ITMO University
Peking University
Training Camp

ITMO大学北京大学训练营

Day 02: Problem Analysis

Maxim Buzdalov

ITMO University,
St. Petersburg, Russia

25.02.2014

Problem Origin

- Problems from my Computational Geometry course
  - some weeks for usual students
- Each dedicated to either:
  - a standard subroutine used on bigger algorithms
  - a single (but probably nontrivial) algorithm idea
- The implementation may be tricky
  - learn how to consider corner cases in CG

Problem A. Segment Intersection

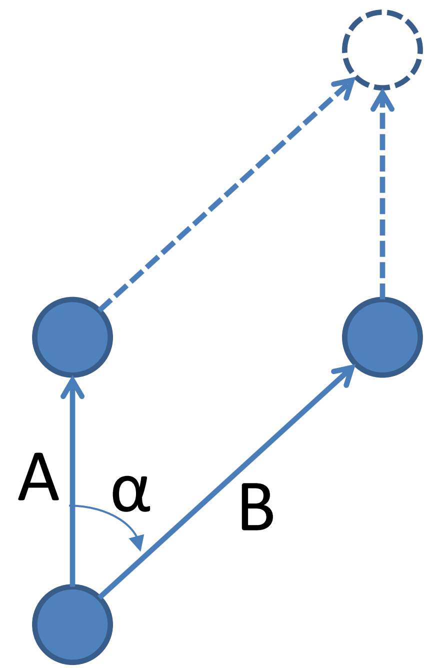
- For two segments, find how many points they have in common
  - Do not need to report these points
  - Coordinates are rather big
- Use precise algorithm implemented in 64-bit integer arithmetic

Case Analysis

- The first segment may be the point
  - Test for a point on a segment
- The second segment may be the point
  - The same
- The segments may be separated
  - Vertical or horizontal lines, answer = 0
- The segments may be collinear
  - One common point or many common points
- The general case

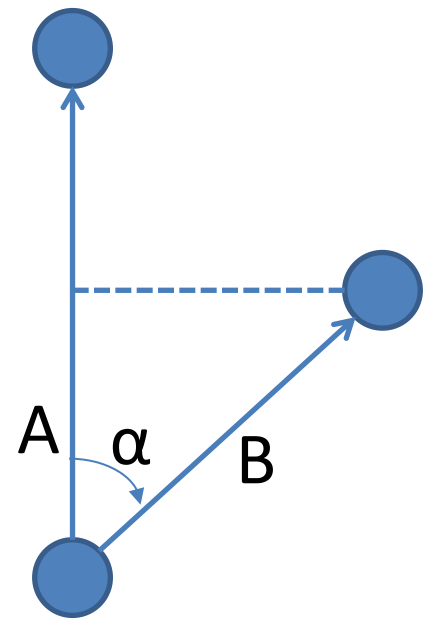
Remember the basics...

- Cross product (in 2D)
  - $A \times B = |A| \cdot |B| \cdot \sin \alpha$
  - angle is directed from A to B
- The semantics
  - Oriented square of a parallelogram
  - Positive when rotation is counterclockwise
- Easy way to compute
  - $A \times B = A.x \cdot B.y - B.x \cdot A.y$



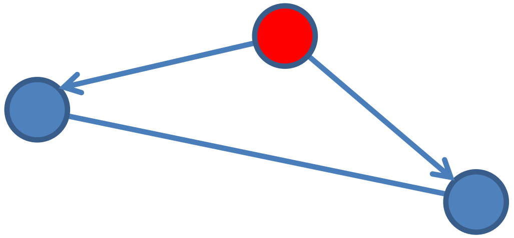
Remember the basics...

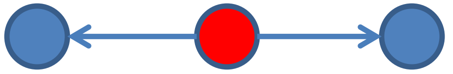
- Scalar product (in 2D)
  - $A \cdot B = |A| \cdot |B| \cdot \cos \alpha$
- The semantics
  - Length of A multiplied by projection of B to A
  - Positive when $-90^o < \alpha < 90^o$
- Easy way to compute
  - $A \cdot B = A.x \cdot B.x + A.y \cdot B.y$



Point on a segment

- Point: P
- Segment: AB
- Collinearity check
  - $(A - P) \times (B - P) = 0$
- Is inside the segment?
  - $(A - P) \cdot (B - P) \le 0$



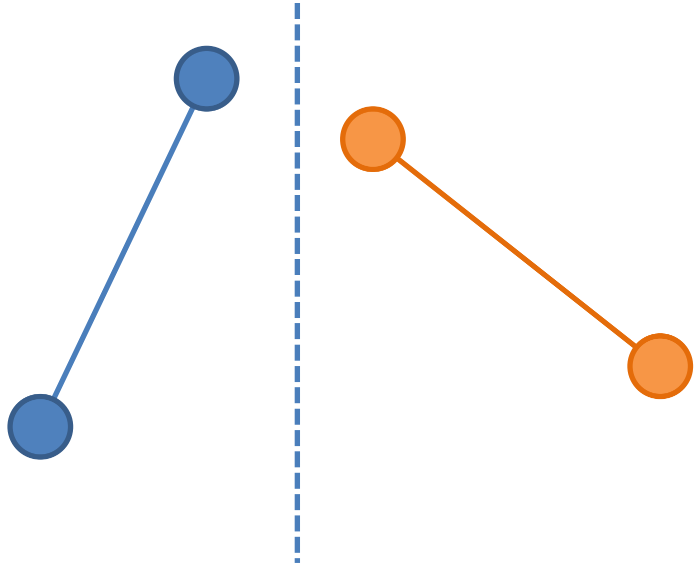


- "×" – the cross product
- "·" – the scalar product

Separation

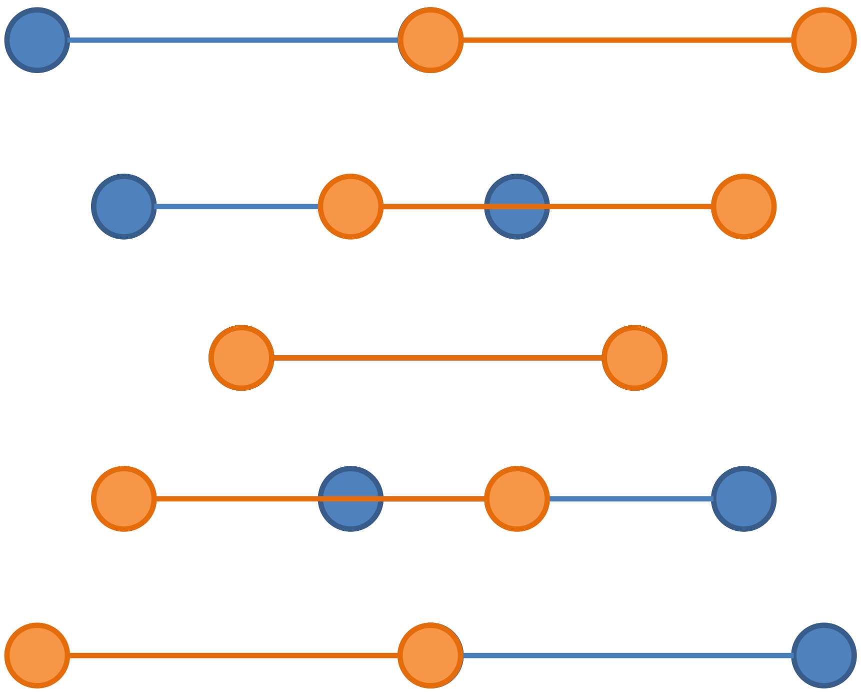
- max(A.x, B.x) < min(C.x, D.x)
- max(A.y, B.y) < min(C.y, D.y)
- max(C.x, D.x) < min(A.x, B.x)
- max(C.y, D.y) < min(A.y, B.y)

- Using these tests, you can reduce the number of bugs in the collinear case
- Especially if cases of one common point and many points are indistinguishable in your problem



Collinear case

- Segments intersect
  - one or many points?
- Point comparison
  - first compare by X
  - if equal, compare by Y
- One point case:
  - min(A, B) = max(C, D)
  - min(C, D) = max(A, B)

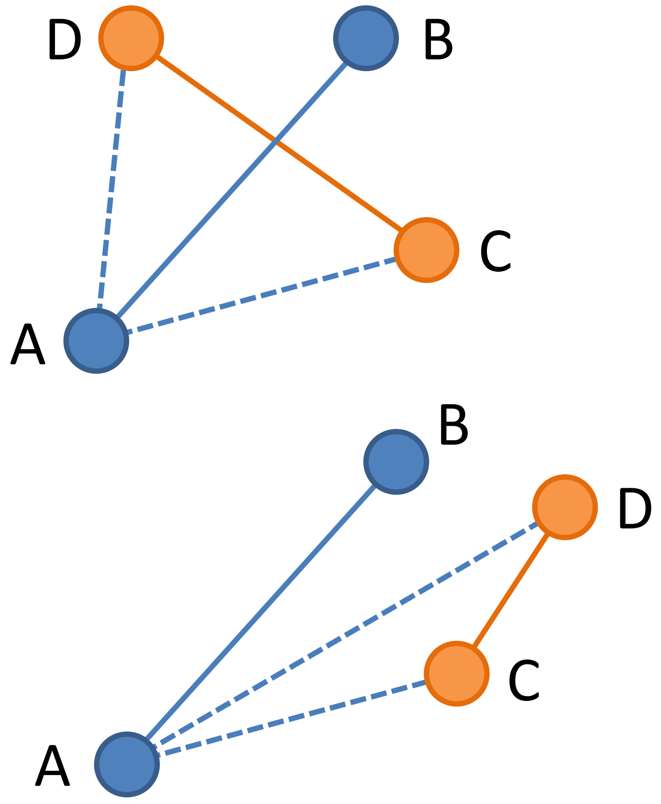


General case

- Consider AB
  - Are C and D to the same side of AB?
  - If yes, then segments do not intersect
- Consider CD
  - Test the same for A, B
- If both tests do not return “yes”, there is exactly one intersection point

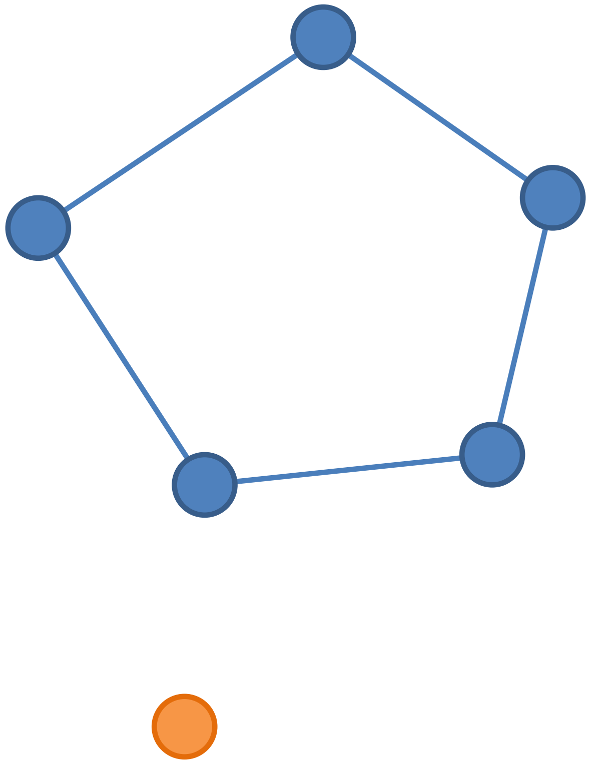
The "same side" test

- Use cross product:
  - $P_C = (B - A) \times (C - A)$
  - $P_D = (B - A) \times (D - A)$
- The same side:
  - $P_C > 0$ and $P_D > 0$
  - $P_C < 0$ and $P_D < 0$
- The final test...
  - $\text{sign}(P_C) \cdot \text{sign}(P_D) > 0$



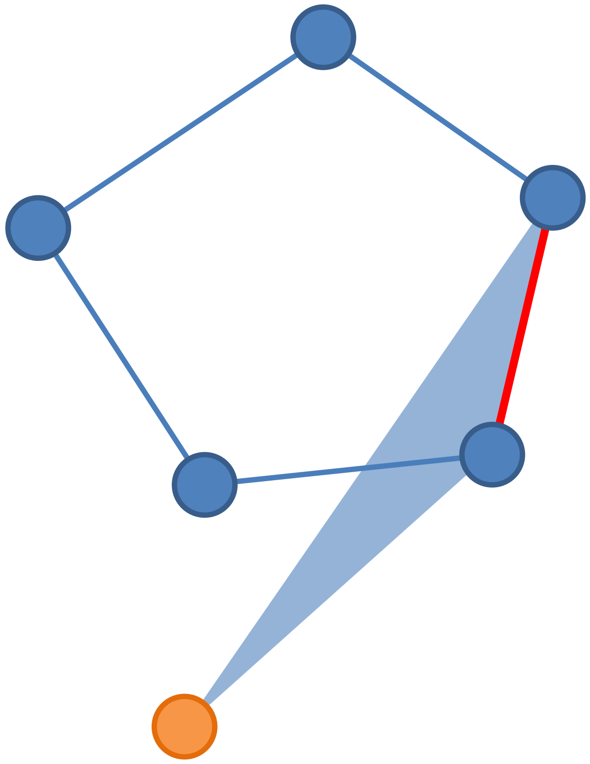
Problem B. Area of a Polygon

- Fix a point P
- For each edge AB:
  - Add (A – P) × (B – P) to the answer
- Take the absolute value of the answer and divide it by 2



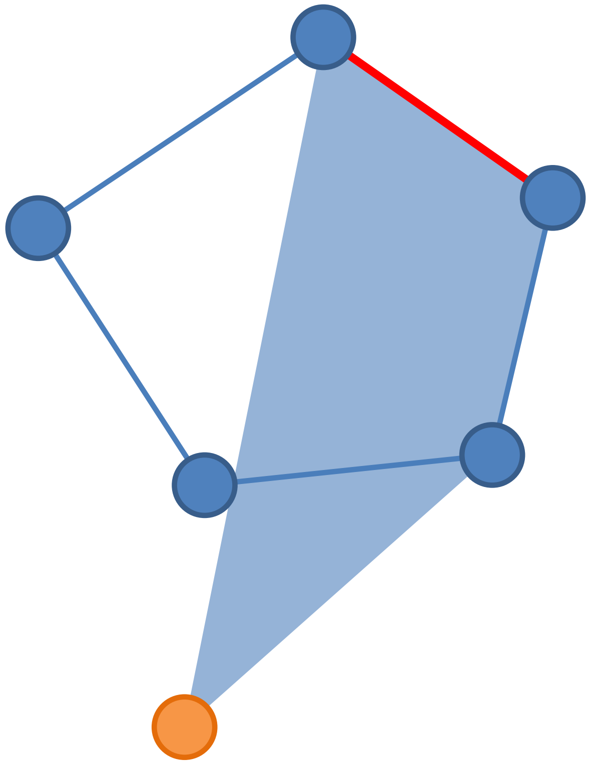
Problem B. Area of a Polygon

- Fix a point P
- For each edge AB:
  - Add (A – P) × (B – P) to the answer
- Take the absolute value of the answer and divide it by 2



