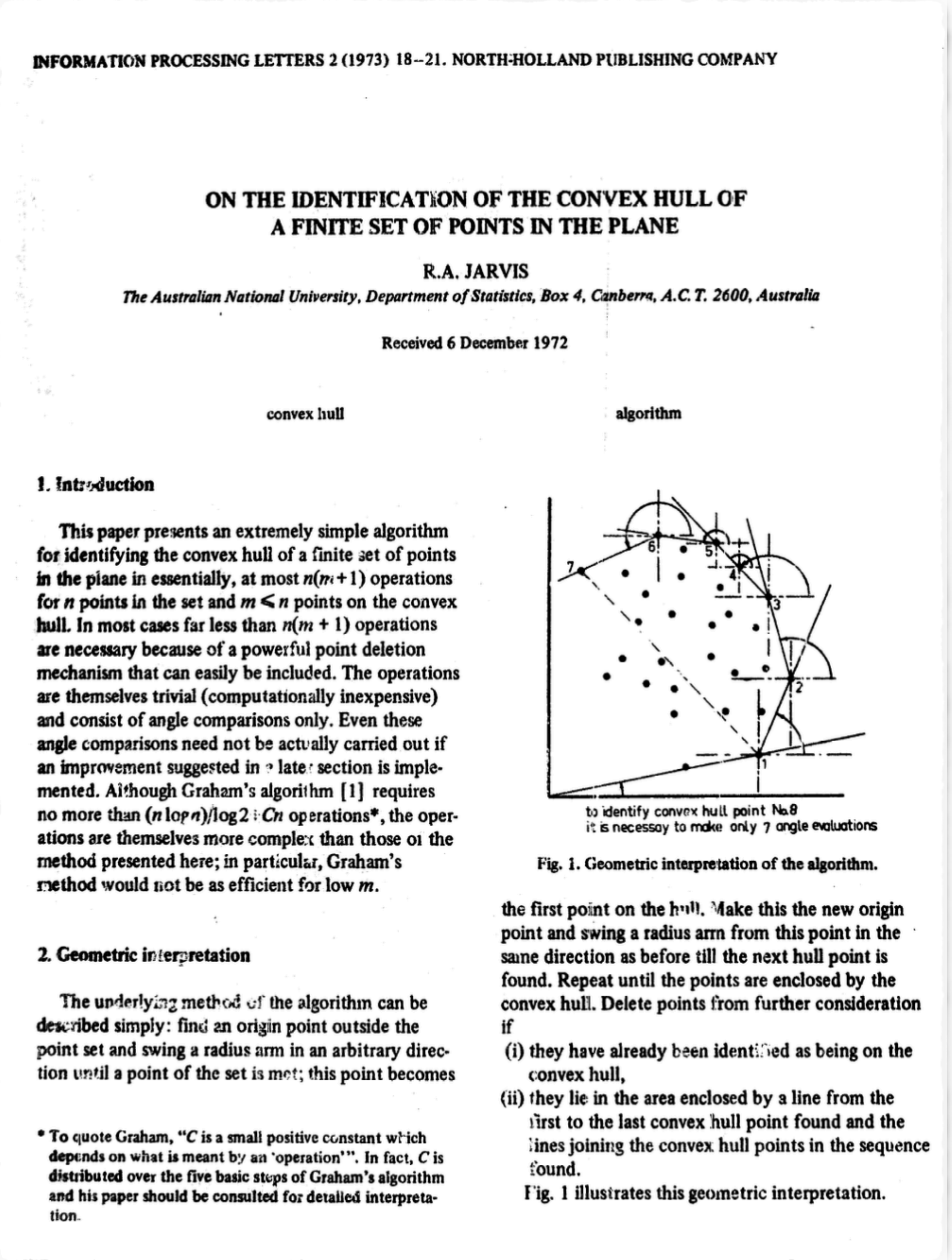
3, 5, 1, 2, 4, 7, 6

The convex hull yields the sorted order of numbers.

Technische Universität Braunschweig

Overview

1. Introduction and Definitions
2. Interlude: Algorithmic Paradigms
3. Jarvis' March
4. Quickhull
5. Divide-and-conquer and incremental construction
6. Graham's Scan
7. Optimal output-sensitive construction



INFORMATION PROCESSING LETTERS 2 (1973) 18–21. NORTH-HOLLAND PUBLISHING COMPANY

ON THE IDENTIFICATION OF THE CONVEX HULL OF A FINITE SET OF POINTS IN THE PLANE

R.A. JARVIS

The Australian National University, Department of Statistics, Box 4, Canberra, A.C.T. 2600, Australia

Received 6 December 1972

convex hull algorithm

1. Introduction

This paper presents an extremely simple algorithm for identifying the convex hull of a finite set of points in the plane in essentially, at most $n(m+1)$ operations for n points in the set and $m \leq n$ points on the convex hull. In most cases far less than $n(m+1)$ operations are necessary because of a powerful point deletion mechanism that can easily be included. The operations are themselves trivial (computationally inexpensive) and consist of angle comparisons only. Even these angle comparisons need not be actually carried out if an improvement suggested in a later section is implemented. Although Graham's algorithm [1] requires no more than $(n \log n)/\log 2 + Cn$ operations*, the operations are themselves more complex than those of the method presented here; in particular, Graham's method would not be as efficient for low m.

2. Geometric interpretation

The underlying method of the algorithm can be described simply: find an origin point outside the point set and swing a radius arm in an arbitrary direction until a point of the set is met; this point becomes the first point on the hull. Make this the new origin point and swing a radius arm from this point in the same direction as before till the next hull point is found. Repeat until the points are enclosed by the convex hull. Delete points from further consideration if

(i) they have already been identified as being on the convex hull,

(ii) they lie in the area enclosed by a line from the first to the last convex hull point found and the lines joining the convex hull points in the sequence found.

Fig. 1 illustrates this geometric interpretation.

to identify convex hull point No.8
it is necessary to make only 7 angle evaluations

Fig. 1. Geometric interpretation of the algorithm.

---

* To quote Graham, "C is a small positive constant which depends on what is meant by an 'operation'". In fact, C is distributed over the five basic steps of Graham's algorithm and his paper should be consulted for detailed interpretation.

