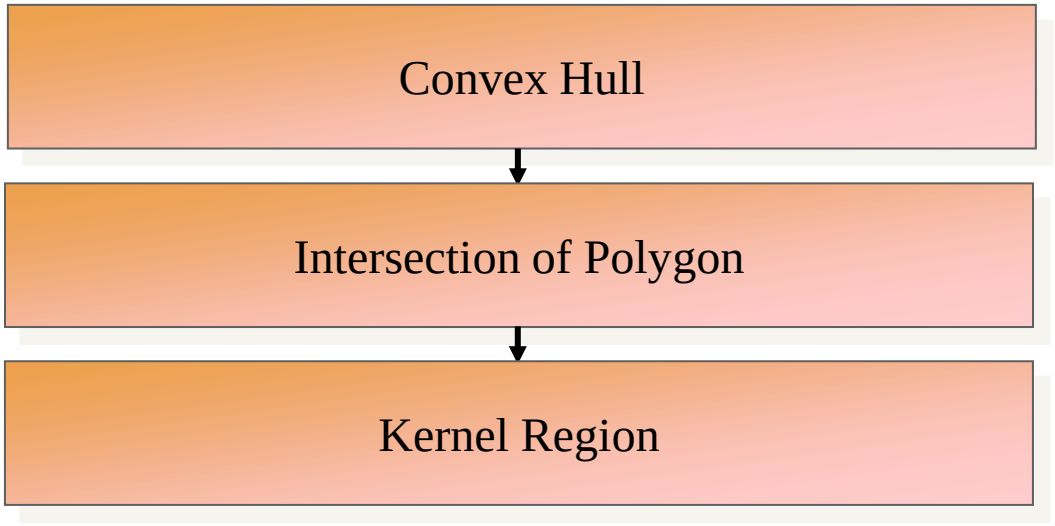
NCKU Programming Contest Training Course
2013/08/07

Pin-Chieh Huang (free999)

http://myweb.ncku.edu.tw/~p76014143/20130807.rar

Department of Computer Science and Information Engineering
National Cheng Kung University
Tainan, Taiwan

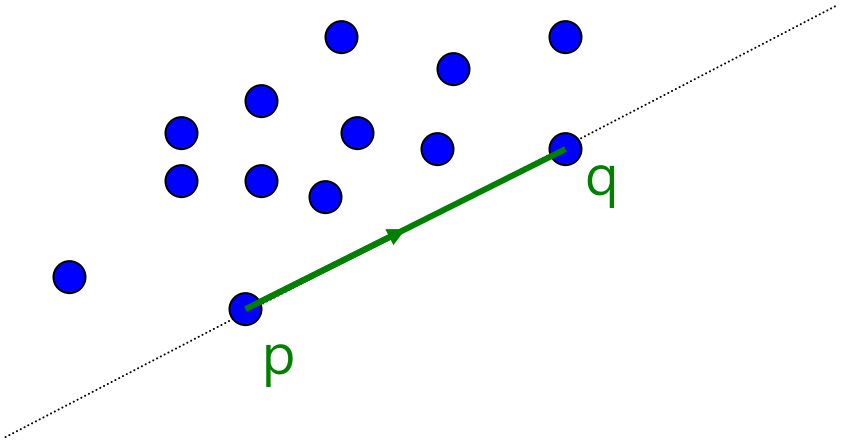
Outline



Convex Hull

- Definition
  - The convex hull of a set Q of points is the smallest convex polygon P for which each point in Q is either on the boundary of P or in this interior
- Algorithm
  - Brute Force
  - Gift-Wrap
  - Quick Hull
  - **Graham-Scan**

Brute Force



Algorithm CH(P)

E ← ∅ (* edge-list of CH(P) *)

for all ordered pairs (p,q) ∈ P×P, p ≠ q do

  supporting ← true

  for all points r ∈ P-{p,q} do

    if r is on the right side of pq then supporting ← false

  if supporting then add directed edge pq to E

From the (un-ordered) edge-list E, construct the list of vertices of CH(P) in CCW order in O(n) time. (How?)

end

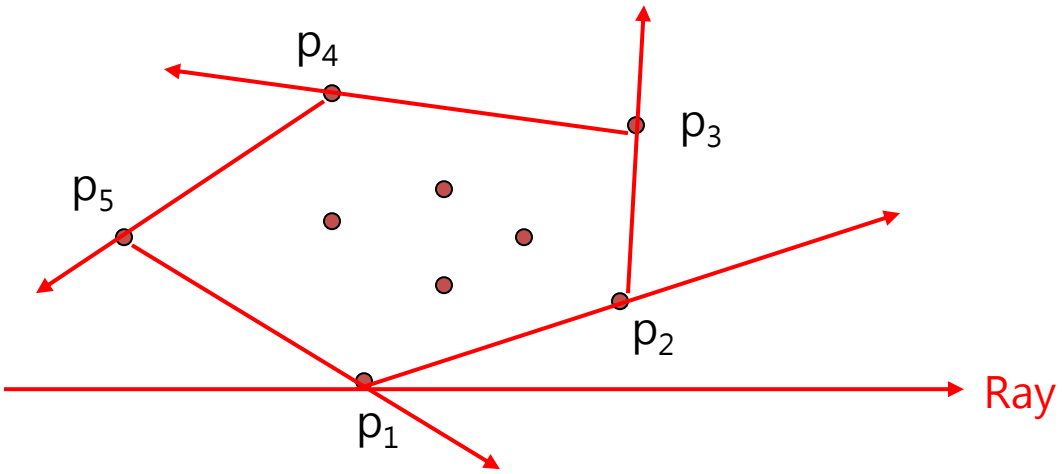
$O(n^3)$ still too slow!

Gift Wrap

Step 1: Let p_1 be the point with minimum y-coordinate (lex.)

Step 2: Anchor ray at current point and rotate to next anchor point.

Repeat.



Output-sensitive: O(nh) time.

n = # input points, h = # hull vertices (output size)

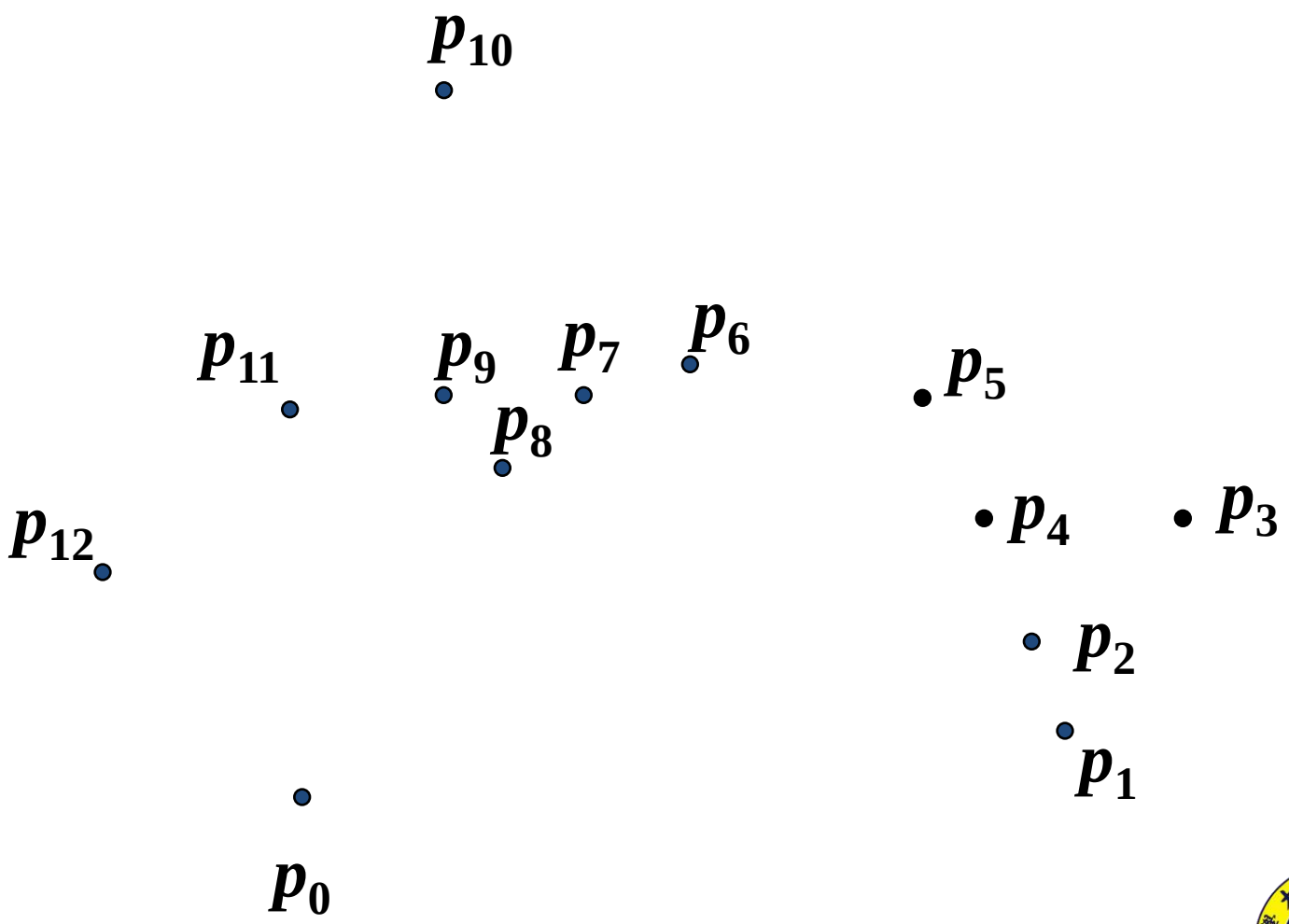
($3 \le h \le n$ if $n \ge 3$ and not all points collinear)

