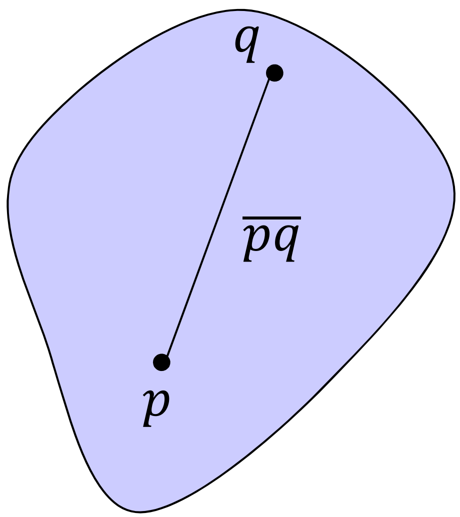
Computational Geometry

2. Convex Hull

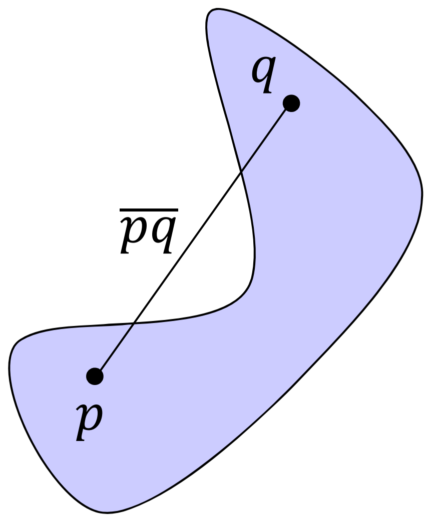
2.1 Definitions

Definition (Convex Set)

A set $S \subseteq \mathbb{R}^d$ is called **convex**, iff for all points $p, q \in S$ the connecting line segment $\overline{pq} = \{x: x = \lambda p + (1-\lambda)q, \lambda \in [0,1]\}$ is contained in S, i.e. $\overline{pq} \subseteq S$.



convex



non-convex

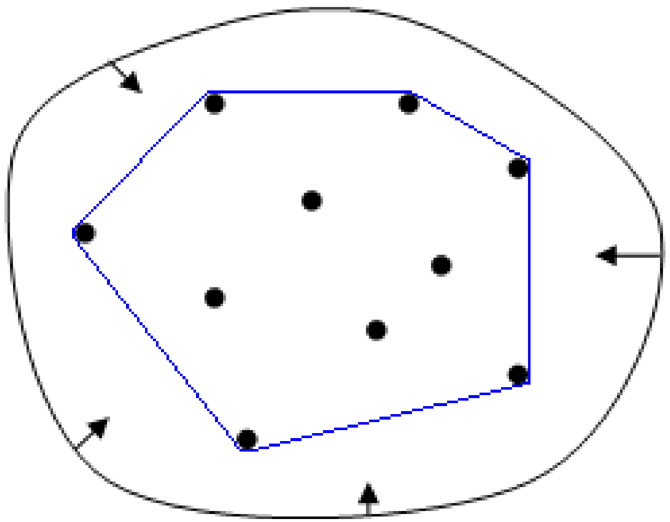
2.1 Definitions

Definition (Convex Hull)

The **convex hull** $\mathcal{CH}(S)$ of a set $S \subseteq \mathbb{R}^d$ is the smallest convex set containing S,

➡ i.e. it is the intersection of all convex sets containing S.

Rubber band metaphor



2.1 Definitions

- For $d = 2$ there is an alternative (informal) definition:

  The convex hull $\mathcal{CH}(P)$ of a finite set P of n points is the unique, convex polygon with corners from P containing all points of P.

- This definition is used to develop an algorithm computing the convex hull of a set of points.

Remark: Convex hull algorithms are to Computational Geometry what sorting is to discrete algorithms.