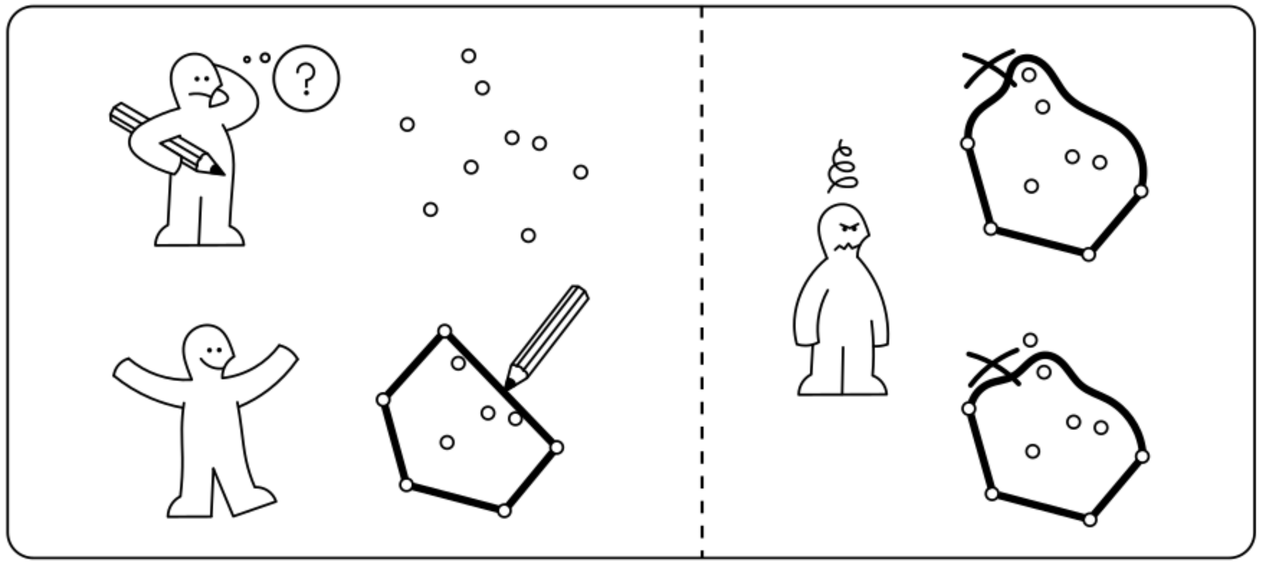
IDEA

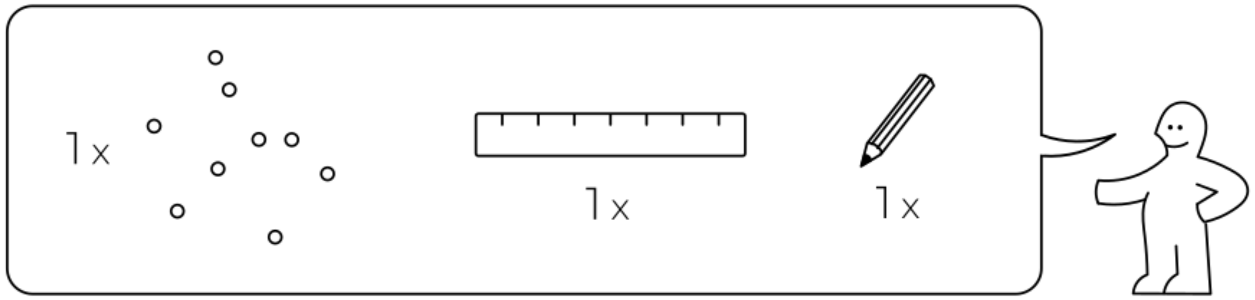
IDEA

A series of nonverbal algorithm assembly instructions.

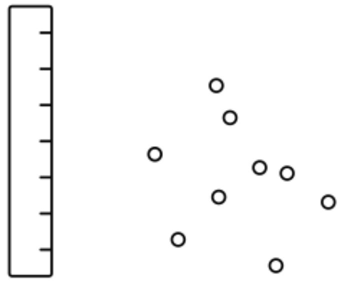
GIFT WRÄPPING

idea-instructions.com/gift-wrapping/
Based on a guest contribution by Christoph Hansknecht – v1.1, CC by-nc-sa 4.0

