Computational Geometry

...

The concept

Answering geometric problems using the power of your computer

Examples :

- Does these two lines intersect ? and if yes where ?
- What is the smallest circle that contains all these points ?
- What is the area of this polygon ?
- ...

For Today : Convex Hulls

Program :

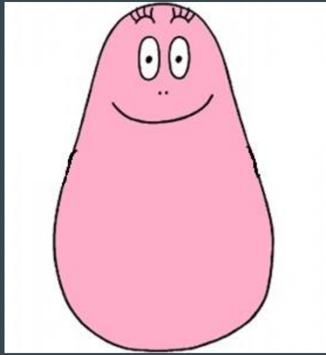
1. Quick maths (quick I promise)
2. Problem specification
3. A naive algorithm (that you should never use again)
4. Jarvis's algorithm
5. The usual algorithm

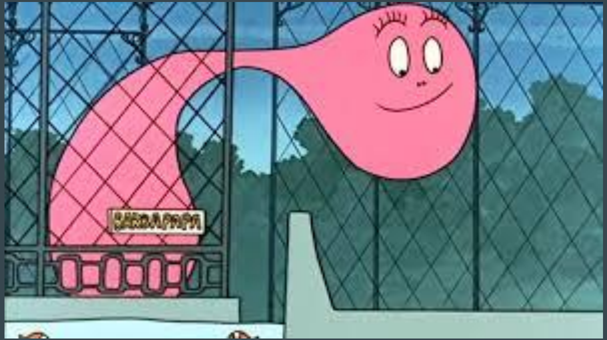
1. Quick maths, what is convexity ?

DEFINITION :

Let A be a subset of the plane, A is CONVEX if for all points a, b in A, the segment [a, b] is included in A

More clearly, A is convex if you can always join two of its points with a straight line that stays in the interior of A

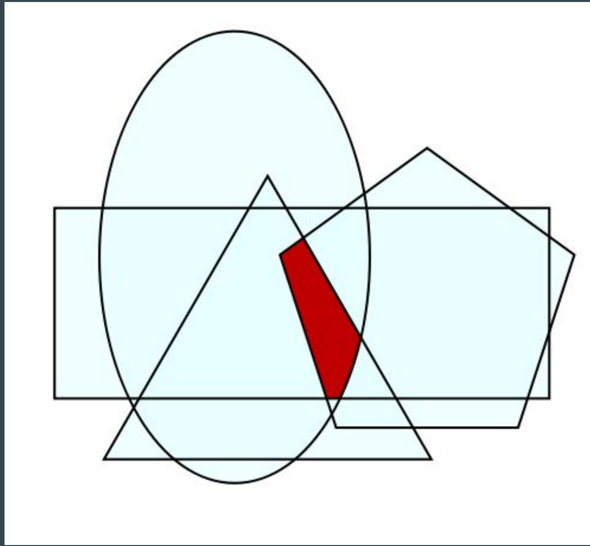




1. Quick maths, what is convexity ?

PROPERTY :

An intersection of convex sets is again a convex



$Let\ (A_i)_{i \in I}\ be\ a\ family\ of\ convex\ sets$

$and\ let\ A = \bigcap_{i \in I} A_i$

$Let\ a, b \in A\ and\ i \in I,\ then\ a \in A_i\ and\ b \in A_i$

$Hence\ since\ A_i\ is\ convex,\ [a, b] \subset A_i$

$Since\ this\ is\ verified\ for\ all\ i \in I,\ [a, b] \subset A$

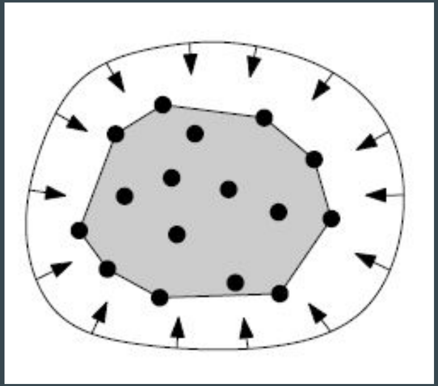
$Hence\ A\ is\ convex$

1. Quick maths, what is convexity ?

DEFINITION :