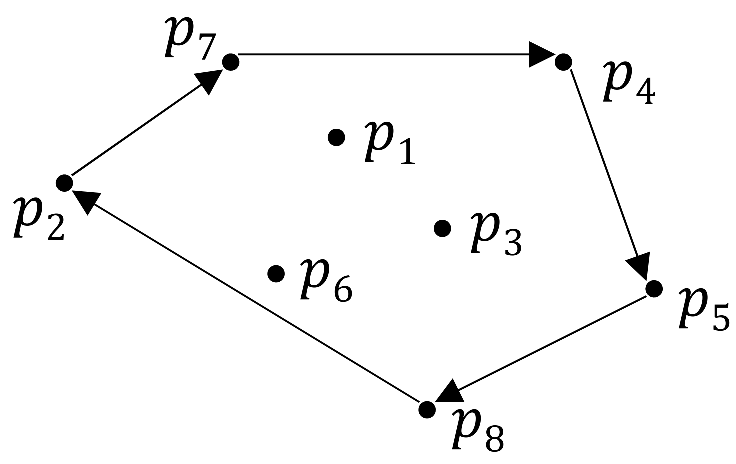
2.2 Simple approach

Input: Set $P = \{p_1, p_2, \dots, p_n\}$ of n mutually distinct points in the plane.

Output: Sorted list of points from P (clockwise), representing the corners of the convex hull $\mathcal{CH}(P)$.

Example:

- Input: $(p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8)$
- Output: $(p_4, p_5, p_8, p_2, p_7)$



2.2 Simple approach

- Idea for a simple algorithm for the convex hull in the plane:

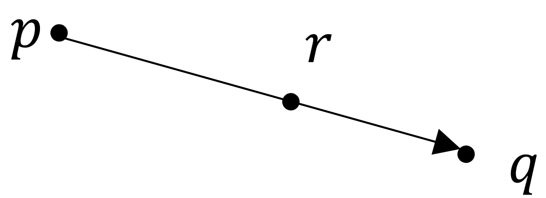
  A directed edge $\overline{pq}$ is part of the convex hull, if no point lies left of it.

2.2 Simple approach

```
SortedList SlowConvexHull(PointSet P)
{
   PointSet E = ∅;
   for all (ordered pairs (p,q)∈P^2 with p≠q){
      valid = true;
      for all (points r∈P\{p,q}){
         if (r lies left of the ordered line pq)
           valid = false;
      }
      if (valid) add the ordered line pq to E;
   }
   SortedList L = ConstructSortedList(E);
   return L;
}
```

2.2 Simple approach

- This algorithm is **(almost)** correct.
  - Neglect rounding errors from floating point arithmetic!
  - Lines 3-11: Yield an unsorted list of all edges of the convex hull.
  - But, there might be degenerate cases!



2.2 Simple approach

- Run time of the algorithm SlowConvexHull:
  - Line 4: $n^2 - n = n(n-1)$ pairs of points.
  - Line 6: $n-2$ points are tested.
  - Line 7: Assume the test to decide if a point lies left or right of a line is a primitive operation, so it is computable in constant time.

  ➡ Lines 3-11: This yields a run time of $O(n^3)$.

  - Line 12: The brute force approach has run time $O(n^2)$.

➡ Total run time: $O(n^3) + O(n^2) = O(n^3)$.

- This can be improved!

2.3 Lower Bound

- The minimal run time to compute the convex hull of n points in 2d is $\Omega(n \log n)$.

Proposition 1

The sorting problem can be transformed in linear time to the convex hull problem.

Thus, the convex hull of n points in the plane can be computed in $\Omega(n \log n)$.

- **Proof:** see Algorithmentechnik.

2.3 Lower Bound

Remark:

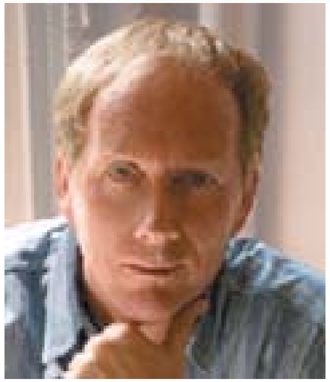
The complexity of the convex hull problem in 2d is $O(n \log n)$.