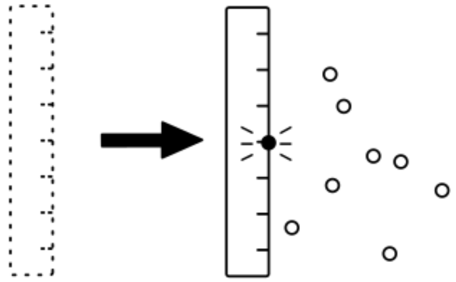




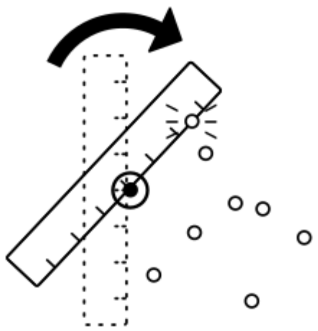
1a



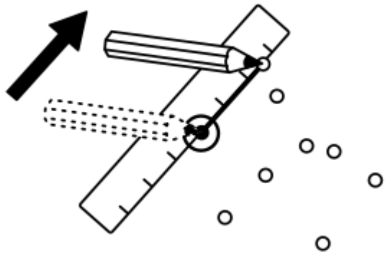
1b



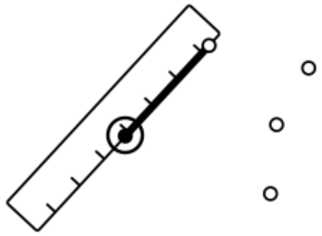
1c



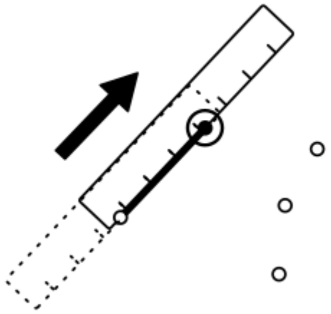
1d



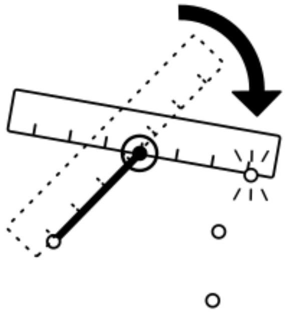
2a



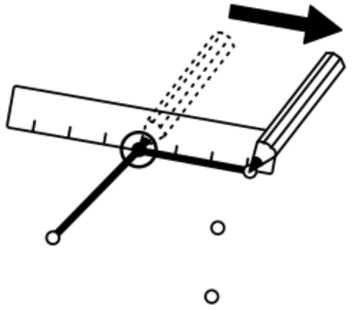
2b



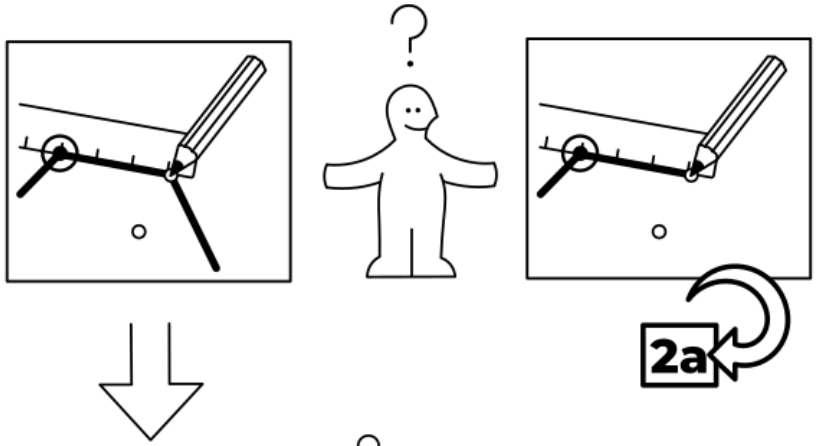
2c



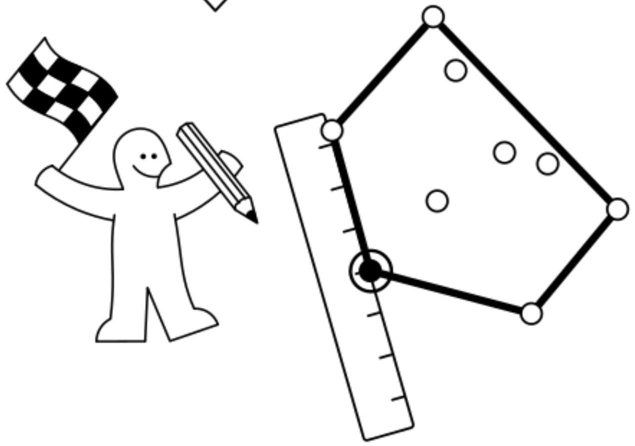
2d



3



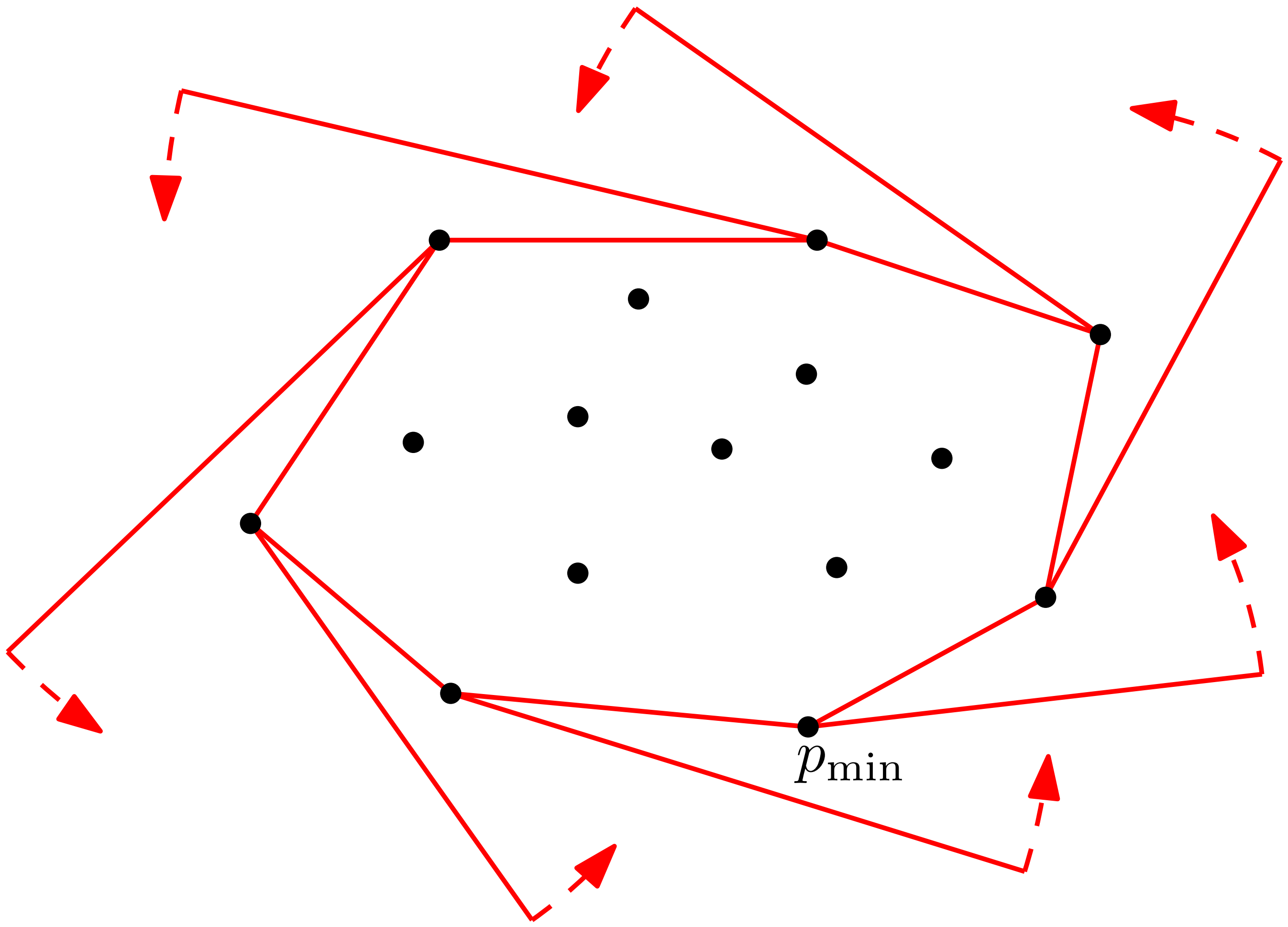
4



Jarvis’ March [Jarvis, 1973]

Basic idea:

- Iteratively find next edge on boundary of $conv(\mathcal{P})$
- Analogy: Selection Sort.
  - Find next element for continuing sorted order
- Start: minimal point $p_{\min}$ wrt $\leq_y$

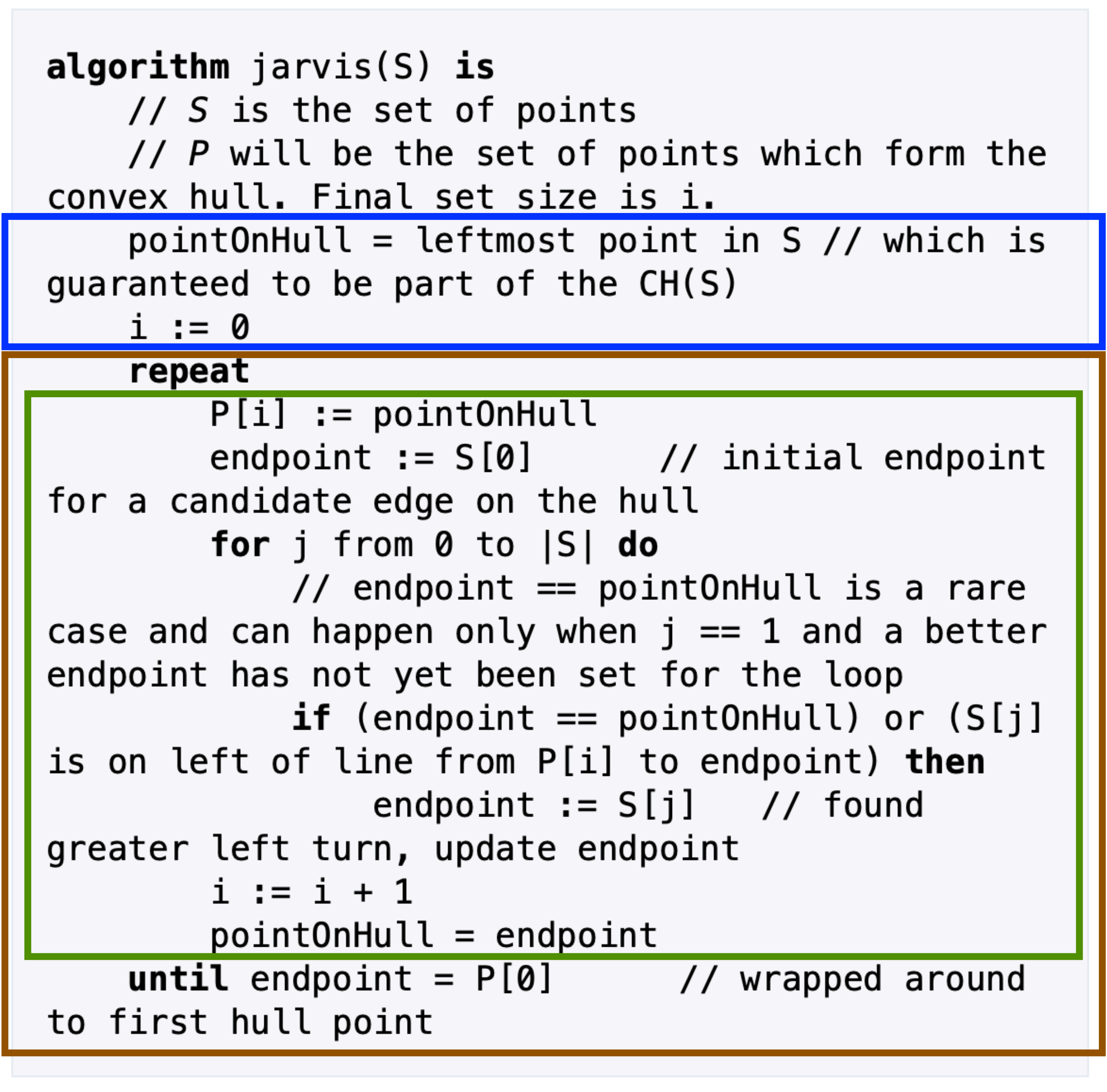


Intuition:

