THINK FAST

Competitive Programming

From Problem 2 Solution in O(1)

Computational Geometry

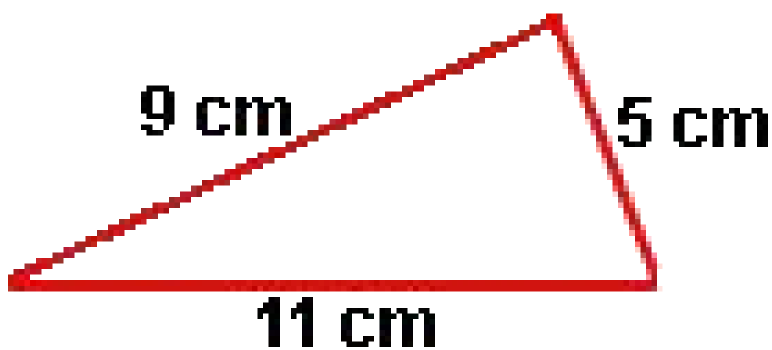
Polygon Area - Centroid - Cut

Mostafa Saad Ibrahim

PhD Student @ Simon Fraser University

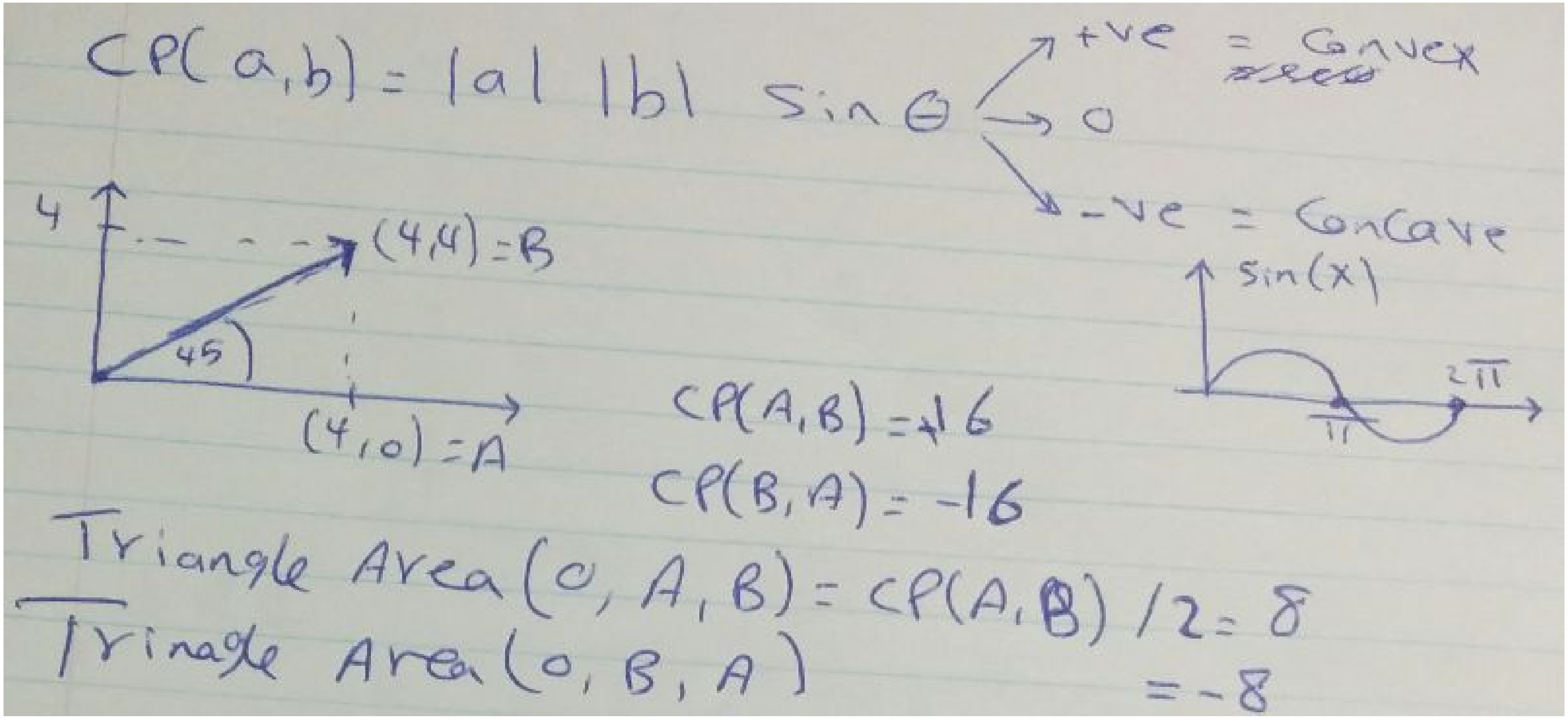
Polygon Perimeter

- It is the sum of the lengths of its sides.
  - Example below = 9 + 5 + 11 = 25 cm