Worst-case: $O(n^2)$ time.

Gift Wrap

```
double AngfOfTwoVector(POINT p1, POINT p2, POINT p3, POINT p4)
{
        POINT v1, v2;
        v1.x = p2.x - p1.x;  v1.y = p2.y - p1.y;
        v2.x = p4.x - p3.x;  v2.y = p4.y - p3.y;
        double dotv = dot(v1, v2), lena, lenb;
        lena = sqrt((double)(sqr(v1.x) + sqr(v1.y)));
        lenb = sqrt((double)(sqr(v2.x) + sqr(v2.y)));
        return acos(dotv/(lena*lenb));
}
```

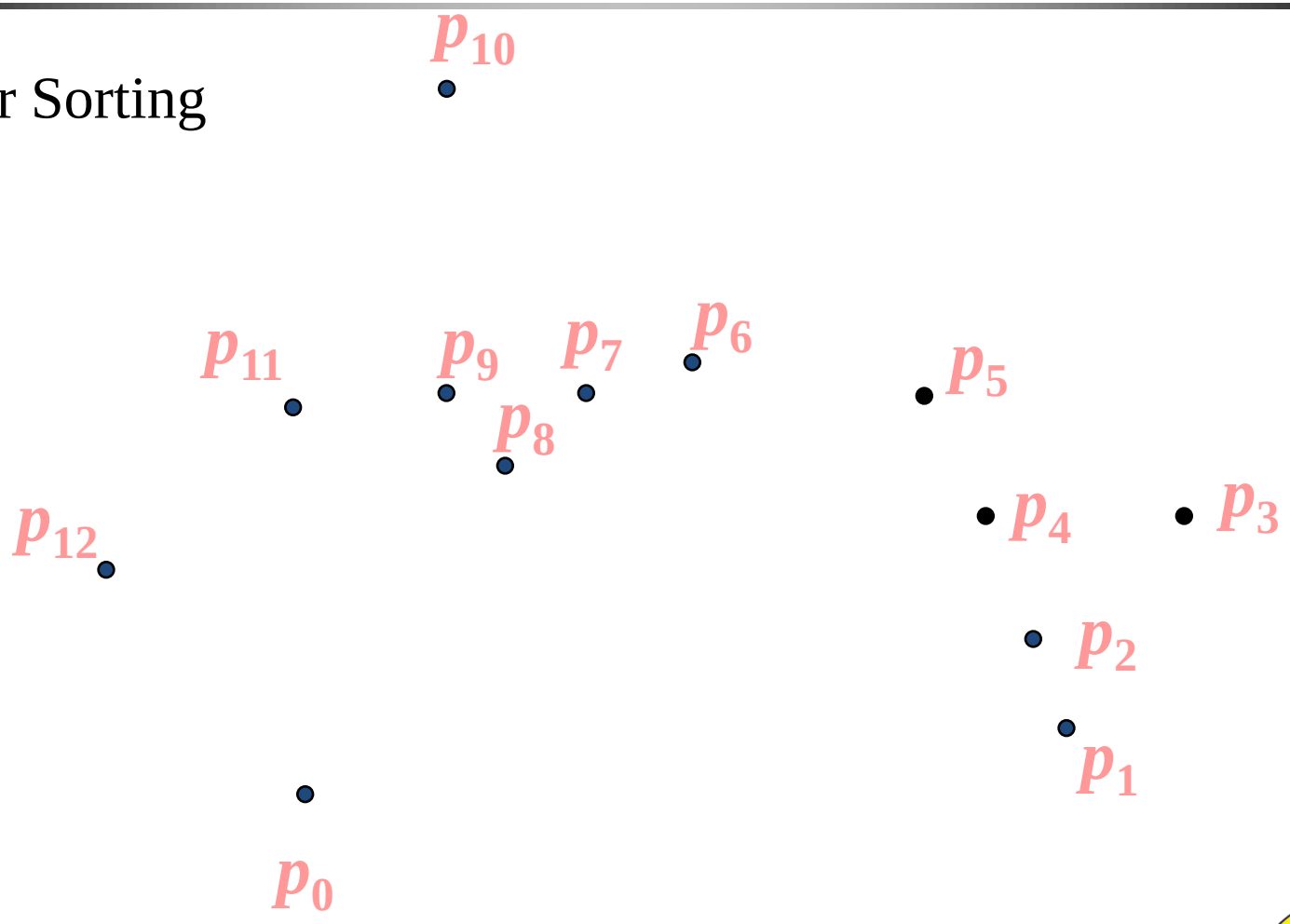
Graham Scan



