(b) the set of all convex combinations of points from P*;*

(c) the set of all convex combinations formed by $d+1$ *or fewer points from* P*;*

(d) the intersection of all convex supersets of P*;*

(e) the intersection of all closed halfspaces containing P*.*

Exercise 4.16 *Prove Theorem 4.15.*

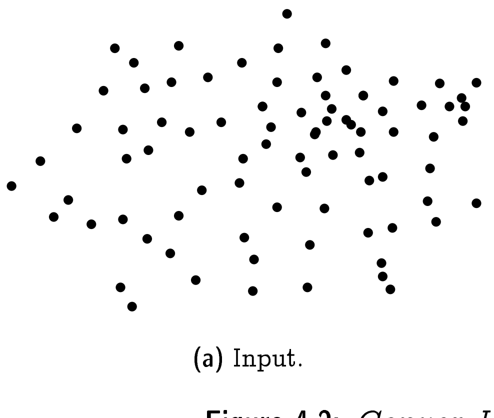
4.3 Planar Convex Hull

Although we know by now what is the convex hull of point set, it is not yet clear how to construct it algorithmically. As a first step, we have to find a suitable representation for convex hulls. In this section we focus on the problem in $\mathbb{R}^2$, where the convex hull of a finite point set forms a convex polygon. A convex polygon is easy to represent, for instance, as a sequence of its vertices in counterclockwise orientation. In higher dimensions finding a suitable representation for convex polytopes is a much more delicate task.

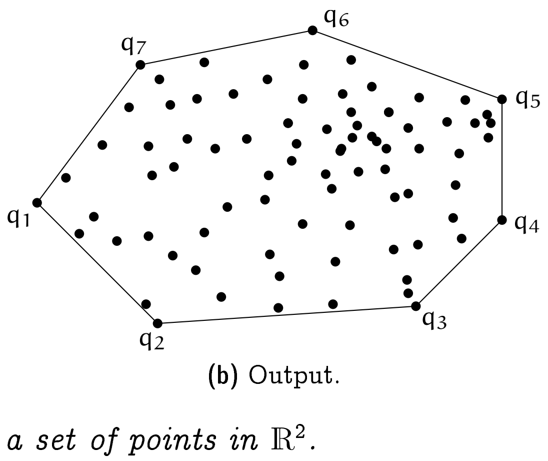
Problem 4.17 (Convex hull)

Input: $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$, $n \in \mathbb{N}$.

Output: Sequence $(q_1, \ldots, q_h)$, $1 \leqslant h \leqslant n$, of the vertices of $\operatorname{conv}(P)$ (ordered counterclockwise).



(a) Input.



(b) Output.

Figure 4.2: *Convex Hull of a set of points in* $\mathbb{R}^2$*.*

Another possible algorithmic formulation of the problem is to ignore the structure of the convex hull and just consider it as a point set.

Problem 4.18 (Extremal points)

Input: $P = \{p_1, \dots, p_n\} \subset \mathbb{R}^2$, $n \in \mathbb{N}$.

Output: Set $Q \subseteq P$ of the vertices of $\operatorname{conv}(P)$.

Degeneracies. A couple of further clarifications regarding the above problem definitions are in order.

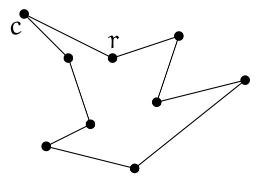
First of all, for efficiency reasons an input is usually specified as a sequence of points. Do we insist that this sequence forms a set or are duplications of points allowed?

What if three points are collinear? Are all of them considered extremal? According to our definition from above, they are not and that is what we will stick to. But note that there may be cases where one wants to include all such points, nevertheless.

By the Separation Theorem, every extremal point p can be separated from the convex hull of the remaining points by a halfplane. If we take such a halfplane and translate its defining line such that it passes through p, then all points from P other than p should lie in the resulting open halfplane. In $\mathbb{R}^2$ it turns out convenient to work with the following "directed" reformulation.

