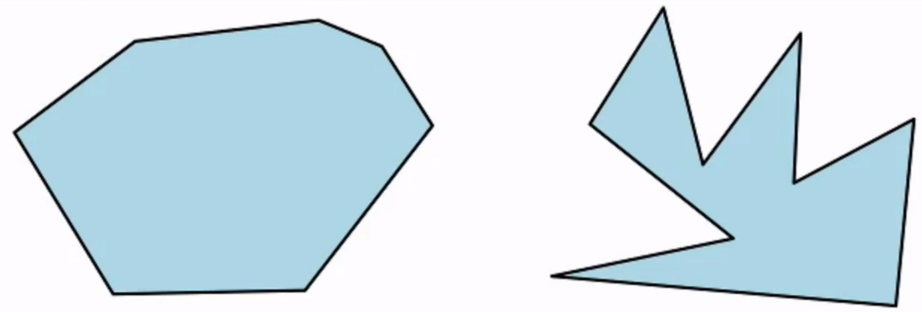
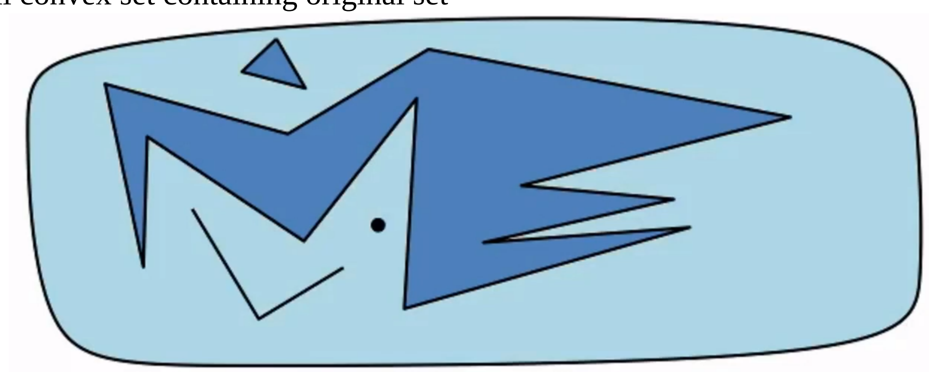
**Motivation**

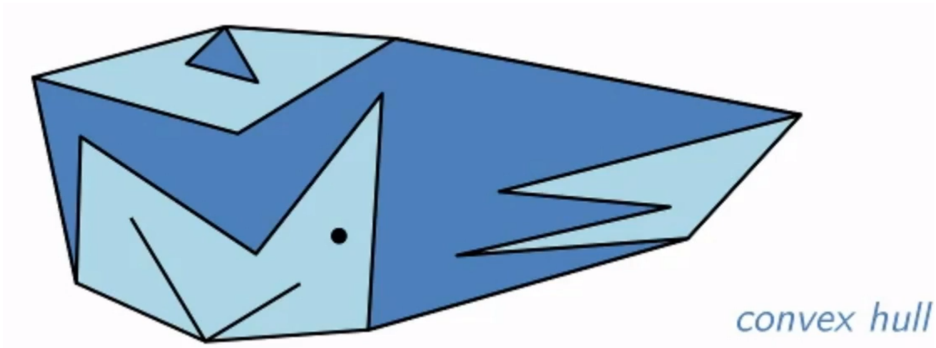
- Polygons represent 2-dimensional parts of the plane
- Complexity can be rather arbitrary
- Convex sets have many important properties
- Convex hull “simplifies” a set.



