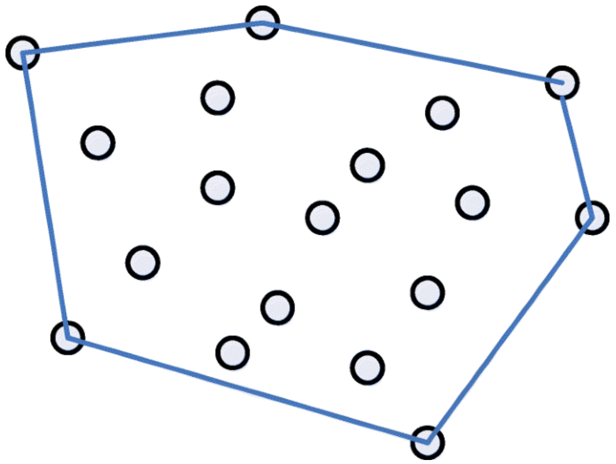
Computational geometry
I – Convex hull



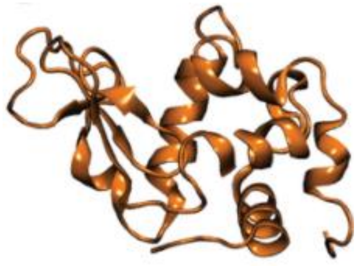
Stefan BORNHOFEN

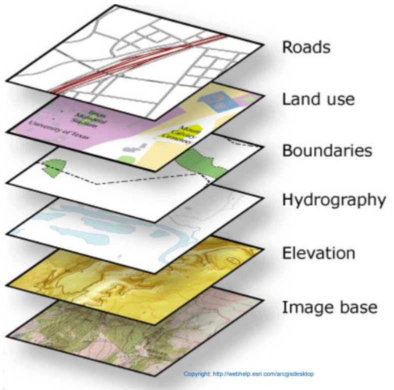
Computational Geometry?

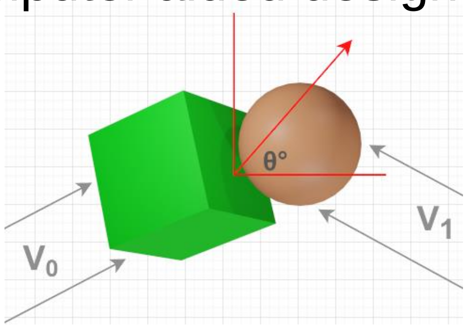
Branch of computer science devoted to the study of algorithms which can be stated in terms of geometry (space, position, distance, size, etc.)

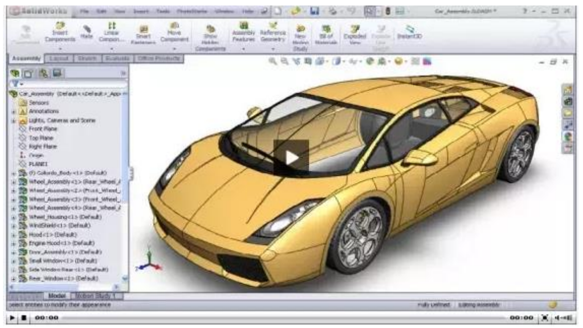
Many application domains, e.g.

- geographic information systems
- molecular biology
- robotics
- physics simulations
- computer aided design



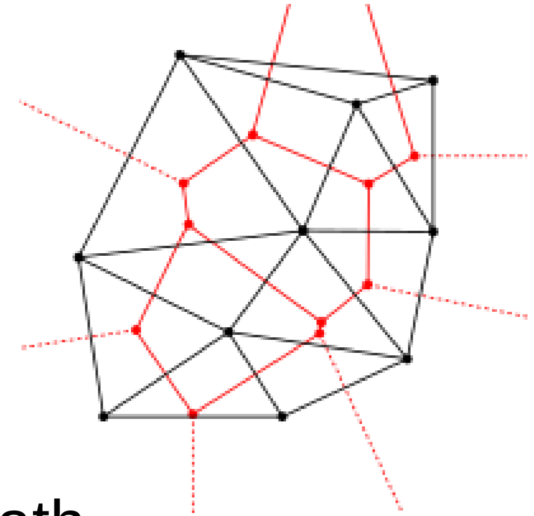




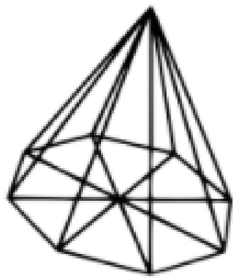


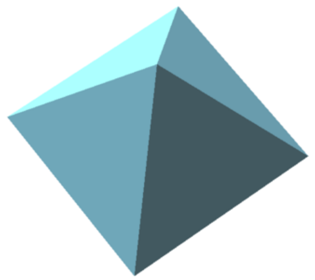
We naturally focus on geometric algorithms specific to computer graphics.

Course objectives



- Improve your skills in 2D and 3D math
- Study some classic 2D geometric algorithms
- Learn concepts about the 3D graphics pipeline

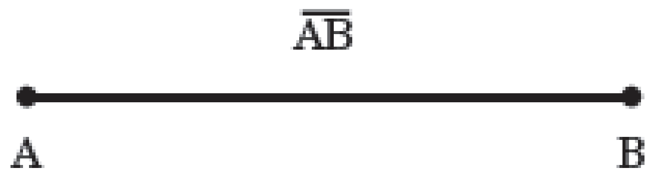




Point and segment

- A point models the exact location in the space, represented by an ordered n-tuple (a1, a2, … , an) where n is the dimension of the space.

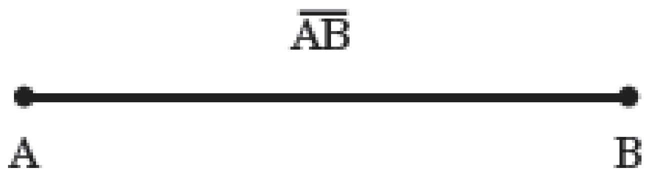
- A segment S is a part of a line that is bounded by two distinct points, and contains every point on the line that is between them.



Read as *line segment AB*.