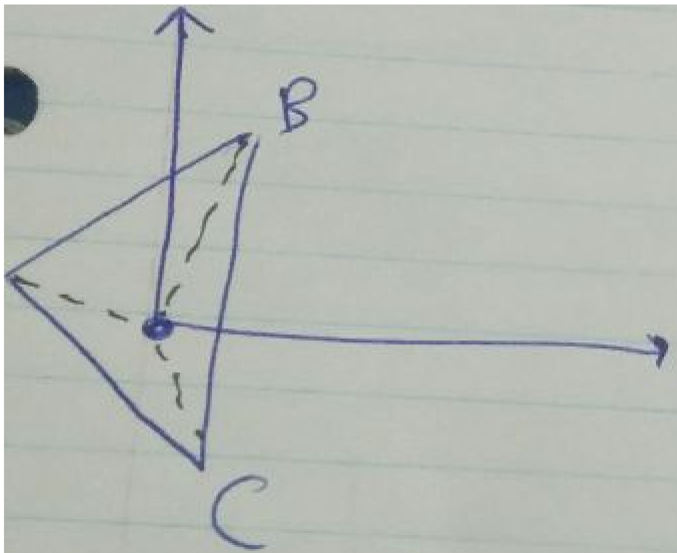


Src: http://www.mathgoodies.com/lessons/vol1/perimeter.html

Recall Cross Product / Triangle



Triangle Area centered at origin



Compute area(A,B,C)