made by electron & free999

Sort by Polar Angle

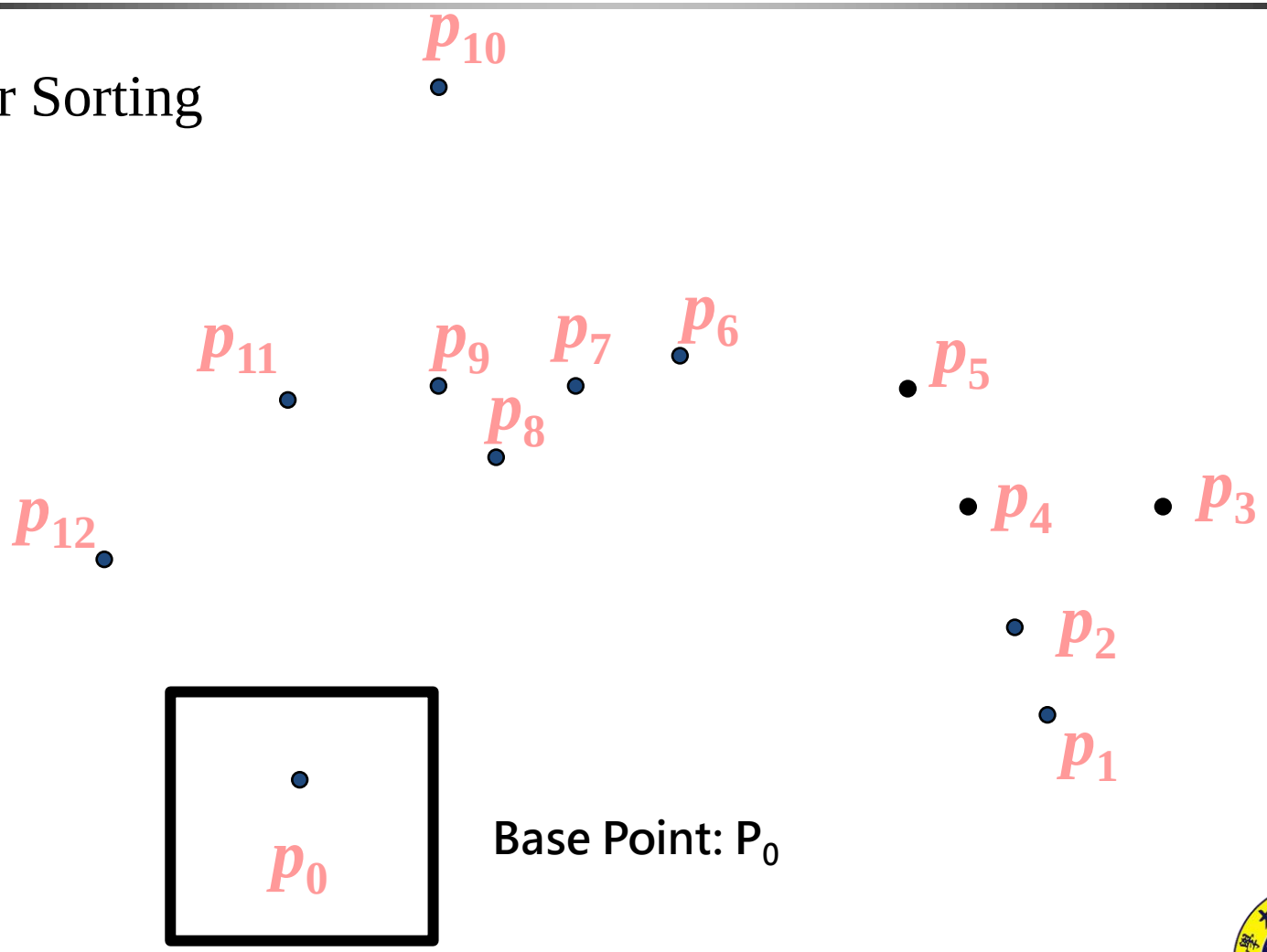
- Polar Sorting



made by electron & free999

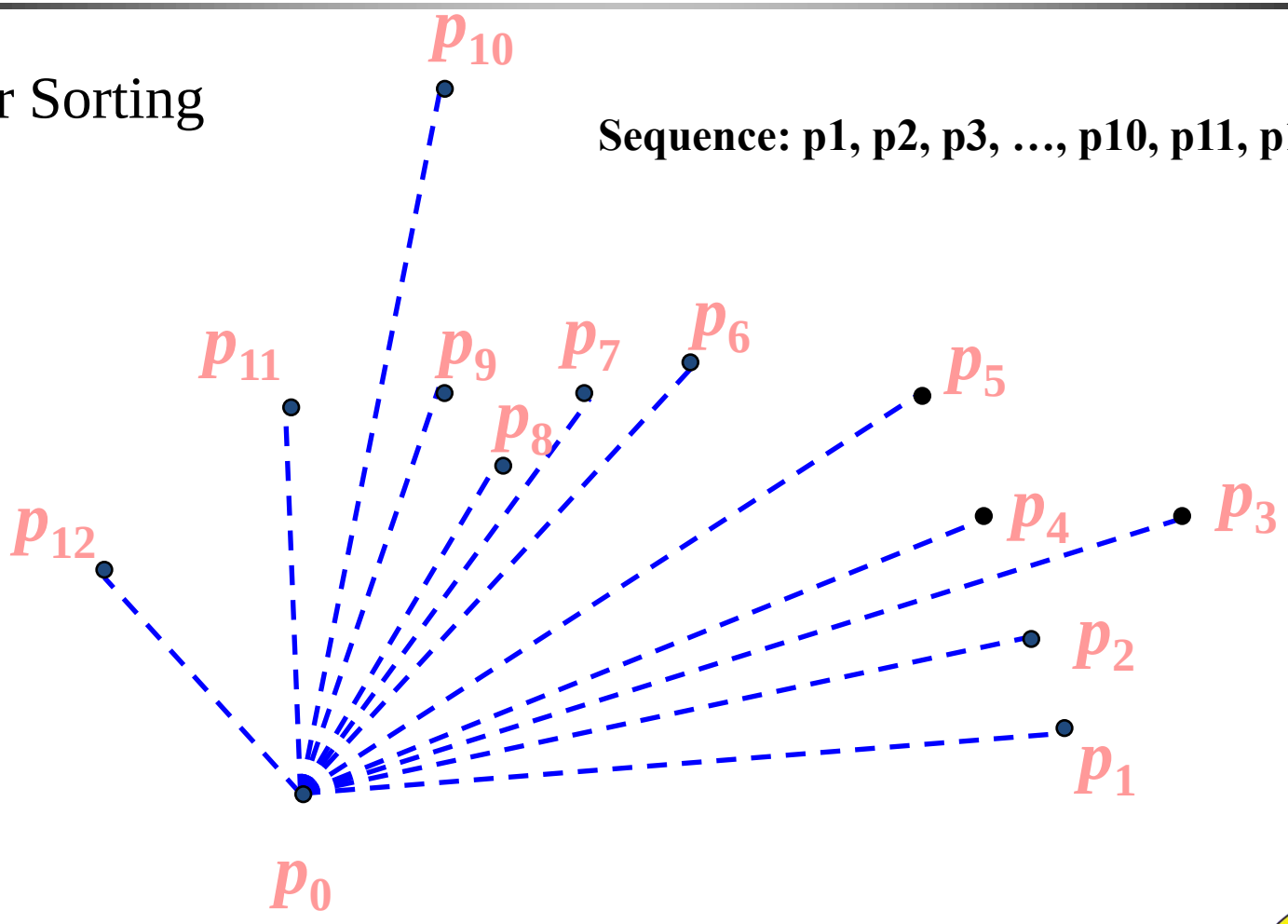
Sort by Polar Angle

- Polar Sorting



Sort by Polar Angle

- Polar Sorting



Sequence: p1, p2, p3, …, p10, p11, p12

made by electron & free999

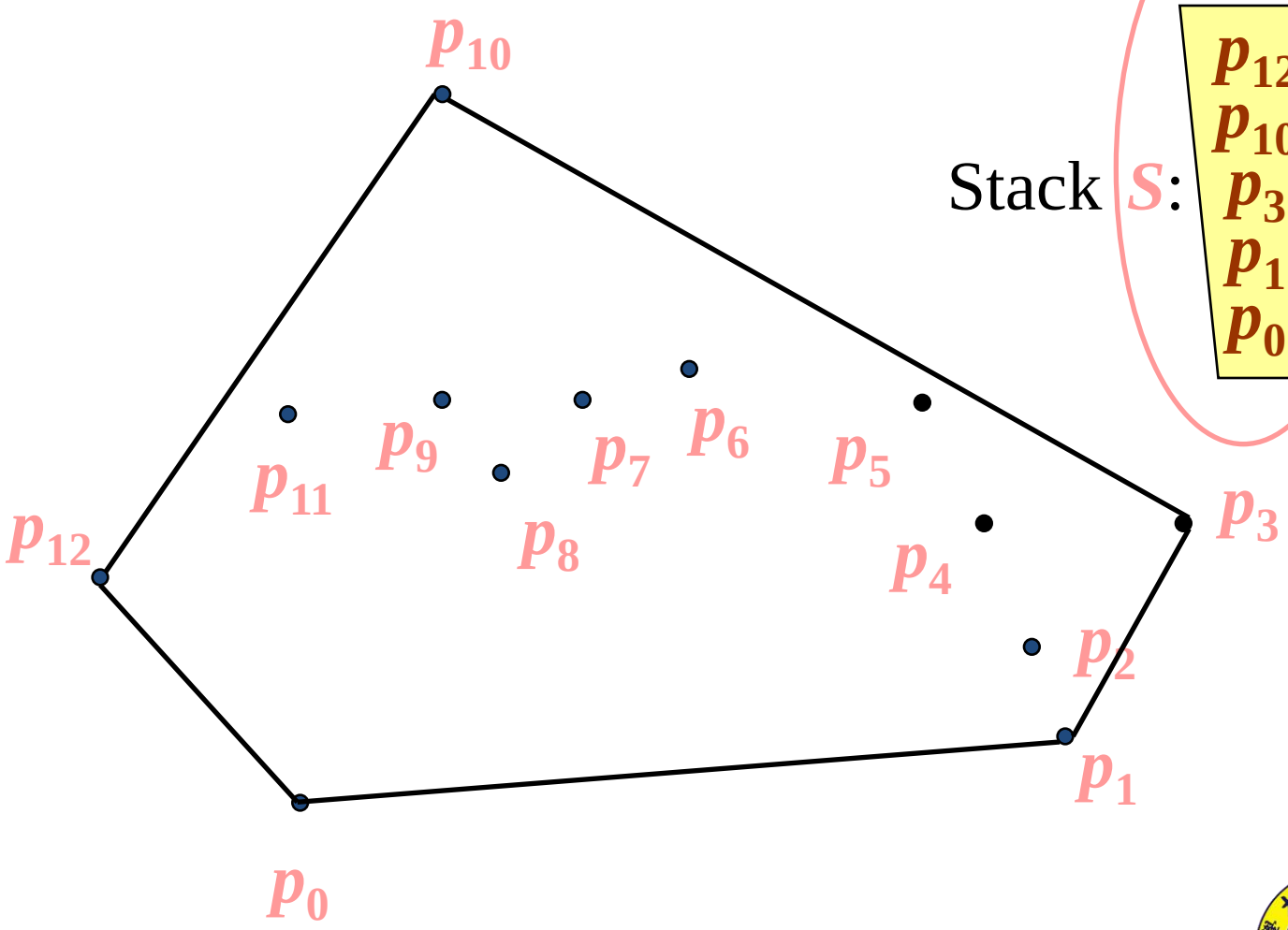
Sort by Polar Angle

- Polar Sorting

Compare Function:

```
p ← base point
bool cmp( point a, point b )
{
    return pa X pb ? 0
}
```