Point and segment

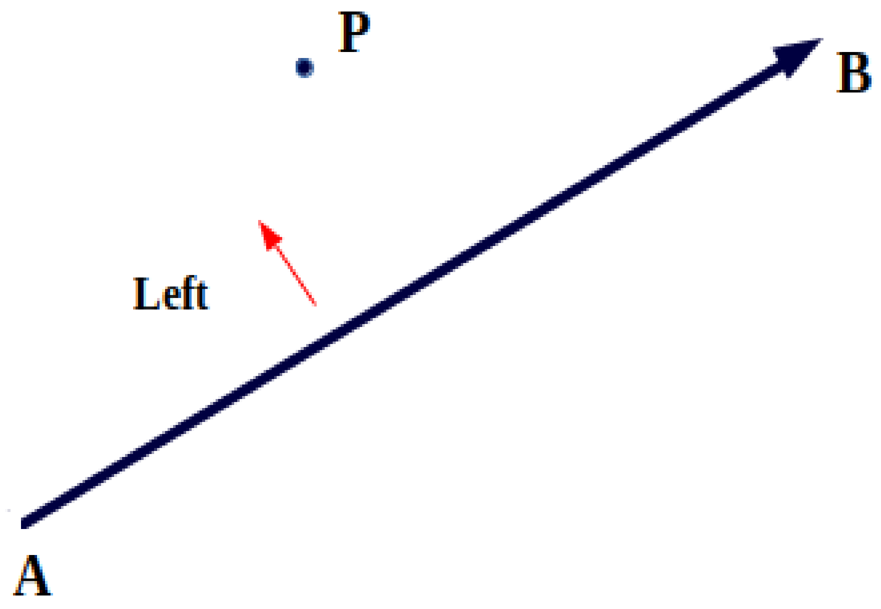
- A point models the exact location in the space, represented by an ordered n-tuple (a1, a2, … , an) where n is the dimension of the space.
- A segment S is a part of a line that is bounded by two distinct points, and contains every point on the line that is between them. S = {A+t(B-A)| 0 ≤ t ≤ 1}
- In computer science, we consider S to be directed, i.e. having the start point A and the end point B.



Read as *line segment AB*.

Orientation point - segment

Given a line segment AB and a point P, what is the orientation of P, i.e. does the point lie to the left, to the right, or on the segment?



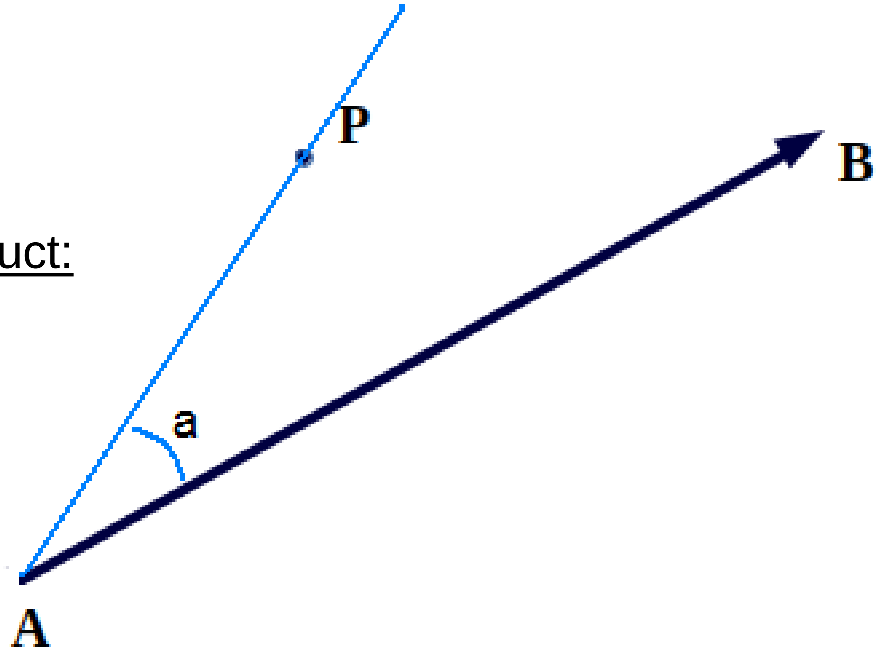
Orientation point - segment

Assess the angle a between AB and AP between]-π, π].

- a>0: P lies to the left
- a<0: P lies to the right
- a=0 or a=π : P lies on the line

Dot product:
<AB,AP> = ||AB||*||AP||*cos(a)

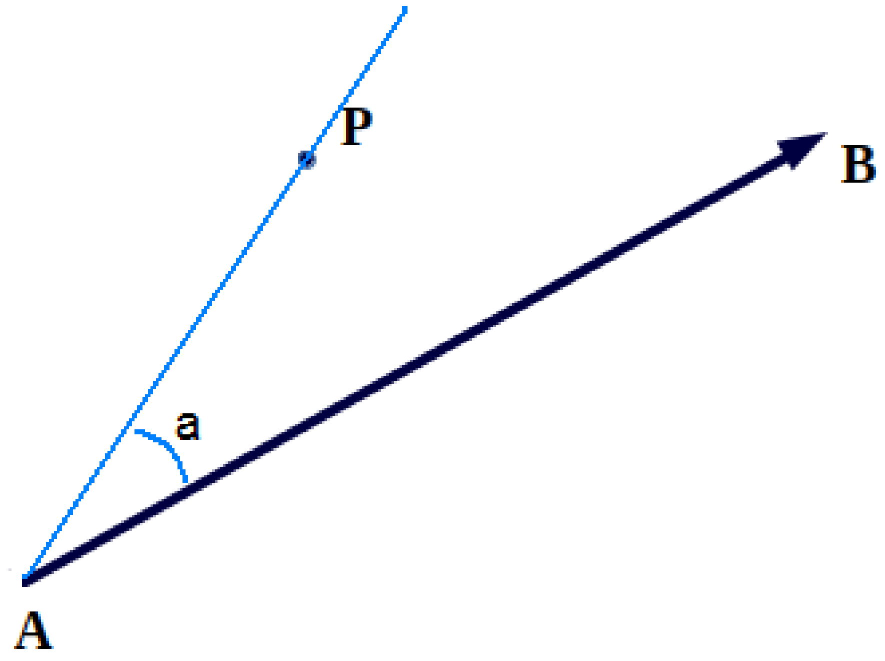
2D version of the 3D cross product:
det(AB,AP) = ||AB||*||AP||*sin(a)



Orientation point - segment

det(AB,AP) = ||AB||*||AP||*sin(a)

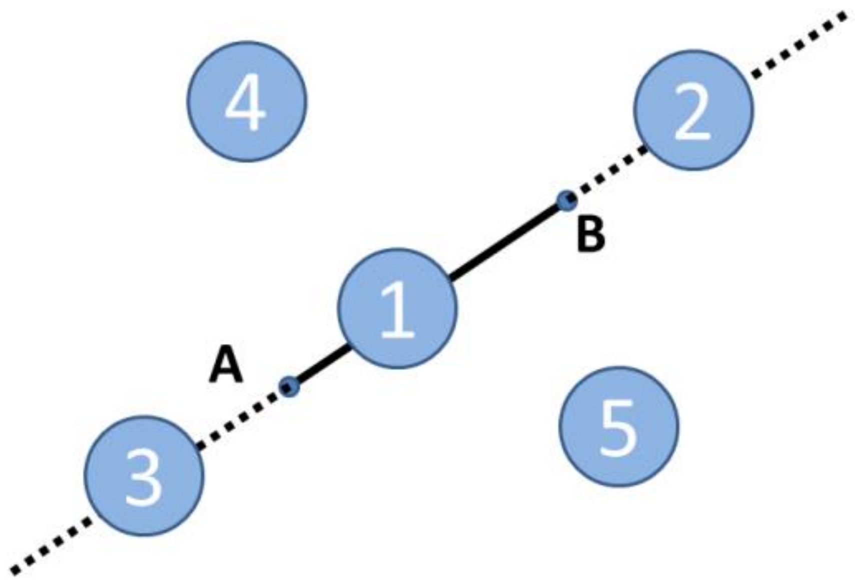
More often than not, we are only interested in the sign and not in the exact value of a. In this case, we only need to compute det(AB,AP).