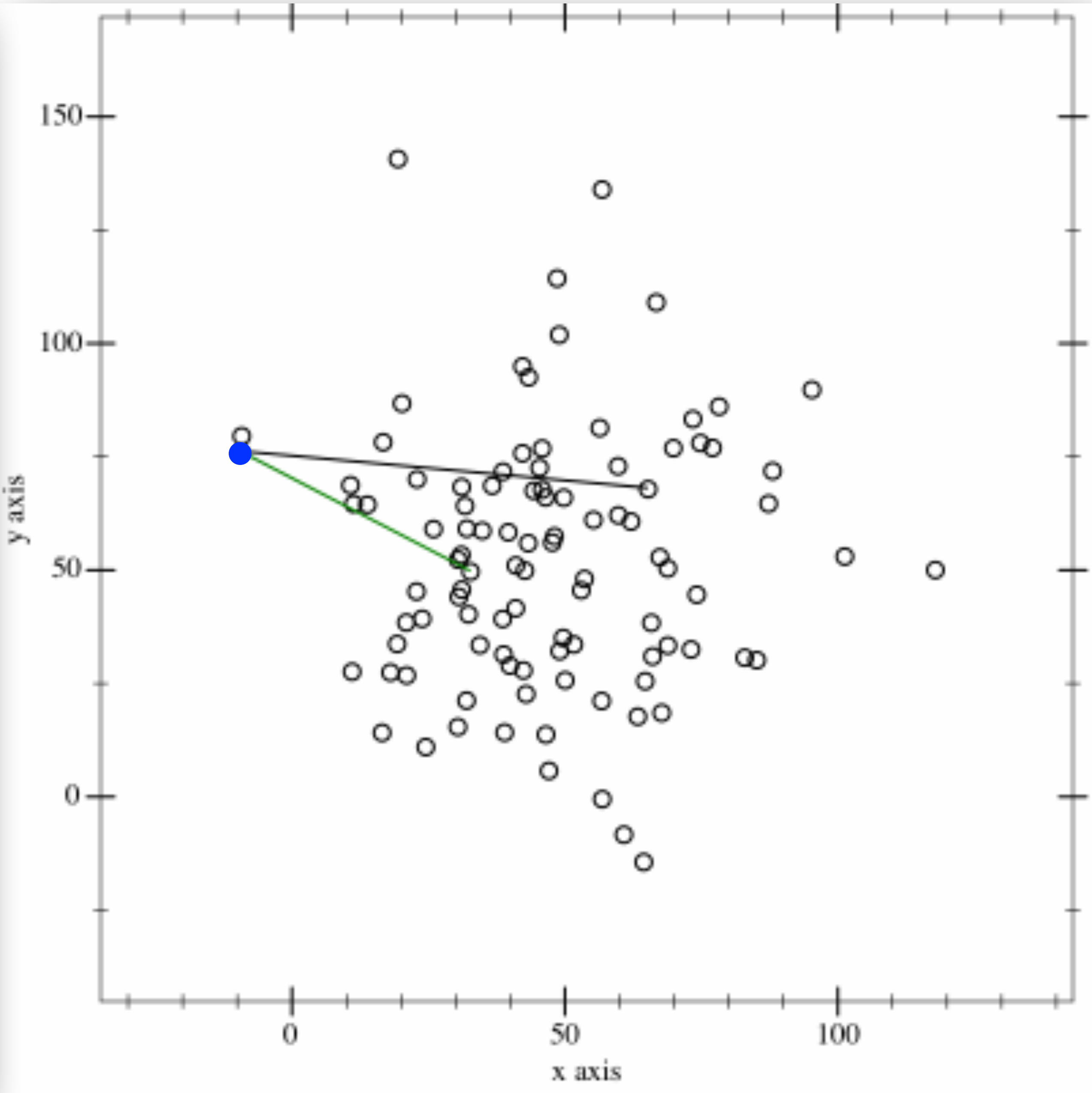
- „Gift wrapping“

Giftwrapping: Pseudocode and Animation

Algorithm 2.9: Compute $conv(\mathcal{P})$ with Jarvis' March.



```
algorithm jarvis(S) is
    // S is the set of points
    // P will be the set of points which form the
convex hull. Final set size is i.
    pointOnHull = leftmost point in S // which is
guaranteed to be part of the CH(S)
    i := 0
    repeat
        P[i] := pointOnHull
        endpoint := S[0]      // initial endpoint
for a candidate edge on the hull
        for j from 0 to |S| do
            // endpoint == pointOnHull is a rare
case and can happen only when j == 1 and a better
endpoint has not yet been set for the loop
            if (endpoint == pointOnHull) or (S[j]
is on left of line from P[i] to endpoint) then
                endpoint := S[j]   // found
greater left turn, update endpoint
        i := i + 1
        pointOnHull = endpoint
    until endpoint = P[0]      // wrapped around
to first hull point
```

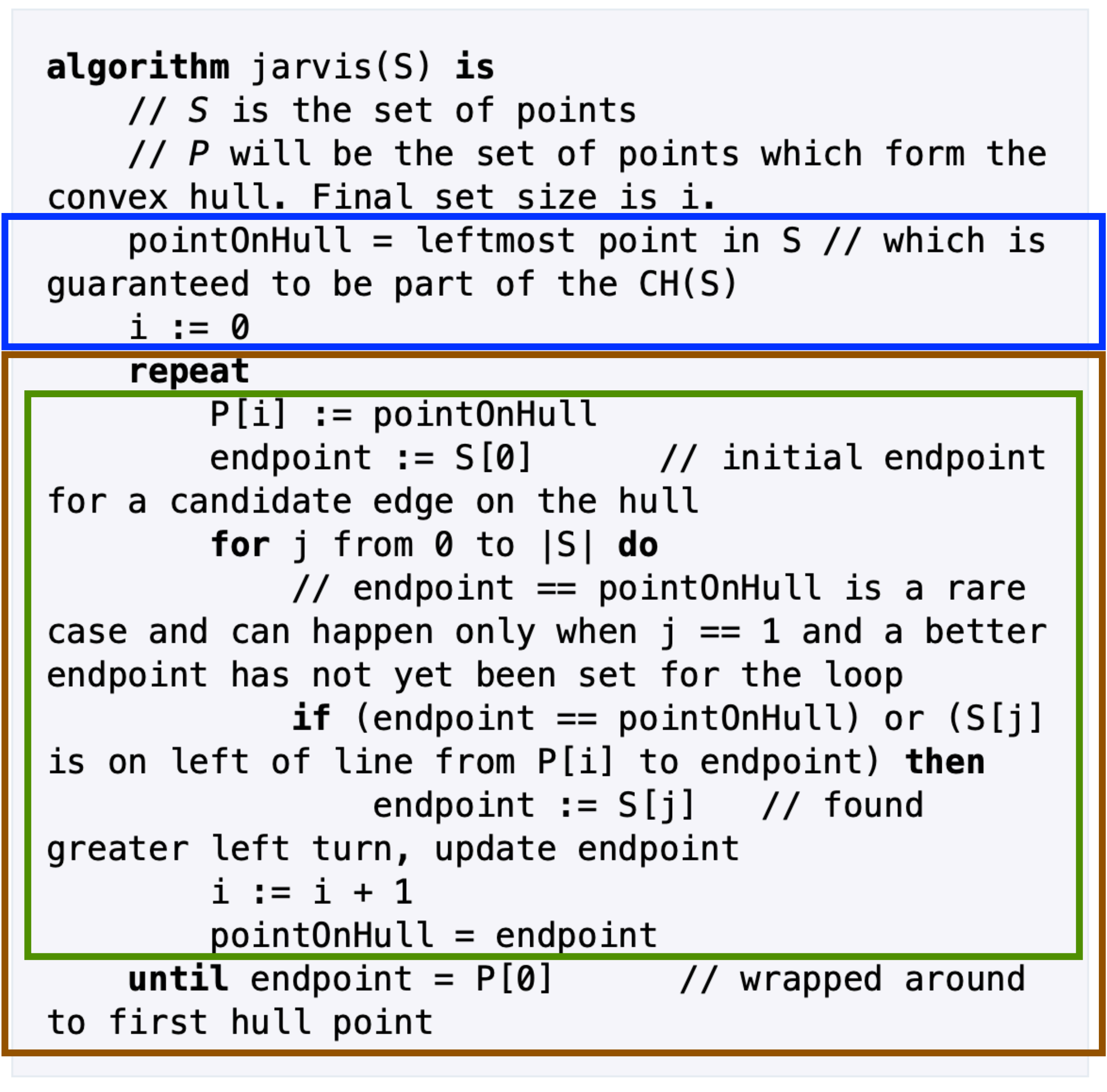


From Wikipedia, the free encyclopedia

By Maonus, CC BY-SA 4.0

Giftwrapping: Pseudocode and Animation

Algorithm 2.9: Compute $conv(\mathcal{P})$ with Jarvis' March.