Triangulate to 3 triangles

OAB, OBC, OCA

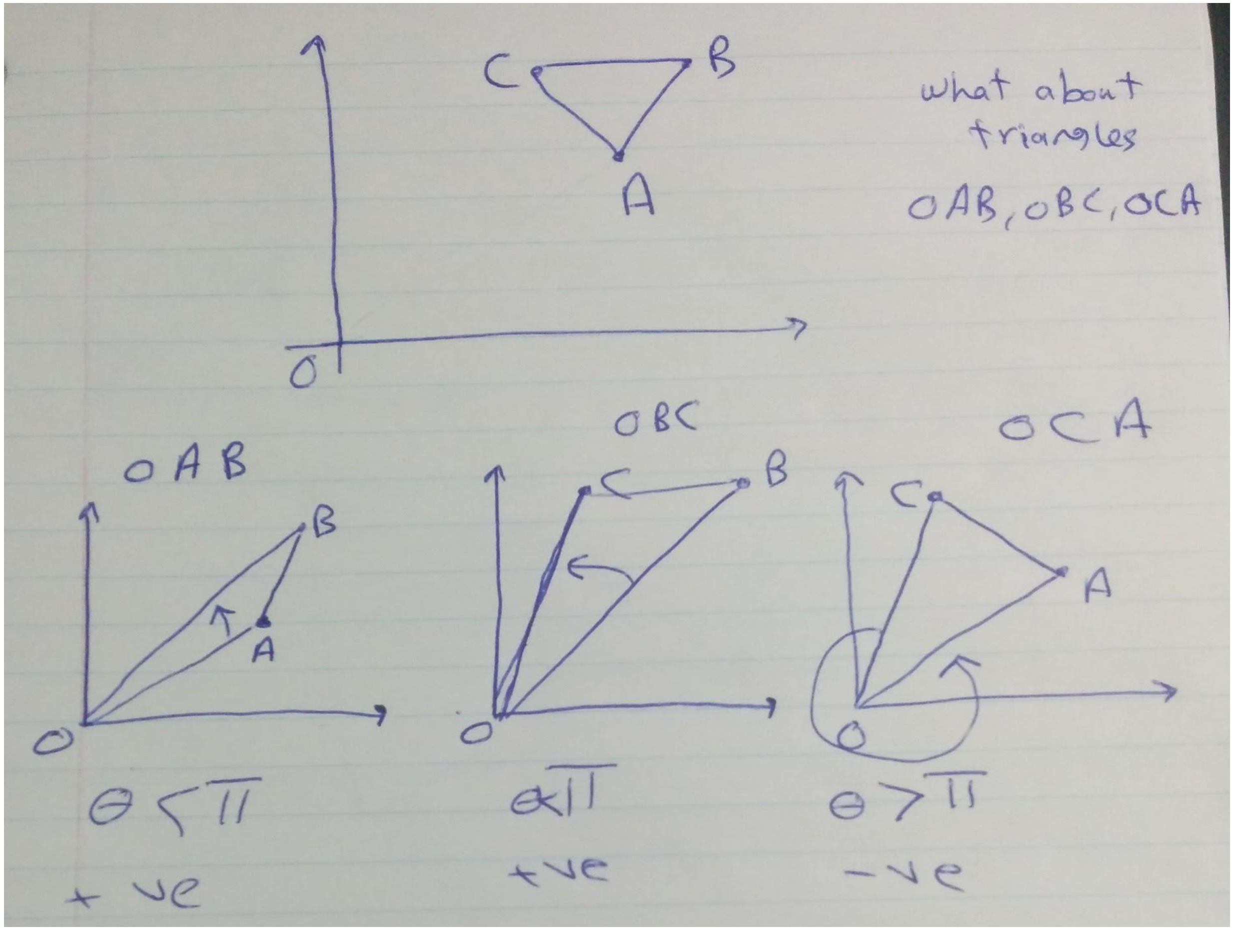
Clearly sum = total

Area = [CP(A,B) + CP(B,C) + CP(C,A)] / 2

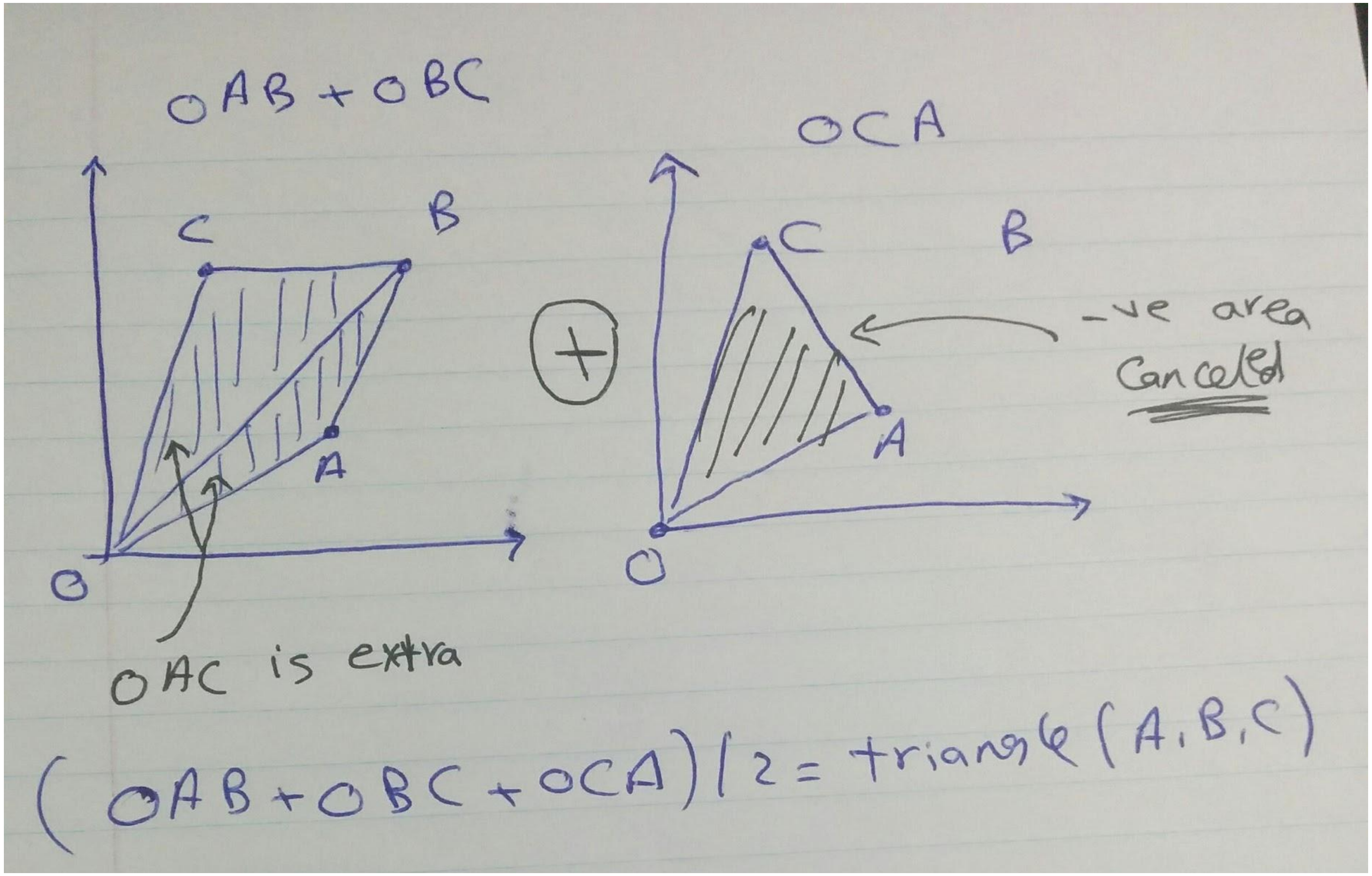
+ve

-ve

Triangle Area far from origin



Triangle Area far from origin



General Polygon Area

- Same handling for same reason
  - For every edge, add its cross product
  - Areas will +ve and -ve (cancel addition)
  - This works for simple, non simple, convex and octave
  - This works for tricky inputs such as duplicate points or collinear ones
- Notes
  - If final area summation > 0, then points are ordered ccw
  - If all coordinates are integers, then area either X or X.5 where X is integer
  - Polygon points must follow some order (ccw / cw)

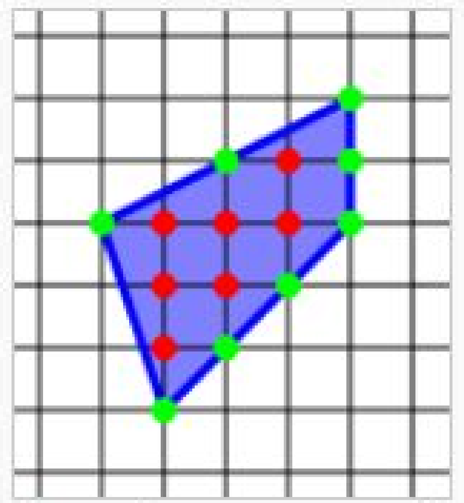
General Polygon Area

```
double polygonArea(vector<point>& points)
{
    double a = 0;
    rep(i, points)
      a += cp(points[i], points[ (i+1) % sz(points)]);