Orientation point - segment

<AB,AP> = ||AB||*||AP||*cos(a)

- Two differentiate between the zones 1/2 and 3, we can compute the dot product <AB,AP>.
- Two differentiate between the zones 1 and 2, we can compare <AB,AP> and <AB,AB>.



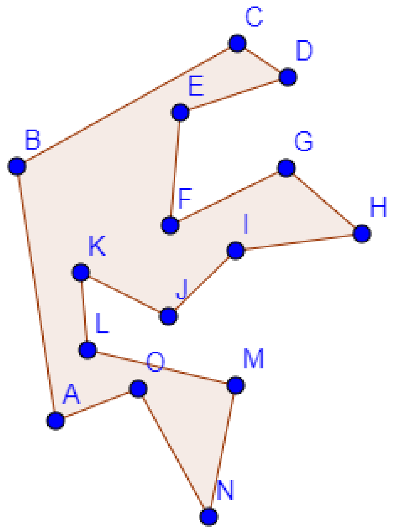
Please note!

- Computing exact angles and distances from coordinates is complex and may lead to numerical inaccuracy.

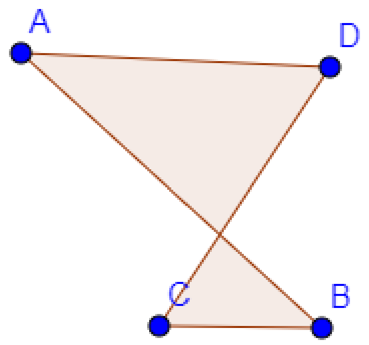
- In our approach we avoided:
  - Dividing two numbers
  - Trigonometric functions (sin, cos)
  - Inverse trigonometric functions (asin, acos)
  - Square roots

Polygons

- A polygon is a closed, connected sequence of line segments.
- A polygon is simple, if it does not intersect itself.



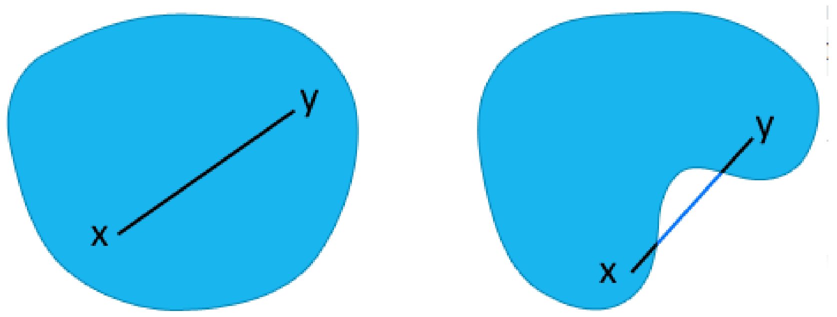
Simple



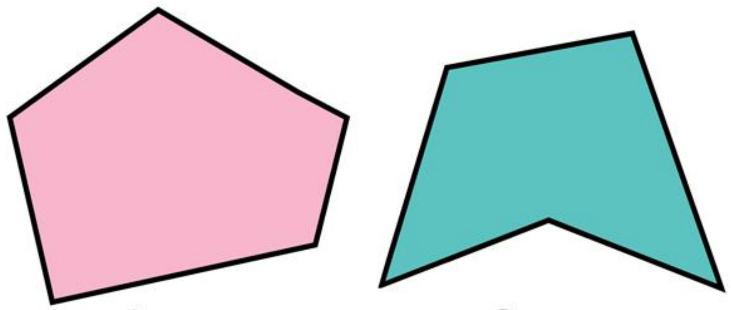
NOT simple

Convex and Concave Polygons

A set S is convex if for every pair of points X and Y in S, the segment XY is also in S; otherwise, it is called concave.



A convex/concave polygon is a polygon that represents the boundary of a convex/concave set.



Convex Concave

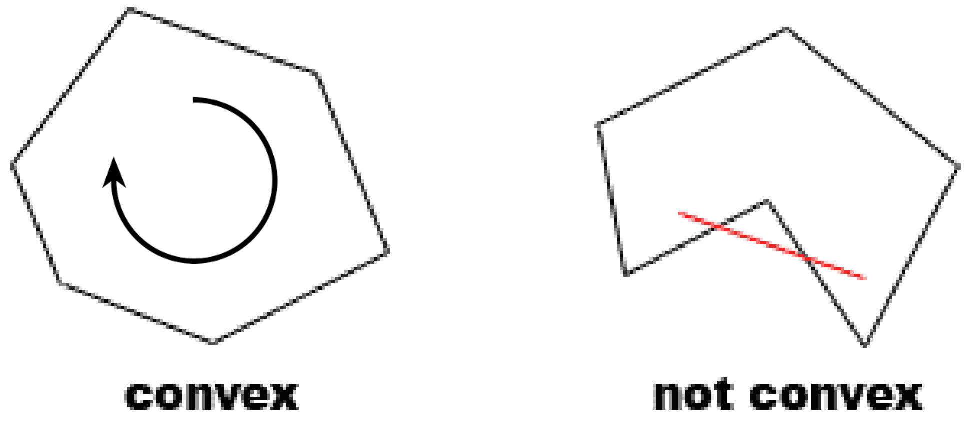
Convexity test

How can we check if a polygon ist convex?

Convexity test

How can we check if a polygon ist convex?

If you move along the edges of the polygon and you only encounter right turns, or only left turns, then it is convex.



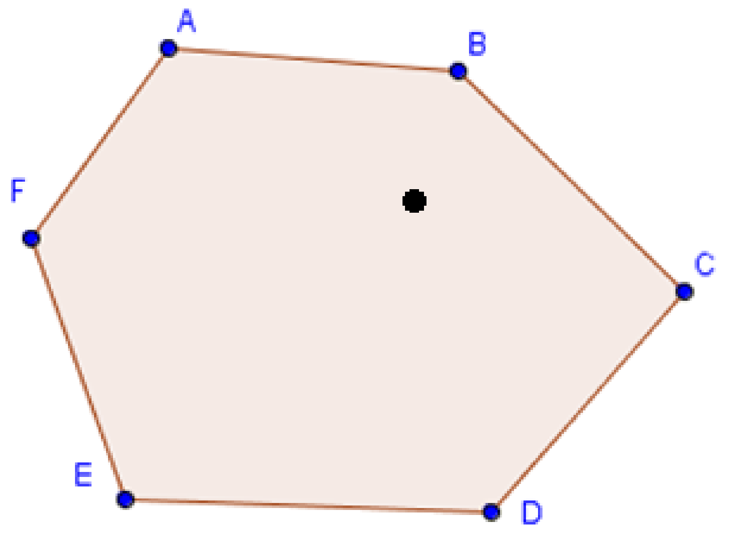
Inside test (convex)

How can we check if a point lies inside a convex polygon?