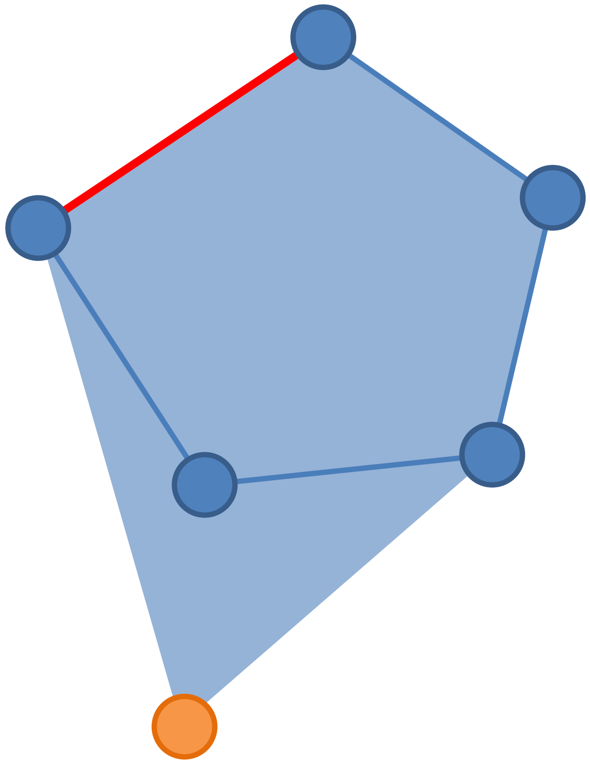
Problem B. Area of a Polygon

- Fix a point P
- For each edge AB:
  - Add $(A - P) \times (B - P)$ to the answer
- Take the absolute value of the answer and divide it by 2



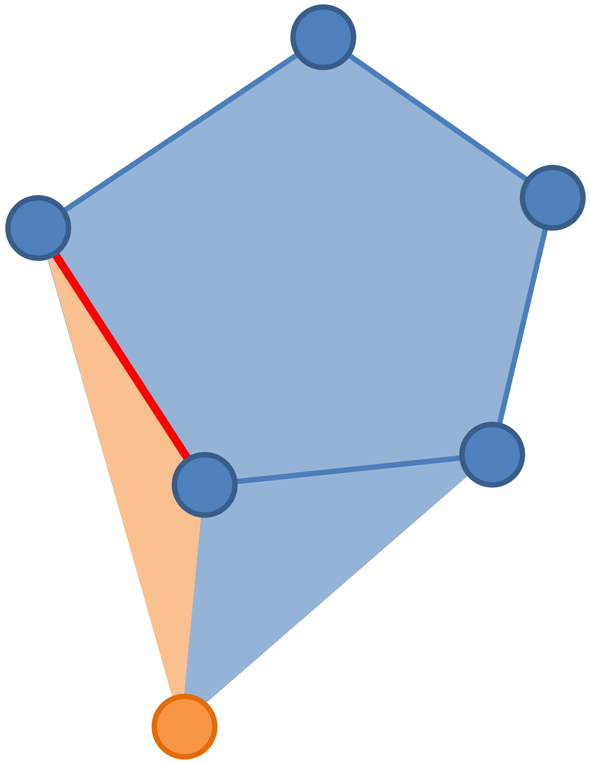
Problem B. Area of a Polygon

- Fix a point P
- For each edge AB:
  - Add $(A - P) \times (B - P)$ to the answer
- Take the absolute value of the answer and divide it by 2



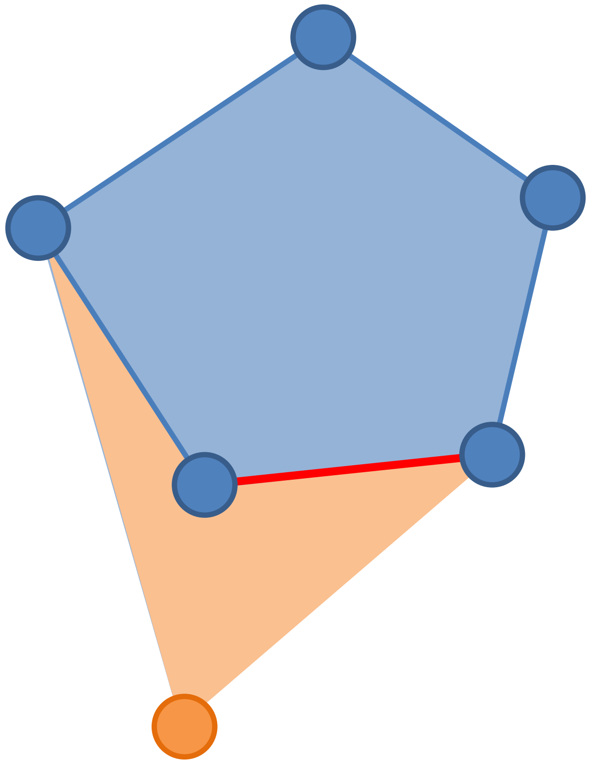
Problem B. Area of a Polygon

- Fix a point P
- For each edge AB:
  - Add $(A - P) \times (B - P)$ to the answer
- Take the absolute value of the answer and divide it by 2



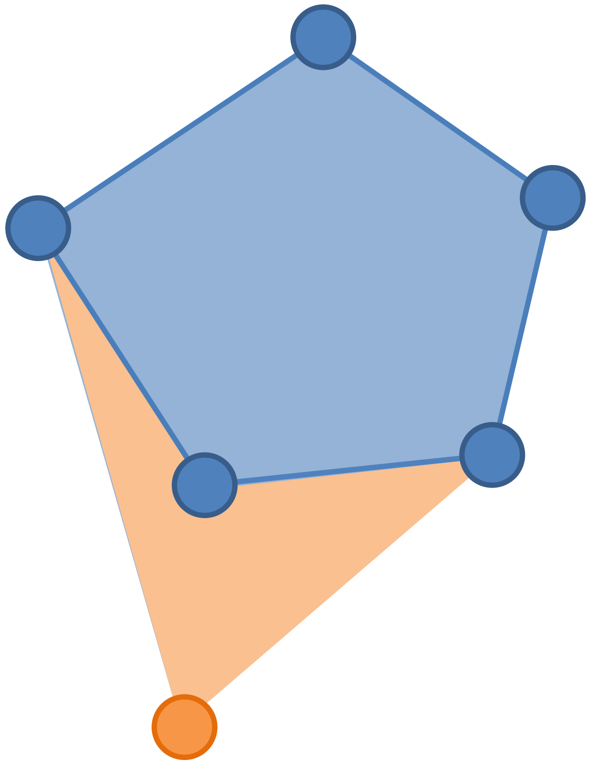
Problem B. Area of a Polygon

- Fix a point P
- For each edge AB:
  - Add $(A - P) \times (B - P)$ to the answer
- Take the absolute value of the answer and divide it by 2



Problem B. Area of a Polygon

- Take the absolute value of the answer and divide it by 2
- If the answer is computed or printed in **double/long double**, precision is not enough
- Compute it in 64-bit int
- Output:
  - answer / 2
  - dot
  - (answer mod 2) · 5



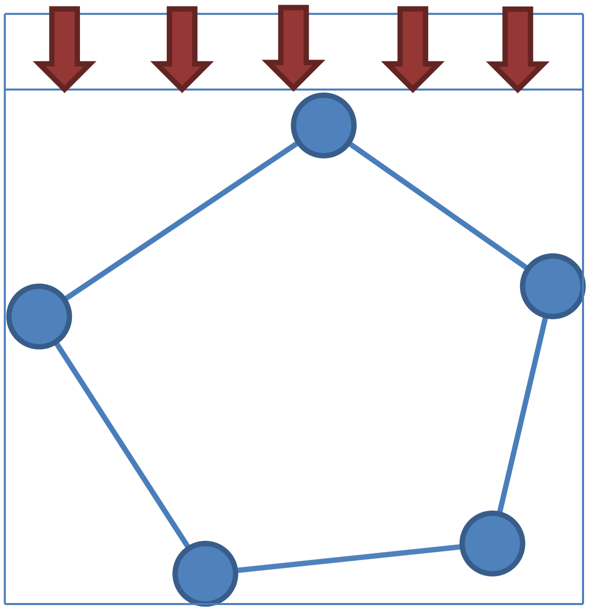
Problem C. Convex Hull

- Any O(N log N) algorithm should work here
  - the time limit is tight however
- Graham algorithm was used by the jury
  - it is a well known algorithm

- Tricky case: all points are equal
  - the answer consists of one point
- All other cases: use the main algorithm

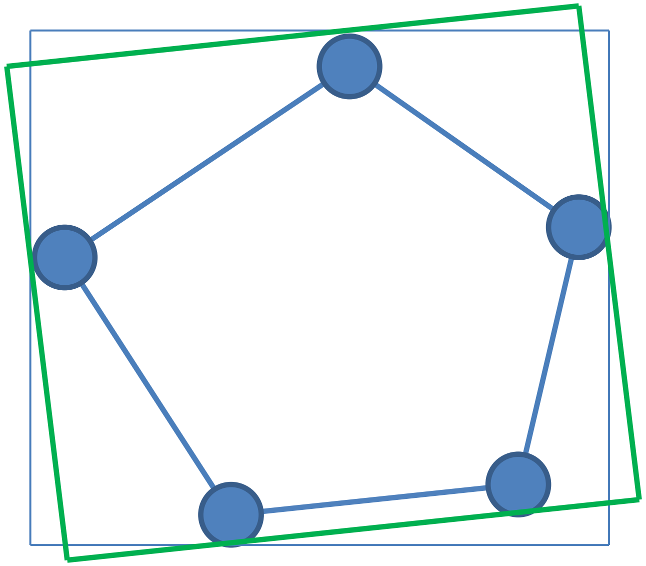
Problem D. Minimum Bounding Box

- Idea 1: A bounding box contains at least one polygon point on each edge
  - otherwise, it can be shrunk, so it is not a BB



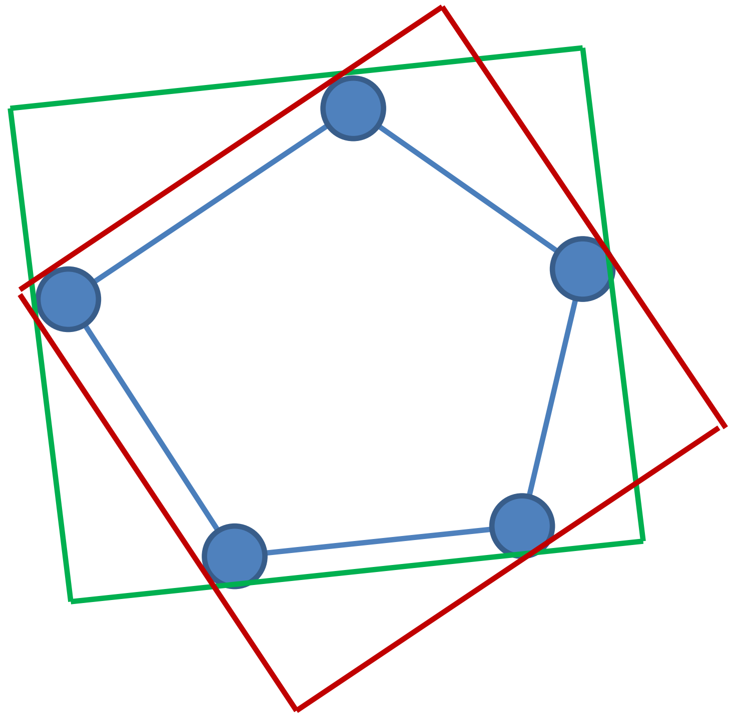
Problem D. Minimum Bounding Box

- Idea 2: A **minimum** bounding box contains at least one polygon **edge** on its edge
  - holds for both minimum area and minimum perimeter
- Proof: exercise



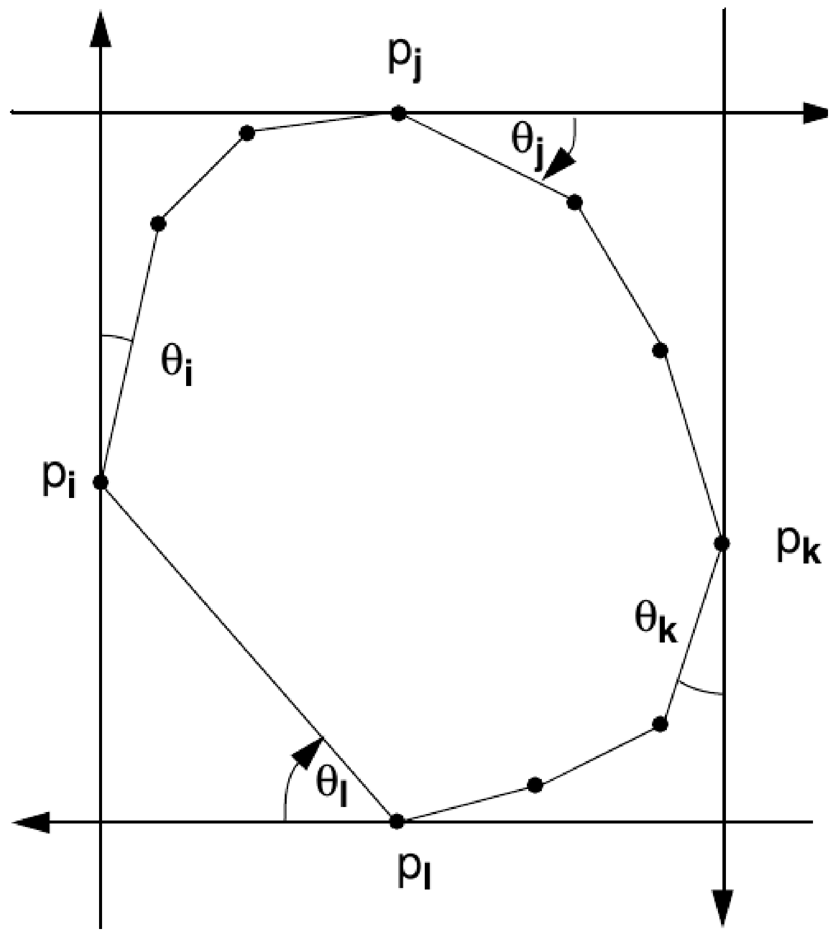
Problem D. Minimum Bounding Box

- Idea 3: Check all bounding boxes with at least one polygon edge on it and find the minimums
- But you should do that in linear time!