    return fabs(a/2.0);    // If a > 0 then points ordered ccw
}
```

- One can also use the first point in the polygon as reference point
- Then do N-1 cross products instead of N

Pick's theorem and Polygon Area

- For a simple polygon of **integer** coordinates.
  - Area(P) = internal_points_cnt + (boundry_points/2) - 1
- # boundary points of a vector can be computed by gcd(x, y)
- If we have Area, boundary of triangle, we can compute its interior points easily
  I = (2*A - b + 2)/2

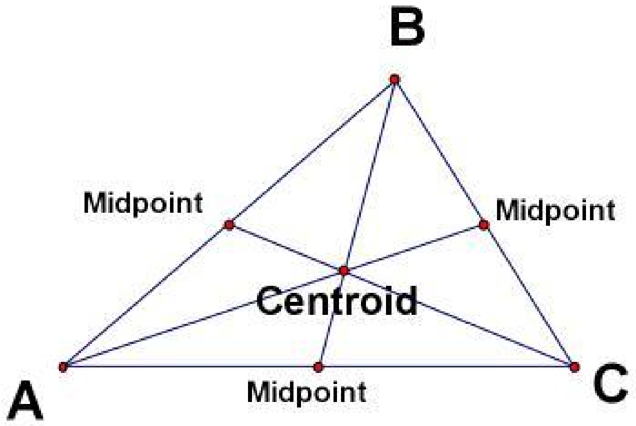


i = 7, b = 8,
A = i + b/2 − 1 = 10

Src: https://en.wikipedia.org/wiki/Pick%27s_theorem

Shape Centroid

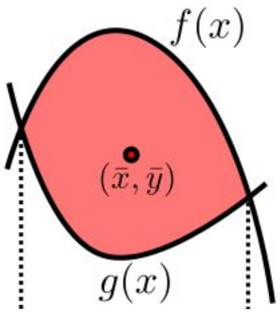
- Centroid point informally, under some conditions, a point where the shape is **balanced** (think in putting it over ***tip of pin***)
- It is the **average** x and y coordinate for **all the points** in the shape
  - NOT average of vertices
- Center of mass/gravity