Inside test (convex)

How can we check if a point lies inside a convex polygon?

Use the « orientation method » (point – segment):
A point lies inside the polygon if it lies to the left or to the right of all edges.



Inside test (concave)

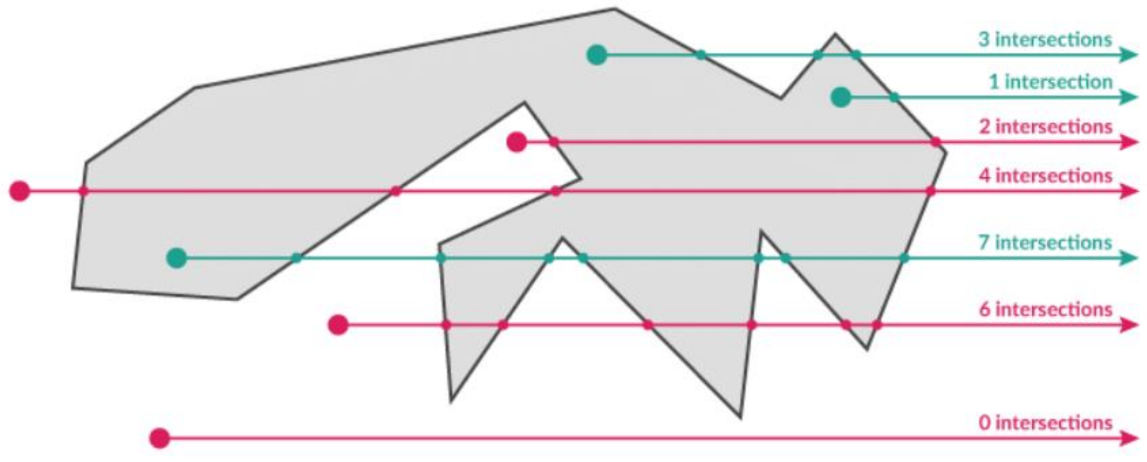
How can we check if a point lies in a concave polygon?

Inside test (concave)

How can we check if a point lies in a concave polygon?

Crossing number algorithm:

Count how many times a ray, starting from the point and going in any fixed direction (horizontal is a smart choice), intersects the edges of the polygon.
If it is an odd number of times => the point is inside the polygon
If it is an even number of times => the point is outside the polygon



Exercise

- Implement a data structure for
  - Point
  - Segment
  - Polygon

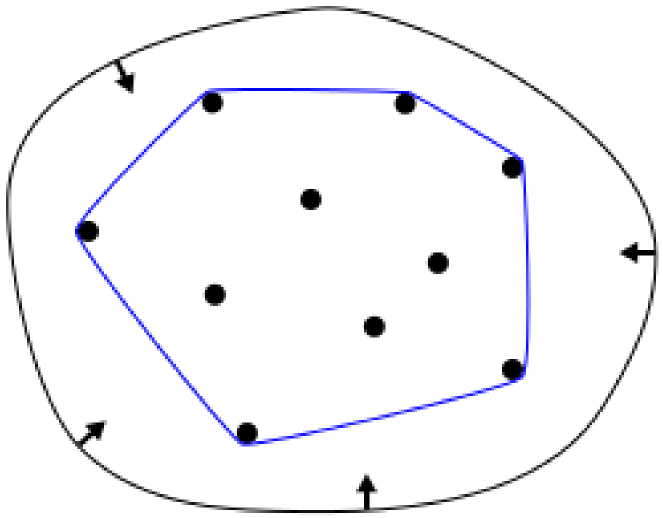
- Implement the following application:
  1. The user defines a polygon with the mouse
  2. Convexity test : the application checks if the polygon is convex.
  3. Inside test: the user can then start clicking on the canvas, and the application checks if the points lie inside or outside the polygon.

     Use the « orientation method » for convex polygons, and the « crossing number method » for concave polygons.

Convex Hull

- Let $P = \{p_1, \dots, p_n\} \subseteq R^2$ be a finite set of points.
- The convex hull of P is the smallest convex set containing all of the points.
- The convex hull is a unique convex polygon whose vertices are points of P and which contains all points from P.
- Think of a rubber band snapping around the points

Note: in the scope of our course, we ignore degenerated forms (3 points on a line, no 4 points on a common circle) which would add annoying special cases to our algorithms.



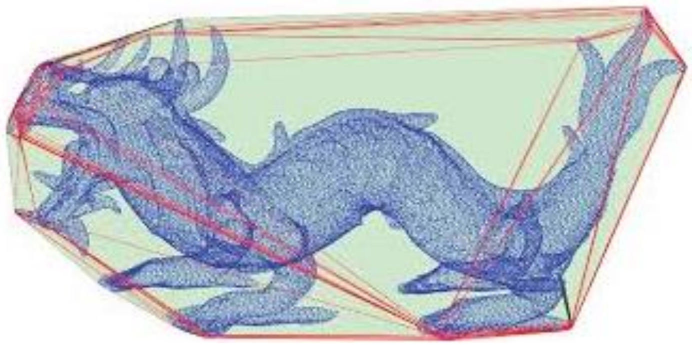
3D Convex Hull

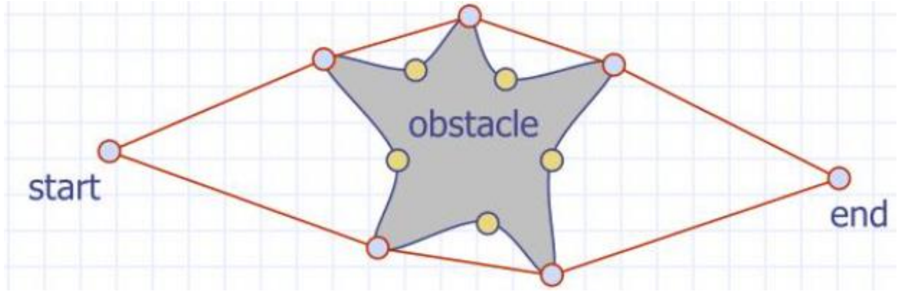
Similarly, the convex hull of points in three dimensions is the shape taken by plastic wrap stretched tightly around the points.

