Competitive Programming

From Problem 2 Solution in O(1)

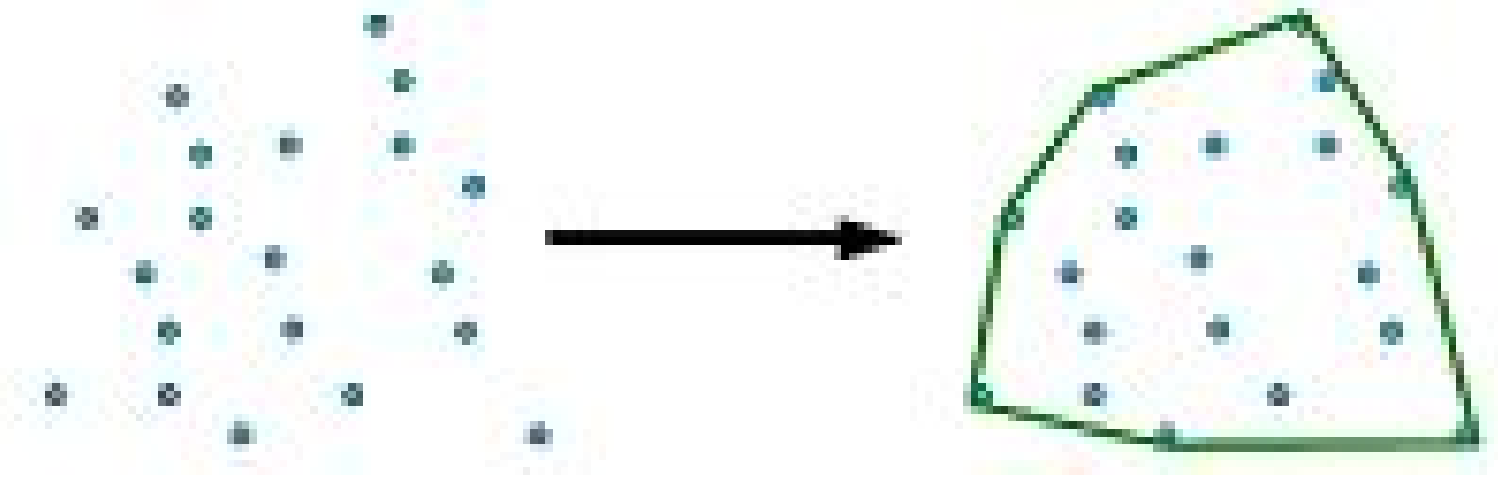
Computational Geometry

Convex Hull

Mostafa Saad Ibrahim
PhD Student @ Simon Fraser University

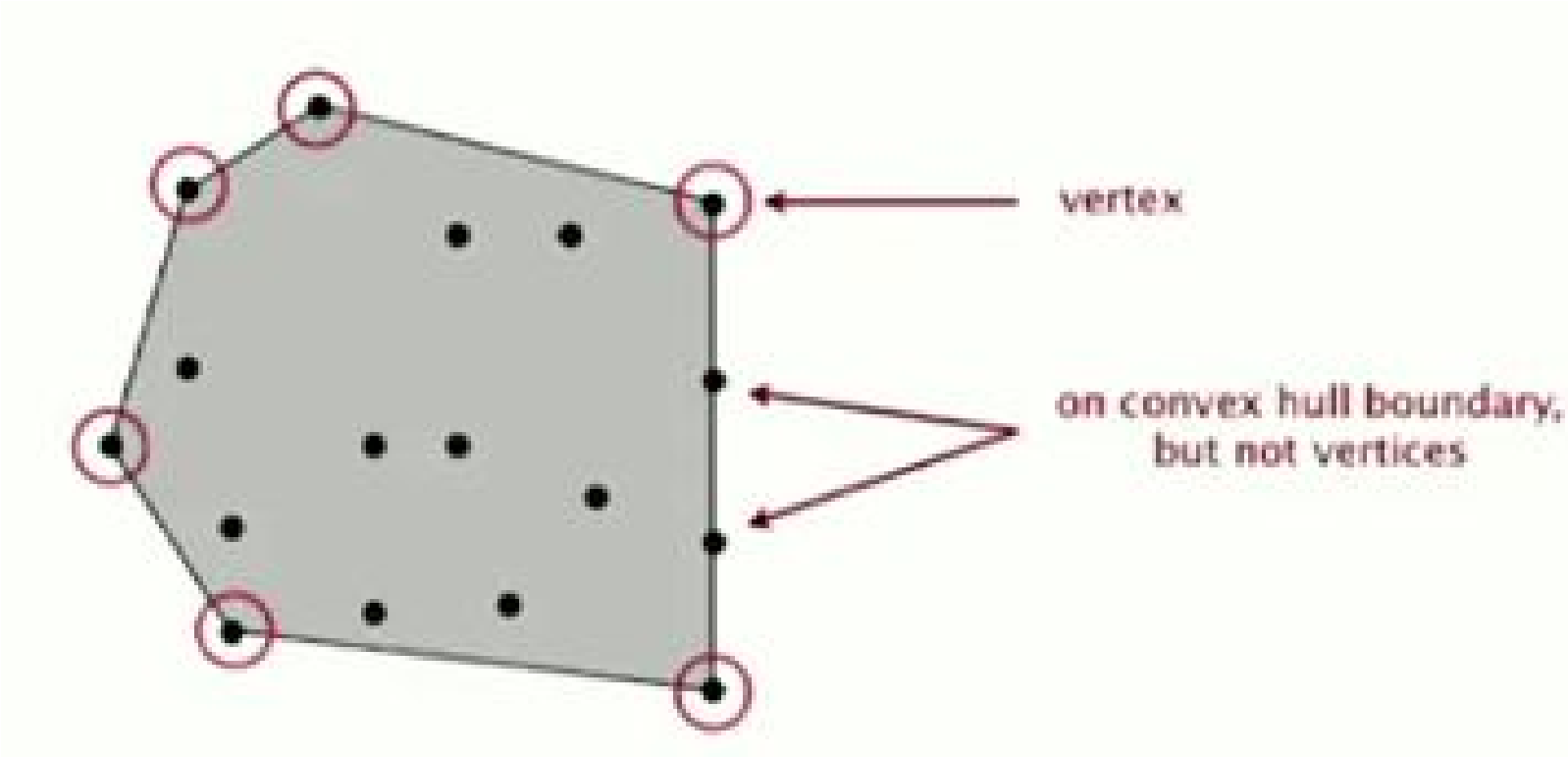
Convex Hull

- Given a set of points: Find is **the smallest convex polygon** that contains all the points of it. (Think a fence wrapping around points)



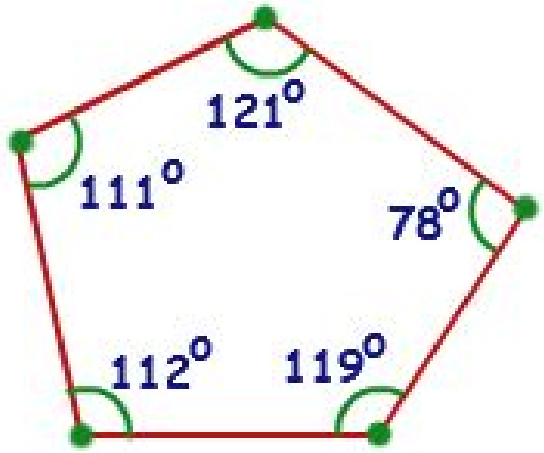
Src: http://www.geeksforgeeks.org/convex-hull-set-2-graham-scan/

