2. Convex Hulls in 2D

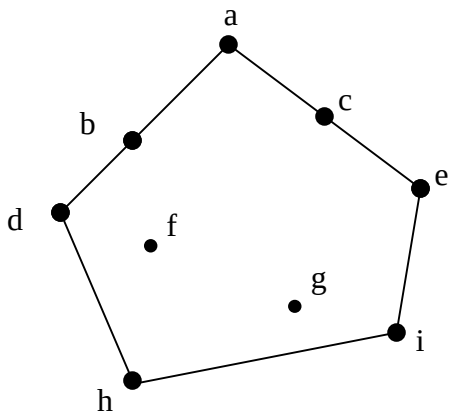
Convex – a set S is convex if $x,y \in S$ implies that the (closed) segment $xy \subseteq S$.

A convex combination of points $x_1,x_2,\ldots, x_k$ is a sum of the form $\alpha_1 x_1 + \ldots + \alpha_k x_k$, with $\alpha_i \geq 0$ for all i and $\alpha_1 + \ldots + \alpha_k = 1$.

Points	Convex combination
2	Line segment
3(noncollinear)	triangle
4 (noncoplanar)	tetrahedron

Theorem The convex hull of a set of points S is the set of all convex combinations of points of S.

The extreme points of a set S of points in the plane are the vertices of the convex hull at which the interior angle is strictly convex, less than π.



Extreme points – ***a, d, e, h*** and ***i***.

An extreme edge is an edge whose endpoints are both extreme vertices. In the above example, the extreme edges are *ad, dh, hi, ie* and *ea*.

We consider the following algorithms and their complexity:

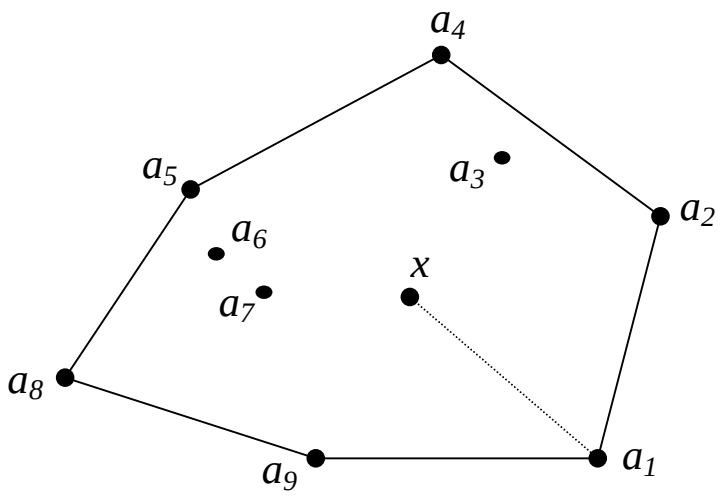
(i) Graham's Algorithm,
(ii) Incremental algorithm.

2.1 Graham's Algorithm (1972)

This algorithm is based on the observation that , for the extreme points $p_1,p_2,p_3, \ldots, p_n$, the point $p_{i+1}$is always at the left side of the extreme edge $p_{i-1}p_i$.

Assume that a point x is interior to the hull and no three points are collinear.

Step 1: Sort the points by angle, counterclockwise about x. The points are given in order by $a_1,a_2,a_3, \ldots, a_{n-1}$



Let S be a stack of extreme points in counterclockwise direction. For instance, in the above example, S may be given by

$S=(a_1, a_2, a_4, a_5, a_8, a_9)$.

We observe that if b_{i-1}, b_i & b_{i+1} are three consecutive points in S, b_{i+1} is at the left side of $b_{i-1}b_i$.

We build up S in the following way:

Step 2: Start S with a_1 and a_2. Points are added to or deleted from S according to the following rule:
Let b_{i-1} be the last but one point and b_i be the last point in S at certain time. For a point a_j which is after b_i in the sorted list, if a_j is at the left side of $b_{i-1}b_i$, then add a_j to S as the last point; otherwise delete b_i.

Step 3: Go back to step 2 and consider a_{j+1} until the sorted list is exhausted.

In the above example , S builds up in this way:

$S= (a_1,a_2), (a_1,a_2,a_3), (a_1,a_2), (a_1,a_2,a_4), (a_1,a_2,a_4,a_5), (a_1,a_2,a_4,a_5,a_6), (a_1,a_2,a_4,a_5,a_6,a_7), (a_1,a_2,a_4,a_5,a_6), (a_1,a_2,a_4,a_5), (a_1,a_2,a_4,a_5,a_8)$, $(a_1,a_2,a_4,a_5,a_8,a_9)$

Algorithm: Graham Scan, Version A

Find the interior point x; label it p_0.
Sort all other points angularly about x;

Label $P_1,\ldots, P_{n-1}$
Stack $S = (P_1, P_2) = (P_{t-1}, P_t)$;
t indexes top
while $i <$ n do
    if P_i is left of (P_{t-1}, P_t)
        then Push (S,i) and increment i
        else Pop(S)

* In the above, we may also assume that P_0 is an extreme vertex.