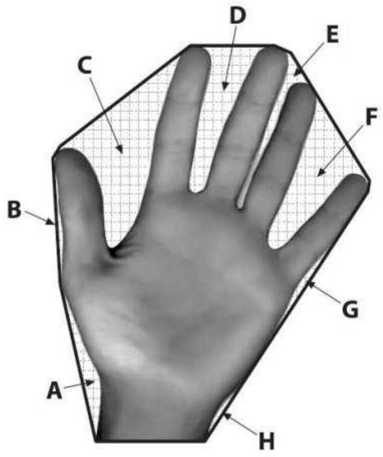
Convex Hull: Applications

Convex hulls have a wide range of applications, for example

- Collision detection
- Path finding
- Hand gesture recognition

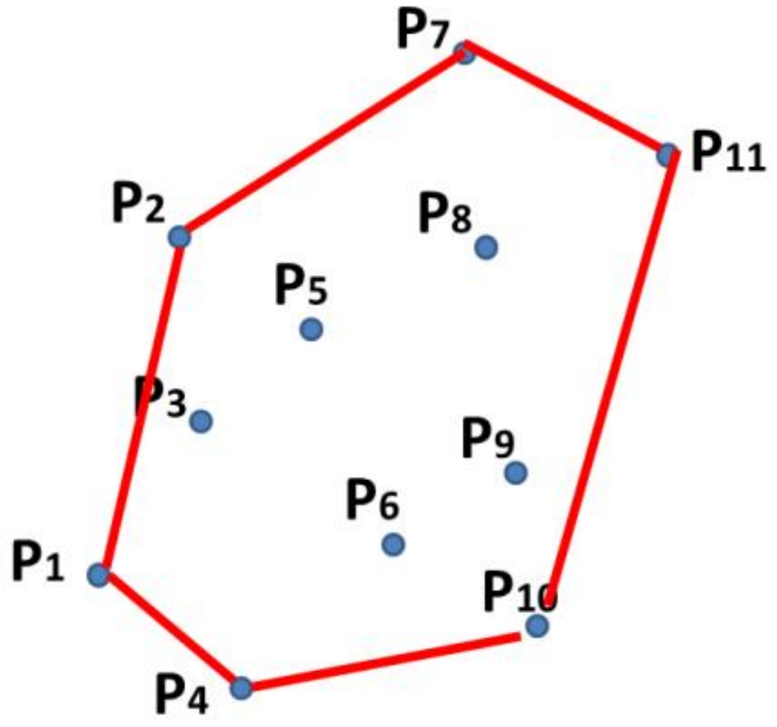






Extreme Edges

The segment pq is part of the convex hull iff all points of P - {p,q} lie to the right or to the left of the line through p and q ("extreme edge").



Extreme Edges

```
For all $p_i$ in $p_1...p_n$
     For all $p_j$ in $p_{i+1}...p_n$
          if $p_ip_j$ is an extreme edge, then
          $p_ip_j$ is part of the convex hull
Stitch the edges together to form a polygon
```

What is the complexity of this algorithm?

Extreme Edges: Complexity

- For each of the $O(n^2)$ pair of points, the test for extremeness costs $O(n)$
- Hence the complexity is $O(n^3)$

More efficient algorithms

- Jarvis’ March (1970)
- Graham’s Scan (1972)
- Quick Hull (1977)
- Divide and Conquer (1977)
- Monotone Chain (1979)
- Incremental (1984)
- Marriage before Conquest (1986)
- Chan (1996)

Jarvis' March

Also called "Gift Wrapping" algorithm. Put the paper in contact with the gift and continue to wrap around from one surface to the next until you get all the way around.

```
Start with the lowest point $p_1$ and a horizontal direction d
Find a point $p_2$ which minimizes the angle between d and $p_1p_2$
d = direction of $p_1p_2$
Find a point p3 which minimizes the angle between d and $p_2p_3$
...
Stop when the polygon is closed.
```

Compare two angles without explicitetly computing their value

What is the complexity of this algorithm?

Jarvis' March: Complexity

- Find the lowest point: O(n)
- Find the next hull edge: O(n)
- Let h be the number of hull edges
- The overall complexity is O(n*h)

Algorithms whose complexity depends on the size of the output are called "output-sensitive".

Graham’s Scan

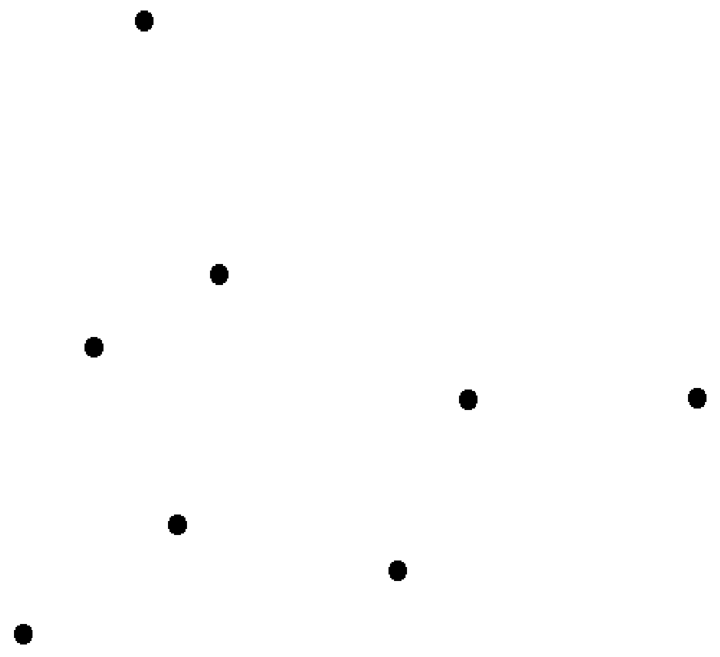
Find the lowest point p_1

Sort the other points by the angle at which they lie as seen from the starting point

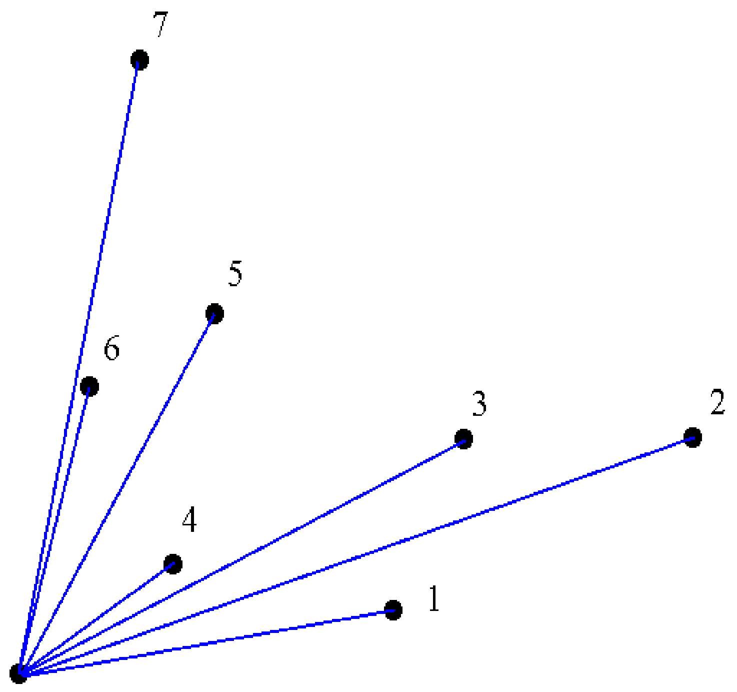
Sort the points without explicitetly computing the angles

Go through sorted points
- Keep vertices of points that have left turns (“convex corner”)
- Drop points that have right turns (“concave corner”).