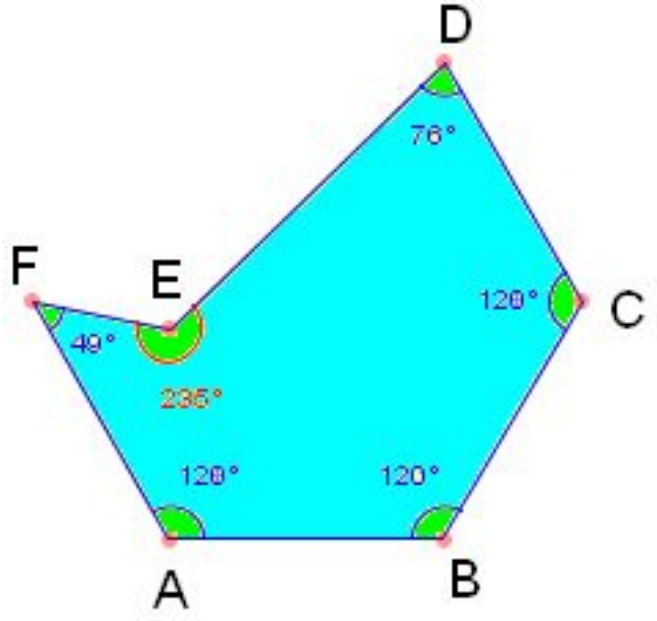
Convex Hull



Src: https://www.youtube.com/watch?v=0HZaRu5IupM

Recall Convex vs Concave Polygon

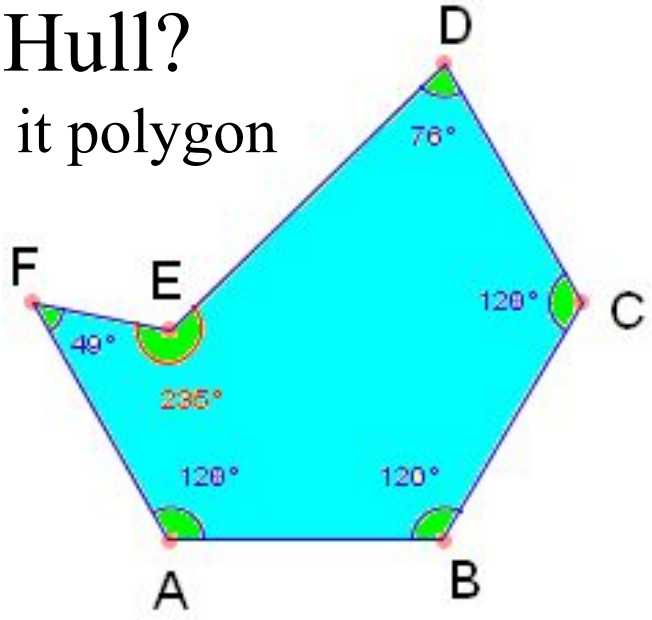




Src: http://image.tutorvista.com/cms/images/38/concave-polygon.jpg http://image.mathcaptain.com/cms/images/108/convex-polygon.PNG