Proposition 4.19 *A point $p \in P = \{p_1, \dots, p_n\} \subset \mathbb{R}^2$ is* **extremal** *for $P \iff$ there is a directed line g through p such that $P \setminus \{p\}$ is to the left of g.*

The *interior angle* at a vertex v of a polygon P is the angle between the two edges of P incident to v whose corresponding angular domain lies in P°. If this angle is smaller than π, the vertex is called *convex*; if the angle is larger than π, the vertex is called *reflex*. For instance, the vertex c in the polygon depicted to the right is a convex vertex, whereas the vertex labeled r is a reflex vertex.



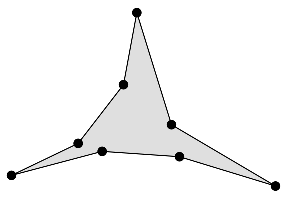
Exercise 4.20
A set $S \subset \mathbb{R}^2$ is star-shaped *if there exists a point $c \in S$, such that for every point $p \in S$ the line segment $\overline{cp}$ is contained in S. A simple polygon with exactly three convex vertices is called a pseudotriangle (see the example shown on the right).*
In the following we consider subsets of $\mathbb{R}^2$. Prove or disprove:

a) *Every convex vertex of a simple polygon lies on its convex hull.*

b) *Every star-shaped set is convex.*

c) *Every convex set is star-shaped.*

d) *The intersection of two convex sets is convex.*



e) The union of two convex sets is convex.

f) The intersection of two star-shaped sets is star-shaped.

g) The intersection of a convex set with a star-shaped set is star-shaped.

h) Every triangle is a pseudotriangle.

i) Every pseudotriangle is star-shaped.

4.4 Trivial algorithms

One can compute the extremal points using Carathéodory's Theorem as follows: Test for every point $p \in P$ whether there are $q, r, s \in P \setminus \{p\}$ such that p is inside the triangle with vertices q, r, and s. Runtime $O(n^4)$.

Another option, inspired by the Separation Theorem: test for every pair $(p, q) \in P^2$ whether all points from $P \setminus \{p, q\}$ are to the left of the directed line through p and q (or on the line segment $\overline{pq}$). Runtime $O(n^3)$.

Exercise 4.21 *Let* $P = (p_0, \dots, p_{n-1})$ *be a sequence of* n *points in* $\mathbb{R}^2$. *Someone claims that you can check by means of the following algorithm whether or not* P *describes the boundary of a convex polygon in counterclockwise order:*

```
bool is_convex($p_0, \dots, p_{n-1}$) {
    for $i = 0, \dots, n-1$:
        if ($p_i$, $p_{(i+1) \bmod n}$, $p_{(i+2) \bmod n}$) form a rightturn:
            return false;
    return true;
}
```

Disprove the claim and describe a correct algorithm to solve the problem.

Exercise 4.22 *Let* $P \subset \mathbb{R}^2$ *be a convex polygon, given as an array p[0]...p[n-1] of its* n *vertices in counterclockwise order.*

a) Describe an $O(\log(n))$ *time algorithm to determine whether a point* q *lies inside, outside or on the boundary of* P.

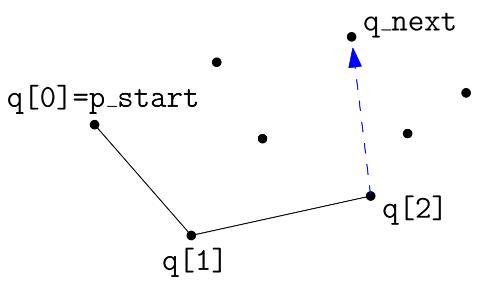
b) Describe an $O(\log(n))$ *time algorithm to find a (right) tangent to* P *from a query point* q *located outside* P. *That is, find a vertex* $p[i]$, *such that* P *is contained in the closed halfplane to the left of the oriented line* $qp[i]$.

4.5 Jarvis' Wrap

We are now ready to describe a first simple algorithm to construct the convex hull. It works as follows:

> Find a point p_1 that is a vertex of $\text{conv}(P)$ (e.g., the one with smallest x-coordinate). "Wrap" P starting from p_1, i.e., always find the next vertex of $\text{conv}(P)$ as the one that is rightmost with respect to the direction given by the previous two vertices.