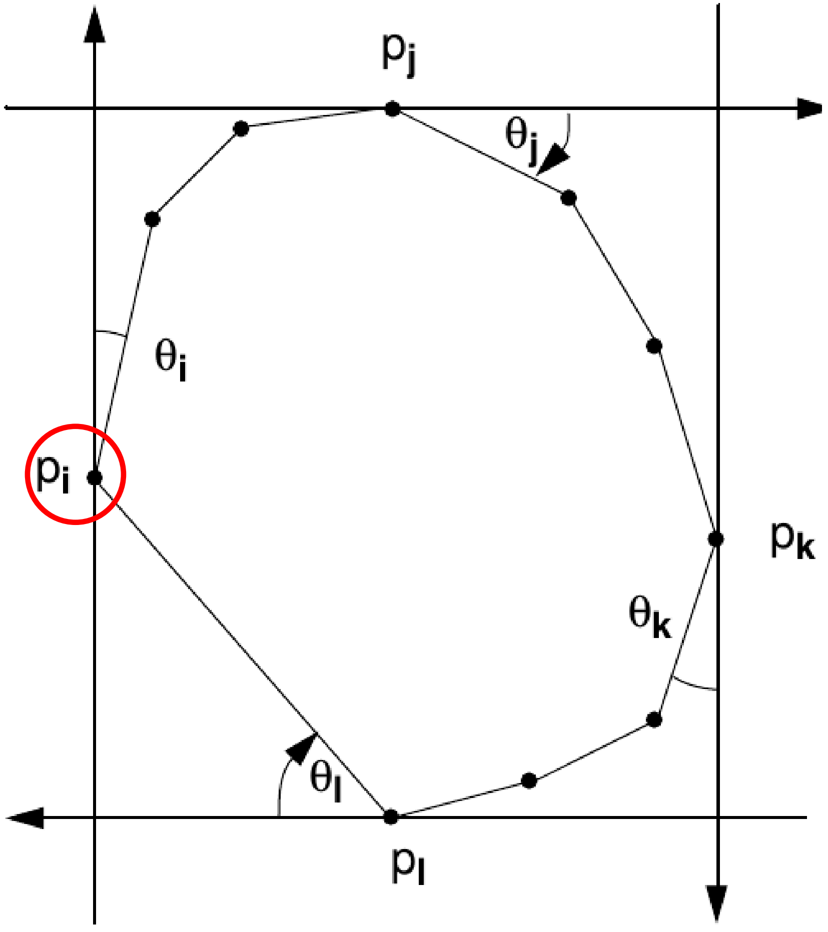
Rotating Calipers technique

- Four orthogonal lines
  - Each touches a vertex
- Rotation:
  - consider the minimum of angles between each line and the polygon
  - rotate the lines by that angle
  - advance the vertex
- There are O(N) possible line positions
  - so the traversal is O(N) too



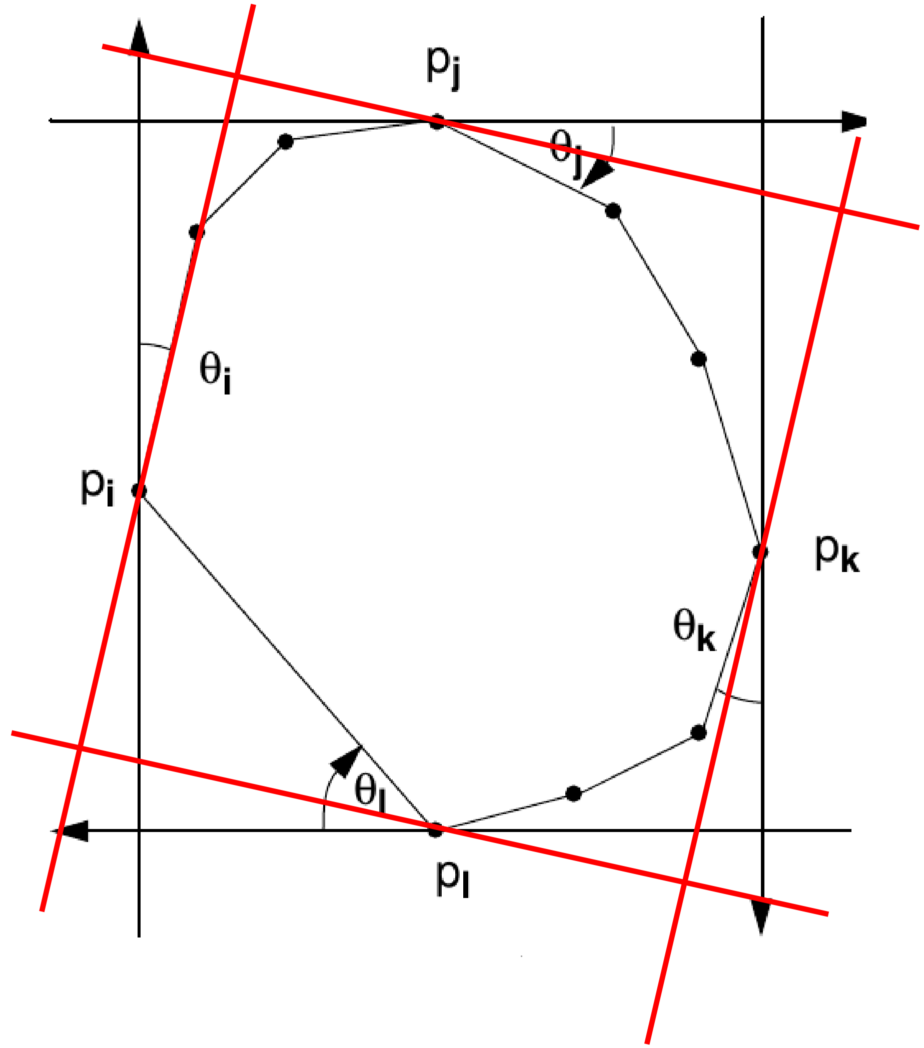
Rotating Calipers technique

- Four orthogonal lines
  - Each touches a vertex
- Rotation:
  - consider the minimum of angles between each line and the polygon
  - rotate the lines by that angle
  - advance the vertex
- There are O(N) possible line positions
  - so the traversal is O(N) too



Rotating Calipers technique

- Four orthogonal lines
  - Each touches a vertex
- Rotation:
  - consider the minimum of angles between each line and the polygon
  - rotate the lines by that angle
  - advance the vertex
- There are O(N) possible line positions
  - so the traversal is O(N) too



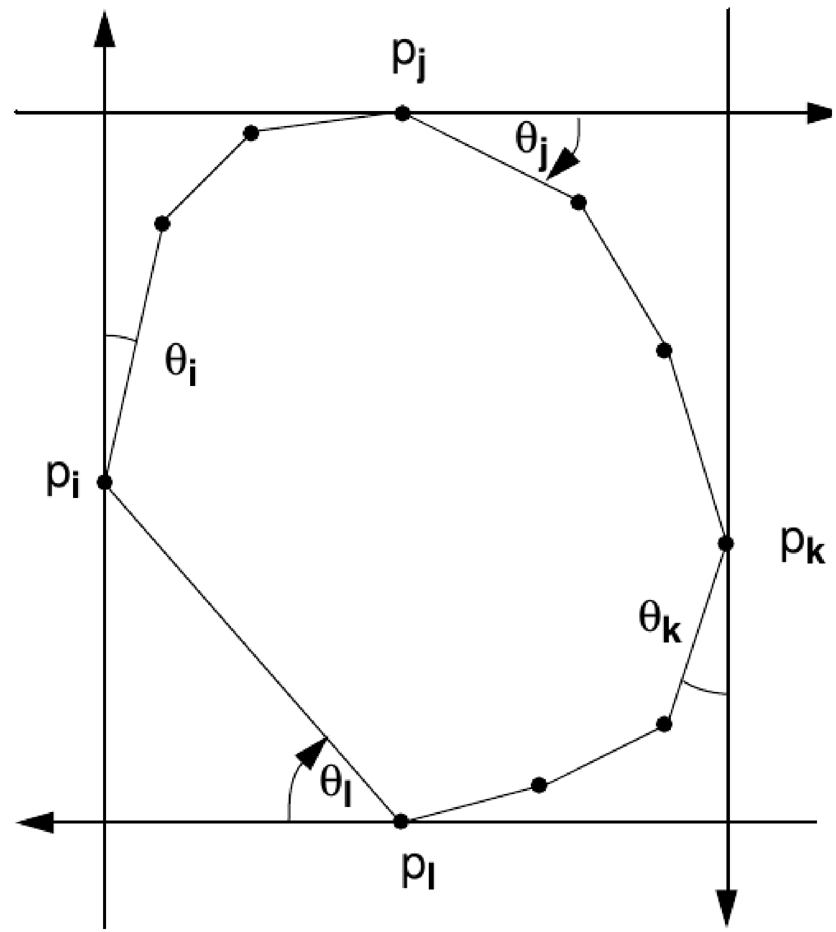
How to compare angles?

Case 1: Compare θ_i and θ_k

- Reflect the edge vector from p_k
- Draw it from p_i
- Compare using cross product

Case 2: Compare θ_i and θ_j

- Rotate the vector clockwise,
  then do the same



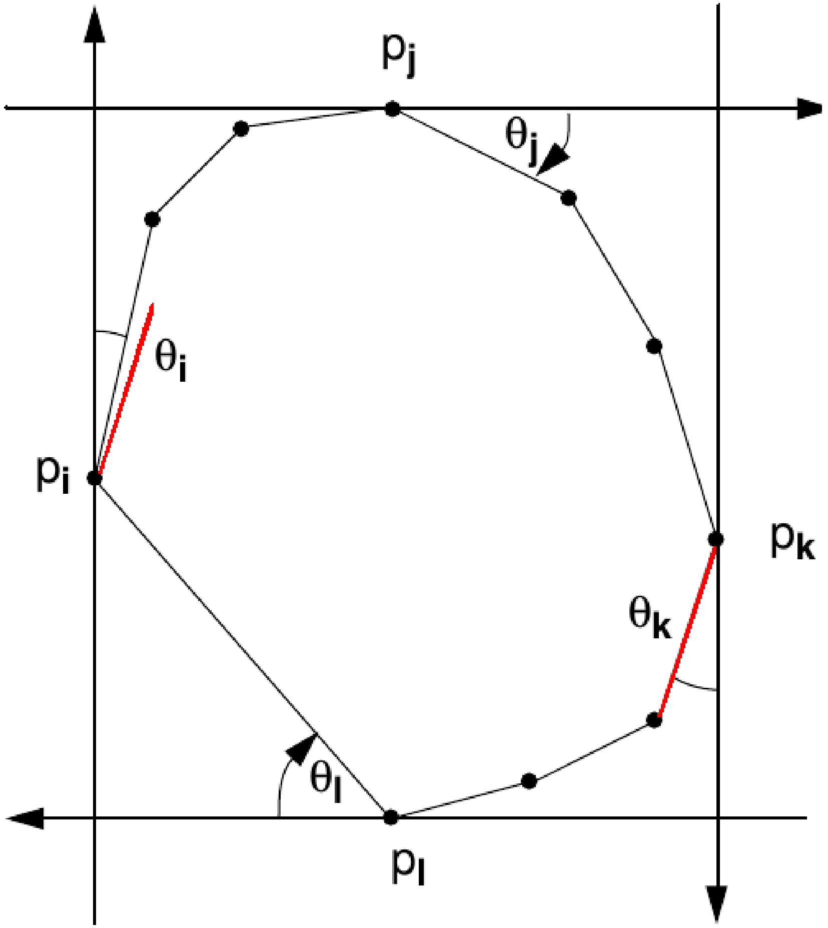
How to compare angles?

Case 1: Compare θ$_i$ and θ$_k$

- Reflect the edge vector from p_k
- Draw it from p_i
- Compare using cross product

Case 2: Compare θ$_i$ and θ$_j$

- Rotate the vector clockwise, then do the same



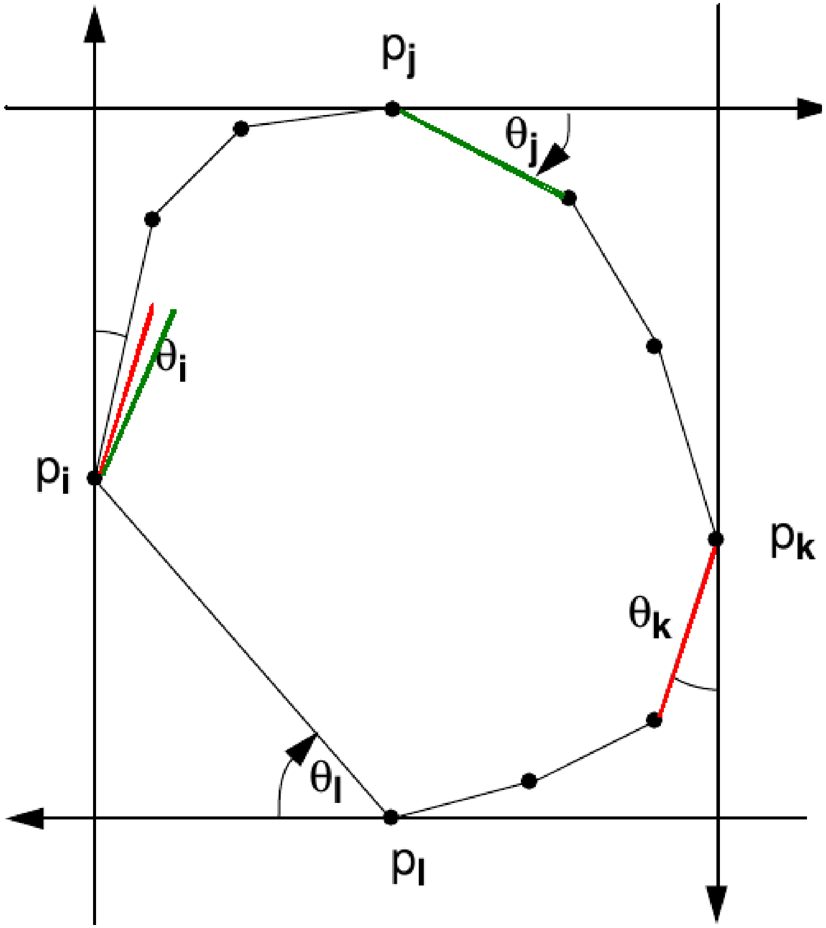
How to compare angles?