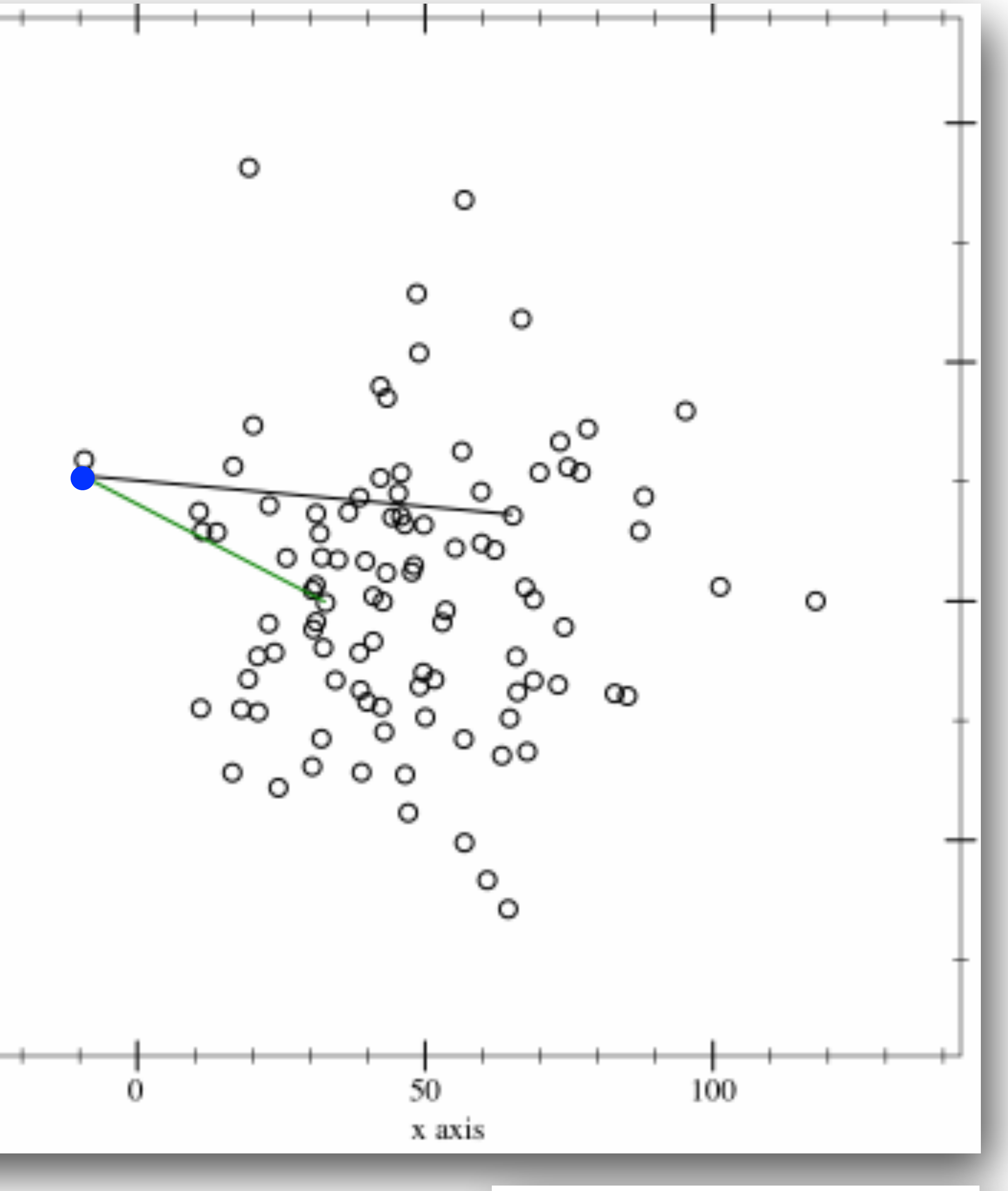
```
algorithm jarvis(S) is
    // S is the set of points
    // P will be the set of points which form the convex hull. Final set size is i.
    pointOnHull = leftmost point in S // which is guaranteed to be part of the CH(S)
    i := 0
    repeat
        P[i] := pointOnHull
        endpoint := S[0]      // initial endpoint for a candidate edge on the hull
        for j from 0 to |S| do
            // endpoint == pointOnHull is a rare case and can happen only when j == 1 and a better endpoint has not yet been set for the loop
            if (endpoint == pointOnHull) or (S[j] is on left of line from P[i] to endpoint) then
                endpoint := S[j]   // found greater left turn, update endpoint
        i := i + 1
        pointOnHull = endpoint
    until endpoint = P[0]      // wrapped around to first hull point
```