Besides comparing x-coordinates, the only geometric primitive needed is an *orientation* test: Denote by $\texttt{rightturn}(p, q, r)$, for three points $p, q, r \in \mathbb{R}^2$, the predicate that is true if and only if r is (strictly) to the right of the oriented line pq.



Code for Jarvis' Wrap.

p[0..N) contains a sequence of N points.
p_start point with smallest x-coordinate.
q_next some *other* point in p[0..N).

```
int h = 0;
Point_2 q_now = p_start;
do {
  q[h] = q_now;
  h = h + 1;

  for (int i = 0; i < N; i = i + 1)
    if (rightturn_2(q_now, q_next, p[i]))
      q_next = p[i];

  q_now = q_next;
  q_next = p_start;
} while (q_now != p_start);
```

q[0,h) describes a convex polygon bounding the convex hull of p[0..N).

Analysis. For every output point the above algorithm spends n rightturn tests, which is $\Rightarrow O(nh)$ in total.

Theorem 4.23 [6] *Jarvis' Wrap computes the convex hull of* n *points in* $\mathbb{R}^2$ *using* $O(nh)$ *rightturn tests, where* h *is the number of hull vertices.*

In the worst case we have $h = n$, that is, $O(n^2)$ rightturn tests. Jarvis' Wrap has a remarkable property that is called *output sensitivity*: the runtime depends not only on the size of the input but also on the size of the output. For a huge point set it constructs the convex hull in optimal linear time, if the convex hull consists of a constant number of vertices only. Unfortunately the worst case performance of Jarvis' Wrap is suboptimal, as we will see soon.

Degeneracies. The algorithm may have to cope with various degeneracies.

- Several points have smallest x-coordinate $\Rightarrow$ lexicographic order:

$$(p_x, p_y) < (q_x, q_y) \iff p_x < q_x \vee p_x = q_x \wedge p_y < q_y \,.$$

- Three or more points collinear $\Rightarrow$ choose the point that is farthest among those that are rightmost.

Predicates. Besides the lexicographic comparison mentioned above, the Jarvis' Wrap (and most other 2D convex hull algorithms for that matter) need one more geometric predicate: the rightturn or—more generally—orientation test. The computation amounts to evaluating a polynomial of degree two, see the exercise below. We therefore say that the orientation test has *algebraic degree* two. In contrast, the lexicographic comparison has degree one only. The algebraic degree not only has a direct impact on the efficiency of a geometric algorithm (lower degree $\leftrightarrow$ less multiplications), but also an indirect one because high degree predicates may create large intermediate results, which may lead to overflows and are much more costly to compute with exactly.

Exercise 4.24 *Prove that for three points* $(p_x, p_y), (q_x, q_y), (r_x, r_y) \in \mathbb{R}^2$*, the sign of the determinant*

$$\begin{vmatrix} 1 & p_x & p_y \\ 1 & q_x & q_y \\ 1 & r_x & r_y \end{vmatrix}$$

determines if r *lies to the right, to the left or on the directed line through* p *and* q.

Exercise 4.25 *The InCircle predicate is: Given three points* $p, q, r \in \mathbb{R}^2$ *that define a circle* C *and a fourth point* s*, is* s *located inside* C *or not? The goal of this exercise is to derive an algebraic formulation of the incircle predicate in form of a determinant, similar to the formulation of the orientation test given above in*

Exercise 4.24. To this end we employ the so-called parabolic lifting map, which will also play a prominent role in the next chapter of the course.

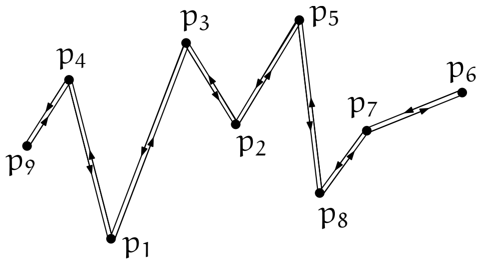
The parabolic lifting map $\ell : \mathbb{R}^2 \to \mathbb{R}^3$ *is defined for a point* $p = (x, y) \in \mathbb{R}^2$ *by* $\ell(p) = (x, y, x^2 + y^2)$. *For a circle* $C \subseteq \mathbb{R}^2$ *of positive radius, show that the "lifted circle"* $\ell(C) = \{\ell(p) \mid p \in C\}$ *is contained in a unique plane* $h_C \subseteq \mathbb{R}^3$. *Moreover, show that a point* $p \in \mathbb{R}^2$ *is strictly inside (outside, respectively) of* C *if and only if the lifted point* $\ell(p)$ *is strictly below (above, respectively)* h_C.