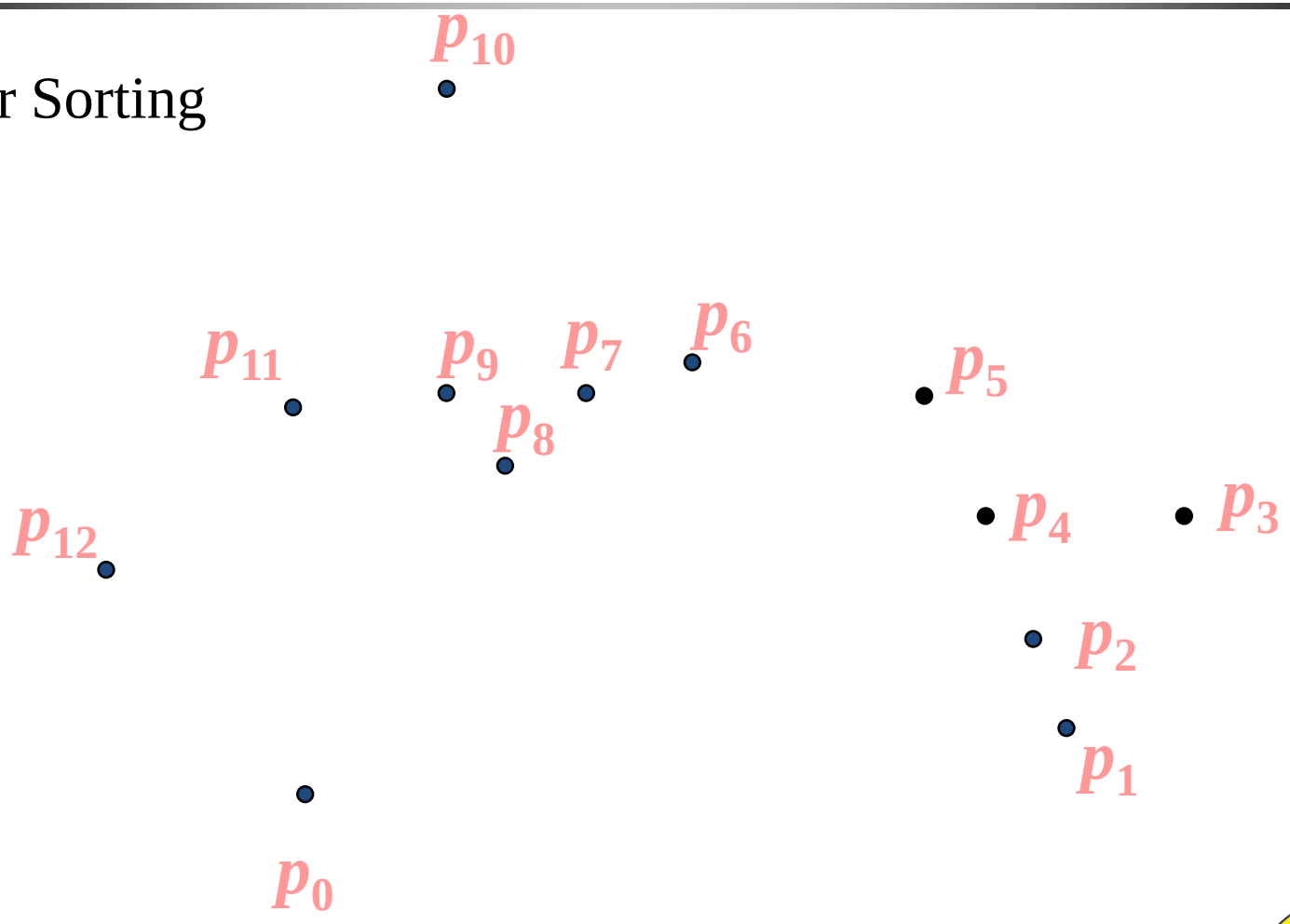
Graham Scan



made by electron & free999

Convex Hull

- Polar Sorting

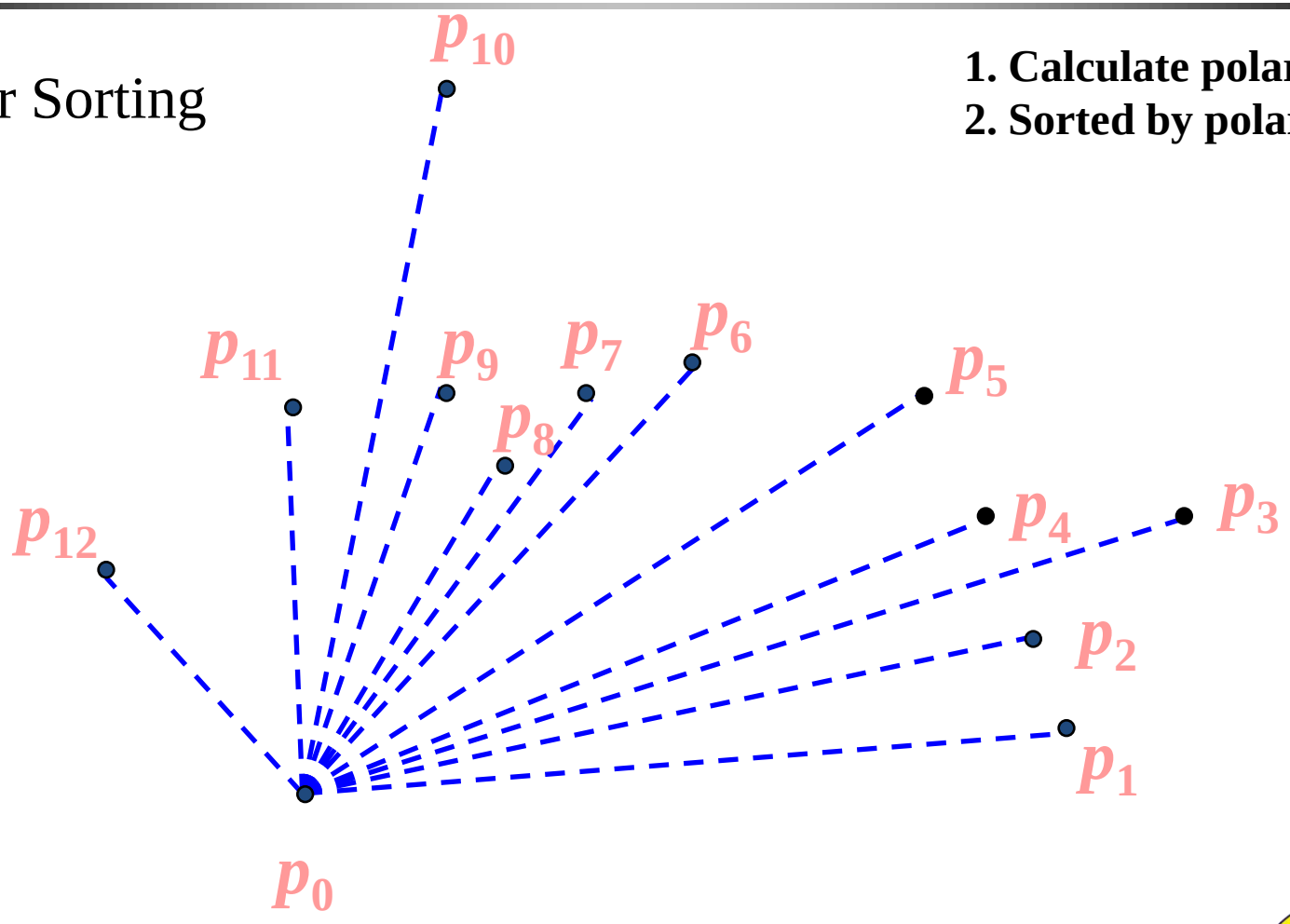


made by electron & free999

Convex Hull

- Polar Sorting