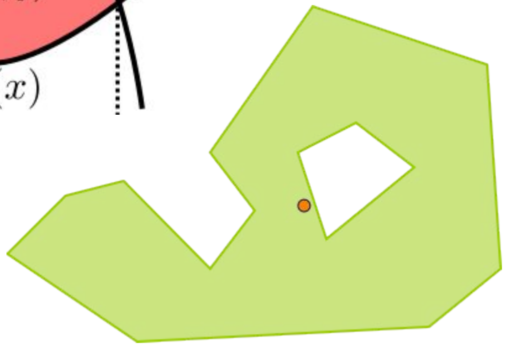


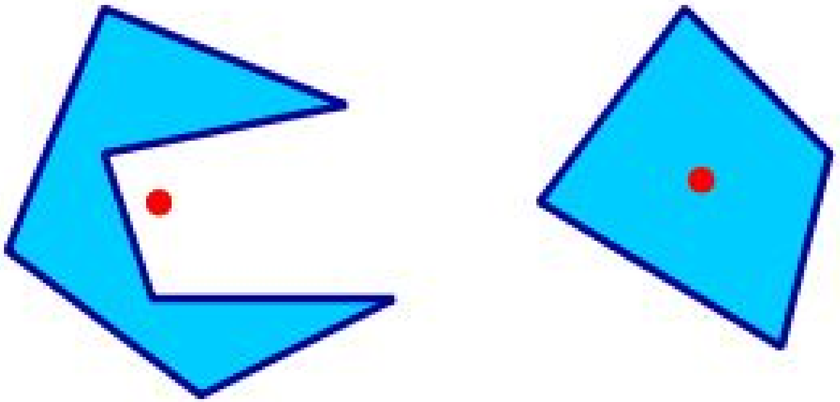
Src: http://www.mathwords.com/c/c_assets/centroid.jpg

Shape Centroid

- Centroid of some shapes (e.g. concave polygons, such as C shapes) may be **outside** it



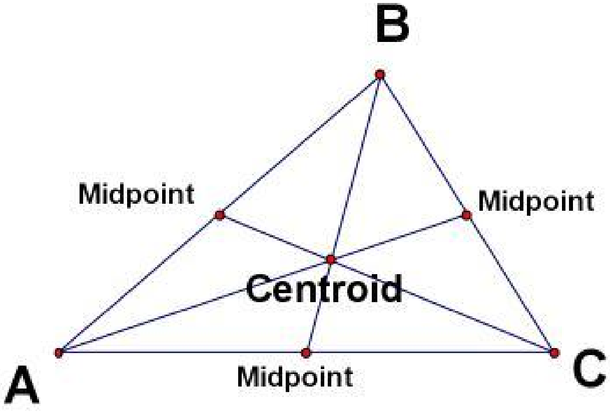




Src: http://suite.opengeo.org/docs/latest/processing/processes/vector/centroid.html http://www.boost.org/doc/libs/1_61_0/libs/geometry/doc/html/img/algorithms/centroid.png

Triangle Centroid

- The intersection of the three medians of the triangle
- Also, the average of the 3 vertices (will be used for polygon centroid)



Src: http://www.mathwords.com/c/c_assets/centroid.jpg

Locating the centroid

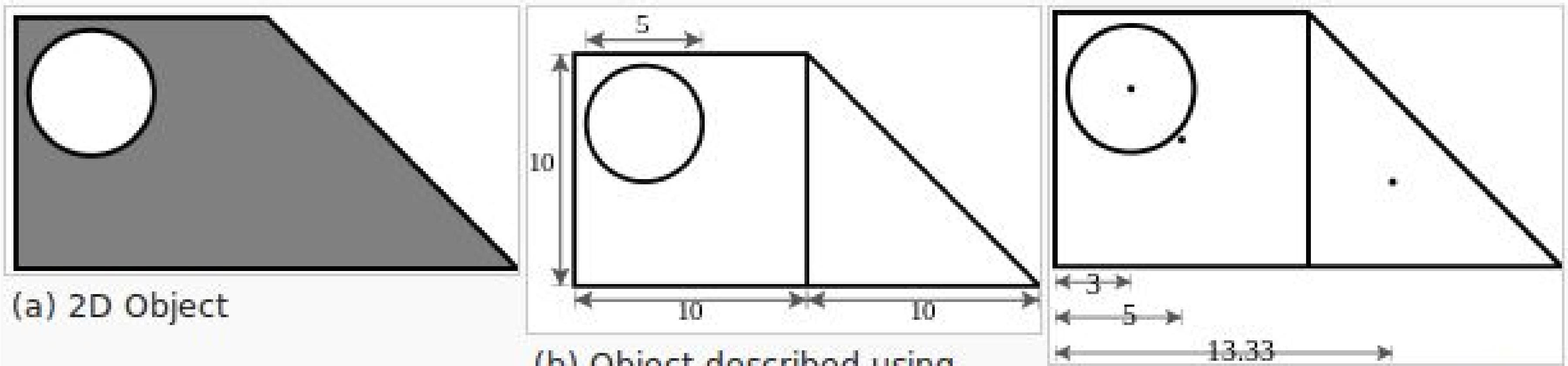
- Theoretically, **double integrate** and sum over all shape points and divide by area
- Some shapes has easy centroids
  - Triangle, Square, Circle