Use these insights to formulate the InCircle predicate for given points $(p_x, p_y), (q_x, q_y), (r_x, r_y$ $\mathbb{R}^2$ *as a determinant.*

4.6 Graham Scan (Successive Local Repair)

There exist many algorithms that exhibit a better worst-case runtime than Jarvis' Wrap. Here we discuss only one of them: a particularly elegant and easy-to-implement variant of the so-called *Graham Scan* [5]. This algorithm is referred to as *Successive Local Repair* because it starts with some polygon enclosing all points and then step-by-step repairs the deficiencies of this polygon, by removing non-convex vertices. It goes as follows:

Sort points lexicographically and remove duplicates: $(p_1, \ldots, p_n)$.



$p_9\ p_4\ p_1\ p_3\ p_2\ p_5\ p_8\ p_7\ p_6\ p_7\ p_8\ p_5\ p_2\ p_3\ p_1\ p_4\ p_9$

As long as there is a (consecutive) triple (p, q, r) such that r is to the right of or on the directed line $\overrightarrow{pq}$, remove q from the sequence.

Code for Graham Scan.

`p[0..N)` lexicographically sorted sequence of pairwise distinct points, $N \geqslant 2$.

```
q[0] = p[0];
int h = 0;
// Lower convex hull (left to right):
for (int i = 1; i < N; i = i + 1) {
  while (h>0 && !leftturn_2(q[h-1], q[h], p[i]))
    h = h - 1;
  h = h + 1;
```

```
    q[h] = p[i];
  }

  // Upper convex hull (right to left):
  for (int i = N-2; i >= 0; i = i - 1) {
    while (!leftturn_2(q[h-1], q[h], p[i]))
      h = h - 1;
    h = h + 1;
    q[h] = p[i];
  }
```

`q[0,h)` describes a convex polygon bounding the convex hull of `p[0..N)`.

Analysis.

Theorem 4.26 *The convex hull of a set $P \subset \mathbb{R}^2$ of n points can be computed using $O(n \log n)$ geometric operations.*

Proof.

1. Sorting and removal of duplicate points: $O(n \log n)$.
2. At the beginning we have a sequence of $2n - 1$ points; at the end the sequence consists of h points. Observe that for every positive orientation test, one point is discarded from the sequence for good. Therefore, we have exactly $2n - h - 1$ such shortcuts/positive orientation tests. In addition there are at most $2n - 2$ negative tests (#iterations of the outer `for` loops). Altogether we have at most $4n - h - 3$ orientation tests.

In total the algorithm uses $O(n \log n)$ geometric operations. Note that the number of orientation tests is linear only, but $O(n \log n)$ lexicographic comparisons are needed. □

4.7 Lower Bound

It is not hard to see that the runtime of Graham Scan is asymptotically optimal in the worst-case.

Theorem 4.27 *$\Omega(n \log n)$ geometric operations are needed to construct the convex hull of n points in $\mathbb{R}^2$ (in the algebraic computation tree model).*

Proof. Reduction from sorting (for which it is known that $\Omega(n \log n)$ comparisons are needed in the algebraic computation tree model). Given n real numbers $x_1, \ldots, x_n$, construct a set $P = \{p_i \mid 1 \leqslant i \leqslant n\}$ of n points in $\mathbb{R}^2$ by setting $p_i = (x_i, x_i^2)$. This construction can be regarded as embedding the numbers into $\mathbb{R}^2$ along the x-axis and then projecting the resulting points vertically onto the unit parabola. The order in which the points appear along the lower convex hull of P corresponds to the sorted order of the x_i. Therefore, if we could construct the convex hull in $o(n \log n)$ time, we could also sort in $o(n \log n)$ time. □

Clearly this reduction does not work for the Extremal Points problem. But using a reduction from Element Uniqueness (see Section 1.1) instead, one can show that $\Omega(n \log n)$ is also a lower bound for the number of operations needed to compute the set of extremal points only. This was first shown by Avis [1] for linear computation trees, then by Yao [9] for quadratic computation trees, and finally by Ben-Or [2] for general algebraic computation trees.