**Convex Set**

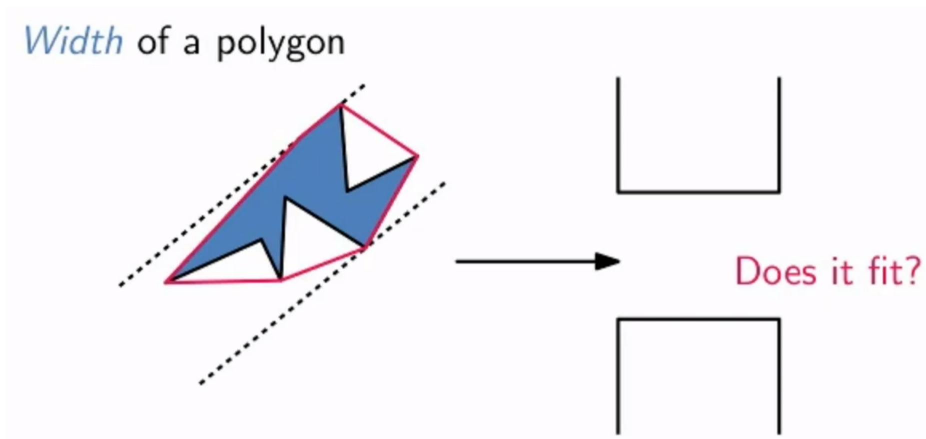
- **Definition:** A set $P \subseteq \mathbb{R}^d$ is *convex* if the segment between two points of $P$ is also contained in $P$.

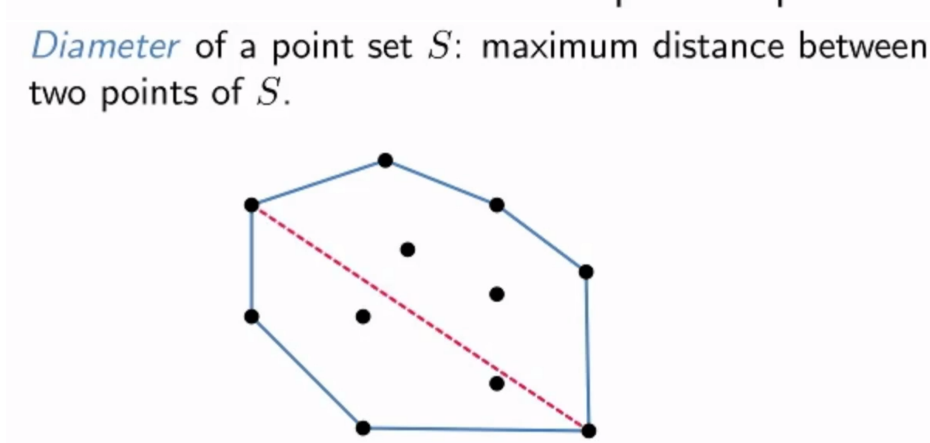
  Equivalent: intersection of $P$ with a line is connected.
- intersection of convex sets also convex
- sets can be approximated with convex set
  - small convex set containing original set
    * 
- convex hull
  - smallest convex set