$\mathcal{O}(n)$

$\mathcal{O}(n)$

$\mathcal{O}(h)$

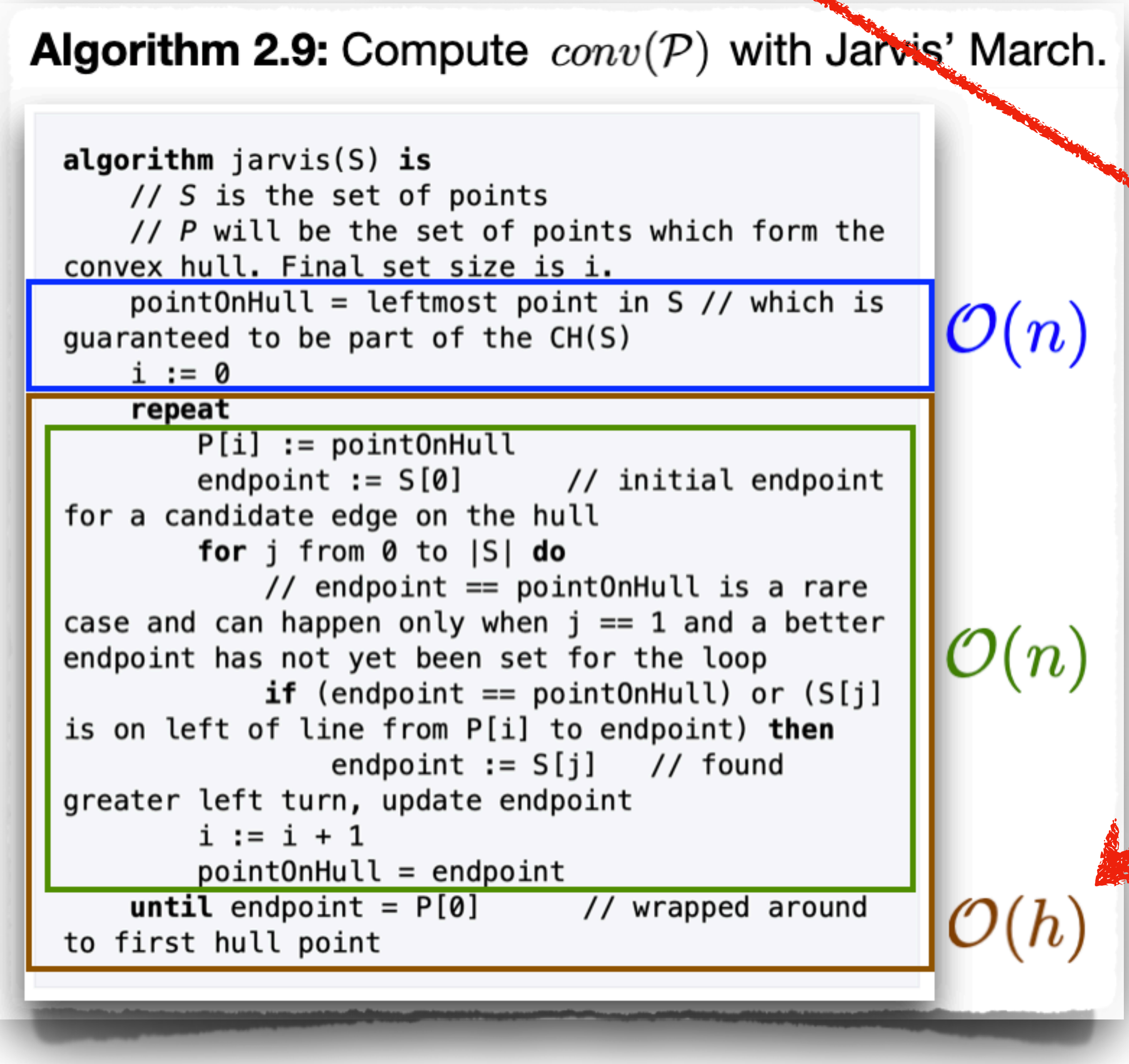
From Wikipedia, the free encyclopedia



By Maonus, CC BY-SA 4.0

Theorem 2.10

Jarvis' March computes the h vertices of $conv(\mathcal{P})$ in $\mathcal{O}(hn)$.



Algorithm 2.9: Compute $conv(\mathcal{P})$ with Jarvis' March.

```
algorithm jarvis(S) is
    // S is the set of points
    // P will be the set of points which form the
convex hull. Final set size is i.
    pointOnHull = leftmost point in S // which is
guaranteed to be part of the CH(S)
    i := 0
    repeat
        P[i] := pointOnHull
        endpoint := S[0]      // initial endpoint
for a candidate edge on the hull
        for j from 0 to |S| do
            // endpoint == pointOnHull is a rare
case and can happen only when j == 1 and a better
endpoint has not yet been set for the loop
            if (endpoint == pointOnHull) or (S[j]
is on left of line from P[i] to endpoint) then
                endpoint := S[j]   // found
greater left turn, update endpoint
        i := i + 1
        pointOnHull = endpoint
    until endpoint = P[0]      // wrapped around
to first hull point
```

$\mathcal{O}(n)$

$\mathcal{O}(n)$

$\mathcal{O}(h)$

output-sensitive

Overview

1. Introduction and Definitions
2. Interlude: Algorithmic Paradigms
3. Jarvis' March
4. **Quickhull**
5. Divide-and-conquer and incremental construction
6. Graham's Scan
7. Optimal output-sensitive construction

Quickhull [Eddy, 1977]

A New Convex Hull Algorithm for Planar Sets

WILLIAM F. EDDY
Carnegie-Mellon University

A new algorithm, CONVEX, that determines which points of a planar set are vertices of the convex hull of the set is presented. It is shown that CONVEX operates in a fashion similar to the sorting algorithm QUICKERSORT. Evidence is given which indicates that in some situations CONVEX is preferable to earlier algorithms. A Fortran implementation, intended to minimize execution time, is presented and an alternative, which minimizes storage requirements, is discussed.

Key Words and Phrases: convex hull, QUICKERSORT, partitioning, sorting
CR Categories: 5.30, 5.31
The Algorithm: CONVEX, A New Convex Hull Algorithm for Planar Sets. *ACM Trans. Math. Software 3*, 4 (Dec. 1977), 411–412.

INTRODUCTION

The convex hull of a planar set is the minimum-area convex polygon containing the planar set. A convex polygon is clearly determined by its vertices. Graham [1] suggests an algorithm for determining which points of a planar set are vertices of its convex hull. Because his algorithm requires sorting the points, if there are N points then at least $O(N \log N)$ operations are needed to determine the vertices. Recently, Preparata and Hong [3, 4] have shown that there exist sets of points for which every algorithm requires at least $O(N \log N)$ operations to determine the vertices of the convex hull. Jarvis [2] gives an algorithm which requires $O(N \cdot C)$ operations, where C is the number of vertices. For some configurations of the points in the plane (those with small values of C) the algorithm given by Jarvis will be faster than the algorithm of Graham; for other configurations it may be slower. An adaptive algorithm, CONVEX, is presented here which never requires more than $O(N \cdot C)$ operations to determine the vertices of the convex hull and may require substantially fewer. However, CONVEX may require more operations than Graham's algorithm for some configurations of points. Evidence is presented which suggests that in applications CONVEX is preferable to the "sorting" algorithms [1, 3, 4] and to Jarvis's algorithm [2].

METHOD

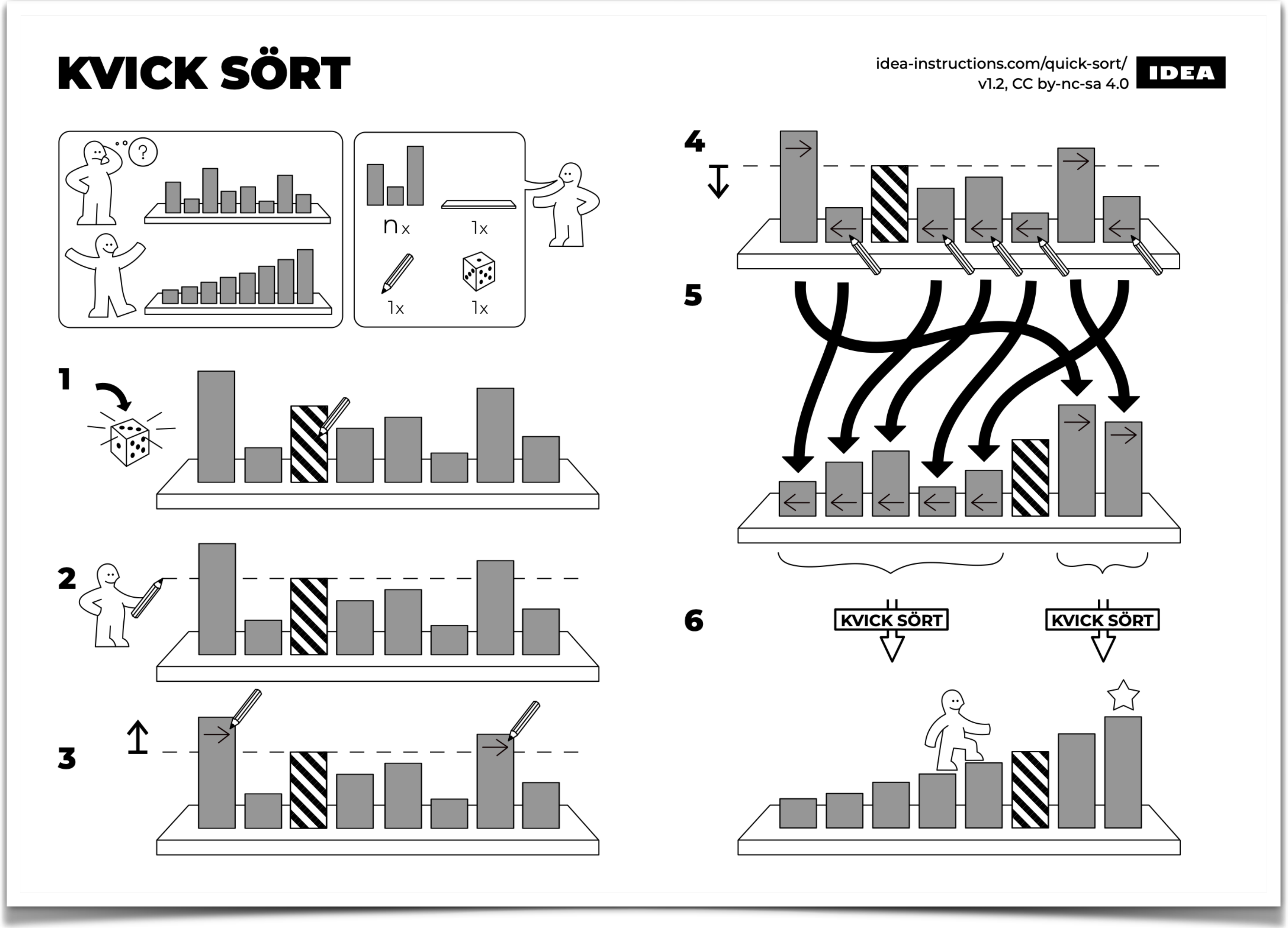
Operationally, this algorithm is analogous to the sorting algorithm QUICKERSORT [5]. At each step QUICKERSORT partitions the input array with respect to a