$$\overline{x} = \frac{\iint_R x\,dx\,dy}{\iint_R dx\,dy}$$

- By geometric decomposition
  - Divide shape to figures that you know their Centroid and Aares. Then do **weighted average**. Areas can be -ve

$$C_x = \frac{\sum C_{i_x} A_i}{\sum A_i}, C_y = \frac{\sum C_{i_y} A_i}{\sum A_i}$$

By geometric decomposition



(a) 2D Object

(b) Object described using simpler elements

(c) Centroids of elements of the object

$$x = \frac{5 \times 10^2 + 13.33 \times \frac{1}{2}10^2 - 3 \times \pi 2.5^2}{10^2 + \frac{1}{2}10^2 - \pi 2.5^2} \approx 8.5 \text{ units.}$$

$$C_x = \frac{\sum C_{i_x} A_i}{\sum A_i}$$

Src: https://en.wikipedia.org/wiki/Centroidv

Polygon Centroid

- Let’s **decompose** it to set of triangles (origin, side 2 points) such as in polygon area
  - But this contains extra areas?
  - Use signed area to cancel extra areas
- Triangle area = cross product / 2
- Triangle centroid = sum points / 3
- Let Polygon total area A
- CenterX = Sum 1/6A *
- (Sum X’s) * Cross Product

Polygon Centroid

$$C_x = \frac{1}{6A} \sum_{i=0}^{n-1} (x_i + x_{i+1})(x_i\ y_{i+1} - x_{i+1}\ y_i)$$

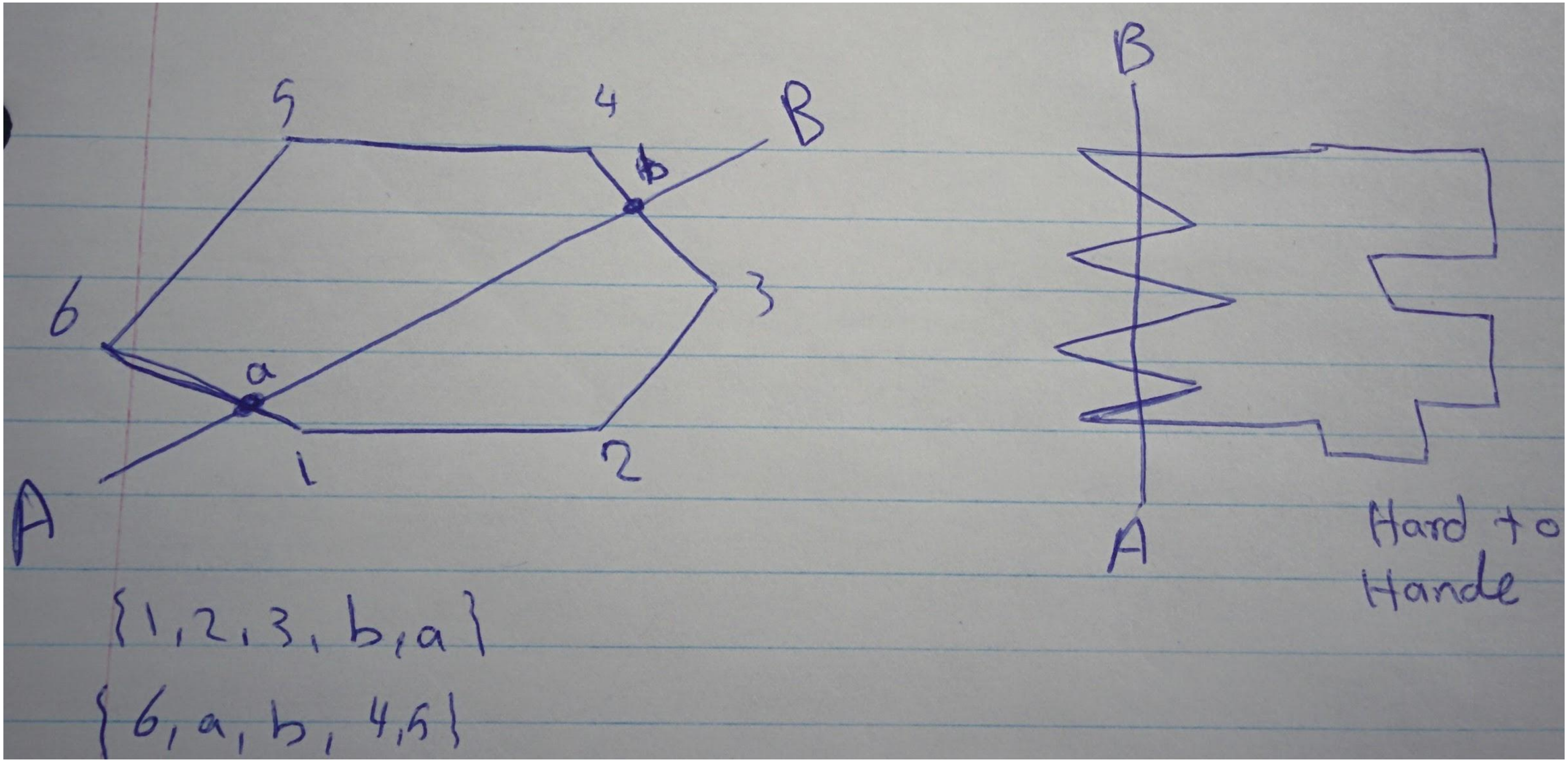
$$C_y = \frac{1}{6A} \sum_{i=0}^{n-1} (y_i + y_{i+1})(x_i\ y_{i+1} - x_{i+1}\ y_i)$$