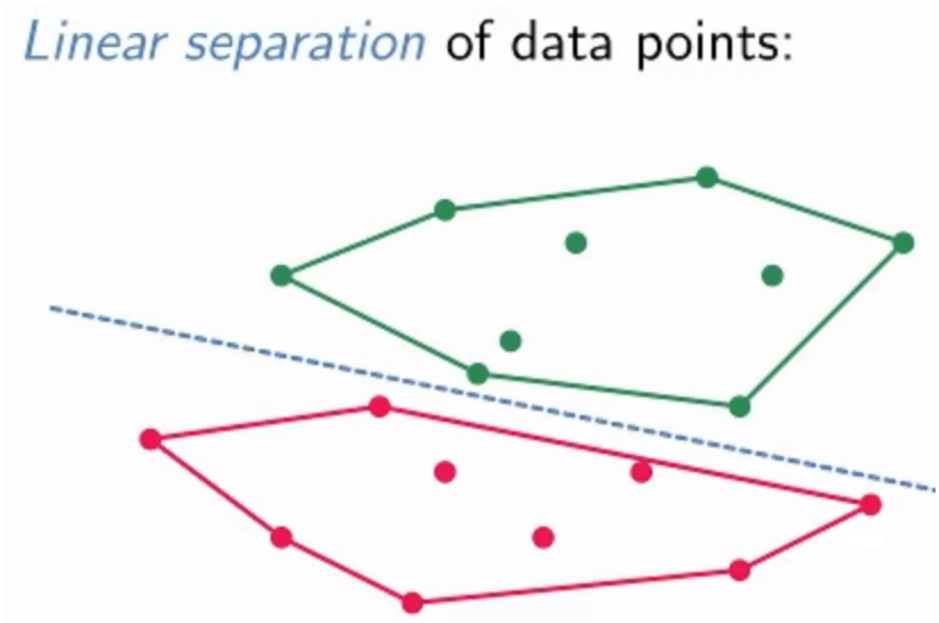
    * 


  - **Definition:** The *convex hull* conv($P$) of a set $P \subseteq \mathbb{R}^d$ is the intersection of all convex supersets of $P$.
  - contains boundary and interior

**Motivation**

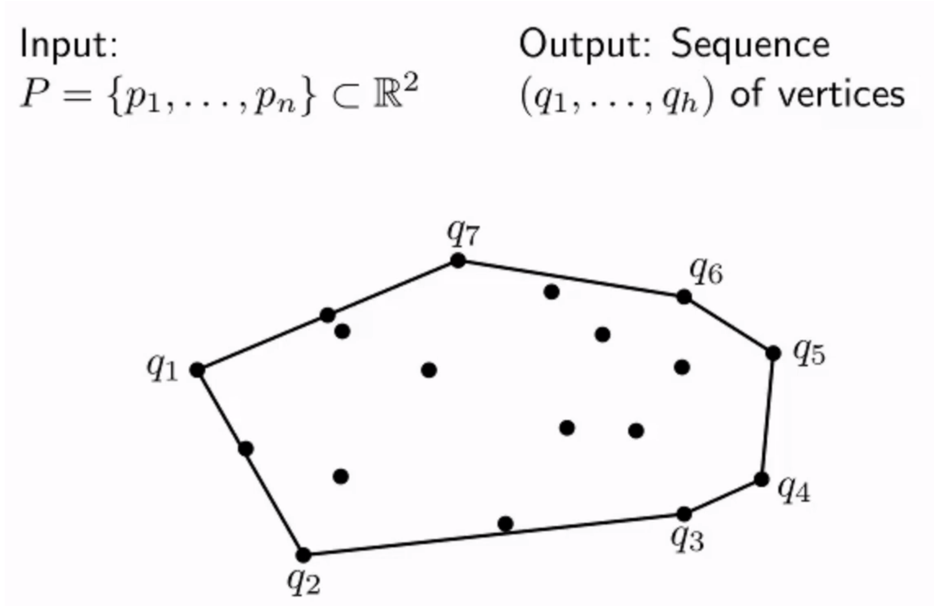
- important geometric [[Datenstrukturen]]
- applications
  - 

