# Geometric Algorithms



The area of geometric algorithms (also known as computational geometry) is concerned with algorithms for geometric problems. We will discuss convex hulls, triangulations, segment intersections, and maybe Delaunay triangulations all in $\mathbb{R}^2$.

## 1. Convex hulls.

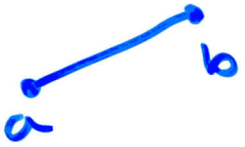
Intuitively one gets the convex hull of a point set by putting nails in a board and letting loose a rubber band stretched around all nails.

$R \subseteq \mathbb{R}^2$ is convex if $a, b \in R$ implies the line segment $ab$ lies in $R$.

convex non-convex

The convex hull of $S \subseteq \mathbb{R}^2$, $\text{conv}(S)$, is the smallest convex set $R$ so that $S \subseteq R$.

**Convex combination** Convex hull can also be defined as the set of all convex combinations of S.




$$ab = \{ x \in \mathbb{R}^2 \mid x = \lambda_1 a + \lambda_2 b,\ \lambda_1, \lambda_2 \geq 0,\ \lambda_1 + \lambda_2 = 1 \}$$

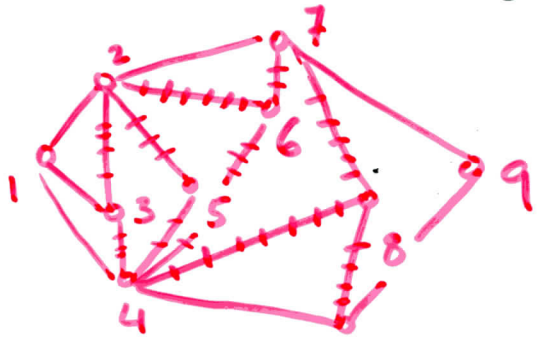
Let $S = \{p_1, p_2, \ldots p_n\} \subseteq \mathbb{R}^2$

$$\text{Conv}(S) = \{ x \in \mathbb{R}^2 \mid x = \sum_{i=1}^{n} \lambda_i p_i,\ \text{all } \lambda_i \geq 0,\ \sum_{i=1}^{n} \lambda_i = 1 \}$$

each such $x$ is called a convex combination of all $p_i$.

## 2. Plane sweep algorithm.

The algorithm constructs conv(S) by adding the points from left to right.

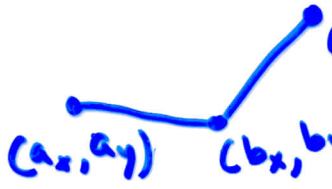


Algorithm:

1. Sort points from left to right $p_1, p_2, \dots p_n$
2. Construct the triangle $\text{conv}(\{p_1, p_2, p_3\})$
3. for $i := 4$ to $n$ do
   add $p_i$ to $\text{conv}(\{p_1, \dots p_{i-1}\})$
   by finding two tangents from $p_i$
   endfor.

Recall that <u>abc</u> is a left turn iff

$$\det \begin{vmatrix} a_x & a_y & 1 \\ b_x & b_y & 1 \\ c_x & c_y & 1 \end{vmatrix} > 0.$$



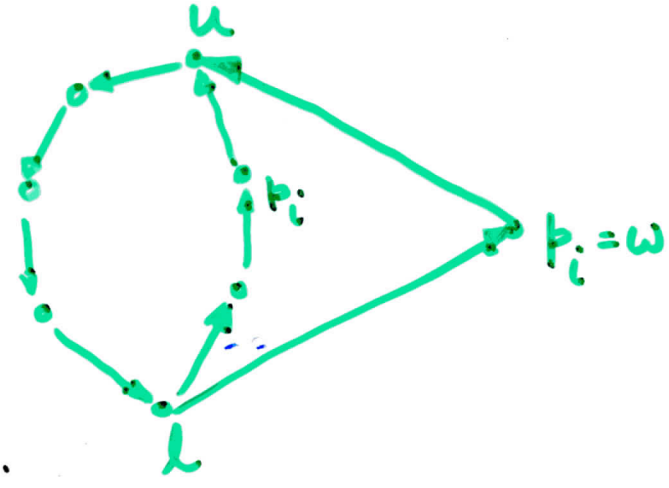

if the determinant is negative - then abc is a right-turn and if it vanishes then a,b,c are collinear.