The CONVEX HULL of a subset S of the plane is the intersection of all the convex sets that contains S

aka the smallest convex set that contains S



When S is a set of points, the convex hull is always a polygon which vertices belong to S

2. Problem specification

Given S a set of points in the plane, compute its convex hull

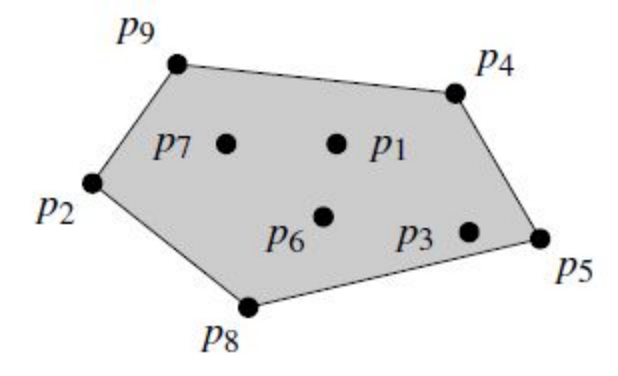
i.e. the list of the vertices of the polygon

Input

{p1, p2, p3, p4, p5, p6, p7, p8, p9}

Output

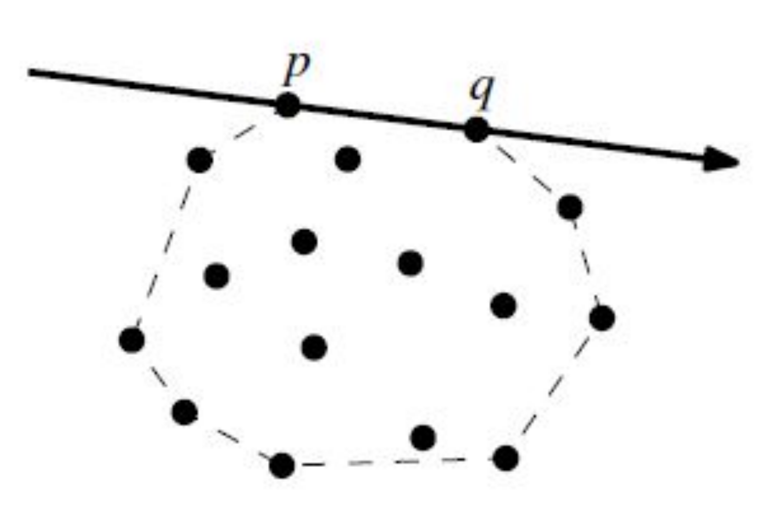
[p2, p9, p4, p5, p8]



3. A naive (and very bad) algorithm

A segment [p, q] is an edge of the convex hull iff all points of S lie to the right of [p, q]

1. compute E the set of segments that satisfy this property
2. from E, compute the convex hull by sticking back the segments



3. A naive (and very bad) algorithm

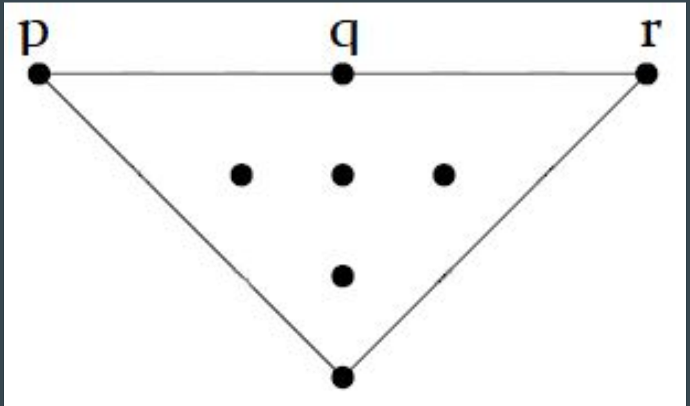
Animation !

Complexity ?

3. Why it is so bad ? First : the edge cases

If there are three aligned points p, q, r in the convex hull,

E will contain [p, q] [p, r] and [q, r], and the step 2 will fail !