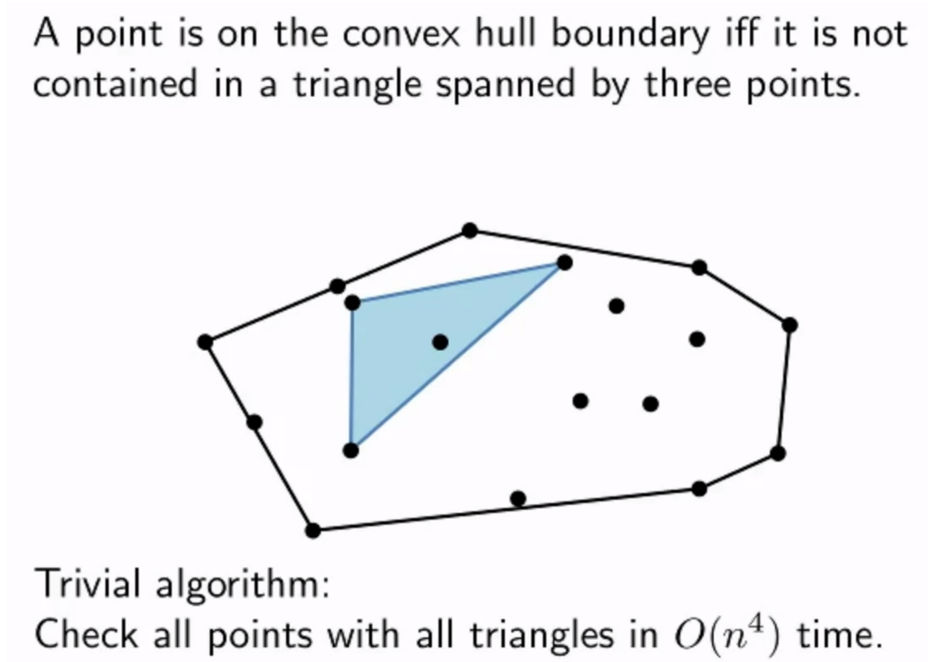
  - 

- 

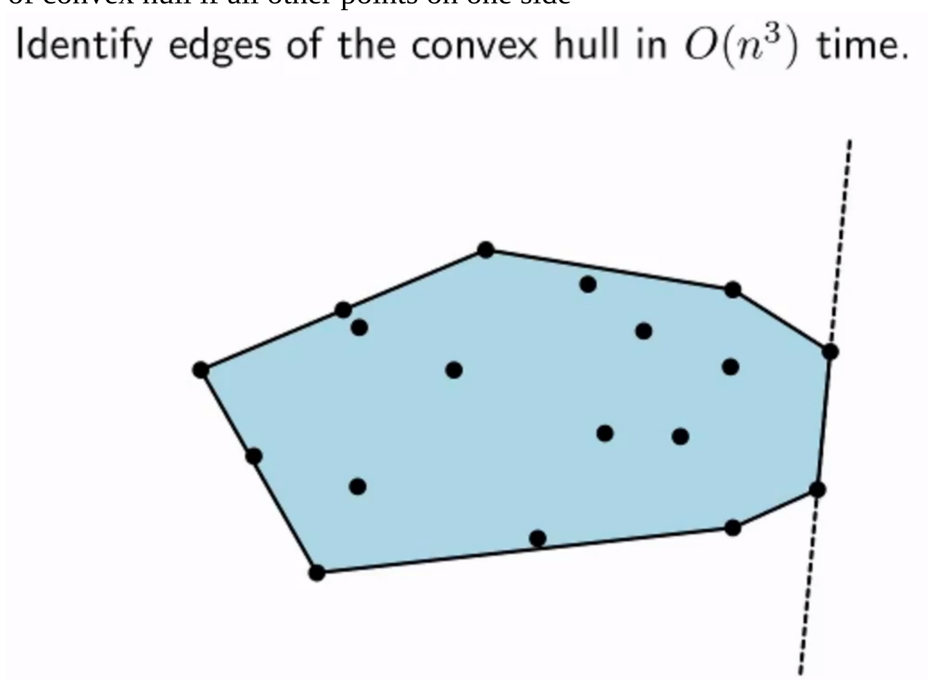
- etc

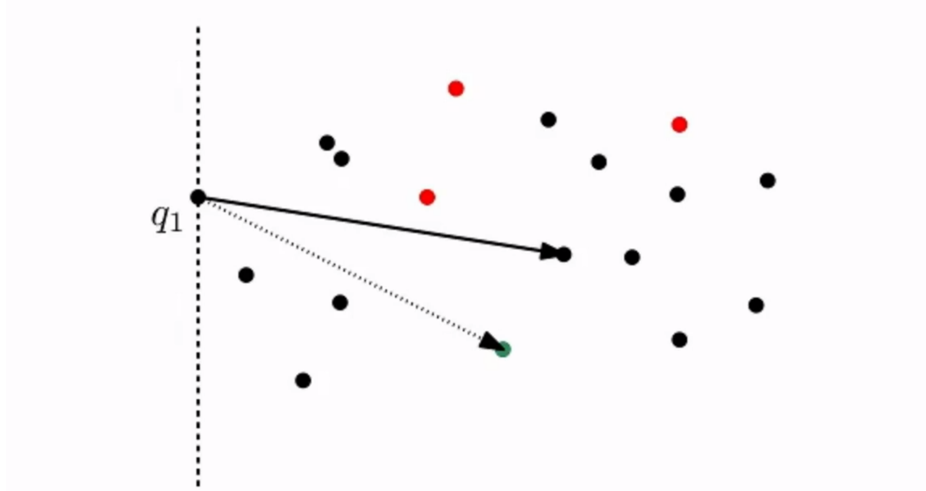
**Planar Convex Hull**

- convex polygon
    - convex hull of point set
- convex hull of vertex set
    - convex hull of polygon
- basic creation algorithms
    - 

- 


**Jarvis' Wrap**

- edge of convex hull if all other points on one side
  - 




- Start with extreme point
- Find edges that have all other points to the left

– Repeat until $q_1$ is reached again

Input:

- Array p[1..N] of points ($N \geq 3$)

Output:

- Array q[1..h] with convex hull vertices in order

Preparation:

- Find the point with smallest $x$-coordinate (q_now).
- q_now is the first known convex hull point.
- Choose a different point (q_next) as the first candidate for the next convex hull point.
- The array q is still empty.

Every round:

- add q_now to array q
- find next convex hull point q_next:
  point such that no other point is right of the directed line from q_now to q_next
- replace q_now by q_next
- replace q_next by new candidate different from q_now

End:

- When the next found convex hull point is equal to q[1] then the convex hull is completed.