1. Calculate polar angle
2. Sorted by polar angle

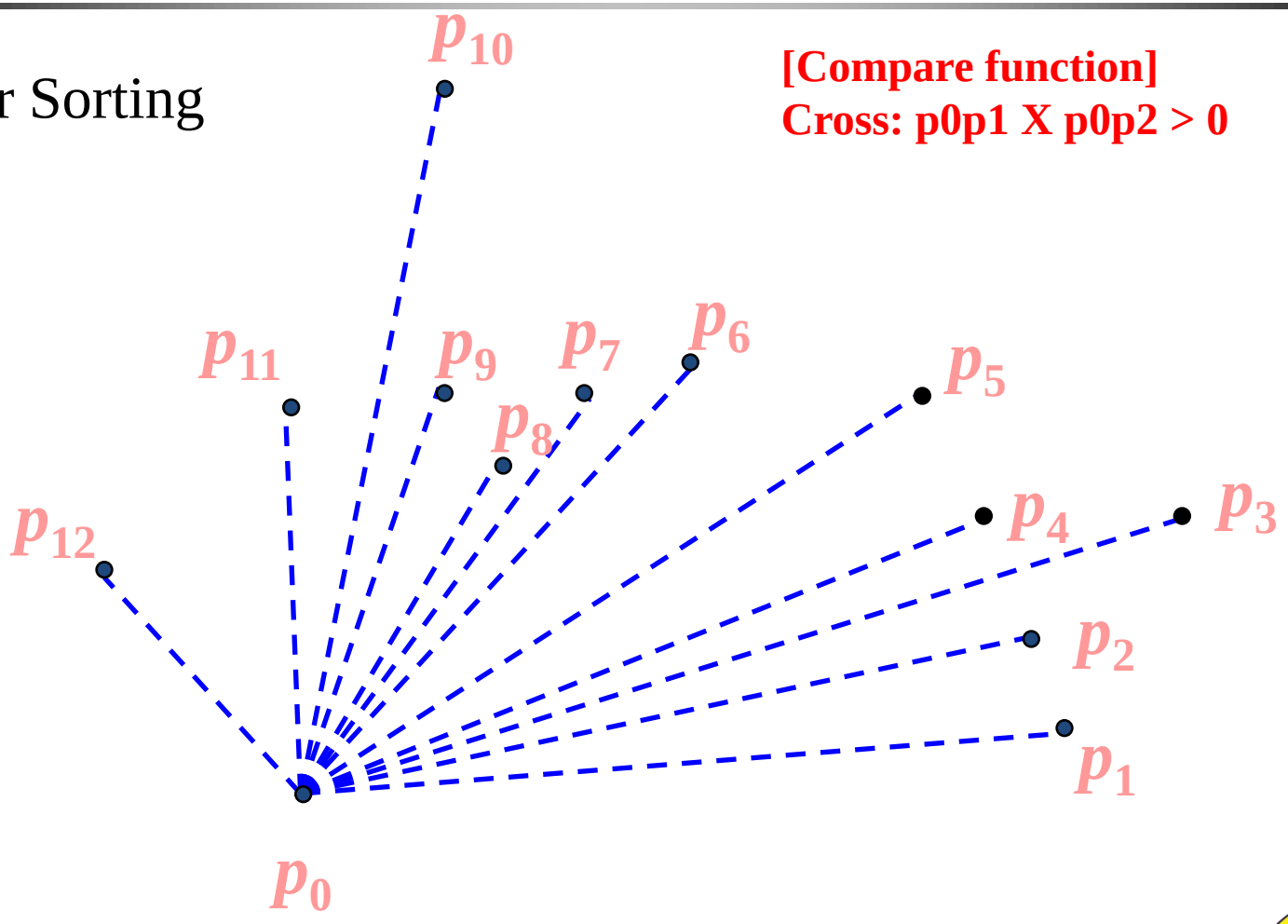


made by electron & free999

Convex Hull

- Polar Sorting

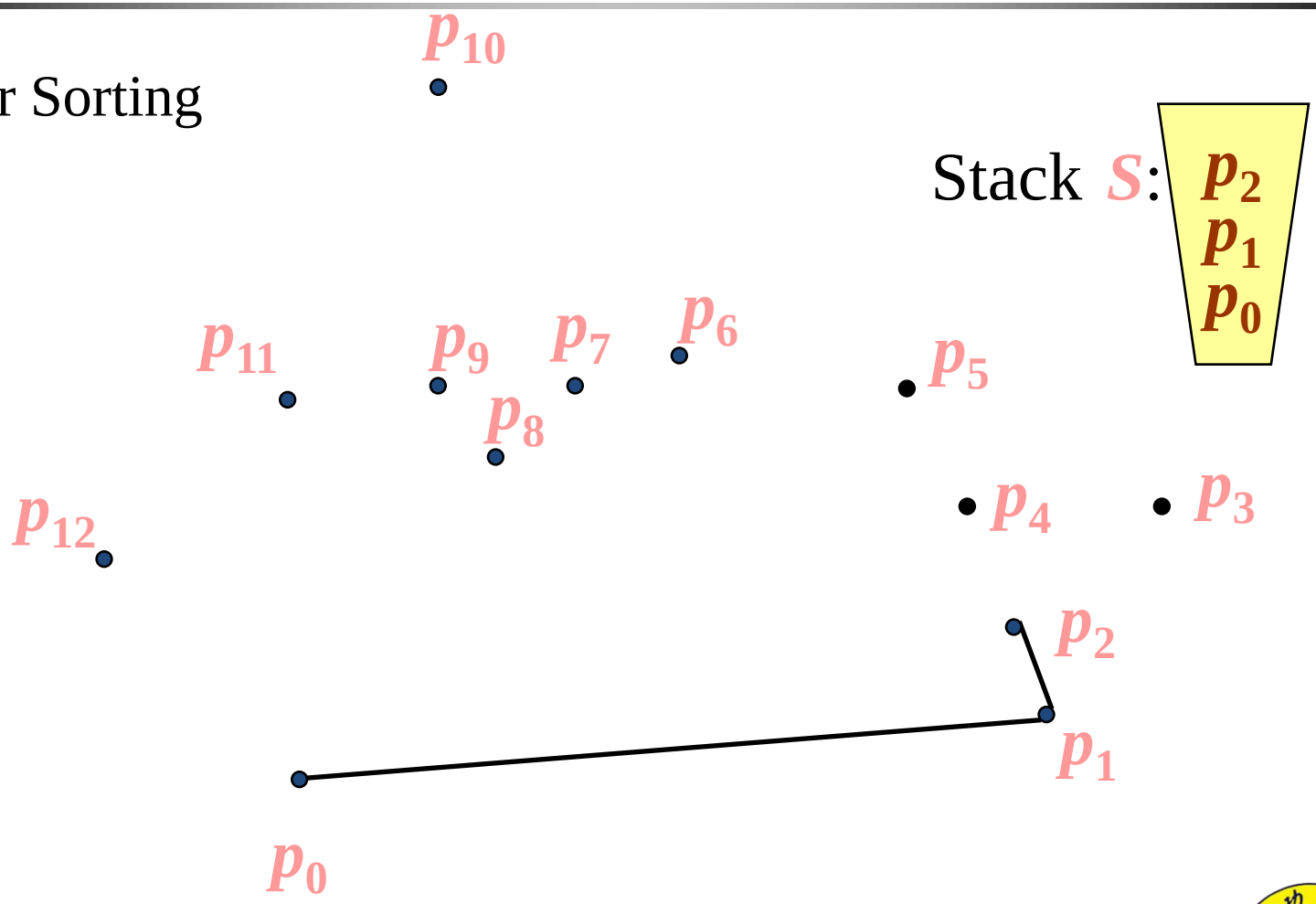
[Compare function]
Cross: p0p1 X p0p2 > 0