and where A is the polygon's signed area,

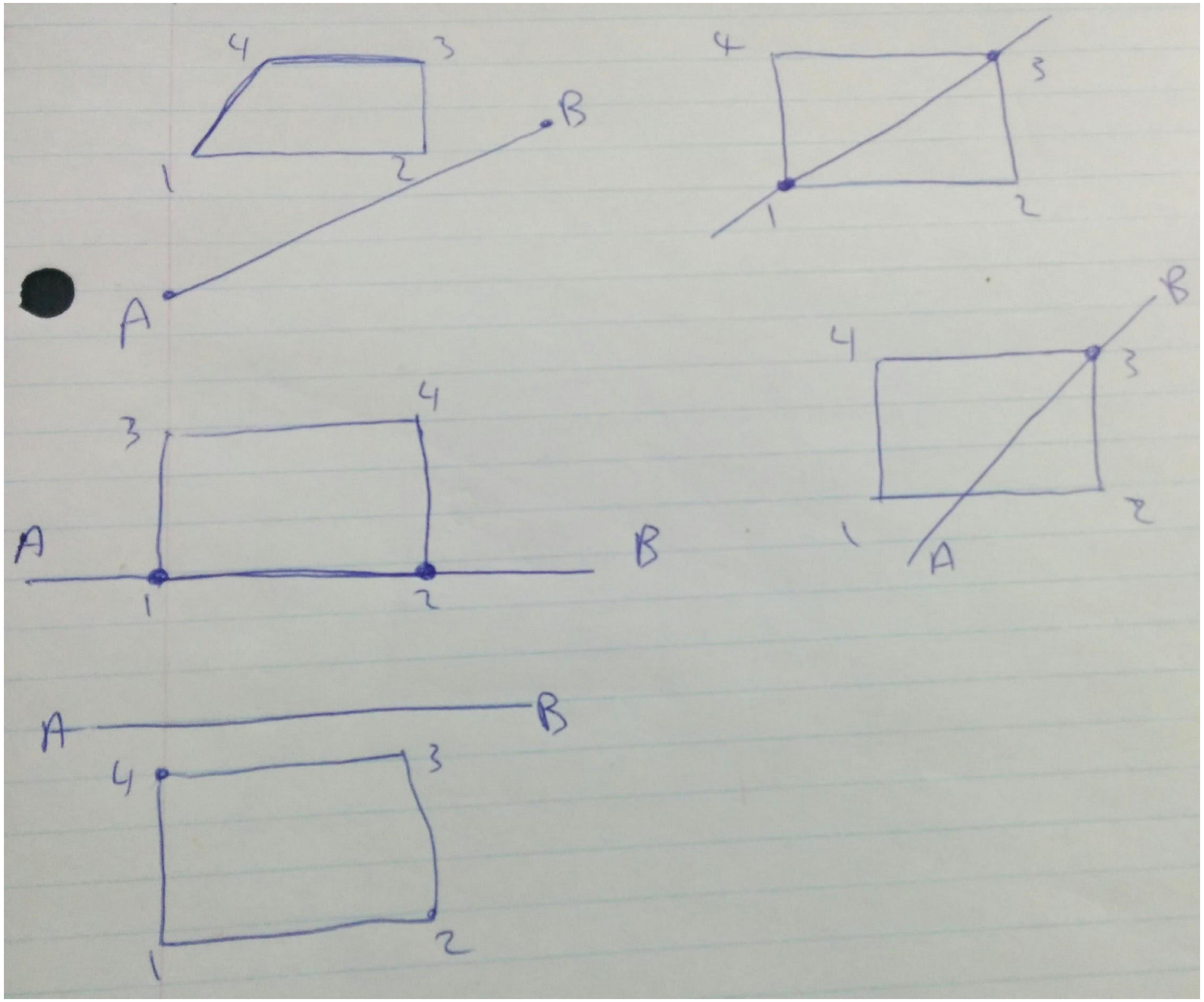
Polygon Centroid

```
point polygonCenteriod(vector<point> points) {
  double x = 0, y = 0, a = 0, c;

  for(int i = 0; i < points.size(): ++i)
  {
    int j = (i + 1) % sz(points);
    c = cp(points[i], points[j]), a += c;
    x += (points[i].X + points[j].X) * c;
    y += (points[i].Y + points[j].Y) * c;
  }
  if (dcmp(a, 0) == 0)
    return (points[0] + points.back()) * 0.5;   // Line
  a /= 2, x /= 6 * a, y /= 6 * a;

  // Fix values in case
  if (dcmp(x, 0) == 0)  x = 0;
  if (dcmp(y, 0) == 0)  y = 0;

  return point(x, y);
}
```