Case 1: Compare θ_i and θ_k

- Reflect the edge vector from p_k
- Draw it from p_i
- Compare using cross product

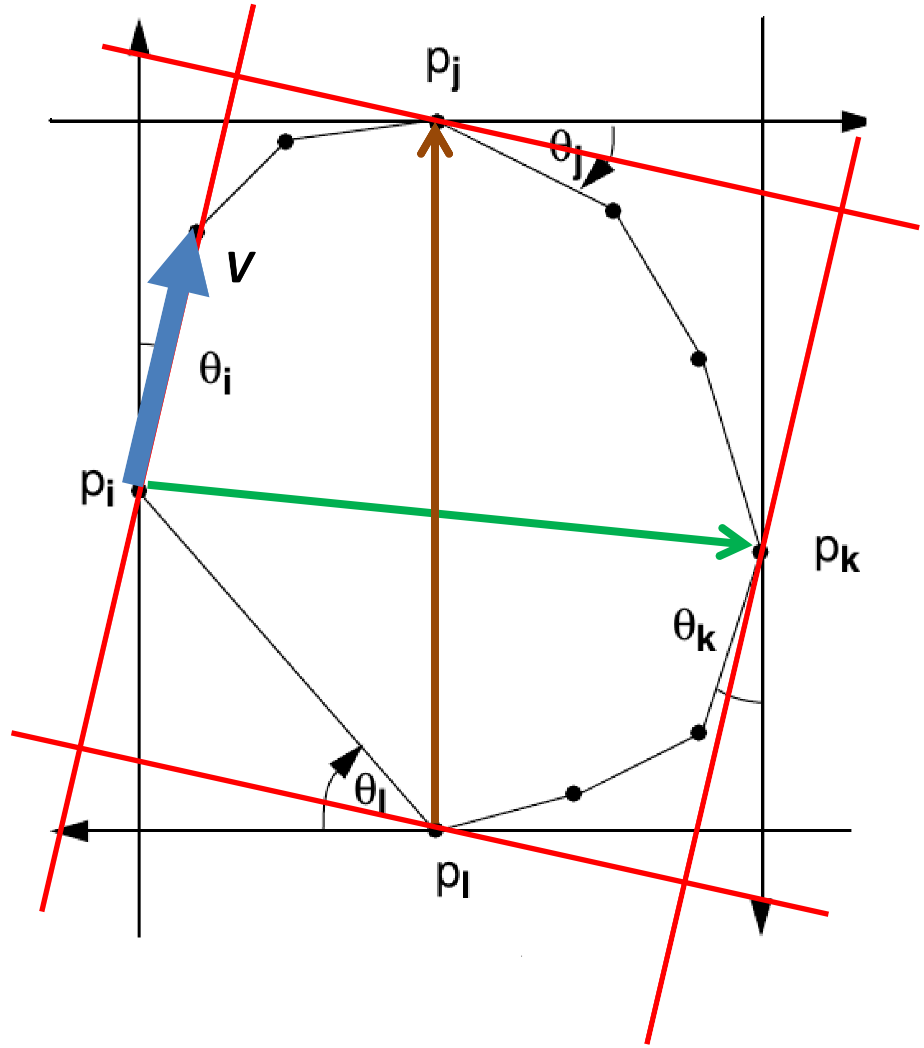
Case 2: Compare θ_i and θ_j

- Rotate the vector clockwise, then do the same



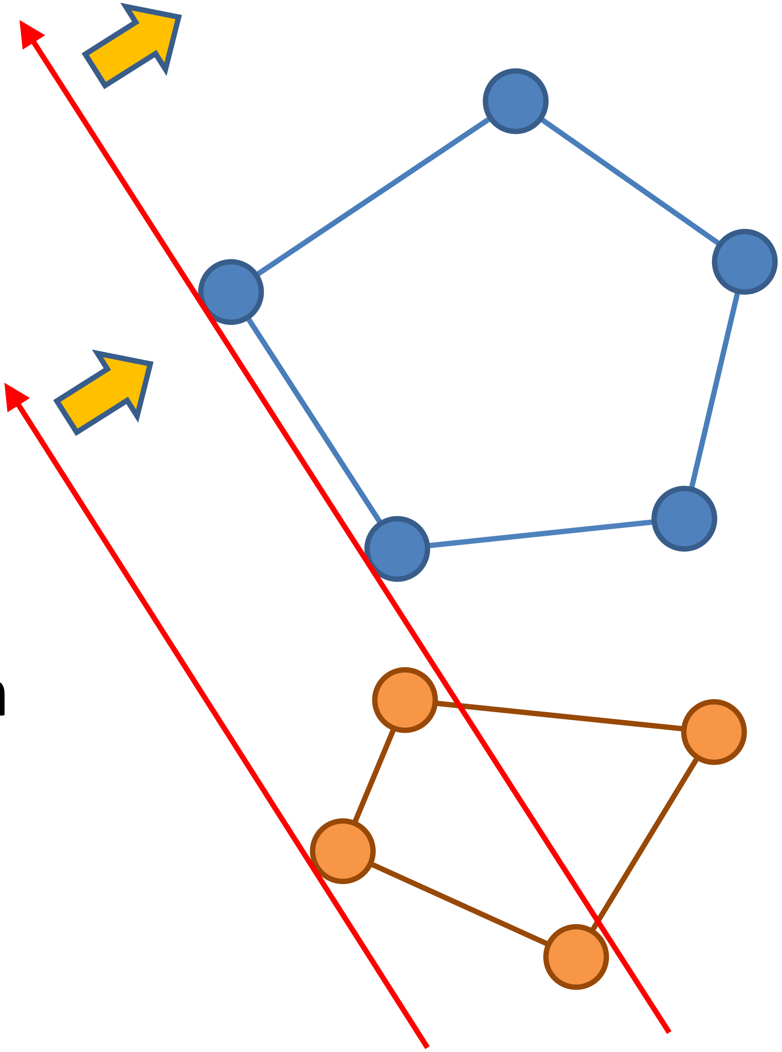
Finding the answers

- You know the points
- You know the vector *V*
- W: $\left|\frac{V\times(p_k-p_i)}{|V|}\right|$
- H: $\left|\frac{V\cdot(p_j-p_l)}{|V|}\right|$
- Area: W · H
- Perimeter: 2(W + H)



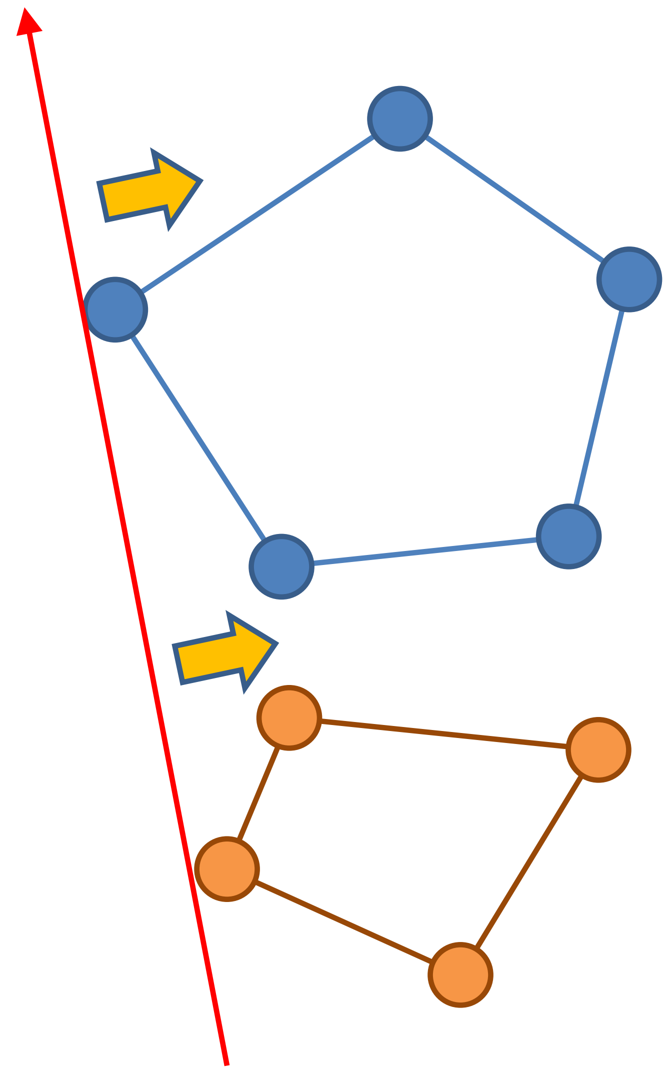
Problem E. Common Tangents 1

- This is one more problem on rotating calipers
  - but this time you do them on two polygons simultaneously
- A common tangent is a where calipers for both polygons coincide
  - maybe in motion



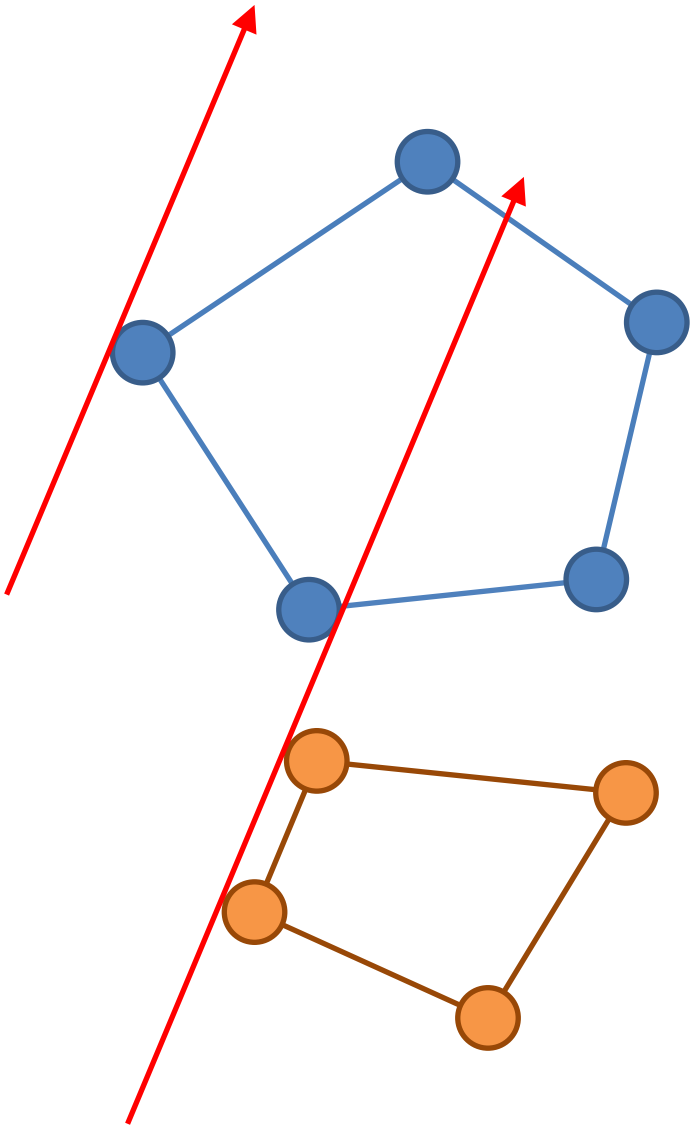
Problem E. Common Tangents 1

- This is one more problem on rotating calipers
  - but this time you do them on two polygons simultaneously
- A common tangent is a where calipers for both polygons coincide
  - maybe in motion



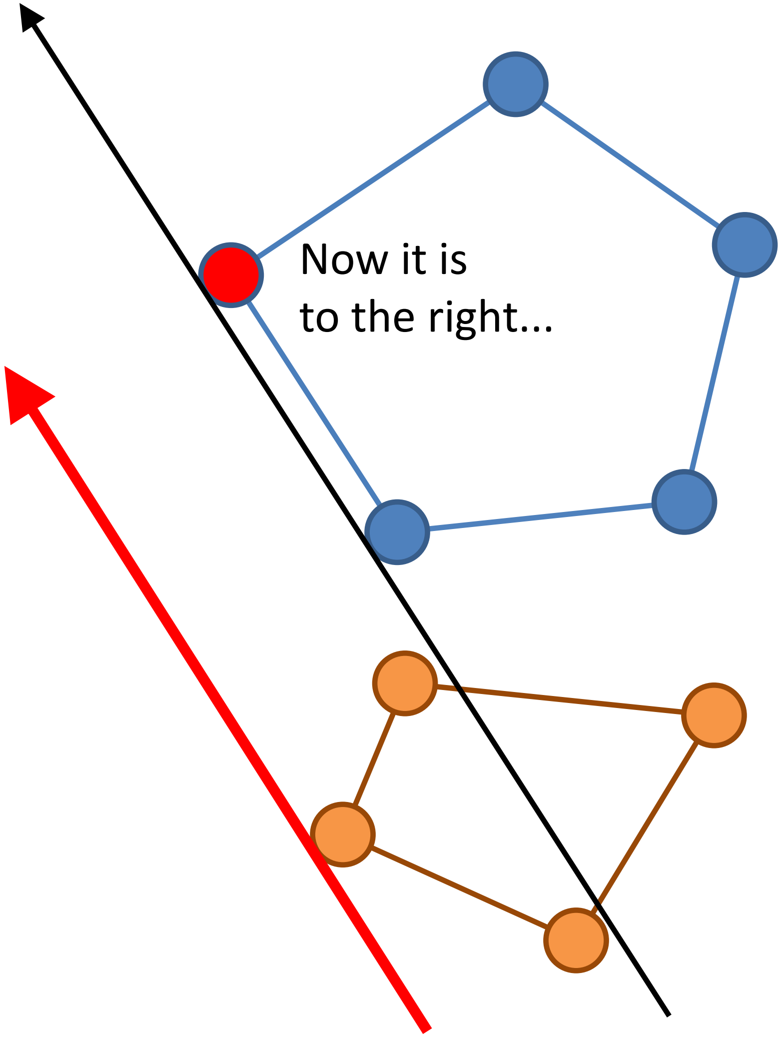
Problem E. Common Tangents 1

- This is one more problem on rotating calipers
  - but this time you do them on two polygons simultaneously
- A common tangent is a where calipers for both polygons coincide
  - maybe in motion