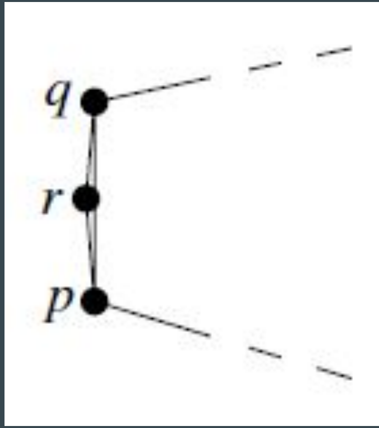


But we can fix this by making the condition of insertion in E more restrictive

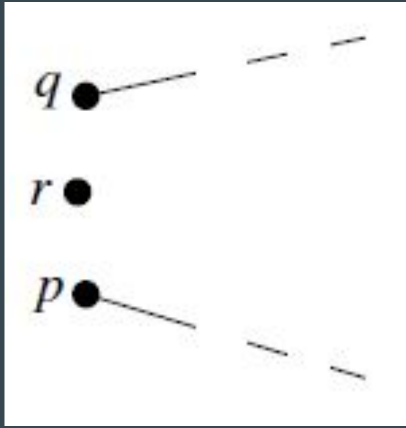
3. Why it is so bad ? Second : the float rounding error

If we are using floats for the coordinates of the points, we may compute a wrong set E due to the rounding error

We may have too many edges

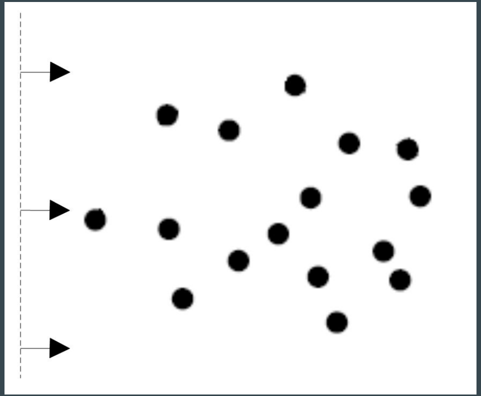


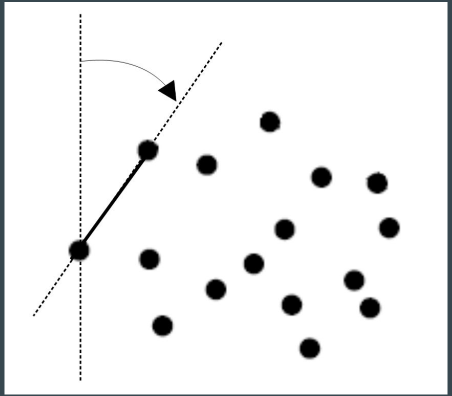
or not enough !

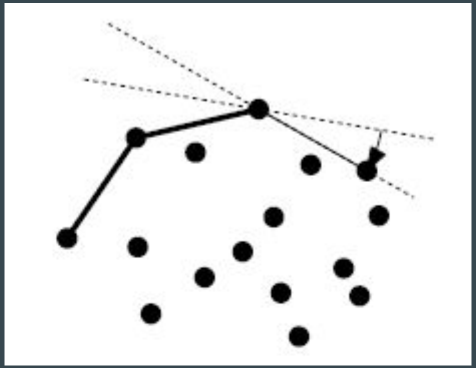


4. Jarvis's algorithme, le papier cadeau in French

Imagine constructing the convex hull using a vertical line that sweeps the plane from left to right and toggles when it hits a point of S





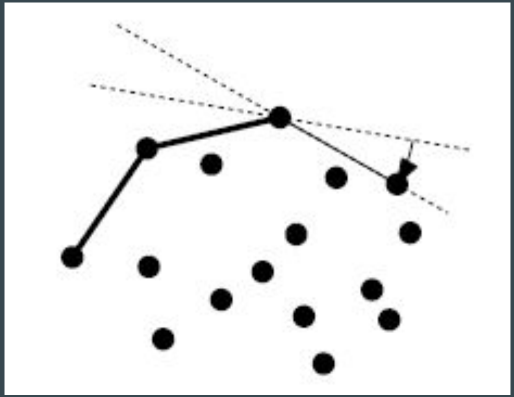


4. Jarvis's algorithme

Let's make it an algorithm

JARVIS(S):

1. Let p0 be the leftmost point of S
2. CH = []
3. p = NEXT_POINT(p0)
4. push p0 into CH
5. while p != p0:
   1. push p into CH
   2. p = NEXT_POINT(p)
6. return CH