Complexity

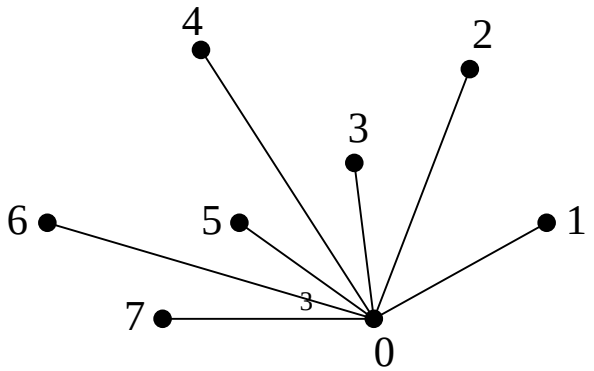
- Sorting of n points angulary	O (n*log*n)
- Push (S,i)	n
- Pop (S)	< n

∴ "Worst-case optimal" = O(n*log*n).

Difficulties to overcome

1. Sorting of points

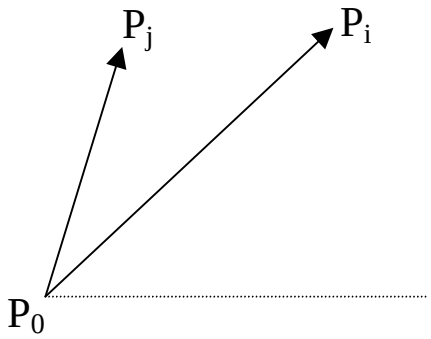
Instead of looking for an interior point, we choose an extreme point as a new sorting origin. We will use the lowest point (the rightmost of the lowest if there are several lowest points), which is clearly on the hull.



Sorting of points angularly about P_0

In the sorting

$$P_i < P_j \Leftrightarrow \text{angle } (P_0P_i) < \text{angle } (P_0P_j)$$



Instead of computing the angles, we use the following simple method:

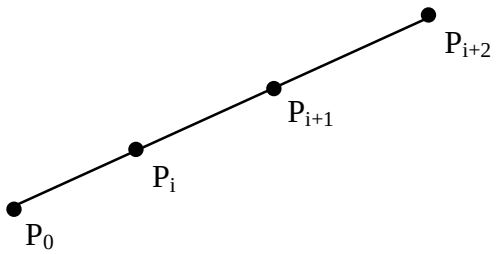
$$Pi < Pj \Leftrightarrow \mathcal{A}(P_0,P_i,P_j) > 0$$

area

2. Collinearities :

In the sorting of collinear points, we use the rule that closer points to P_0 are treated as earlier in the sorting sequence, i.e.

$$a < b \Leftrightarrow | a - P_0 | < | b - P_0 |$$

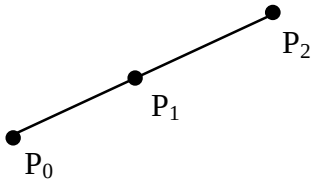


If P_{t-1} and P_t are the top two points on the stack, we require a strict left turn (P_{t-1}, P_t, P_i) to push P_i onto the stack.

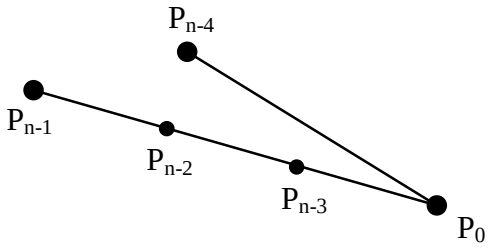
If P_t is collinear with P_{t-1} and P_i, it will be deleted.

3. Start and Stop of Loop:

To start the loop, the first two points in the stack must be extreme points. In the way that we sort the points, P_0, P_1 and P_{n-1} must be on the hull. P_0 is an extreme point. P_1 may not be as shown below,



P_{n-1} is also an extreme point from the way that we label collinear points:



We initialize $S = (P_{n-1}, P_0)$ so that the stack will always contain at least two points.

4. With the above rules, P_{n-1} appears twice on the final stack: the first and the last element. One final stack pop is necessary.