- This is the case when q_now equals q[1] after a round.

- pseudo code
  - preparation

```
for (i = 2 to N)
   if (p[i].x < p[1].x)
      swap(p[1], p[i])
q_now = p[1]
q_next = p[2]
h = 0
```

  *

```
do
  h = h+1
  q[h] = q_now
  for (i = 2 to N)
    if (rightturn(q_now, q_next, p[i]))
      q_next = p[i]
  q_now = q_next
  q_next = p[1]
while (q_now != q[1])
```

  –

- time complexity
  - Preparation: $\Theta(n)$ time
  - Outer loop is processed $h \leq n$ times
  - Inner loop is processed $\Theta(n)$ times each round.
  - $\Theta(nh) = O(n^2)$ rightturn tests
  - Worst case: $h = \Theta(n)$
  - Output sensitive (good for small hulls)
  - Asymptotically not optimal

  –

- space complexity
  - $O(n)$ in addition to input:
    q and constantly many extra variables.

  –

- correctness

- Before round $k$,
  - q contains $k-1$ vertices of the convex hull in order,
  - `q_now` is the next vertex on the convex hull.
- At the end of round $k$, `q_now` is the next convex hull vertex after `q[h]`: edge from `q[h]` to `q_now` has no other point to the right.
- When `q_now` equals `q[1]` the convex hull is complete.