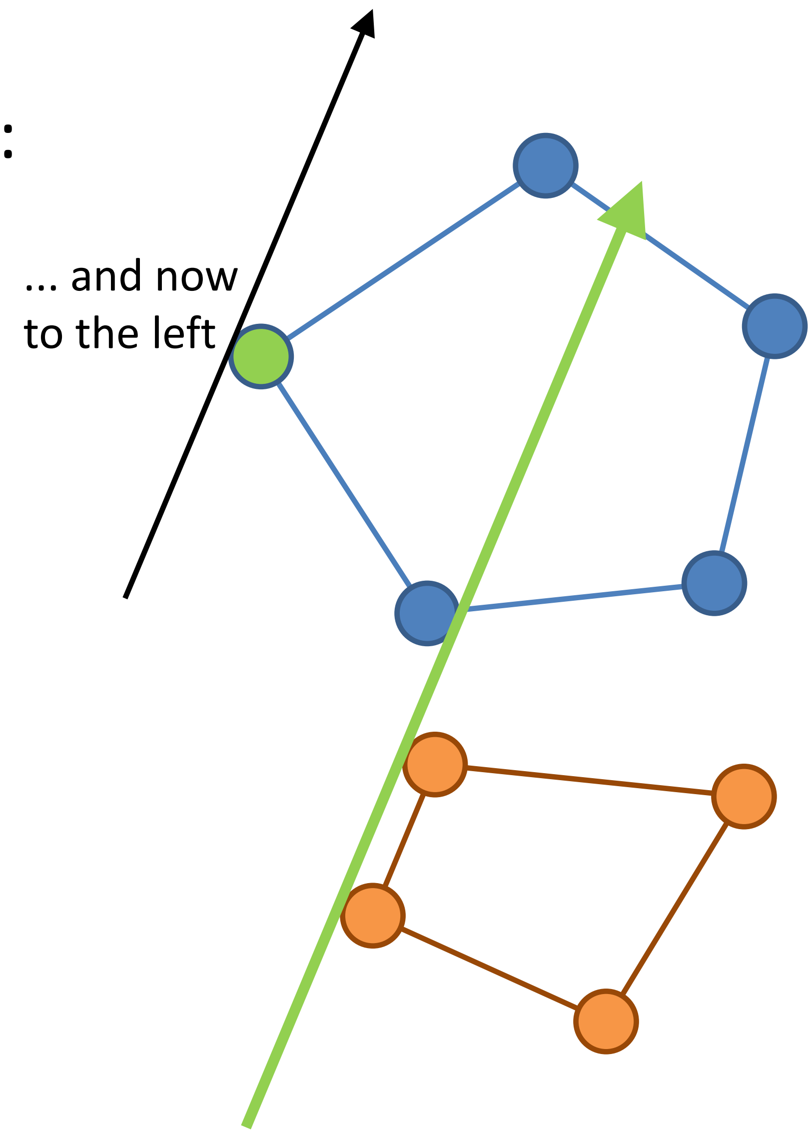
Discrete test for caliper coincidence

- Consider the caliper 1:
  - base point
  - initial vector
  - final vector
- Test the location of the base point of the caliper 2 to the caliper 1
  - changed its side – this is the tangent



Discrete test for caliper coincidence

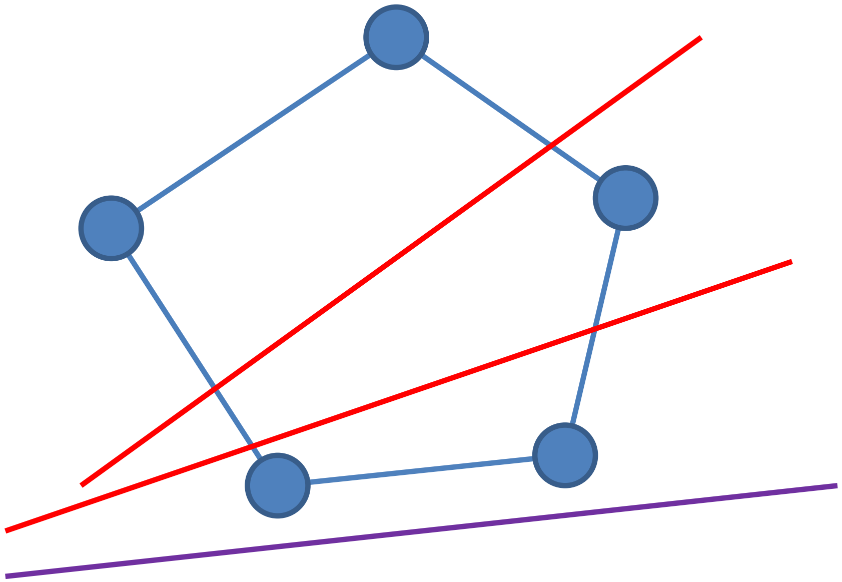
- Consider the caliper 1:
  - base point
  - initial vector
  - final vector
- Test the location of the base point of the caliper 2 to the caliper 1
  - changed its side – this is the tangent



Problem F. Polygon and Lines

Naive solution:

- for each line query
- for each polygon edge
- check if they intersect

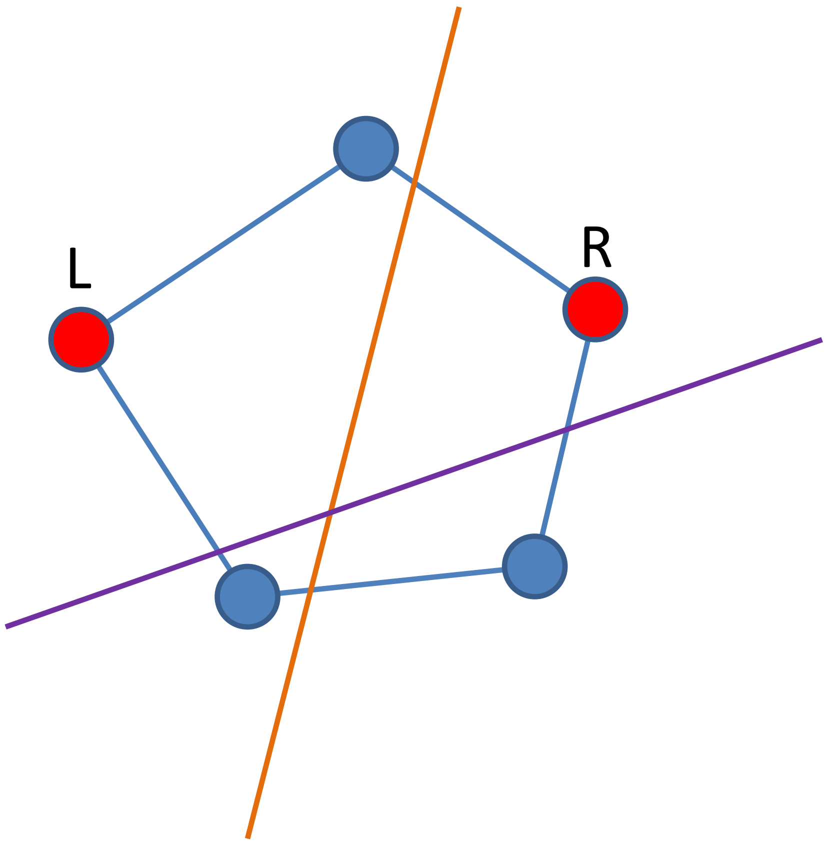


- Works in O($N \cdot Q$)
  - way too slow

Better solution

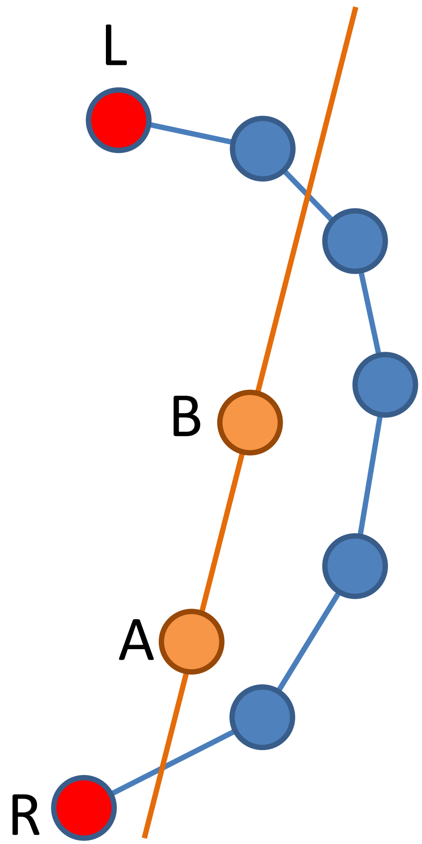
Ternary search
on convex polygon!

- Find leftmost and rightmost vertices of the polygon
- Given a line...
  - has L and R at different sides – must intersect
  - has L and R at the same side – ...



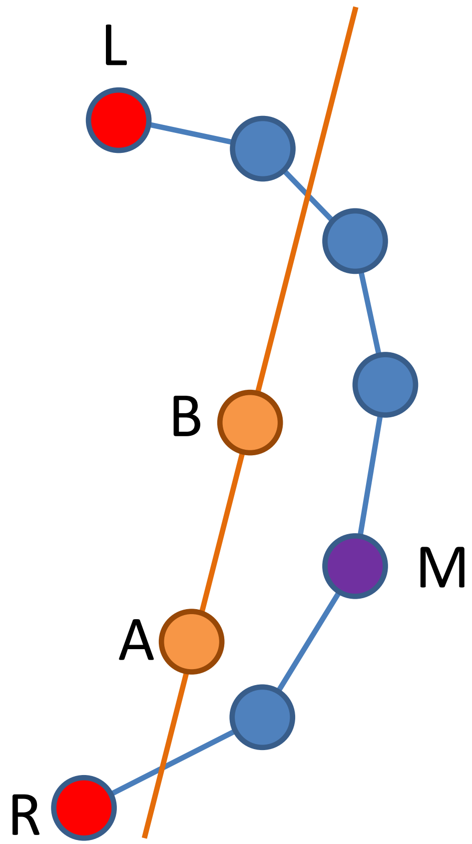
Ternary search

- Let A and B be two points on the line
- Polygon is convex – for all points between *L* and *R* in a certain chain $(p – A) \times (B – A)$ either decreases then increases or vice versa
  - starts with increase if $(L – A) \times (B – A) < 0$



Ternary search

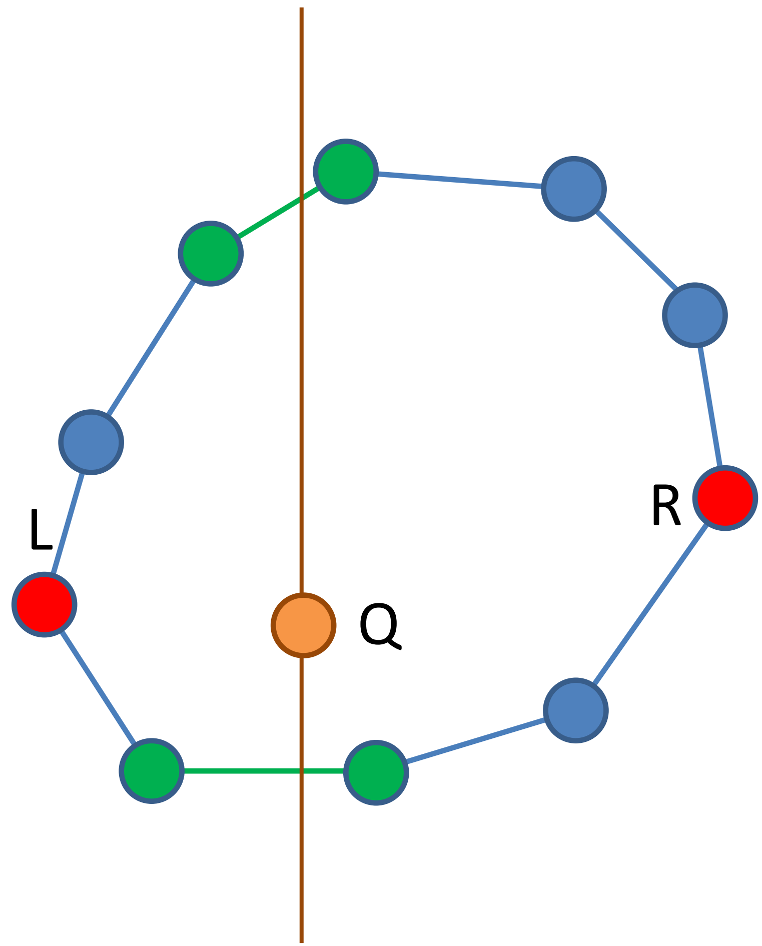
- Let $(L - A) \times (B - A) < 0$
- Find a point M such that $(M - A) \times (B - A)$ is maximum possible
  - using ternary search
  - in $O(\log N)$
- If M is at the other side than L, then the polygon is intersected



Problem G. Polygon and Points

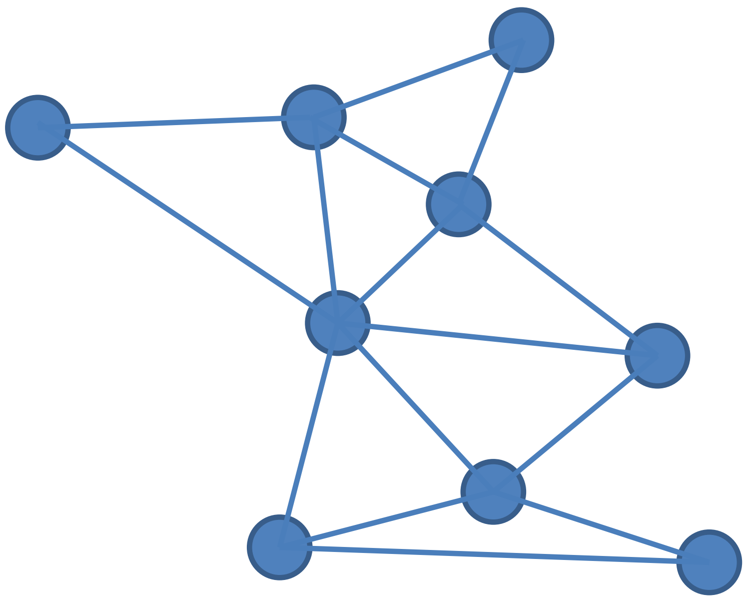
Binary search on convex polygon

- find the *L* and *R* points
- for each query point *Q*:
  - if to the left of L or to the right of R – then “0”
  - find a segment on which Q is projected to
    - in the upper chain
    - in the lower chain
    - using binary search
  - test where Q lies