made by electron & free999

Convex Hull

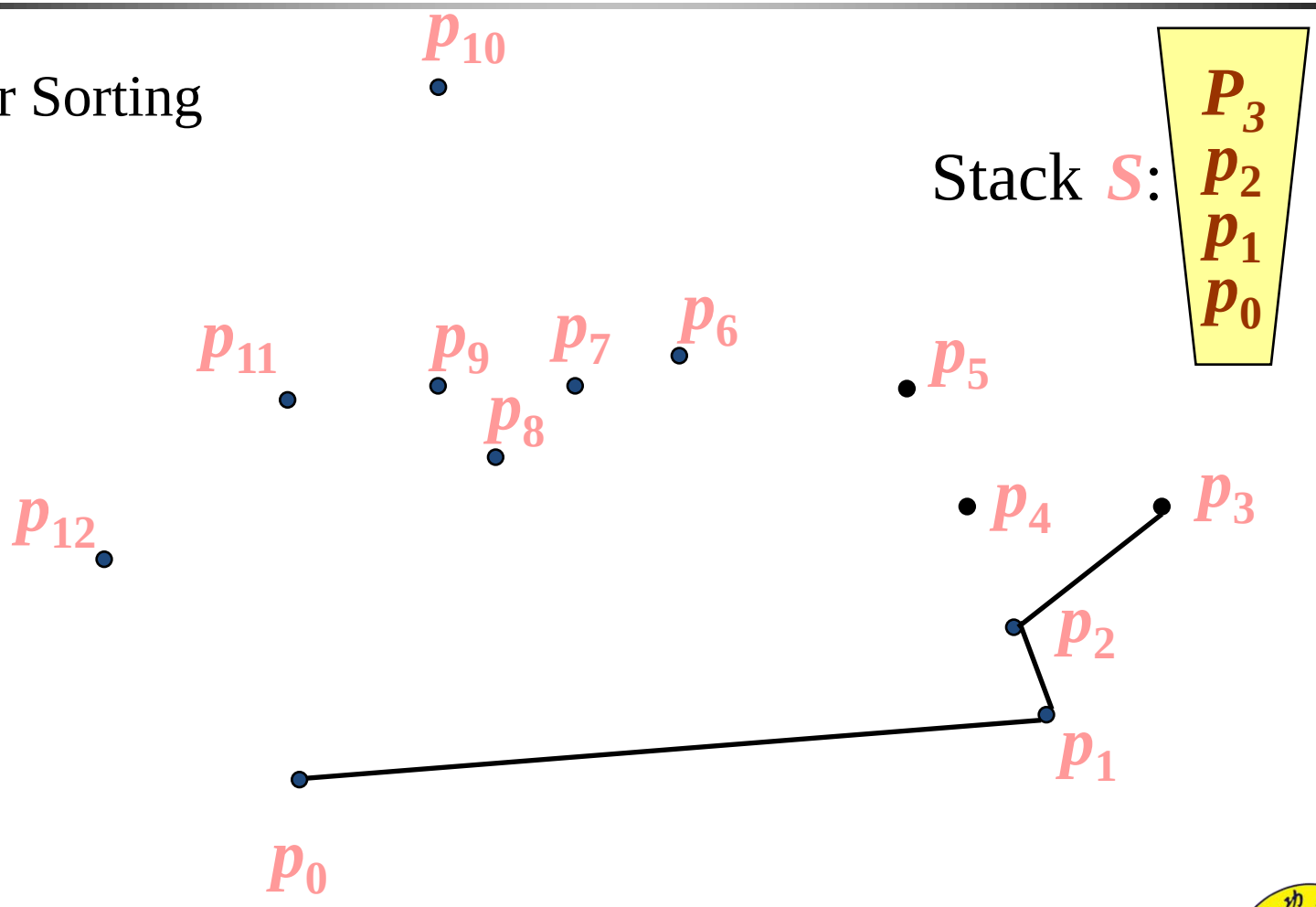
- Polar Sorting



made by electron & free999

Convex Hull

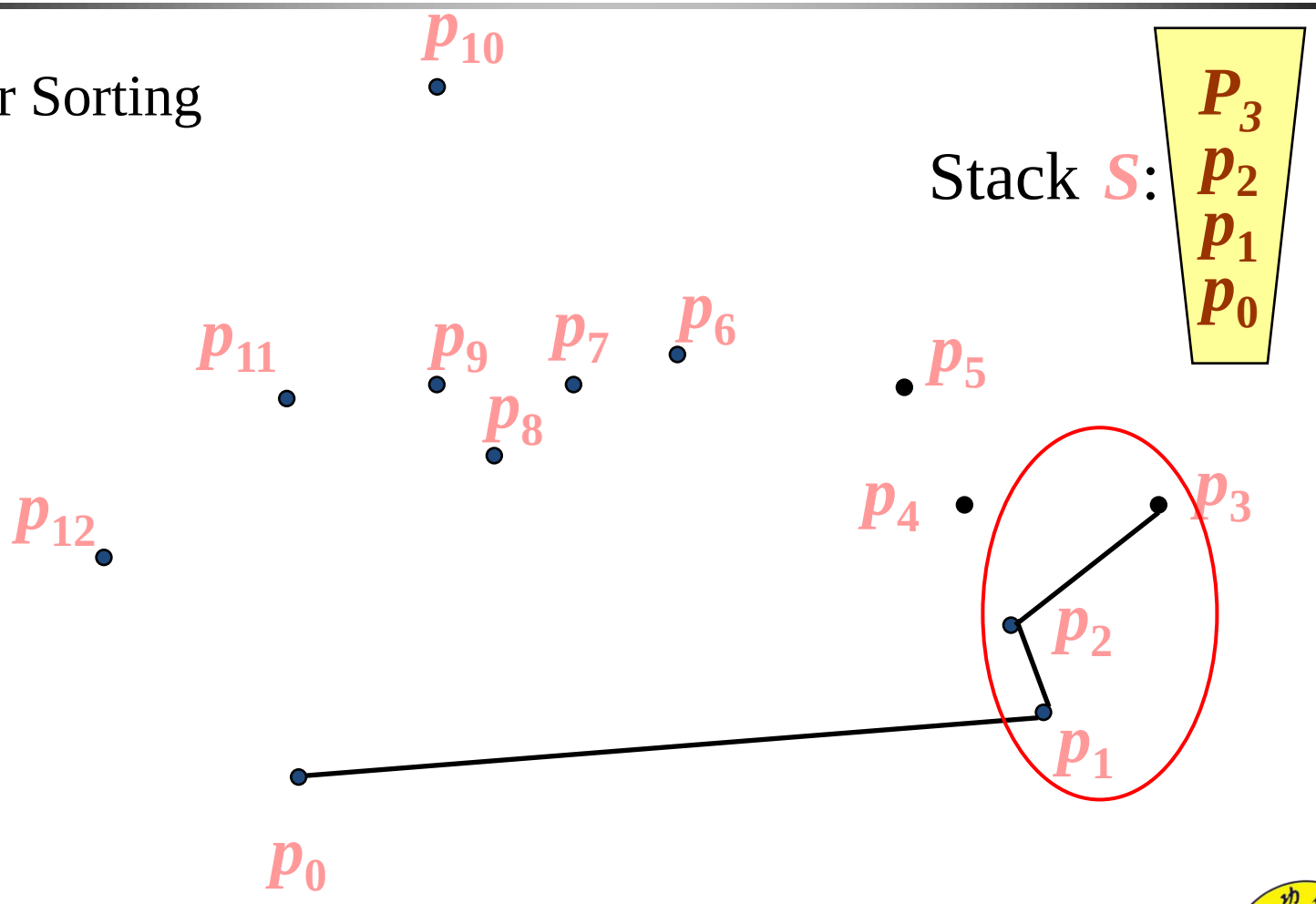
- Polar Sorting



made by electron & free999