–

**Degeneracies:**

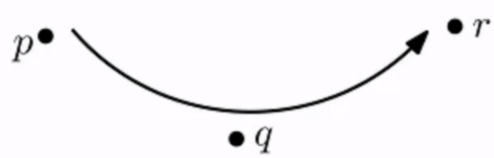
- Several points with smallest $x$-coordinate
- More than two points on a line

**Orientation Test**

**Do three points make a left turn?**

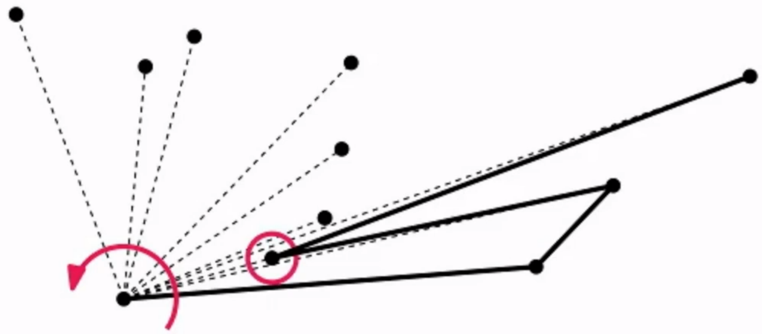
- Needed by many algorithms
- Evaluate degree 2 polynomial: *algebraic degree* 2
- Practical relevance

$$\operatorname{sign}\left(\begin{vmatrix} 1 & p_x & p_y \\ 1 & q_x & q_y \\ 1 & r_x & r_y \end{vmatrix}\right)$$



**Graham Scan**

- Create hull by "Successive Local Repair"
- sequence sorted around extreme point: remove points not making a left turn



- Input:
  - Array p[1..N] of points ($N \geq 3$)

  Output:
  - Array q[1..h] with convex hull vertices (in order)

  Preparation:
  - Place the point with smallest $y$-coordinate into p[1]
  - Sort all other points counterclockwise around p[1]: p[i] is larger than p[j] if p[i] is left of the directed line from p[1] to p[j]
  - p[1] and p[2] are the first two convex hull vertices: Add them in this order to q.

-

```
for (i = 2 to N)
  if (p[i].y < p[1].y)
    swap(p[1], p[i])
sort p[2..N] counterclockwise around p[1]
q[1] = p[1]
q[2] = p[2]
h = 2
```

Process the remaining points from p[3] to p[n].

Processing point p[i]:

- While from the last edge of the convex hull there is no left turn to p[i], remove the last point from q.
- Add p[i] to q.

End:

- After p[n] has been processed, the convex hull vertices are stored in order in q.

```
for (i = 3 to N)
  while (h>1 and not leftturn(q[h-1],q[h],p[i]))
    h = h - 1
  h = h + 1
  q[h] = p[i]
```

- time complexity
  - Preparation in $O(n \log n)$ time due to sorting.
  - Buliding the convex hull: $\Theta(n)$ time. Why?

```
for (i = 3 to N)
  while (h>1 and not leftturn(q[h-1],q[h],p[i]))
    h = h - 1
  h = h + 1
  q[h] = p[i]
```

  → in total $O(n \log n)$ time.

- space complexity
  - $O(n)$ in addition to input

- correctness
  - Before the round for p[i], q contains the vertices of the convex hull of p[1..i-1] in order. ⇐ initially true
  - After the round for p[i], q contains the vertices of the convex hull of p[1..i] in order:
    - Point p[i] is the "last" vertex of the convex hull.
    - The last point q[h] is removed iff it lies in the triangle $\Delta$ q[1] q[h-1] p[i].
      If it does not lie in this triangle, it is on the convex hull of p[1..i].

**Lower Bound**

- The convex hull of $n$ points in $\mathbb{R}^2$ can be computed in
  - $O(nh)$ time by Jarvis' Wrap:
    output sensitive, but quadratic in the worst case.
  - $O(n \log n)$ time by Graham's scan:
    worst-case optimal, but not output-sensitive.
- Lower bound of $\Omega(n \log n)$ time in the worst case follows from $\Omega(n \log n)$ worst case time for sorting.

**Remark:** The convex hull of $n$ points in $\mathbb{R}^2$ can be computed in $O(n \log h)$ time by *Chan's algorithm* (a clever combination of Graham's scan and Jarvis' wrap).