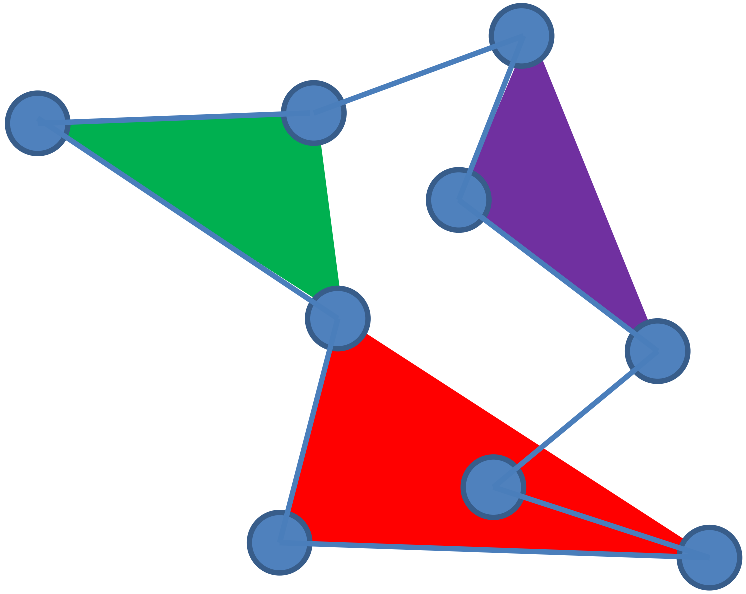
Problem H. Triangulation

- The limits allow $O(N^2)$ algorithms
  - although the triangulation of a polygon can be done in $O(N)$
- Consider “ear clipping” algorithm



Ear clipping algorithm

- An “ear” is a triangle formed by three consecutive vertices, with angle at the middle one less than 180º, which does not contain other points inside or on the boundary
- For a single triangle, ears {1,2,3}, {2,3,1}, {3,1,2} are the different ears
- An ear; not an ear; not an ear



Ear clipping algorithm

- Theorem: Every polygon has at least two ears
- Proof by induction.
- Induction base: N = 3
  - Three ears
- Induction step:
  - Take any vertex with an angle less than 180^o
    - the vertex is B, its neighbors are A and C
  - It is an ear – cut it, by induction the remaining polygon has at least one more non-coincident ear
  - It is not an ear:
    - find a point P inside ABC that is farthest from AC
    - no edges can intersect BP – link B with P to get two polygons
    - these polygons have two ears each, at least one not incident to BP

Ear clipping algorithm

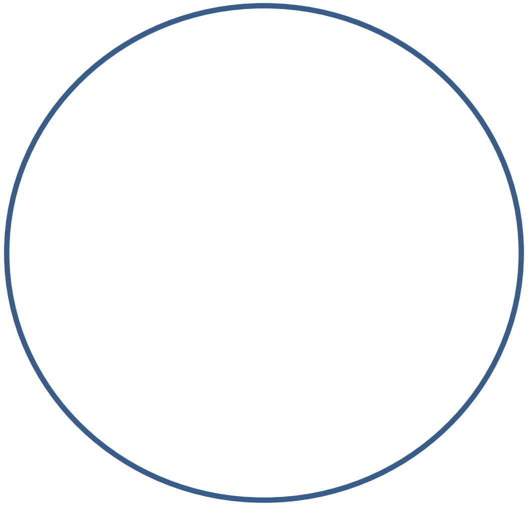
- Algorithm: Follow the proof of the theorem

Problem I. Circle Intersection

Case 0. Circles coincide

- $x_1 = x_2$
- $y_1 = y_2$
- $r_1 = r_2$

The answer is -1

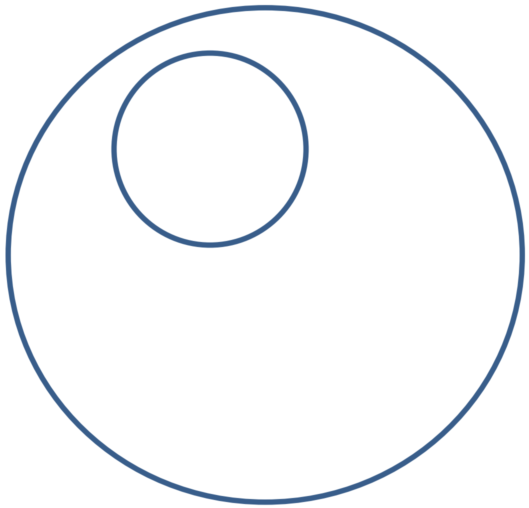


Problem I. Circle Intersection

Case 1. Circles are strictly one inside another

- $d^2 = (x_1 - x_2)^2 + (y_1 - y_2)^2$
- $d^2 < (r_1 - r_2)^2$

The answer is 0

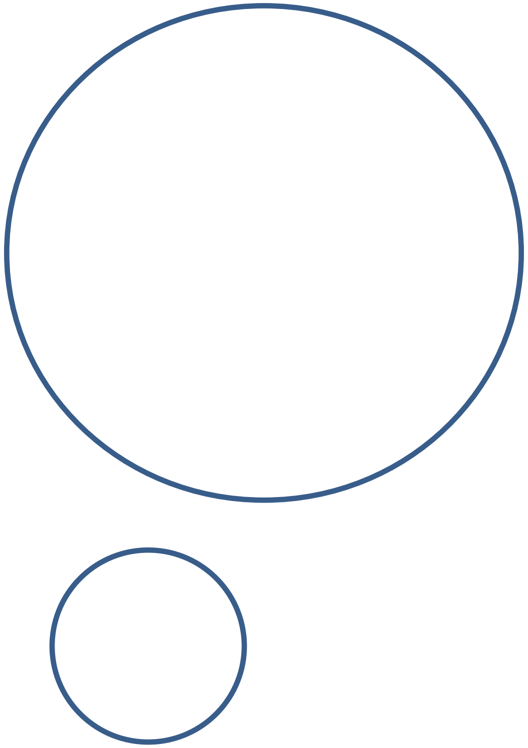


Problem I. Circle Intersection

Case 2. Circles are strictly separated

- $d^2 = (x_1 - x_2)^2 + (y_1 - y_2)^2$
- $d^2 > (r_1 + r_2)^2$

The answer is 0



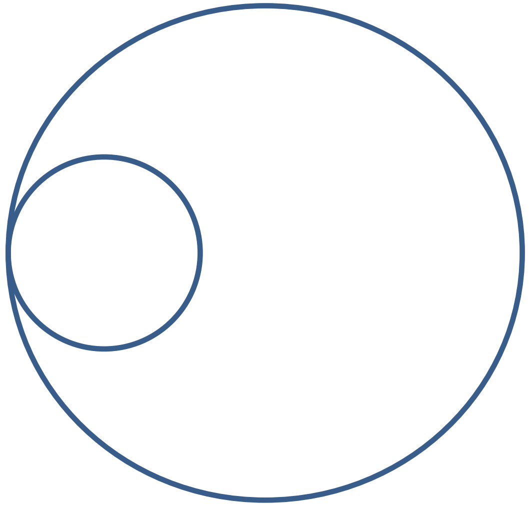
Problem I. Circle Intersection

Case 3. Circles have an inner touch

- $d^2 = (x_1 - x_2)^2 + (y_1 - y_2)^2$
- $d^2 = (r_1 - r_2)^2$, $d \neq 0$

The answer is 1:

- $x = x_1 - (x_2 - x_1)\ r_1 / d$
- $y = y_1 - (y_2 - y_1)\ r_1 / d$



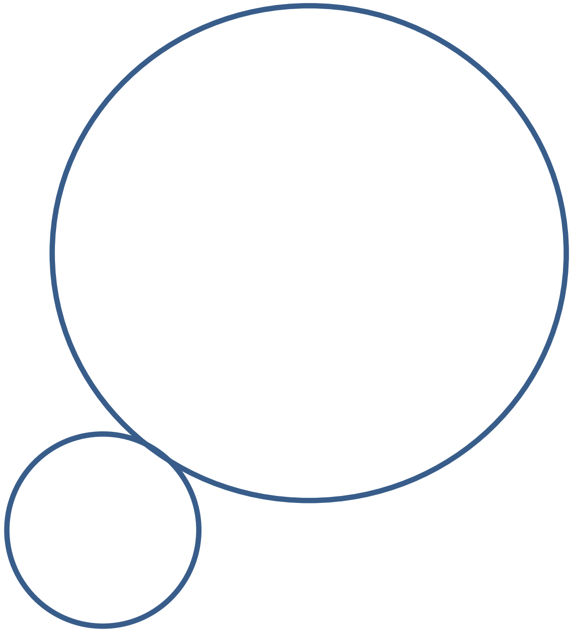
Problem I. Circle Intersection

Case 4. Circles have an outer touch

- $d^2 = (x_1 - x_2)^2 + (y_1 - y_2)^2$
- $d^2 = (r_1 + r_2)^2$

The answer is 1:

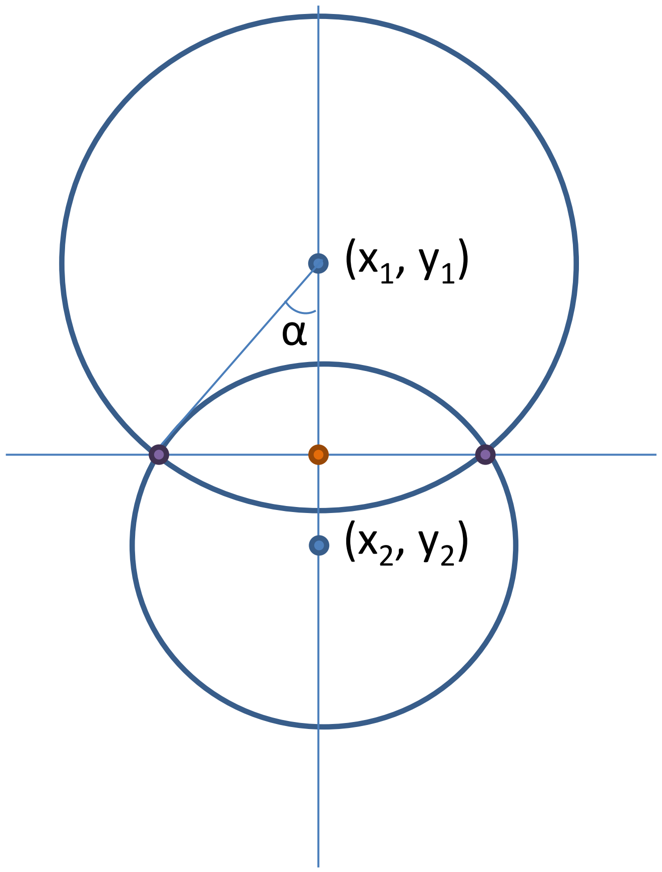
- $x = x_1 + (x_2 - x_1)\ r_1 / d$
- $y = y_1 + (y_2 - y_1)\ r_1 / d$



Problem I. Circle Intersection

Case 5. The general case

- The answer is 2
- $d^2 = (x_1 - x_2)^2 + (y_1 - y_2)^2$
- $\cos\alpha = (d^2 + r_1^2 - r_2^2)/(2r_1 d)$
- $\sin\alpha = \sqrt{1 - (\cos\alpha)^2}$
- $x_M = x_1 + (x_2 - x_1)\cos\alpha$
- $y_M = y_1 + (y_2 - y_1)\cos\alpha$
- $x_D = (x_2 - x_1)\sin\alpha$
- $y_D = (y_2 - y_1)\sin\alpha$
- $X = x_M \pm y_D$
- $Y = y_M \mp x_D$

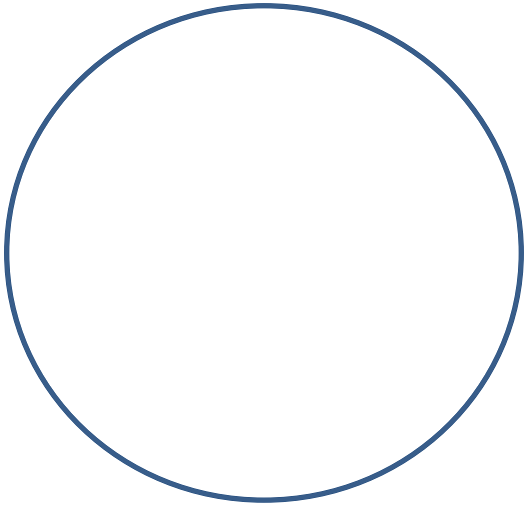


Problem J. Common Tangents 2

Case 0. Circles coincide

- $x_1 = x_2$
- $y_1 = y_2$
- $r_1 = r_2$

The answer is -1

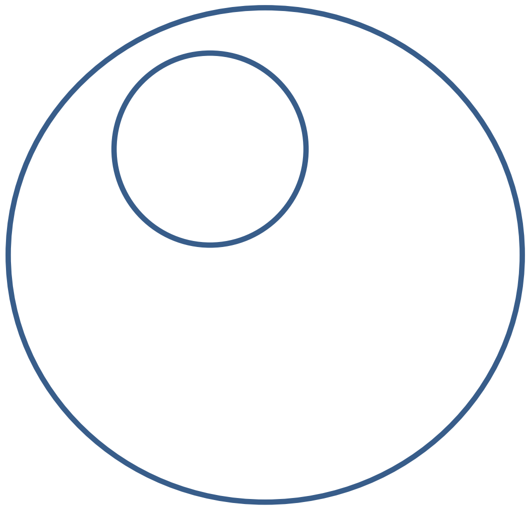


Problem J. Common Tangents 2

Case 1. Circles are strictly one inside another

- $d^2 = (x_1 - x_2)^2 + (y_1 - y_2)^2$
- $d^2 < (r_1 - r_2)^2$

The answer is 0



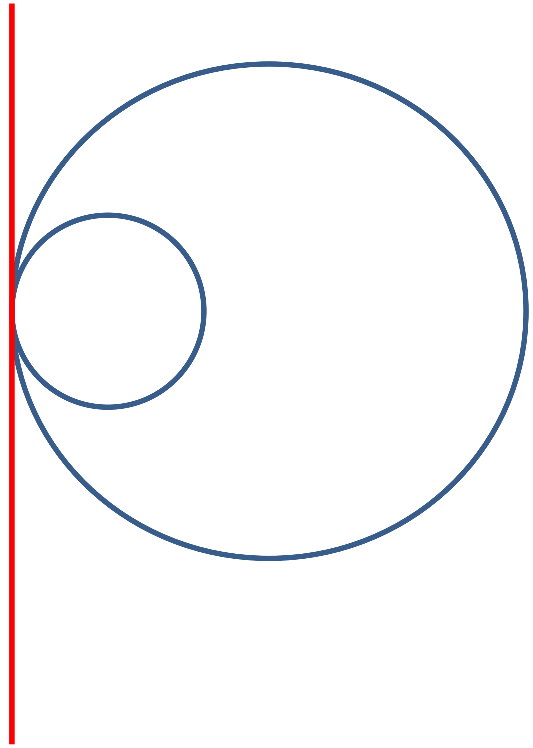
Problem J. Common Tangents 2

Case 2. Circles have an inner touch

- $d^2 = (x_1 - x_2)^2 + (y_1 - y_2)^2$
- $d^2 = (r_1 - r_2)^2$, $d \neq 0$

The answer is 1

- find the touch point as in Problem I
- the line direction vector is $(y_1 - y_2, x_2 - x_1)$