4.8 Chan's Algorithm

Given matching upper and lower bounds we may be tempted to consider the algorithmic complexity of the planar convex hull problem settled. However, this is not really the case: Recall that the lower bound is a worst case bound. For instance, the Jarvis' Wrap runs in $O(nh)$ time an thus beats the $\Omega(n \log n)$ bound in case that $h = o(\log n)$. The question remains whether one can achieve both output dependence and optimal worst case performance at the same time. Indeed, Chan [4] presented an algorithm to achieve this runtime by cleverly combining the "best of" Jarvis' Wrap and Graham Scan. Let us look at this algorithm in detail. The algorithm consists of two steps that are executed one after another.

Divide. *Input:* a set $P \subset \mathbb{R}^2$ of n points and a number $H \in \{1, \dots, n\}$.

1. Divide P into $k = \lceil n/H \rceil$ sets $P_1, \dots, P_k$ with $|P_i| \leqslant H$.

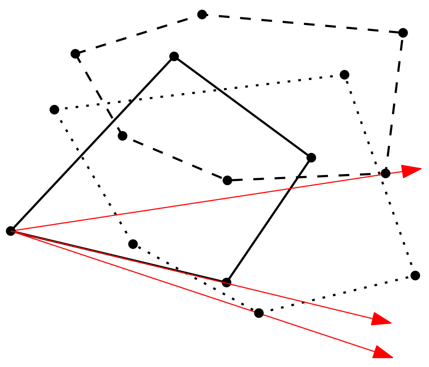
2. Construct $\text{conv}(P_i)$ for all i, $1 \leqslant i \leqslant k$.

Analysis. Step 1 takes $O(n)$ time. Step 2 can be handled using Graham Scan in $O(H \log H)$ time for any single P_i, that is, $O(n \log H)$ time in total.

Conquer. *Output:* the vertices of conv(P) in counterclockwise order, if conv(P) has less than H vertices; otherwise, the message that conv(P) has at least H vertices.

1. Find the lexicographically smallest point in $\text{conv}(P_i)$ for all i, $1 \leqslant i \leqslant k$.

2. Starting from the lexicographically smallest point of P find the first H points of conv(P) oriented counterclockwise (simultaneous Jarvis' Wrap on the sequences $\text{conv}(P_i)$).



Determine in every wrap step the point q_i of tangency from the current point of $\mathrm{conv}(P)$ to $\mathrm{conv}(P_i)$, for all $1 \leqslant i \leqslant k$. We have seen in Exercise 4.22 how to compute q_i in $O(\log|\mathrm{conv}(P_i)|) = O(\log H)$ time. Among the k candidates $q_1, \ldots, q_k$ we find the next vertex of $\mathrm{conv}(P)$ in $O(k)$ time.

Analysis. Step 1 takes $O(n)$ time. Step 2 consists of at most H wrap steps. Each wrap step needs $O(k \log H + k) = O(k \log H)$ time, which amounts to $O(Hk \log H) = O(n \log H)$ time for Step 2 in total.

Remark. Using a more clever search strategy instead of many tangency searches one can handle the conquer phase in $O(n)$ time, see Exercise 4.28 below. However, this is irrelevant as far as the asymptotic runtime is concerned, given that already the divide step takes $O(n \log H)$ time.

Exercise 4.28 *Consider* k *convex polygons* $P_1, \ldots P_k$*, for some constant* $k \in \mathbb{N}$*, where each polygon is given as a list of its vertices in counterclockwise orientation. Show how to construct the convex hull of* $P_1 \cup \ldots \cup P_k$ *in* $O(n)$ *time, where* $n = \sum_{i=1}^{k} n_i$ *and* n_i *is the number of vertices of* P_i*, for* $1 \leqslant i \leqslant k$.