Recall Convex vs Concave Polyg

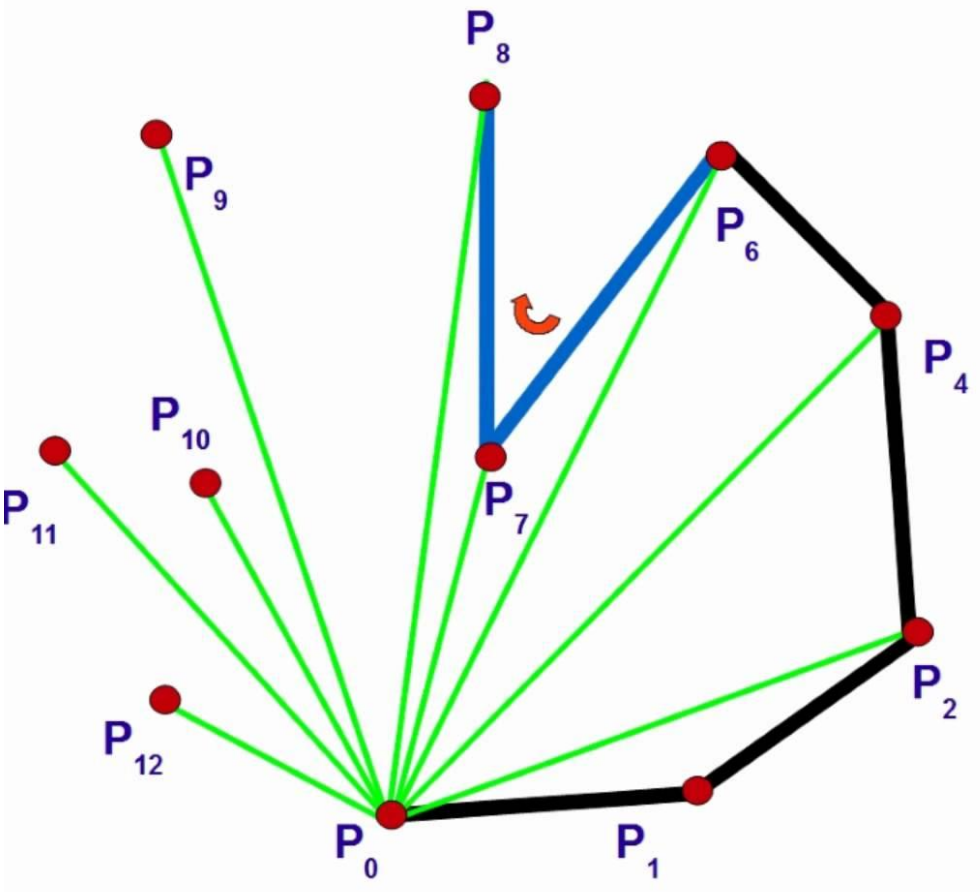
- Given a polygon with points ordered CCW
  - Assume we build CH for points A-E and adding F
- For triple points: D-E-F, Can we decide if E is **NOT** part of final Convex Hull?
  - If angle(DEF) > 180 ⇒ NOT in it polygon
  - This is equal to IsLeft point (e.g. using cross product)



Points definitely in the Hull

- Can we identify some initial point that is definitely part of the final convex hull?
- It must be a very boundary point
- 4 points satisfy that (think rectangle boundary)
  - Most of: top left, top right, bottom left, bottom right
- Most Bottom left point
  - It has the minimum Y
  - It has the smallest X if multiple points with same Y
  - It creates angles <= 180 with other points / edges
  - These are great info for ordering a polygon ccw

Most Bottom Left Point



Src: https://i.ytimg.com/vi/dw120Yclav8/maxresdefault.jpg

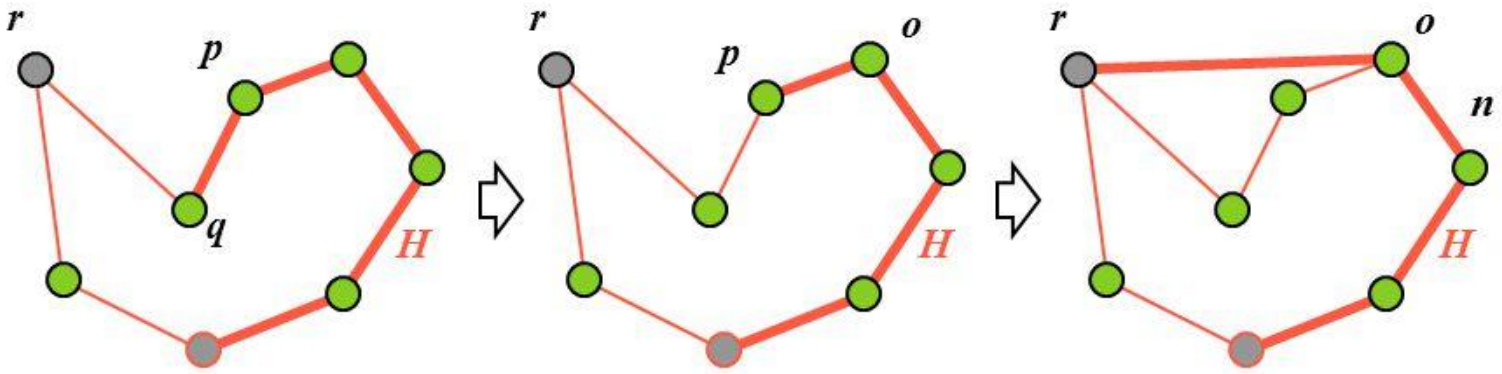
- **P0** is the most bottom left point
- Think in its angle with others
- Think in sorting point based on angle with P0