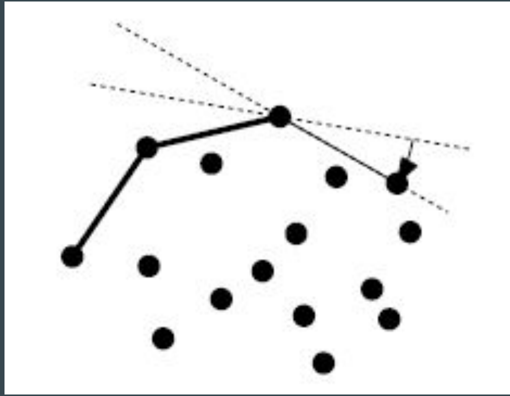


4. Jarvis's algorithme

The function NEXT_POINT(p) is just the search of a maximum for a particular order relation

NEXT_POINT(p):

1. let q0 be a point of S \ {p}
2. for all points q in S \ {p}:
    2.1. if q lies to the left of [p -> q0]:
        2.1.1. q0 = q
3. return q0



4. Jarvis's algorithm

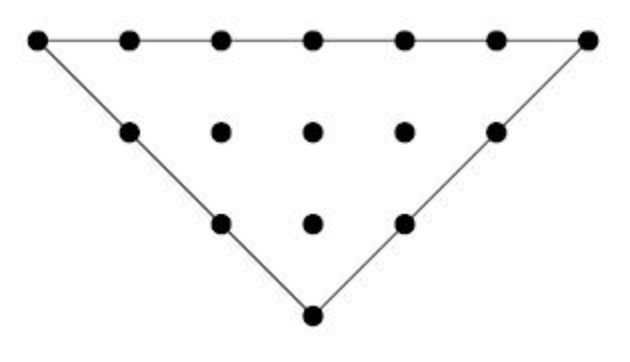
Animation !

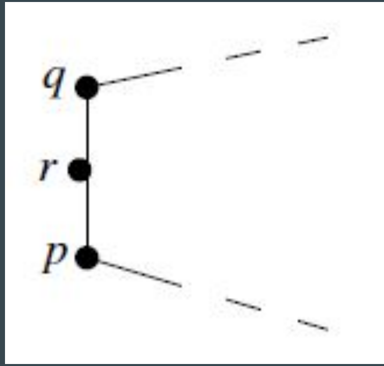
Complexity ?

4. Jarvis's algorithme, robustness

How does Jarvis's algorithm deals with aligned points ?

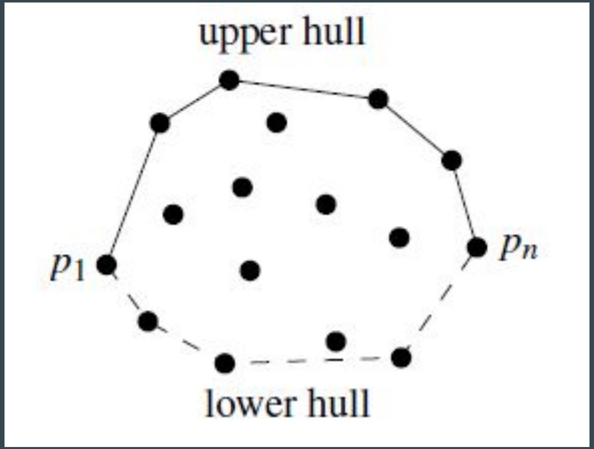
And with floating rounding error ?





5. The usual algorithm

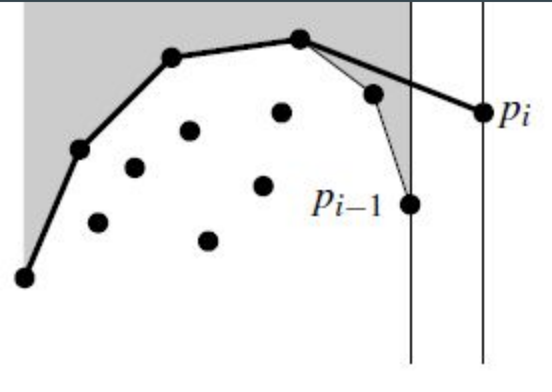
The usual approach computes independently the upper part and the lower part of the convex hull