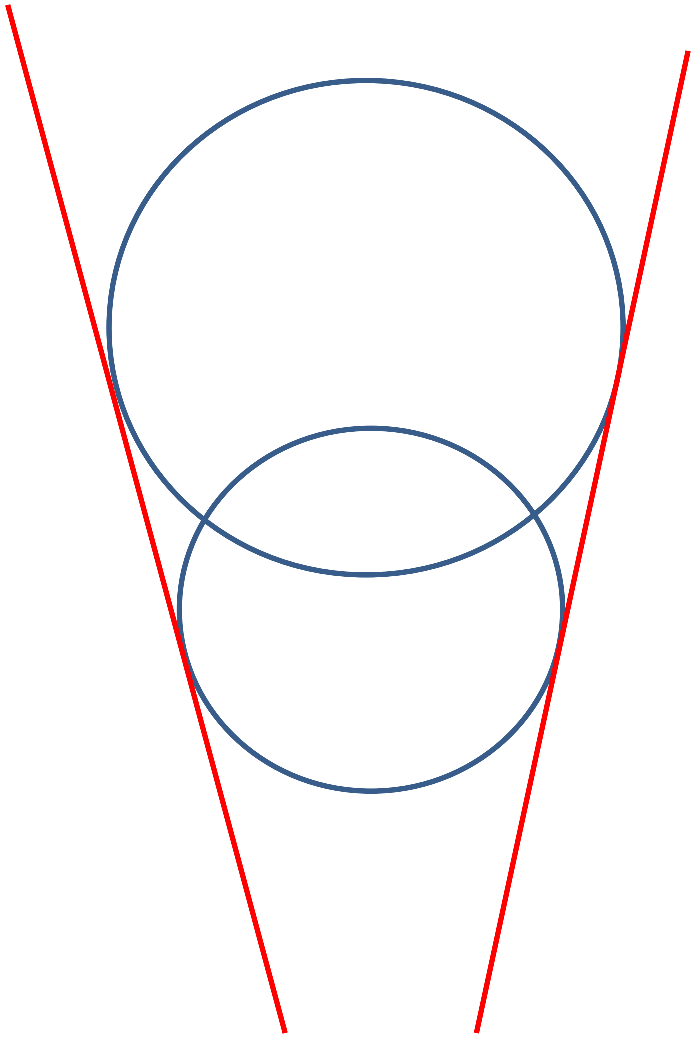
Problem J. Common Tangents 2

Case 3. Circle intersect

- $d^2 = (x_1 - x_2)^2 + (y_1 - y_2)^2$
- $(r_1 - r_2)^2 < d^2 < (r_1 + r_2)^2$

The answer is 2:

- two outer tangents



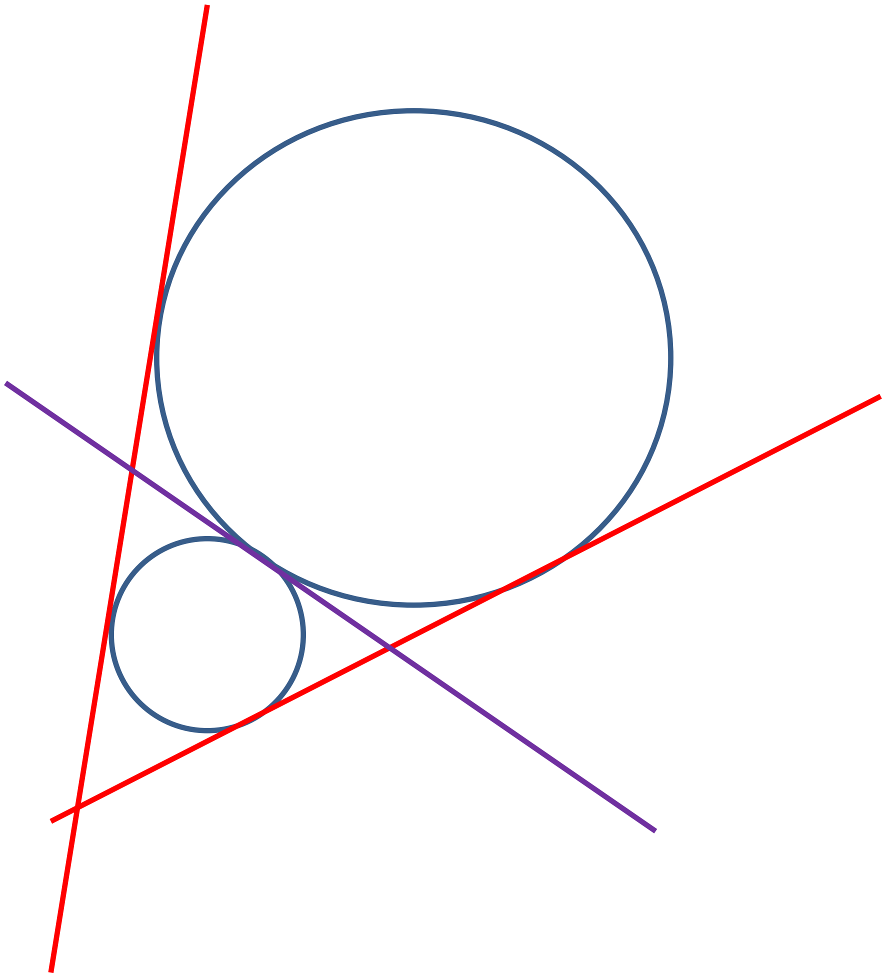
Problem J. Common Tangents 2

Case 4. Circles have an outer touch

- $d^2 = (x_1 - x_2)^2 + (y_1 - y_2)^2$
- $d^2 = (r_1 + r_2)^2$

The answer is 3:

- two outer tangents
- one inner tangent



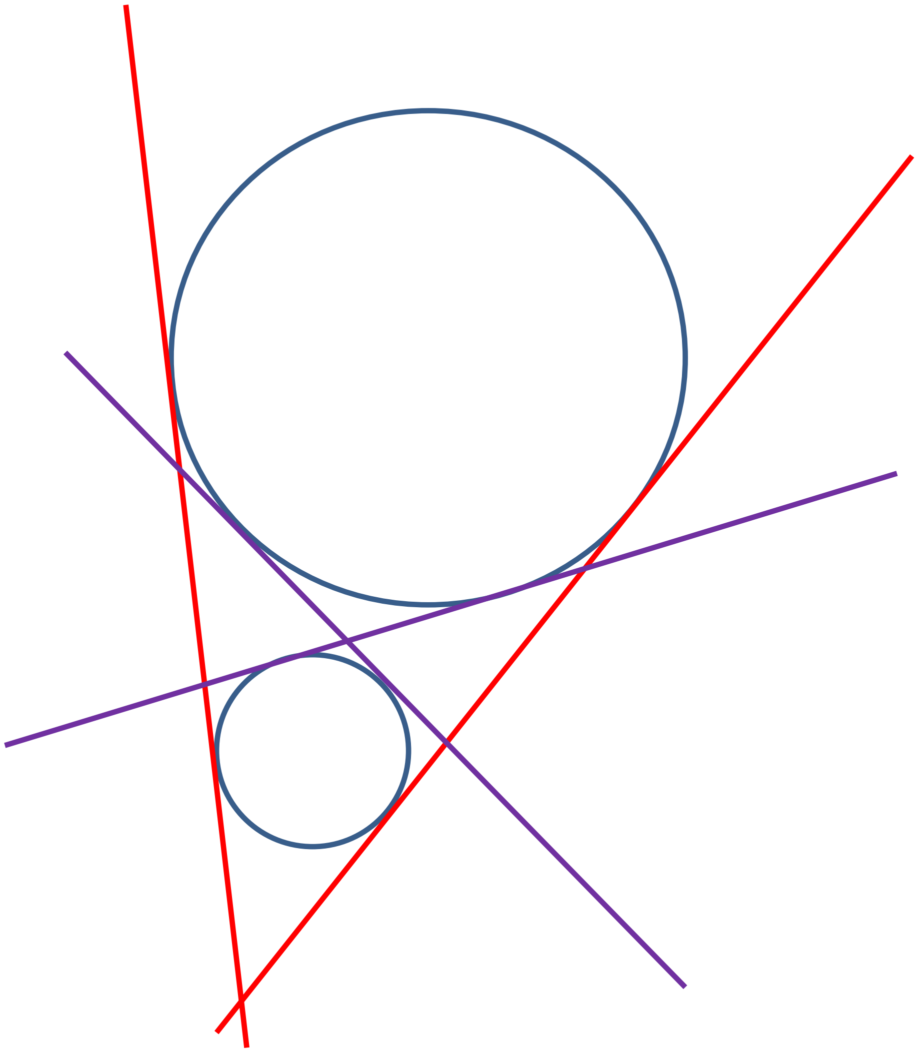
Problem J. Common Tangents 2

Case 5. Circles are strictly separated

- $d^2 = (x_1 - x_2)^2 + (y_1 - y_2)^2$
- $d^2 > (r_1 + r_2)^2$

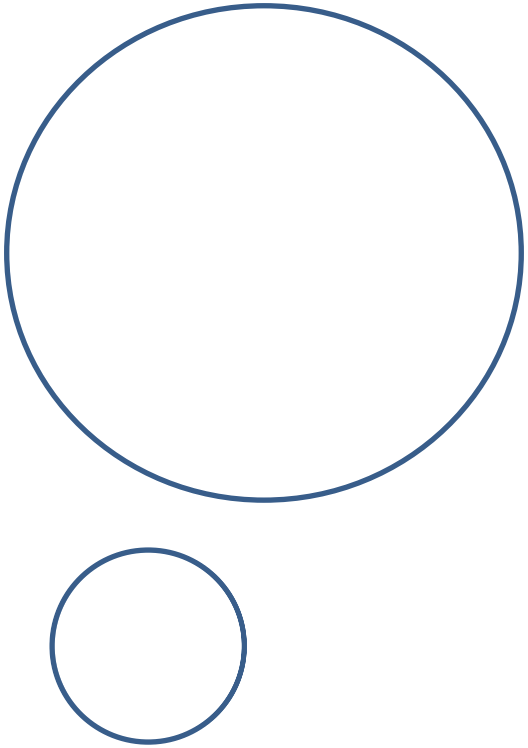
The answer is 4:

- two outer tangents
- two inner tangents



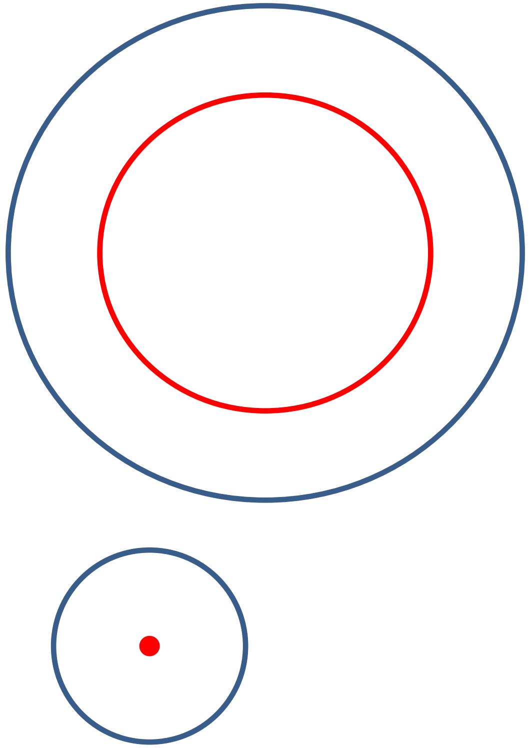
How to find the outer tangents?

- Let $r_1 \le r_2$
- Subtract r_1 from both circles
- Find the tangents from a point (x_1, y_1) to the new circle centered at (x_2, y_2)
- Move the tangents outside by r_1 in perpendicular direction



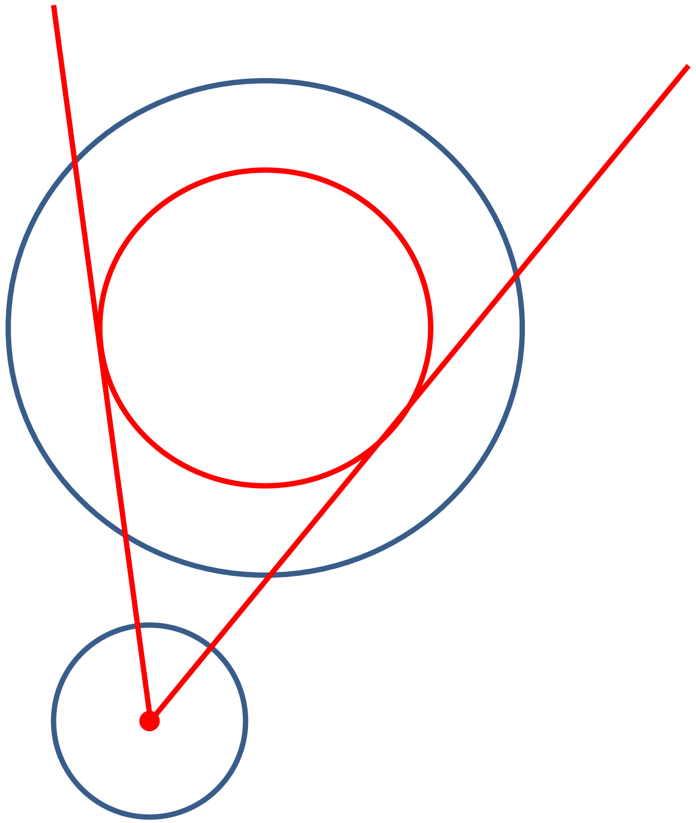
How to find the outer tangents?