Searching for h. While the runtime bound for $H = h$ is exactly what we were heading for, it looks like in order to actually run the algorithm we would have to know h, which—in general—we do not. Fortunately we can circumvent this problem rather easily, by applying what is called a *doubly exponential search*. It works as follows.

Call the algorithm from above iteratively with parameter $H = \min\{2^{2^t}, n\}$, for $t = 0, \ldots$, until the conquer step finds all extremal points of P (i.e., the wrap returns to its starting point).

Analysis: Let 2^{2^s} be the last parameter for which the algorithm is called. Since the previous call with $H = 2^{2^{s-1}}$ did not find all extremal points, we know that $2^{2^{s-1}} < h$, that is, $2^{s-1} < \log h$, where h is the number of extremal points of P. The total runtime is therefore at most

$$\sum_{i=0}^{s} cn \log 2^{2^i} = cn \sum_{i=0}^{s} 2^i = cn(2^{s+1} - 1) < 4cn \log h = O(n \log h),$$

for some constant $c \in \mathbb{R}$. In summary, we obtain the following theorem.

Theorem 4.29 *The convex hull of a set* $P \subset \mathbb{R}^2$ *of* n *points can be computed using* $O(n \log h)$ *geometric operations, where* h *is the number of convex hull vertices.*

Questions

13. *How is convexity defined? What is the convex hull of a set in* $\mathbb{R}^d$*?* Give at least three possible definitions.

14. *What does it mean to compute the convex hull of a set of points in $\mathbb{R}^2$?* Discuss input and expected output and possible degeneracies.

15. *How can the convex hull of a set of n points in $\mathbb{R}^2$ be computed efficiently?* Describe and analyze (incl. proofs) Jarvis' Wrap, Successive Local Repair, and Chan's Algorithm.

16. *Is there a linear time algorithm to compute the convex hull of n points in $\mathbb{R}^2$?* Prove the lower bound and define/explain the model in which it holds.

17. *Which geometric primitive operations are used to compute the convex hull of n points in $\mathbb{R}^2$?* Explain the two predicates and how to compute them.

Remarks. The sections on Jarvis' Wrap and Graham Scan are based on material that Emo Welzl prepared for a course on "Geometric Computing" in 2000.

References

[1] David Avis, Comments on a lower bound for convex hull determination. *Inform. Process. Lett.*, **11**, 3, (1980), 126, URL `http://dx.doi.org/10.1016/0020-0190(80)90125-8`.

[2] Michael Ben-Or, Lower bounds for algebraic computation trees. In *Proc. 15th Annu. ACM Sympos. Theory Comput.*, pp. 80–86, 1983, URL `http://dx.doi.org/10.1145/800061.808735`.

[3] Constantin Carathéodory, Über den Variabilitätsbereich der Fourierschen Konstanten von positiven harmonischen Funktionen. *Rendiconto del Circolo Matematico di Palermo*, **32**, (1911), 193–217, URL `http://dx.doi.org/10.1007/BF03014795`.

[4] Timothy M. Chan, Optimal output-sensitive convex hull algorithms in two and three dimensions. *Discrete Comput. Geom.*, **16**, 4, (1996), 361–368, URL `http://dx.doi.org/10.1007/BF02712873`.

[5] Ronald L. Graham, An efficient algorithm for determining the convex hull of a finite planar set. *Inform. Process. Lett.*, **1**, 4, (1972), 132–133, URL `http://dx.doi.org/10.1016/0020-0190(72)90045-2`.

[6] Ray A. Jarvis, On the identification of the convex hull of a finite set of points in the plane. *Inform. Process. Lett.*, **2**, 1, (1973), 18–21, URL `http://dx.doi.org/10.1016/0020-0190(73)90020-3`.

[7] Jiří Matoušek, *Lectures on discrete geometry.* Springer-Verlag, New York, NY, 2002.

[8] Johann Radon, Mengen konvexer Körper, die einen gemeinsamen Punkt enthalten. *Math. Annalen*, **83**, 1–2, (1921), 113–115, URL `http://dx.doi.org/10.1007/BF01464231`.

[9] Andrew C. Yao, A lower bound to finding convex hulls. *J. ACM*, **28**, 4, (1981), 780–787, URL `http://dx.doi.org/10.1145/322276.322289`.