Polygon Cut with Line

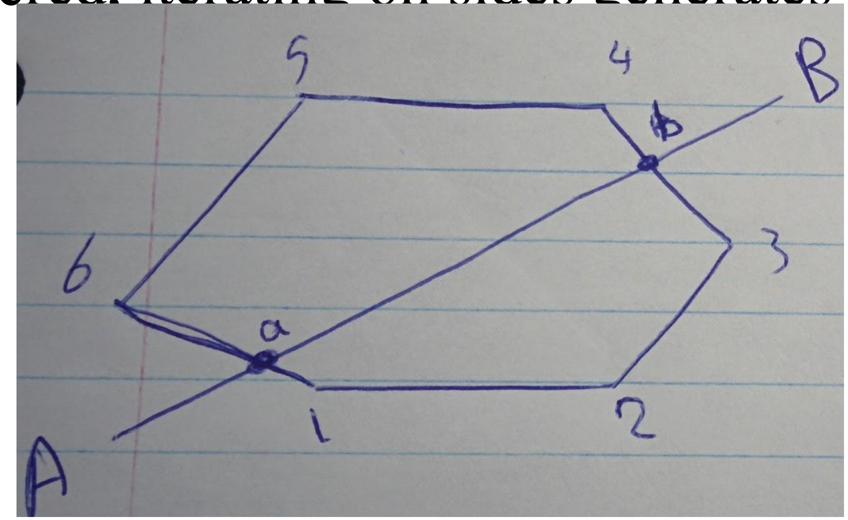


Convex Polygon Cut



Convex Polygon Cut

- Notes
  - Is point above/below line is trivial using cross product
  - Let's allow resulted polygons to have duplicate points
  - If side intersects with line, then this point in 2 polygons
  - Given that polygon is ordered, iterating on sides generates correct sub-polygons



Convex Polygon Cut

```
// P need to be counterclockwise convex polygon
pair<vector<point>, vector<point> > polygonCut(vector<point> &p,
    point A, point B) {

  vector<point> left, right;
  point intersect;

  for (int i = 0; i < sz(p); ++i) {
    point cur = p[i], nxt = p[(i + 1) % sz(p)];

    if ( cp(B-A, cur-A) >= 0)
      right.push_back(cur);

    //NOTE adust intersectSegments should handled AB as line
    if (intersectSegments(A, B, cur, nxt, intersect)) {
      right.push_back(intersect);
      left.push_back(intersect);
    }

    if ( cp(B-A, cur-A) <= 0)
      left.push_back(cur);
  }
  return make_pair(left, right);
}
```

تم بحمد الله

علمكم الله ما ينفعكم

ونفعكم بما تعلمتم

وزادكم علماً