Convex Hull

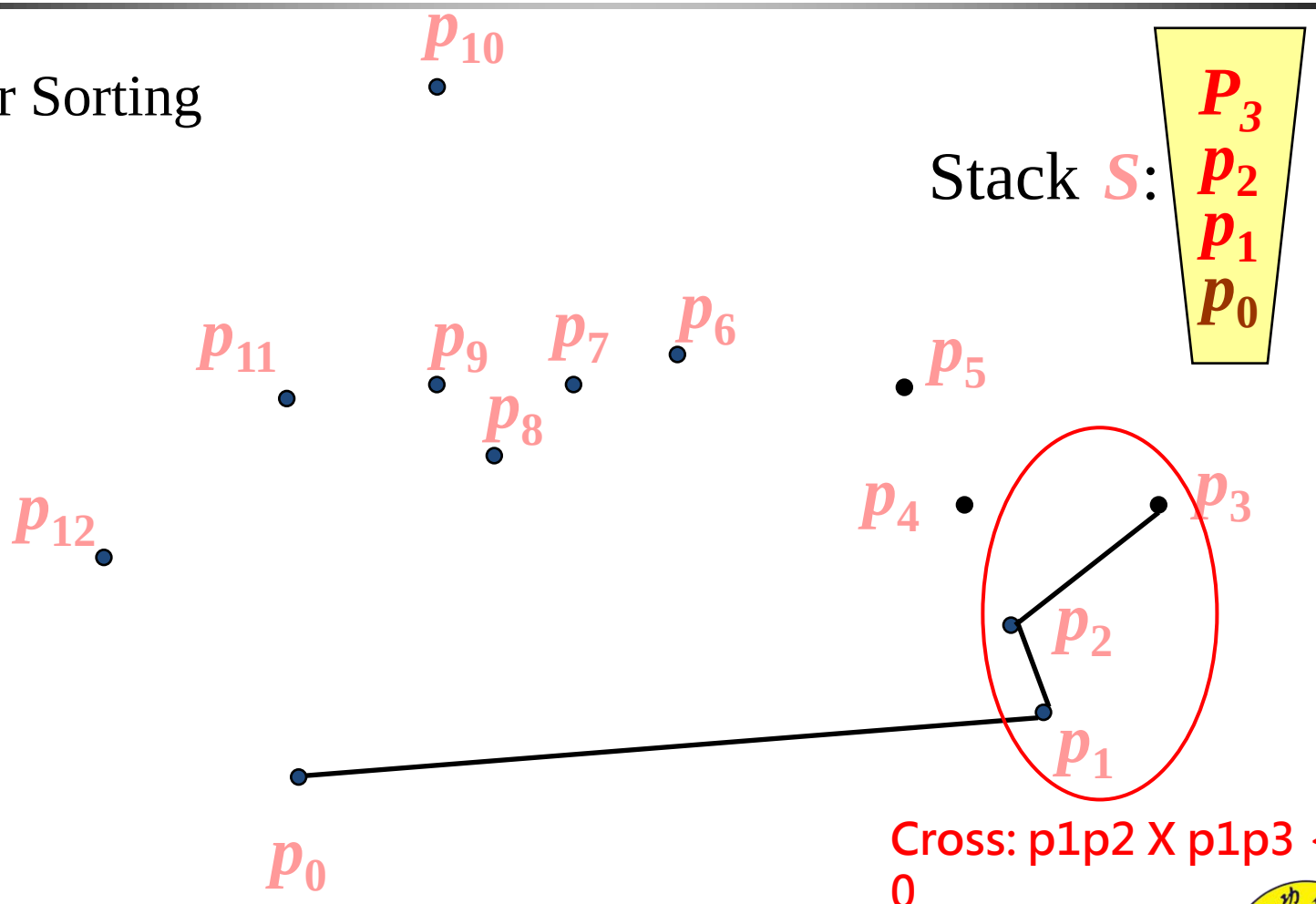
- Polar Sorting



made by electron & free999

Convex Hull

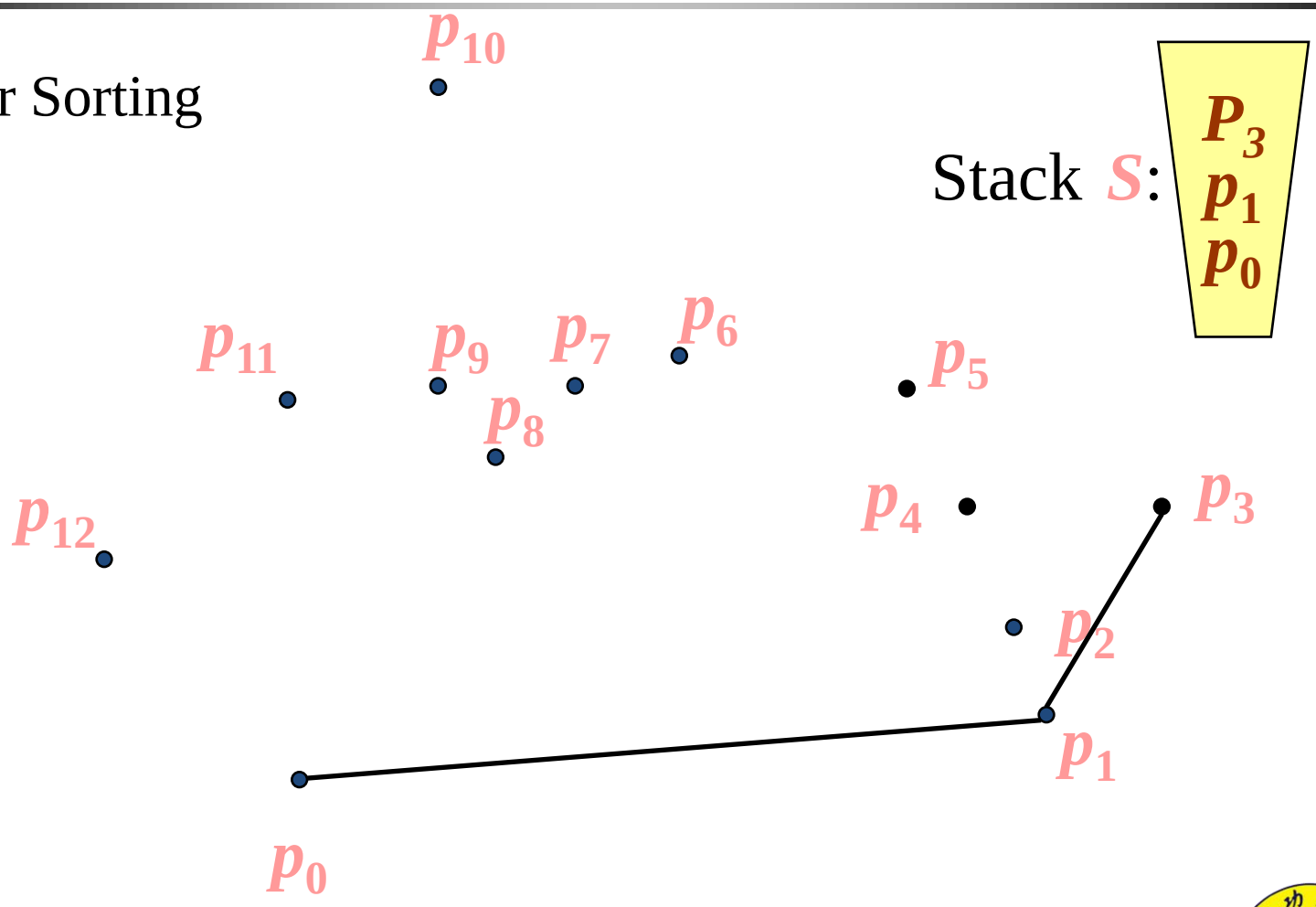
- Polar Sorting



made by electron & free999

Convex Hull