The polygon will be stored in a doubly-linked list with pointers <u>last</u> to point to the point added last, <u>next</u> is a pointer to the CCW next point/vertex and <u>prev</u> points to the CCW next one.

# Finding the two tangents



```
u := last;
while p_i, v(u), v(next(u)) is a right-turn
        do u := next(u)
endwhile
l := last;
while p_i, v(l), v(prev(l)) is a left-turn
        do l := prev(l)
endwhile
set v(w) := p_i; next(w) := u; prev(w) := l;
    prev(u) := w; next(l) := w.
```




If this procedure takes k steps within the 2 while loops, then k-1 points are deleted from convex hull.

Since each point is deleted at most once, the total time to construct conv(S), $|S| = n$, is $O(n)$ after the sorting.

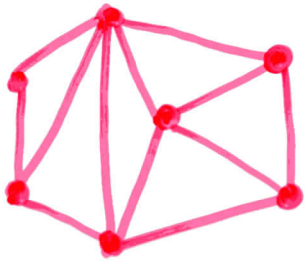
So, the total time is $O(n \log n)$.

Can this be improved ?

# 3. Triangulations.

The convex hull algorithm implicitly builds a triangulation also.

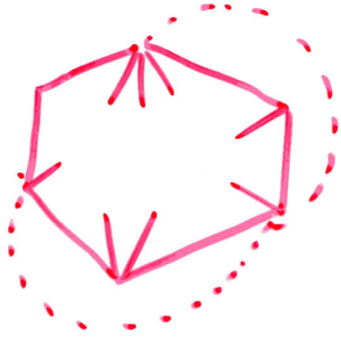
A (geometric) triangulation of a finite set $S \subseteq \mathbb{R}^2$ is a maximally connected straight-line plane graph $(S, E)$



$S$ is the vertex set of the triangulation, and every face is a triangle, except possibly the outer face. Let $|S| = n$ and $h$ be the number of points of $\text{conv}(S)$.

**Claim.** Any (geometric) triangulation of $S$ has $|E| = e = 3n - h - 3$ edges and $f = 2n - h - 1$ faces (or $f - 1 = 2n - h - 2$ triangles).

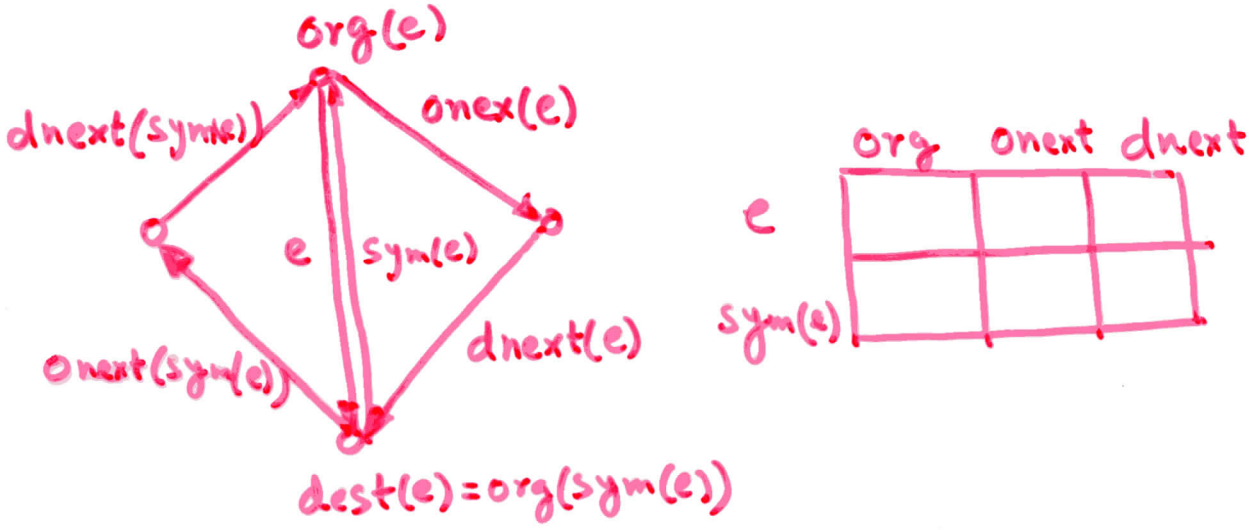
- Convince yourself that the claim is correct. Recall $e = 3n - 6$ and $f = 2n - 4$ for abstract triangulation of $S$. Such abstrat triangulation is obtained by adding $h - 3$ curved edges.