---

This research was supported in part by the National Science Foundation under Grant DCR75-08374.
Author's address: Department of Statistics, Carnegie-Mellon University, Schenley Park, Pittsburgh, PA 15213.

ACM Transactions on Mathematical Software, Vol. 3, No. 4, December 1977, Pages 398–403.

Quickhull [Eddy, 1977]



Technische
Universität
Braunschweig

Quickhull [Eddy, 1977]

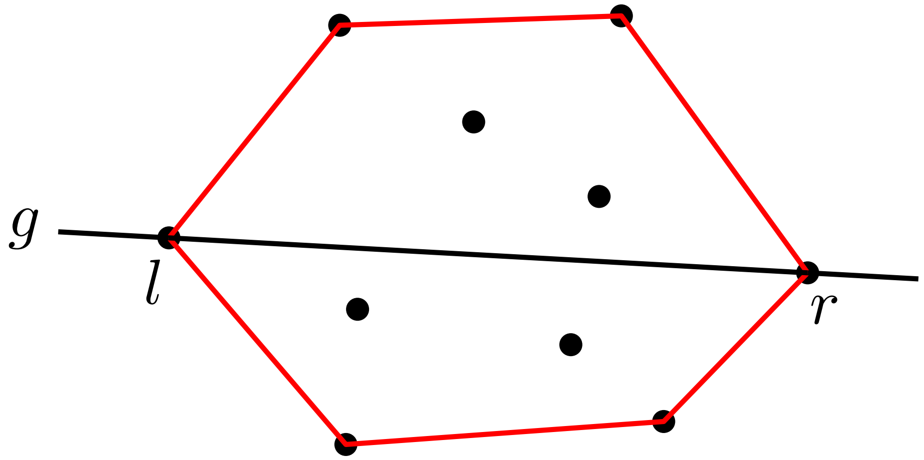
Basic idee:

- Use pivot element for subdivision into independent subsets
- Analogy: Quicksort
  - Pivot element $m \in A$: $A \to A_{<m} \circ A_{=m} \circ A_{>m}$
  - Concatenate subsequences

Transfer to $\mathbb{R}^2$:

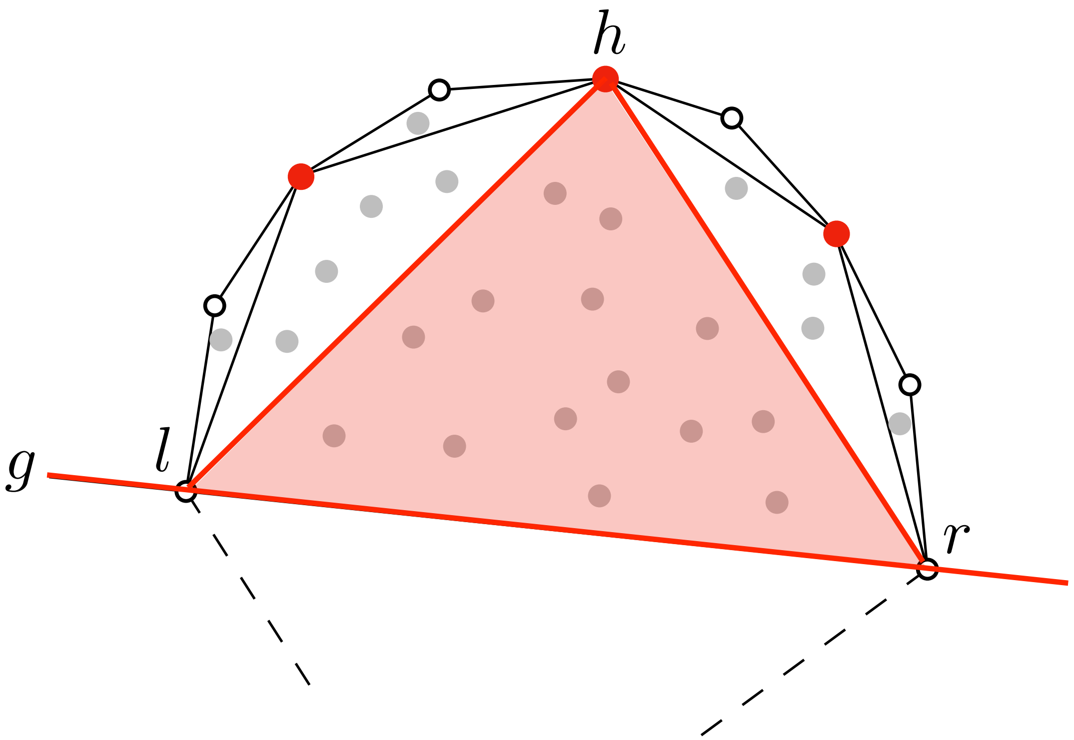
- Separation of $\mathcal{P}$ by line g
- $g :=$ line through extreme points l, r
- Concatenation of recursively computed hull