Remark:

Four approaches:

1. Naïve approach: $O(n^3)$.
2. Divide-and-conquer approach: $O(n \log n)$.
3. Scan line approach, Graham's scan: $O(n \log n)$.
4. Output-size sensitive approach, Jarvis' march: $O(h\, n)$.

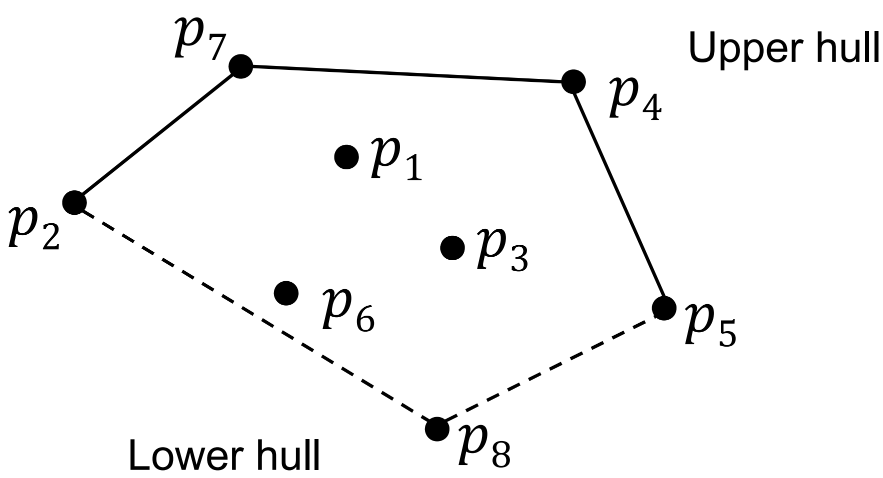
2.4 Graham´s Scan



- Graham's Scan was developed in 1972 by Ronald L. Graham.
- Improved by Andrew in 1979.
- Incremental algorithm
  - Points are analyzed one by one and added successively to the solution.
  - This is done left to right.
  - Therefore, the points are sorted by their x-coordinate.
  - **Problem:** The knots of the convex hull should also be added to the convex hull from left to right.

2.4 Graham´s Scan

- **Solution:** Compute first the knots of the *upper convex hull* and then the knots of the *lower convex hull*.
  - The points to split the convex hull in the upper and lower convex hull are determined by the minimal and maximal x-coordinate.



2.4.1 Algorithm

Algorithm 2 GrahamScan(P)

Sort the points by their x-coordinate $(p_1, \ldots, p_n)$;
Add the points p_1 und p_2 to the list $\mathcal{L}_{upper}$, where p_1 is the first point;
for $(i = 3;\ i \leq n;\ i++)$ $\{$
    Append p_i to the list $\mathcal{L}_{upper}$;
    while ($\mathcal{L}_{upper}$ containes more than two points AND the last three points do not make a right turn) $\{$
        Delete the middle point of the last three points from $\mathcal{L}_{upper}$;
    $\}$
$\}$
$\vdots$

2.4.1 Algorithm

$\vdots$

Add the points p_n und p_{n-1} to the list $\mathcal{L}_{lower}$, where p_n is the first point;

for $(i = n - 2;\ i \geq 1;\ i - -)$ {

    Append p_i to the list $\mathcal{L}_{lower}$;

    while ($\mathcal{L}_{lower}$ containes more than two points AND the last three points do not make a right turn) {

        Delete the middle point of the last three points from $\mathcal{L}_{lower}$;