Algorithm: Graham Scan , Version B

Find rightmost lowest point; label it P_0.
Sort all other points angularly about P_0, break ties in favor of closeness to P_0; label $P_1,\ldots, P_{n-1}$.

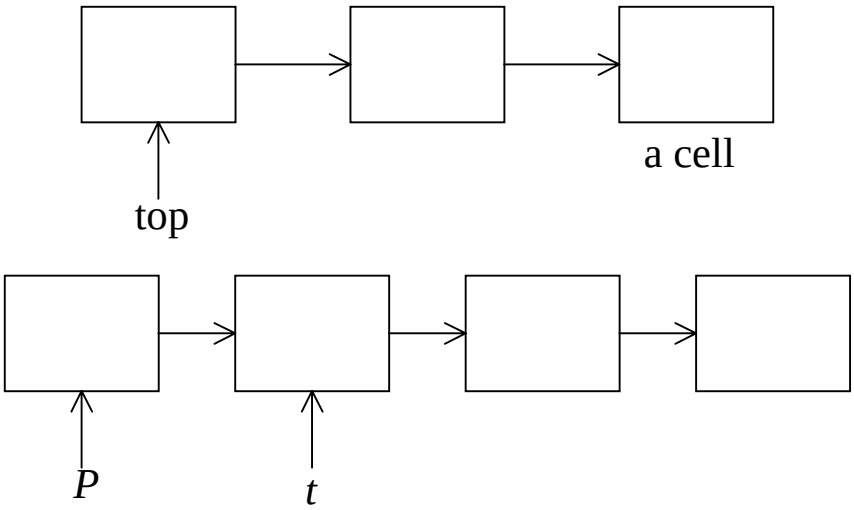
Stack $S = (P_{n-1}, P_0) = (P_{t-1}, P_t)$; t indexes top.
i =1

while i < n do
    if P_i is strictly left of (P_{t-1}, P_t)
        then Push (S,i) and set $i \leftarrow i+1$
        else Pop (S).

Data Representation

Input Points – stored in an integer array *P*: *P*[0],…, *P*[n-i] corresponding to $P_0, P_1 \ldots, P_{n-1}$.(Code 3.1)

Stack— represented by a singly–linked list of cells, each of which "contains" a point. The stack top is head of the list. (Code 3.2)



Push links a new element *P* to the lead (Code 3.3).
Pop removes the top element and returns its storage to the system (Code 3.4).

2.2 Incremental Algorithm

Graham's algorithm has no obvious extension to 3D as the angular sorting has no direct counterpart.

Basic plan: Add the points one at a time, at each step constructing the hull of the first *k* points, and using that hull to incorporate the next point.

Let Q be a convex hull and p be the next point to be considered.

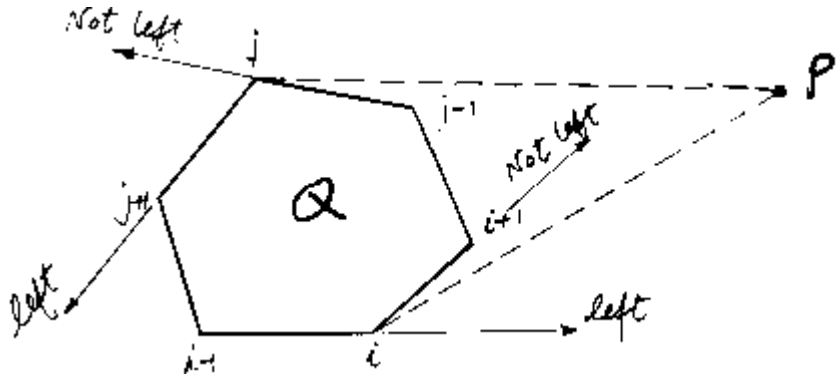
Case 1. $p \in Q$.

$p \in Q \quad \Leftrightarrow$ p is left of or on every directed edge.

Case 2. $p \notin Q$.

There exist two extreme points on Q such that the following statement is wrong:

"p is left or on both of $p_{i-1}p_i$ and p_ip_{i+1}".



The new hull is $(p_0, p_1, \ldots, p_{i-1}, p_i, p, p_j, \ldots, p_{n-1})$.

Algorithm: Tangent Points

for i = 0 to n-1 do
if *xor* (p left or on (p_{i-1}, p_i), p left or on (p_i, p_{i+1}))
then p_i is point of tangency.

Complexity
Number of checking at the k-th hull = *k*.
Total checking of all the n points
= 3 + 4 + 5 + … + n
= $O(n^2)$

The incremental algorithm can easily be extended to 3D.