$\rightarrow conv(\mathcal{P})$

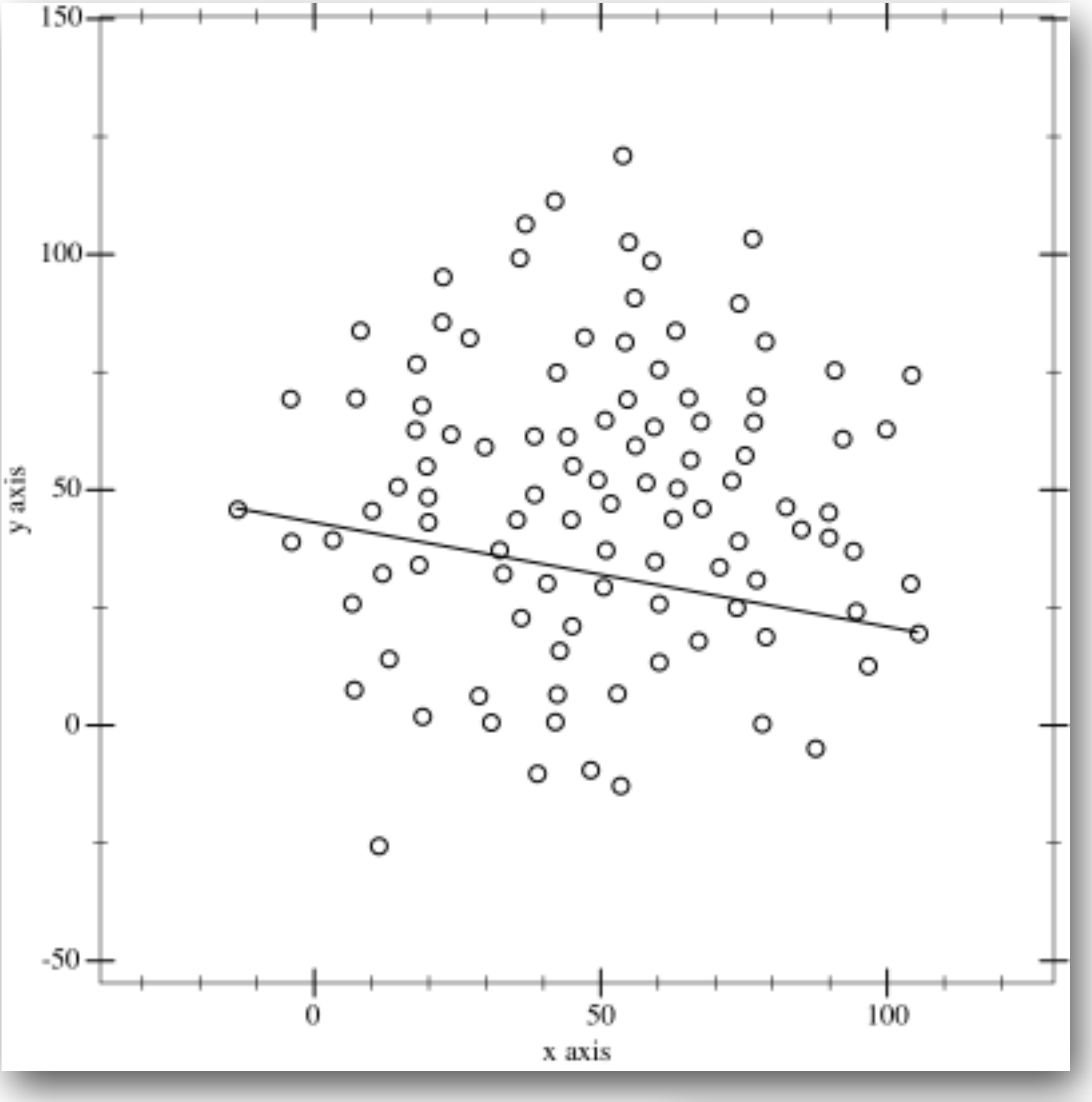


Choosing the pivot element:

- Points above g
- Auxiliary point h: maximal distance to g.
- New pivot elements: $\overline{lh}, \overline{hr}$
- Delete $\mathcal{P} \cap \triangle(l, r, h)$.
- Two recursions:
  Above $\overline{lh}$ and below $\overline{rh}$
- „Exhaustion from inside“
- Analogously below g



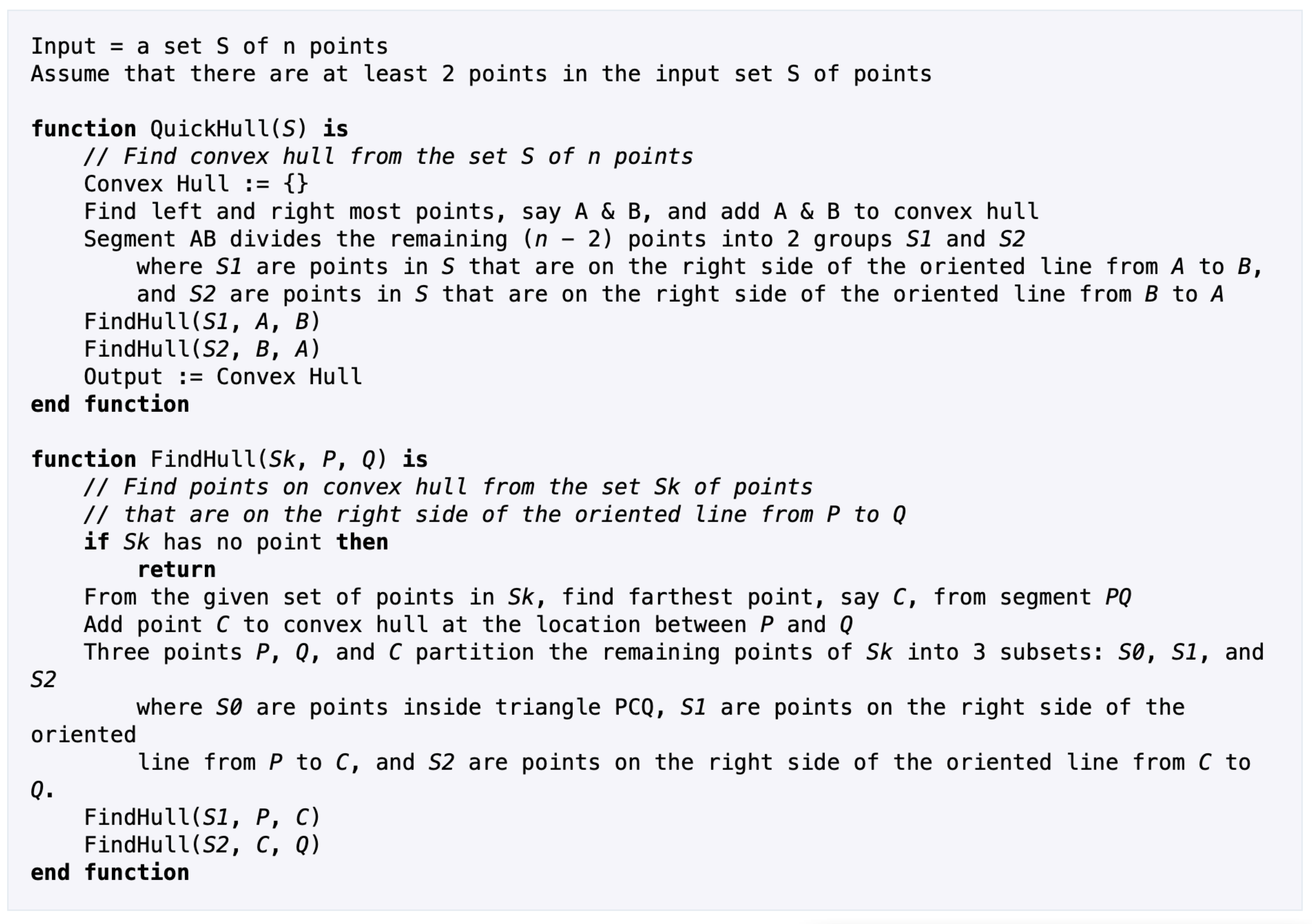
Quickhull: Animation



By Maonus, CC BY-SA 4.0

Quickhull: Pseudocode

Algorithm 2.11: Compute $conv(\mathcal{P})$ with Quickhull.



```
Input = a set S of n points
Assume that there are at least 2 points in the input set S of points

function QuickHull(S) is
    // Find convex hull from the set S of n points
    Convex Hull := {}
    Find left and right most points, say A & B, and add A & B to convex hull
    Segment AB divides the remaining (n − 2) points into 2 groups S1 and S2
        where S1 are points in S that are on the right side of the oriented line from A to B,
        and S2 are points in S that are on the right side of the oriented line from B to A
    FindHull(S1, A, B)
    FindHull(S2, B, A)
    Output := Convex Hull
end function

function FindHull(Sk, P, Q) is
    // Find points on convex hull from the set Sk of points
    // that are on the right side of the oriented line from P to Q
    if Sk has no point then
        return
    From the given set of points in Sk, find farthest point, say C, from segment PQ
    Add point C to convex hull at the location between P and Q
    Three points P, Q, and C partition the remaining points of Sk into 3 subsets: S0, S1, and
S2
        where S0 are points inside triangle PCQ, S1 are points on the right side of the
oriented
        line from P to C, and S2 are points on the right side of the oriented line from C to
Q.
    FindHull(S1, P, C)
    FindHull(S2, C, Q)
end function
```

From Wikipedia, the free encyclopedia

Theorem 2.12
Quickhull computes $conv(\mathcal{P})$ in $\mathcal{O}(n^2)$ worst-case and in $\mathcal{O}(n \log n)$ best-case runtime.

Excercises:

- Details of implementation
- Termination
- Runtime

Thank you!

Technische
Universität
Braunschweig