# Convex Hull

## CS 491 – Competitive Programming

Dr. Mattox Beckman

University of Illinois at Urbana-Champaign
Department of Computer Science

Fall 2023

# Objectives

- The the `ccw` function to check if a polygon is convex or not.
- Compute the perimeter and area or an arbitrary polygon
- Find a polygon (convex hull) that fits a set of points.
- Most code samples from Competitive Programming 3.

# Representation

- Represent a polygon using a vector of points.
- Add them in counterclockwise order.
- The first point should be re-added as the last point.

```
vii mypoly;
mypoly.push_back(ii(2,2));
mypoly.push_back(ii(27,10));
mypoly.push_back(ii(30,25));
mypoly.push_back(ii(6,20));
mypoly.push_back(ii(2,20));
mypoly.push_back(ii(2,2));
```

# Getting the Area

$$A = \frac{1}{2} \begin{bmatrix} x_0 & y_0 \\ x_1 & y_1 \\ x_2 & y_2 \\ \cdots & \cdots \\ x_n & y_n \end{bmatrix}$$

$$= x_0y_1 + x_1y_2 + \cdots + x_ny_0 \quad - x_1y_0 - x_2y_1 - \cdots - x_0y_n$$

```
1  double area(const vector<point> &P) {
2      double result = 0.0, x1, y1, x2, y2;
3      for (int i = 0; i < (int)P.size()-1; i++) {
4          x1 = P[i].x; x2 = P[i+1].x;
5          y1 = P[i].y; y2 = P[i+1].y;
6          result += (x1 * y2 - x2 * y1);
7      }
8      return fabs(result) / 2.0;
9  }
```

# Cross Product and CCW

- Check if all turns on the perimeter turn the same way.

```
1  double cross(vec a, vec b) {
2      return a.x*b.y - a.y*b.x;
3  }
4
5  point ccw(point p, point q, point r) {
6      return cross(toVec(p,q), toVec(p,r)) > EPS;
7  }
```

# IsConvex

```
1  bool isConvex(const vector<point> &P) {
2      int sz = (int)P.size();
3      if (sz <= 3) return false;
4      bool isLeft = ccw(P[0], P[1], P[2]); // start
5      for (int i = 1; i < sz-1; i++)       // compare
6          if (ccw(P[i], P[i+1], P[(i+2) == sz ? 1 : i+2]) != is
7              return false; // different sign -> this polygon is
8      return true;          // this polygon is convex
9  }
```

# Convex Hull

- Given: a bunch of points
- We want: a minimal convex polygon to contain them.
- Basic algorithm:
  - Pick lowest point (and rightmost, if tie) as a *pivot*
  - Sort the points by the angle to the pivot. Call these $p_1, p_2$, etc.
  - Push pivot, $p_1$, to a stack.
  - Repeat from $p_2$:
    - Push point to stack
    - If (angle of top three points) is clockwise, delete midpoint from stack.
- See video for example

# Sorting by Angle

```
1  point pivot(0, 0);
2  bool angleCmp(point a, point b) {// angle-sorting function
3     if (collinear(pivot, a, b)) // special case
4        return dist(pivot, a) < dist(pivot, b);
5   // check which one is closer
6     double d1x = a.x - pivot.x, d1y = a.y - pivot.y;
7     double d2x = b.x - pivot.x, d2y = b.y - pivot.y;
8     return (atan2(d1y, d1x) - atan2(d2y, d2x)) < 0;
9  }
```

# Convex Hull

```
10  vector<point> CH(vector<point> P) {
11      int i, j, n = (int)P.size();
12      if (n <= 3) {
13          // safeguard from corner case
14          if (!(P[0] == P[n-1])) P.push_back(P[0]);
15          return P;
16      }
17
18  // first, find P0 = point with lowest Y and rightmost X
19      int P0 = 0;
20      for (i = 1; i < n; i++)
21          if (P[i].y < P[P0].y ||
22              (P[i].y == P[P0].y && P[i].x > P[P0].x))
23              P0 = i;
24      point temp = P[0]; P[0] = P[P0]; P[P0] = temp; // swap i
25      pivot = P[0];
```

# Convex Hull Code, 2

```
26      // Sort the remainders
27      sort(++P.begin(), P.end(), angleCmp);
28
29      // First three points
30      vector<point> S;
31      S.push_back(P[n-1]);
32      S.push_back(P[0]);
33      S.push_back(P[1]);
34      i = 2;
```

# Convex Hull Code, 2

```
35      while (i < n) {
36       // note: N must be >= 3 for this method to work
37          j = (int)S.size()-1;
38          if (ccw(S[j-1], S[j], P[i]))
39            S.push_back(P[i++]); // left turn, accept
40          else S.pop_back();
41      }
42      return S;
43  }
```