    }

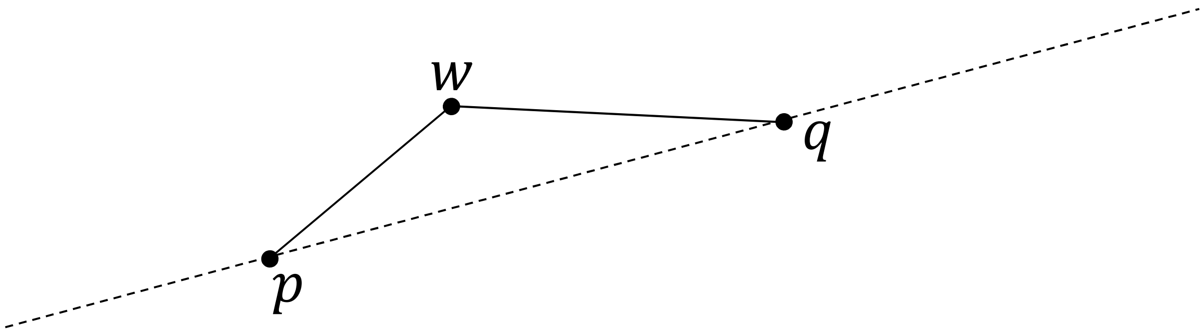
}

Delete the first and last point from $\mathcal{L}_{lower}$ to remove the split points of the upper und lower hull once;

Append $\mathcal{L}_{lower}$ to $\mathcal{L}_{upper}$ resulting in a new list $\mathcal{L}$;

return($\mathcal{L}$);

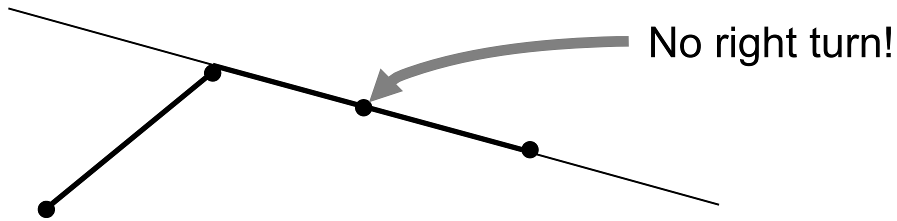
- The test for a right turn is easy to compute:
  - Three points (p, w, q) have a right turn, if w lies above the line segment $\overline{pq}$.



2.4.1 Algorithm

- The Algorithm is not correct!
  - If several points have the same x-coordinate!
    - Thus, the chosen order is not well-defined.
    - Solution: Sorting the points *lexicographically*.

      If two points have equal x-coordinates, sort them by their y-coordinate.
- Another special case occurs if three points in the right-turn-test lie on a line!



2.4.2 Run time

Proposition 2

The convex hull of a set of n points in the plane can be computed in $O(n \log n)$.

- Proof of correctness of the algorithm
  - Induction: number of considered points (→black board)
- Run time
  - Lexicographical sorting: $O(n \log n)$
  - Upper Hull
    - Loop in line 3: $n - 3$
    - Loop in line 5: at most n points are removed in total
    - Total run time of the upper hull: $O(n)$ [lower hull analogously]
  - Total run time: $O(n \log n)$

2.5 Jarvis´ March

- 1973 developed by Ray A. Jarvis.
- „Gift Wrapping“-method
- **Idea:** Wrap a cord around the set of points until it coincides with the convex hull.
  - Similar to Graham´s Scan start with the lexicographically smallest point. This ensures that it lies on the convex hull!
    <u>Assumption:</u> Search first for the smallest y-coordinate and then for the smallest x-coordinate.
  - Jarvis´ March computes a sequence of „extreme“ points of the convex hull in counter-clockwise order.

2.5.1 Algorithm

Algorithm 3 JarvisMarch(P)

```
Find lexicographically smallest point p_1;
q_1 = p_1;
i = 2;
q_i = point with smallest angle to the horizontal line through p_1;
repeat
    i++;
    q_i = point with smallest angle to the line through q_{i-2} and q_{i-1}
    at point q_{i-1};
until (q_i = p_1)
Add all points q_1, ..., q_i to a list L;
return(L);
```

2.5.2 Run time

- Denote by k the number of corners of the convex hull.
  - Computing q_i has runtime $O(n)$, because a smallest angle is calculated by arithmetic operations and comparisons only.
- The total run time is now $O(kn)$
  - For small k this algorithm is very efficient!
  - For large k it is inefficient, because its worst-case run time is $O(n^2)$.
  - Such a behavior is called *output-size sensitive*.