# 4. Quad-edge data structure



In contrast to abstract graphs, every plane graph has faces, and the edges incident to a vertex are ordered around the vertex. We modify the adjacency structure to reflect this additional information. The role of an edge becomes important.




Edges are now interconnected in a richer way.

```
type  Point = record  x, y : integer end;
      Vertex = record  p: Point;  adj : DEDGEINDEX end
      Vertexset = array [1...n] of Vertex;
      DEDGEINDEX = record i:= 1...3n;  j = 0,1 end;
      Edge = array [0,1] of record
                              org : 1,..,n
                              onext, dnext : DEDGEINDEX
                            end;
      Edgeset = array [1..3n] of Edge;
var   V: Vertexset;  E : Edgeset;
```

Access to this data structure is most conveniently provided through a collection of mini-routines.

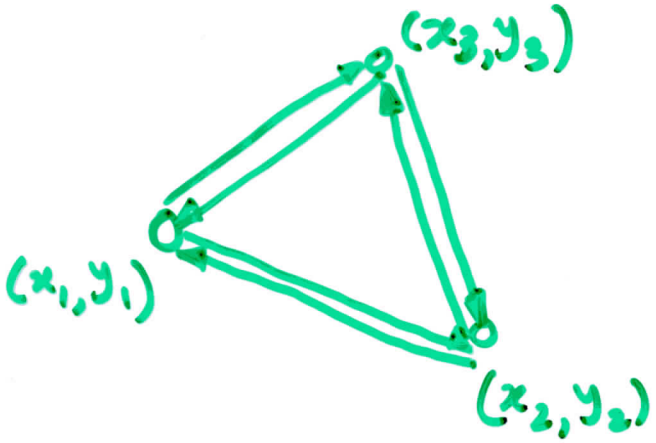
```
function Org (d: EDGEINDEX): 1..n;
    return E[d.i][d.j].org

function Sym (d: EDGEINDEX): DEDGEINDEX
    return (d.i, (d.j + 1) mod 2).

function dest (d: DEDGEINDEX): 1..n;
    return org(Sym(d))

function onext (d: DEDGEINDEX): DEDGEINDEX
    return (E[d.i][d.j].onext.i, E[d.i][d.j].onext.j)
```

Similarly dnext function can be obtained by replacing all onexts by dnexts.

```
procedure set-dnext (d, n : DEDGEINDEX)
    E[d.i][d.j].dnext.i := n.i;
    E[d.i][d.j].dnext.j := n.j;
```

# 5. Initializing a triangle.

As an exercise let us look at the first step of the plane-sweep algorithm: initializing a triangle:




V

| | |
|---|---|
| $x_1$ | 1,0 |
| $y_1$ | |
| $x_2$ | 2,0 |
| $y_2$ | |
| $x_3$ | 3,0 |
| $y_3$ | |

E

| | | |
|---|---|---|
| 1 | 3,1 | 2,1 |
| 2 | 2,0 | 3,0 |
| 2 | 1,1 | 3,1 |
| 3 | 3,0 | 1,0 |
| 3 | 2,1 | 1,1 |
| 1 | 1,0 | 2,0 |

org onext dnext

Check: $f = onext(e)$ iff $e = sym(dnext(sym(f)))$

| | | | | | |
|---|---|---|---|---|---|
| 3,1 | 1,0 | | 1,0 | 1,1 | 3,0 3,1 |
| 2,0 | 1,1 | | 1,1 | 1,0 | 2,1, 2,0 |

etc.

We skip the rest of the algorithm as it is the same as for convex hull, only that it is tedious to get all operations right for the quad-edge data structure.

Summary: A triangulation of a set of $n$ points in $\mathbb{R}^2$ can be computed in time $O(n \log n)$.