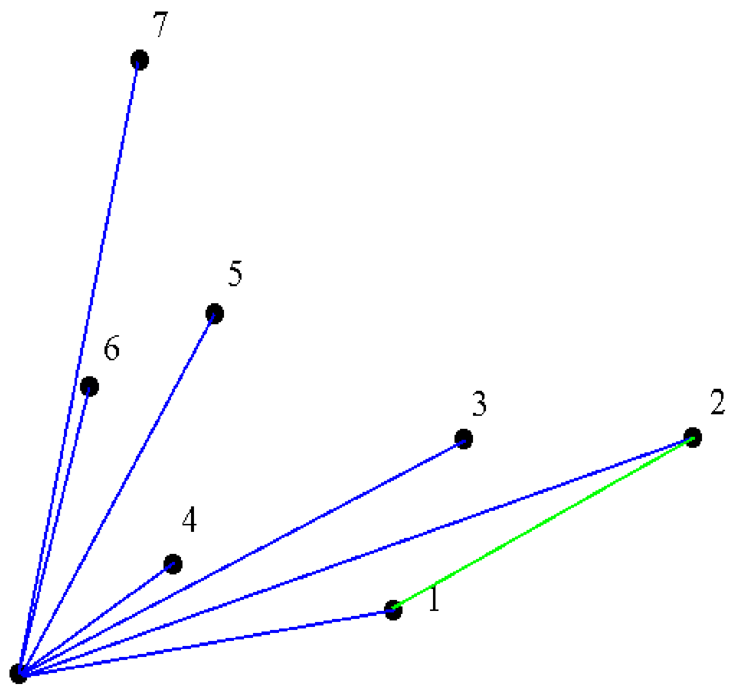
Graham’s Scan



Graham’s Scan

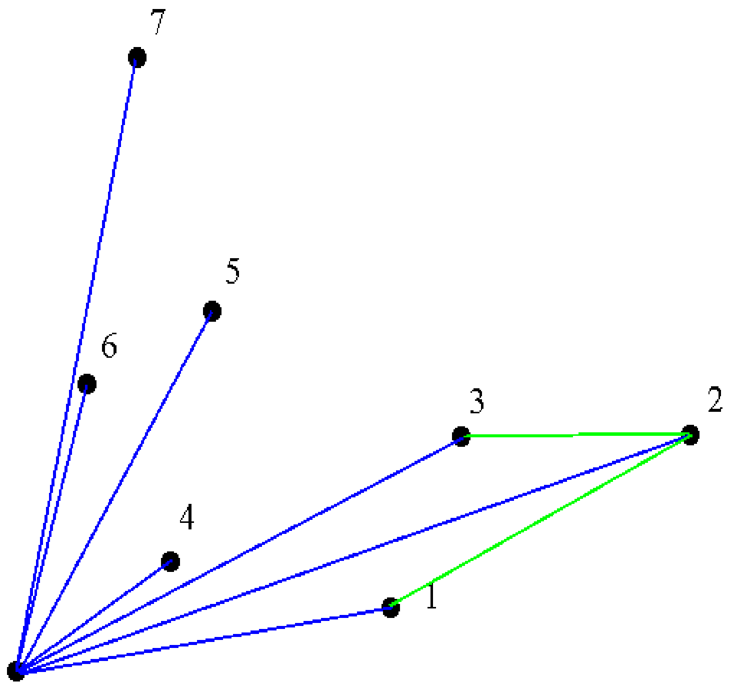


Graham's Scan



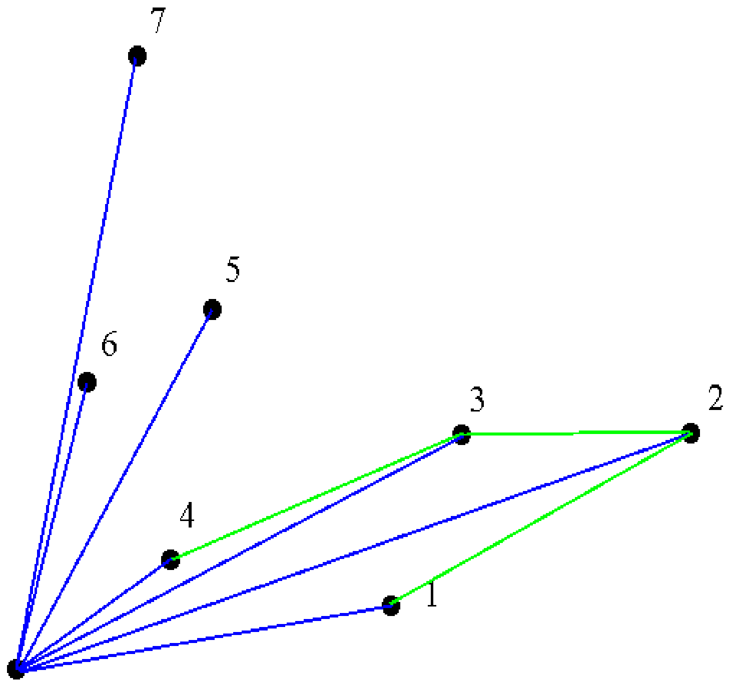
0 - 1 - 2

Graham’s Scan



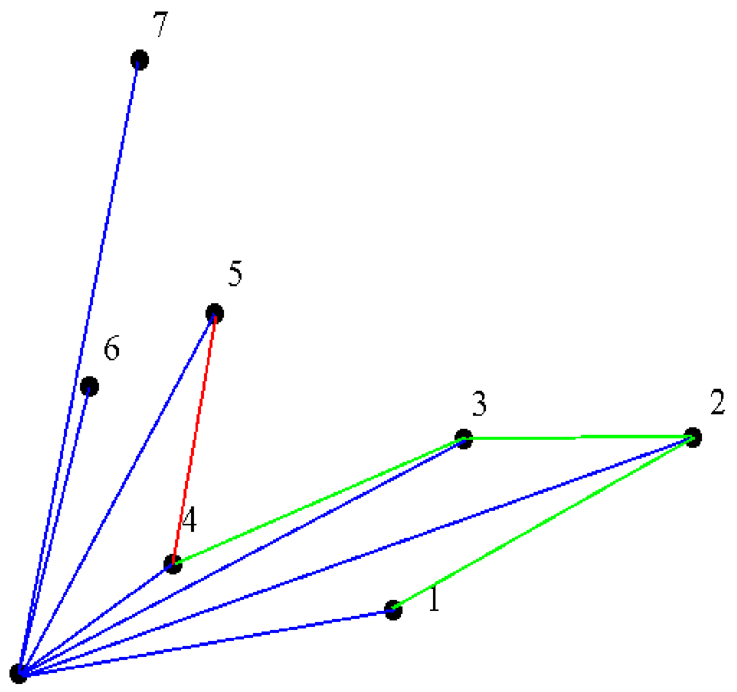
0 - 1 - 2 - 3

Graham’s Scan



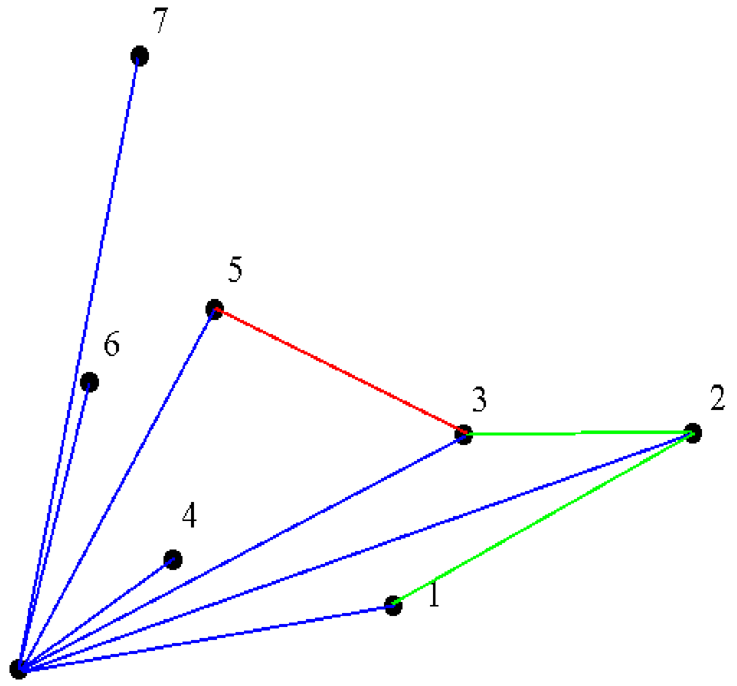
0 - 1 - 2 - 3 - 4

Graham's Scan



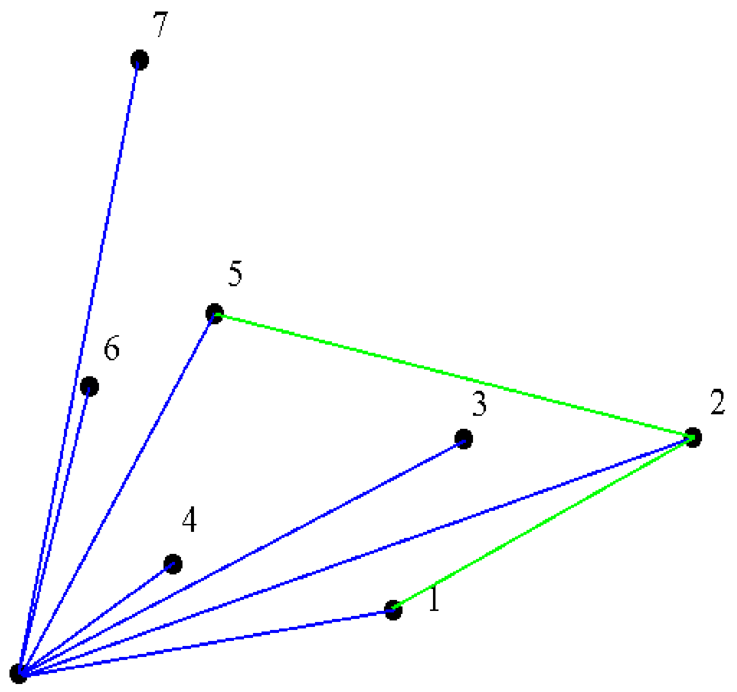
0 - 1 - 2 - 3 - 4 - 5

Graham’s Scan



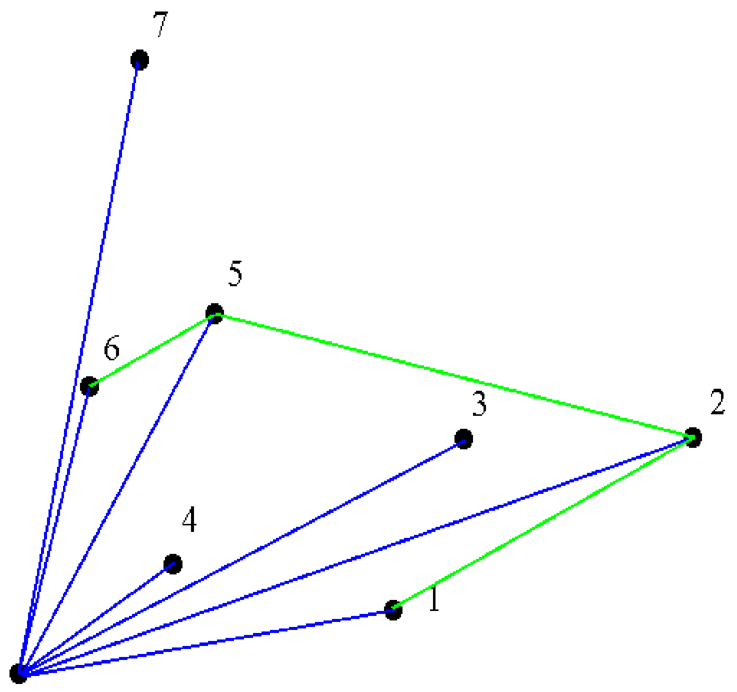
0 - 1 - 2 - 3 - 5

Graham’s Scan



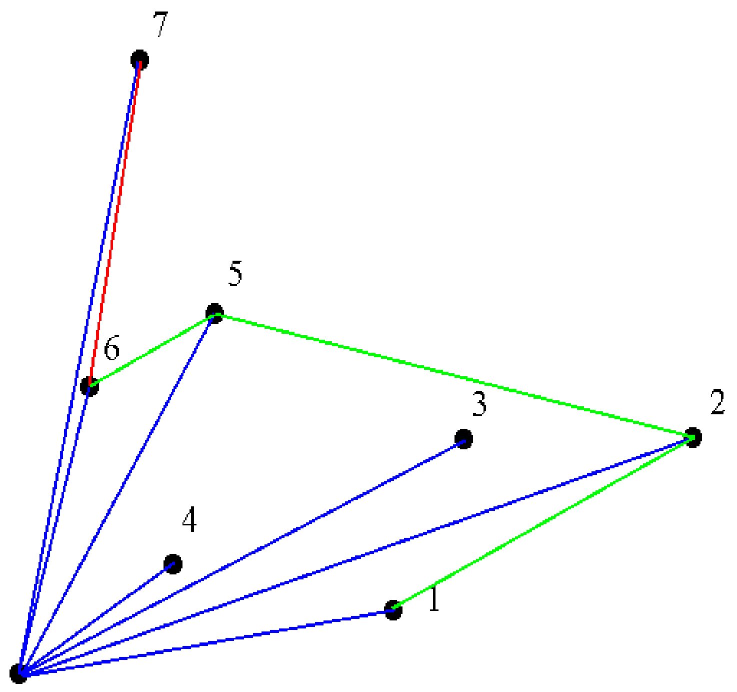
0 - 1 - 2 - 5

Graham’s Scan



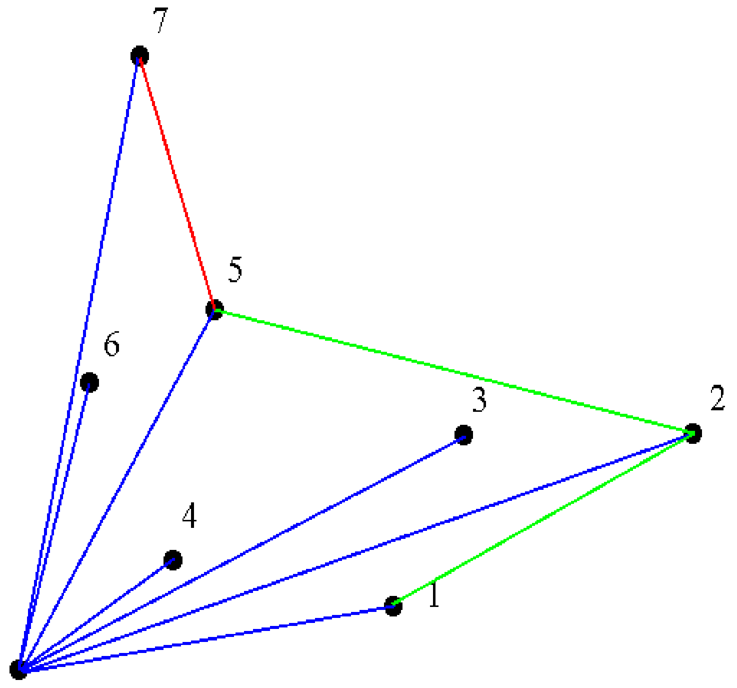
0 - 1 - 2 - 5 - 6

Graham’s Scan



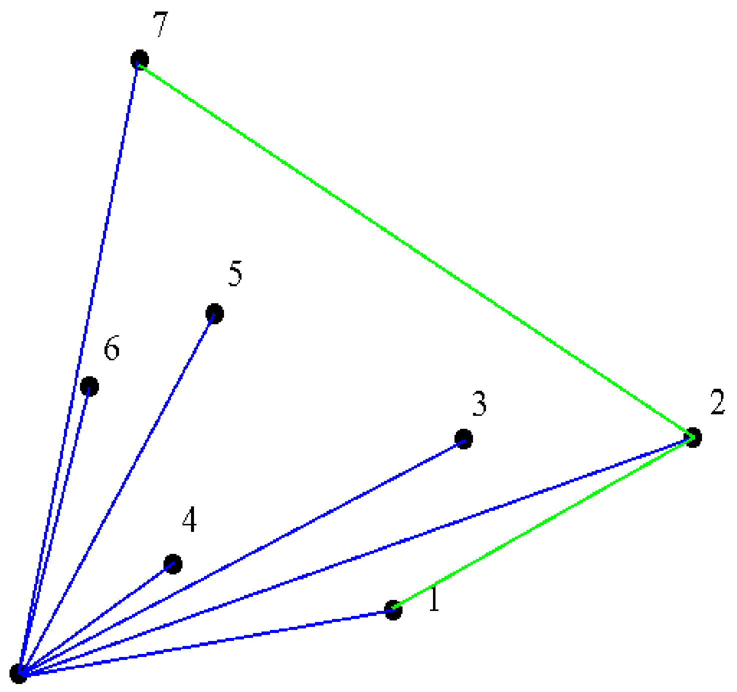