Graham Scan Algorithm

- It is an O(nlogn) algorithm for finding the CH
- Graham plays on the mentioned 2 core points
- Ordering the polyon
  - Find the most bottom right point (must be in final CH)
  - Order the polygon based on the angle with it ⇒ we reduce much hassle of considerations
  - The polygon is ordered counterclockwise
- Then, in incremental manner, identify points that are NOT part of the final convex hull

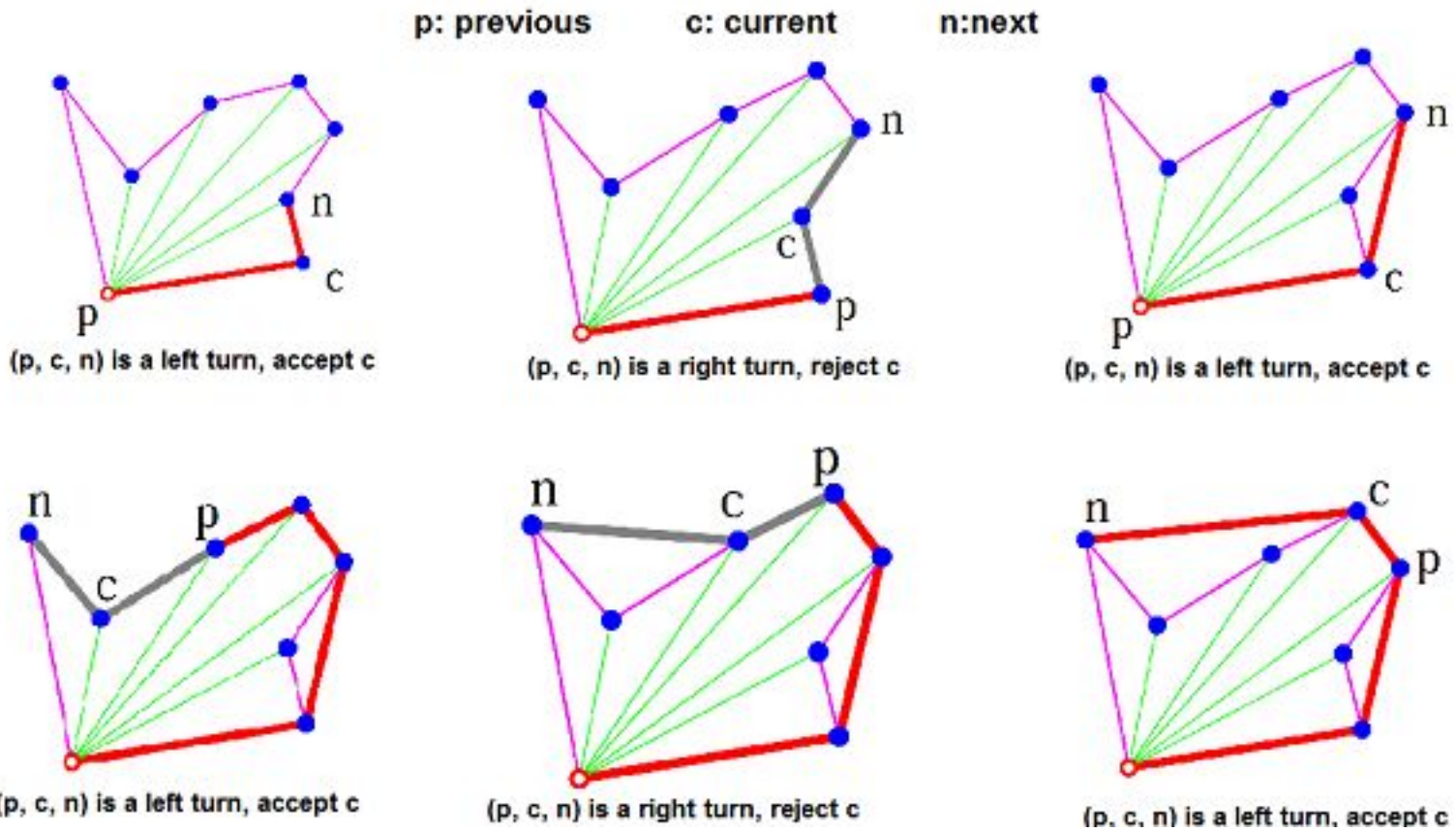
Graham Scan Algorithm

- The polygon is traversed in sorted order and a sequence ***H*** of vertices in the hull is maintained
- For each point ***a***, add ***a*** to ***H***
  - While the last turn is a right turn, remove the second to last point from ***H***
- In the image below, ***p, q, r*** forms a right turn, and thus ***q*** is removed from ***H***. Similarly, ***o, p, r*** forms a right turn, and thus ***p*** is removed from ***H***.



Src: http://images.slideplayer.com/14/4310635/slides/slide_11.jpg

Graham Scan Algorithm



Src: http://www.geeksforgeeks.org/convex-hull-set-2-graham-scan/

Sorting in O(nlogn)

```
struct angleCMP {
    point center;
    angleCMP(point c) : center(c) {}
    bool operator () (const point &p0, const point p1) const {
        if( dcmp(cp(p0-center, p1-center), 0) == 0) {
            if(fabs(p0.Y - p1.Y) < EPS)
                return p0.X < p1.X;
            return p0.Y < p1.Y;
        }
        return  cp(p0-center, p1-center) < 0;
    }
};

vector<point> ConvexHull(vector<point> points) {
    if(sz(points) <= 1) return points;

    // Get most bottom left point