- Let $r_1 \leq r_2$
- Subtract r_1 from both circles
- Find the tangents from a point (x_1, y_1) to the new circle centered at (x_2, y_2)
- Move the tangents outside by r_1 in perpendicular direction



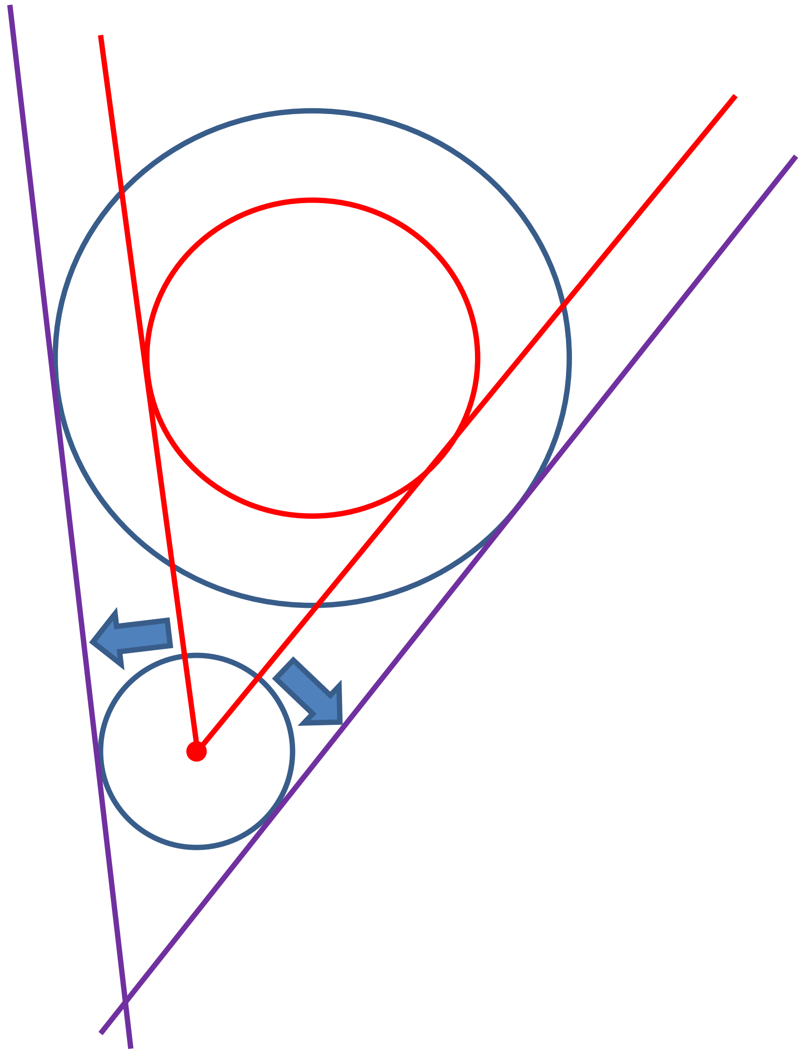
How to find the outer tangents?

- Let $r_1 \le r_2$
- Subtract r_1 from both circles
- Find the tangents from a point (x_1, y_1) to the new circle centered at (x_2, y_2)
- Move the tangents outside by r_1 in perpendicular direction



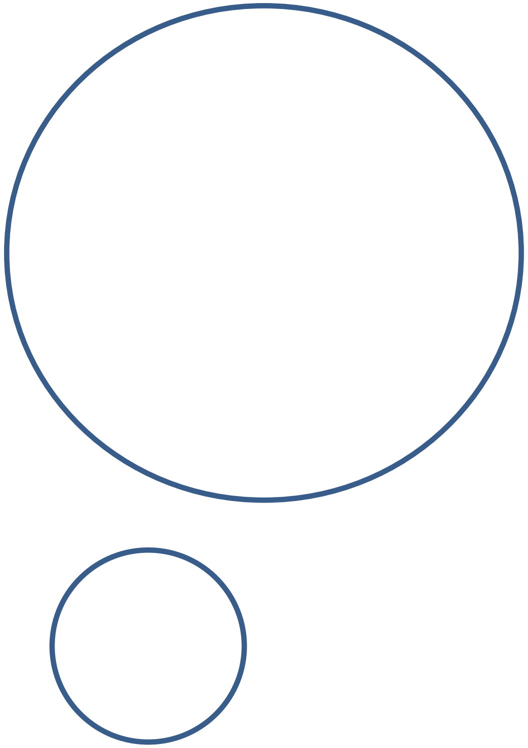
How to find the outer tangents?

- Let $r_1 \leq r_2$
- Subtract r_1 from both circles
- Find the tangents from a point (x_1, y_1) to the new circle centered at (x_2, y_2)
- Move the tangents outside by r_1 in perpendicular direction



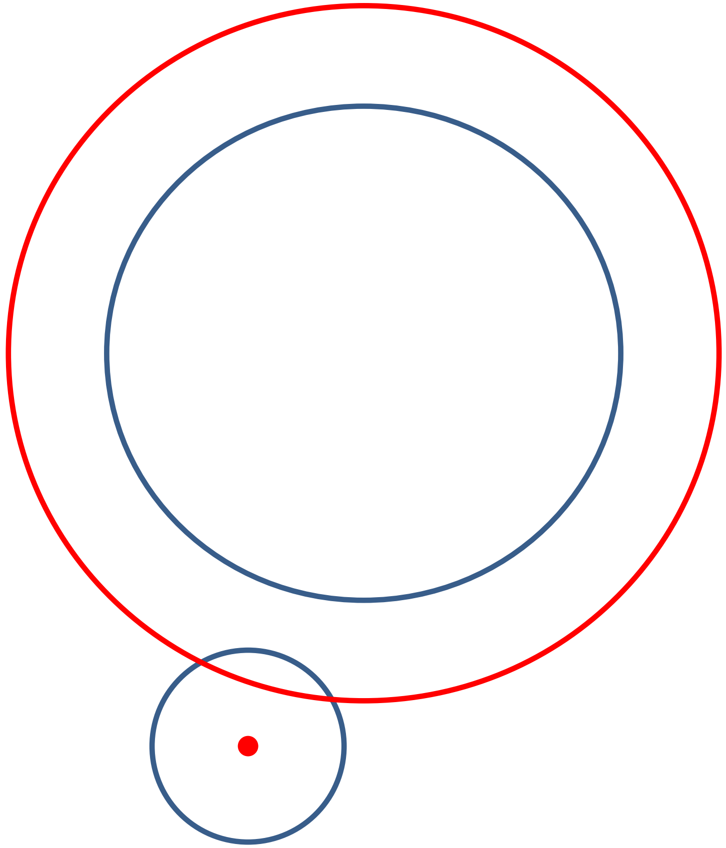
How to find the inner tangents?

- Add r_1 to circle 2
- Find the tangents from a point (x_1, y_1) to the new circle centered at (x_2, y_2)
- Move the tangents inside by r_1 in perpendicular direction



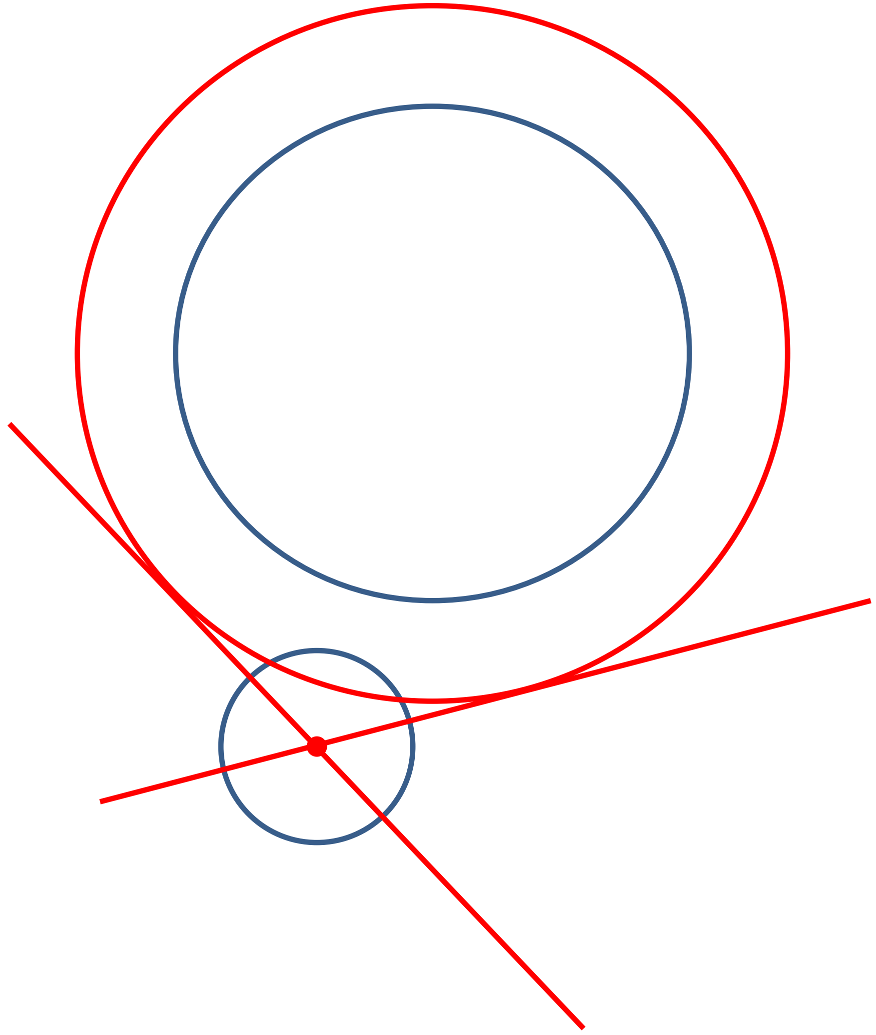
How to find the inner tangents?

- Add r_1 to circle 2
- Find the tangents from a point (x_1, y_1) to the new circle centered at (x_2, y_2)
- Move the tangents inside by r_1 in perpendicular direction



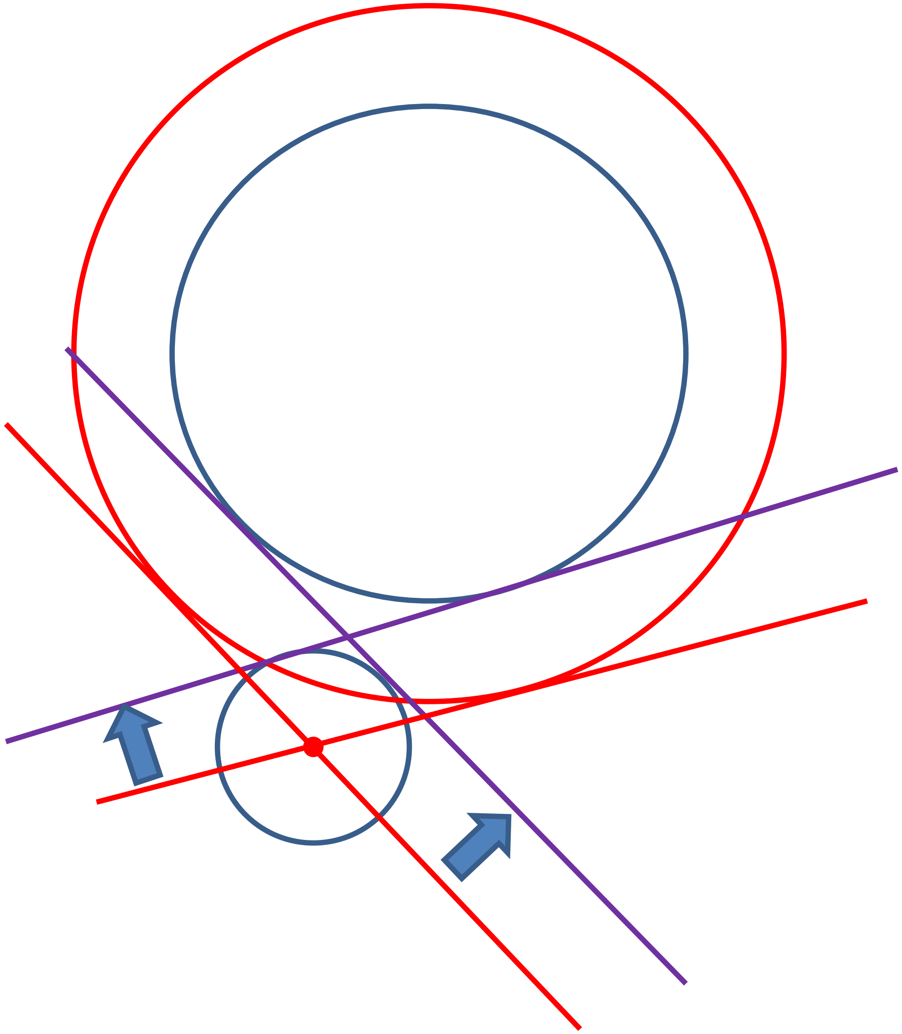
How to find the inner tangents?

- Add r_1 to circle 2
- Find the tangents from a point (x_1, y_1) to the new circle centered at (x_2, y_2)
- Move the tangents inside by r_1 in perpendicular direction



How to find the inner tangents?

- Add r_1 to circle 2
- Find the tangents from a point (x_1, y_1) to the new circle centered at (x_2, y_2)
- Move the tangents inside by r_1 in perpendicular direction



Thank you!

Thank you for your attention!