0 - 1 - 2 - 5 - 6 - 7

Graham's Scan



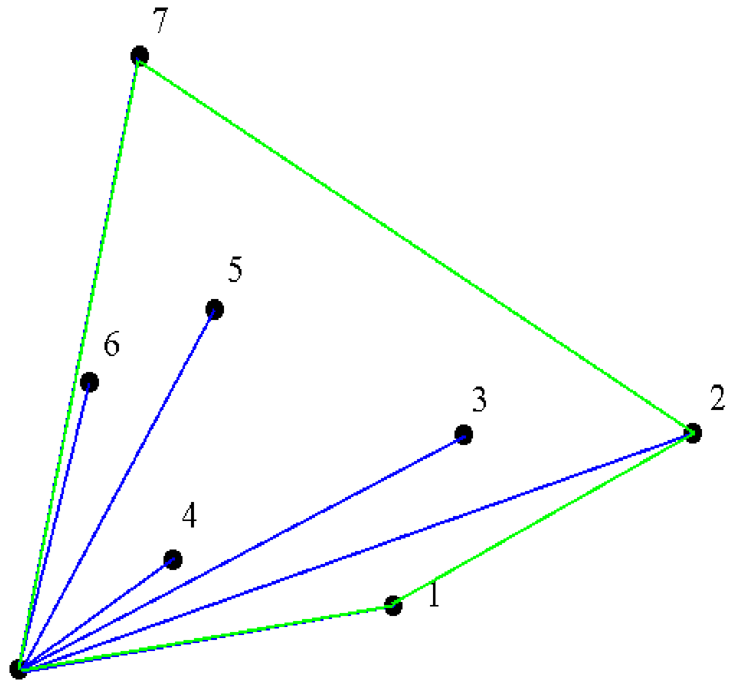
0 - 1 - 2 - 5 - 7

Graham’s Scan



0 - 1 - 2 - 7

Graham's Scan



0 - 1 - 2 - 7 - 0

What is the complexity of this algorithm?

Graham’s Scan: Complexity

- Find the lowest point: O(n)
- Sort n points: O(n*log n)
- The scan is O(n)
    - Each point is inserted once to the sequence
    - Each point is removed at most once from the sequence
- The overall complexity is O(n*log n)

Final Thoughts

- Graham's Scan runs with O(n*log n). It can be shown that this is optimal in worst case scenarios (all points are part of the convex hull).
- Jarvis' March runs with the output-sensitive O(n*h) so Jarvis is O(n^2) in worst cases, but Jarvis beats Graham if h is very small.
- The question remains whether one can achieve both output dependence and optimal worst case performance at the same time.
- Chan (1996) presented such an algorithm by cleverly combining the best of Jarvis' March and Graham Scan.
- Chan runs with O(n*log h)

Exercise

- Implement an application with the following features:
  - Initialize a set of n points at random positions
  - The user can choose to compute the convex hull via
    - the Extrem Edges algorithm
    - Jarvis' March
    - Grahams' Scan

    Display the intermediate steps of each algorithm so that the user can understand how they work.