    rep(i, points)
        if( MP(points[i].Y, points[i].X) < MP(points[0].Y, points[0].X))
            swap(points[0], points[i]);

    sort( points.begin()+1, points.end(), angleCMP(points[0]));
```

Processing in O(n)

```
vector<point> p, ch;

/*
//To remove co-linear points, un-comment this part
rep(i, points) {
    if(sz(p) > 1 && !cp(p.back()-p[0], points[i]-p[0]) ) {
        if( length(points[0]-p.back()) < length(points[0]-points[i]) )
            p.back() = points[i];
    }
    else
        p.push_back( points[i] );
}
points = p;
*/

rep(i, points) {
    while( sz(ch) > 1 &&
        cp(ch[sz(ch)-2]-ch[sz(ch)-1], points[i]-ch[sz(ch)-1]) < 0)
            ch.pop_back();  // ALSO Make it <= to remove co-linear points
    ch.push_back(points[i]);
}
if(sz(ch) >= 3) // Not a line
    ch.push_back(ch[0]);
return ch;
```

Other Algorithms

- There are many other algorithms, however Graham is most probably enough
  - One can still study them to learn about thinking in geometry
  - One way is using sweep line: See1, See2
- Chan's algorithm is faster: O(nlogh)
  - h: # of points in final convex hull
  - This is so fast if h is really small
- Some approximation algorithms are O(n)

Smallest Bounding Containers

- Given set of points find the smallest container
  - Smallest rectangle: trivial - just minimize 4 boundaries
  - Smallest circle - we studied
  - Smallest convex polygon - today session
  - Smallest diamond = Your turn
  - Won't come in competitions :) .. just little thinking

تم بحمد الله

علمكم الله ما ينفعكم

ونفعكم بما تعلمتم

وزادكم علماً