5. The usual algorithm, computing the upper hull

Once again we will use a sweep line, but a vertical one this time

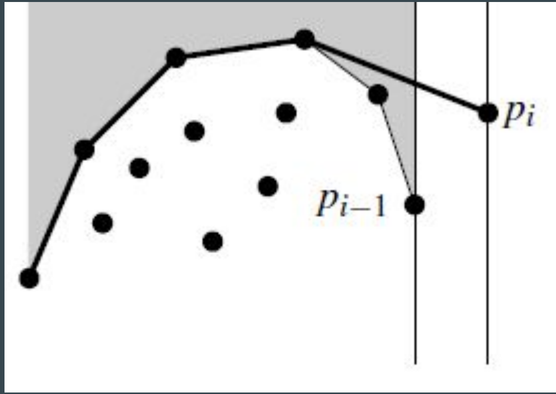
The sweep line will move left to right and compute the upper hull of the points it has already swept



5. The usual algorithm, computing the upper hull

UPPER_HULL(S):

1. Let [p1, ..., pn] be the points of S sorted from left to right
2. UH = [p1, p2]
3. For i = 3 to n:
   3.1. HANDLE_PI(UH, pi)
4. return UH



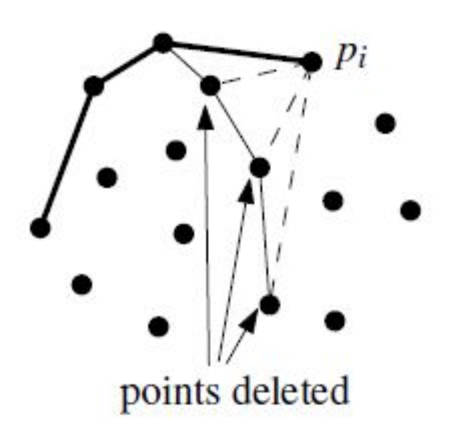
5. The usual algorithm, computing the upper hull

LEFT_TURN(UH, pi):

1. If UH has only one element:
   1.1. Return False
2. Let p and q be the last two elements of UH
3. Return true iff pi lies to the left of [p, q]

HANDLE_PI(UH, pi):

1. While LEFT_TURN(UH, pi):
   1.1. Pop the last element of UH
2. Push back pi at the end of UH

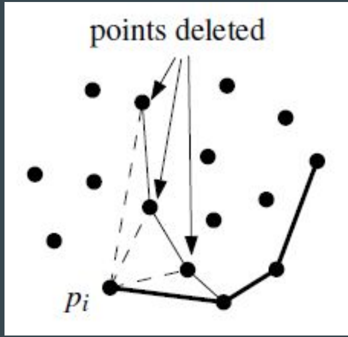


5. The usual algorithm, and the lower hull ?

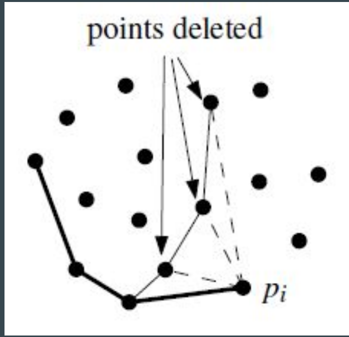
To compute the lower hull, you can either use the same HANDLE_PI function and handle points from right to left instead of left to right,

Or change HANDLE_PI checking if pi lies to the left of [p, q] instead of the right and still handle points from left to right.

Option 1 :



Option 2 :



5. The usual algorithm

Animation !

Complexity ?

The end ! Here are some exercises …

1. Implement Jarvis's algorithm and the usual algorithm
2. Show that the usual algorithm is optimal
3. Implement a divide and conquer convex hull algorithm

You know what, points are boring

4. Implement an algorithm that computes the convex hull of a polygon in linear time
5. Implement an algorithm that computes the convex hull of a set of circles with same radii

Credits

Slides : Félix Castillon for INSAlgo

strongly inspired from

COMPUTATIONAL GEOMETRY, by

Mark De Berg,

Otfried Cheong,

Marc van Kreveld,

Mark Overmars