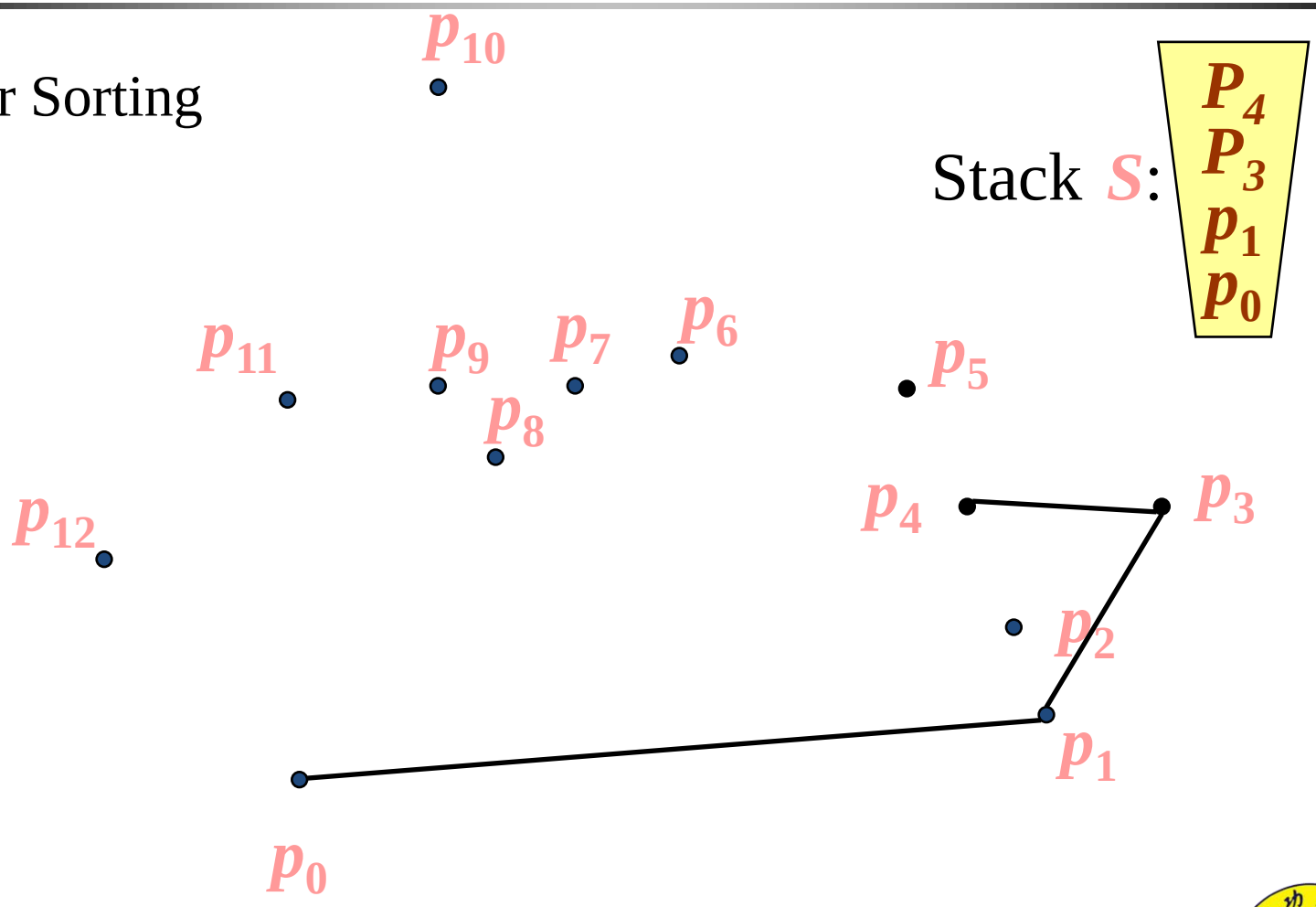
- Polar Sorting



made by electron & free999

Convex Hull

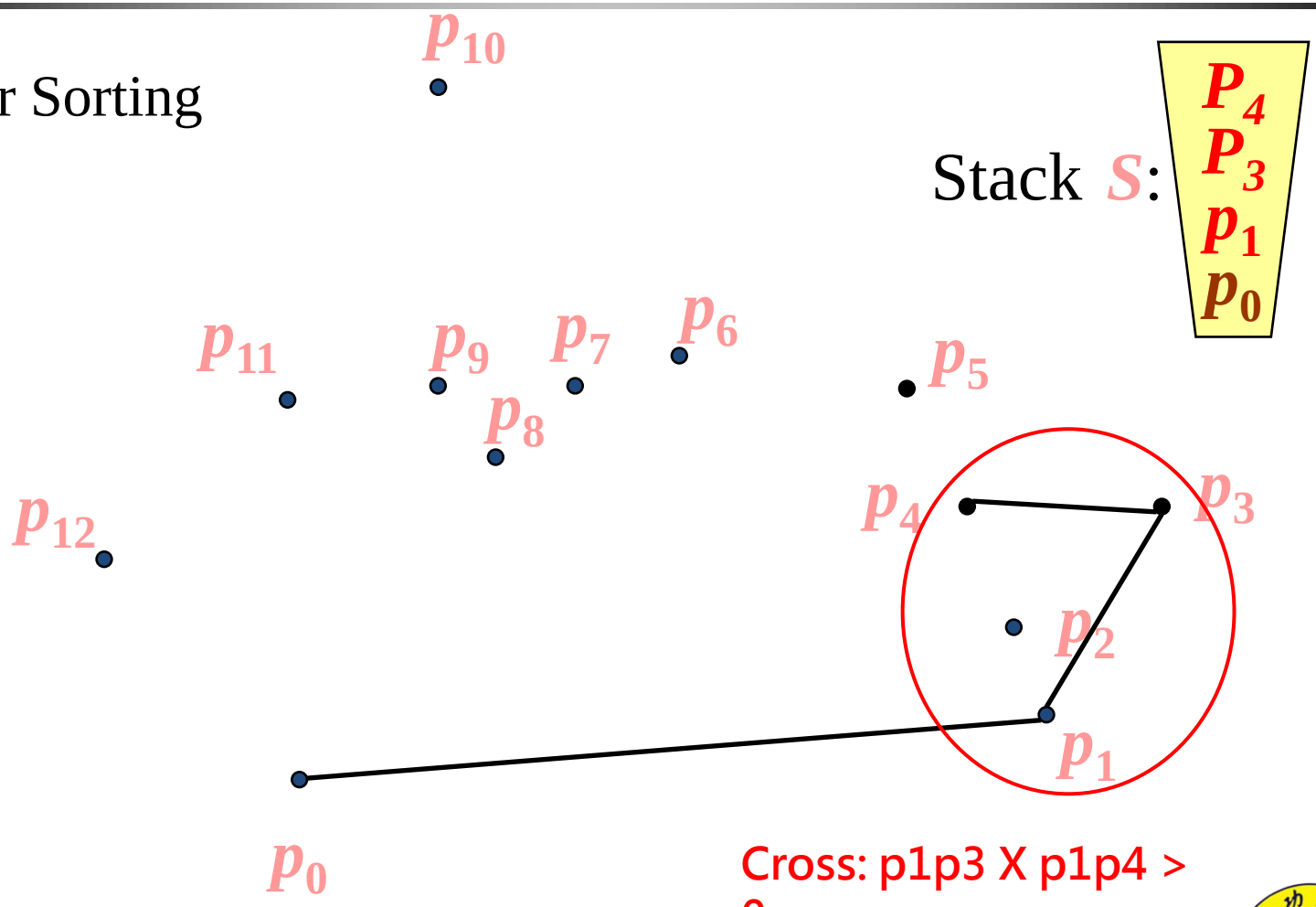
- Polar Sorting



made by electron & free999

Convex Hull

- Polar Sorting

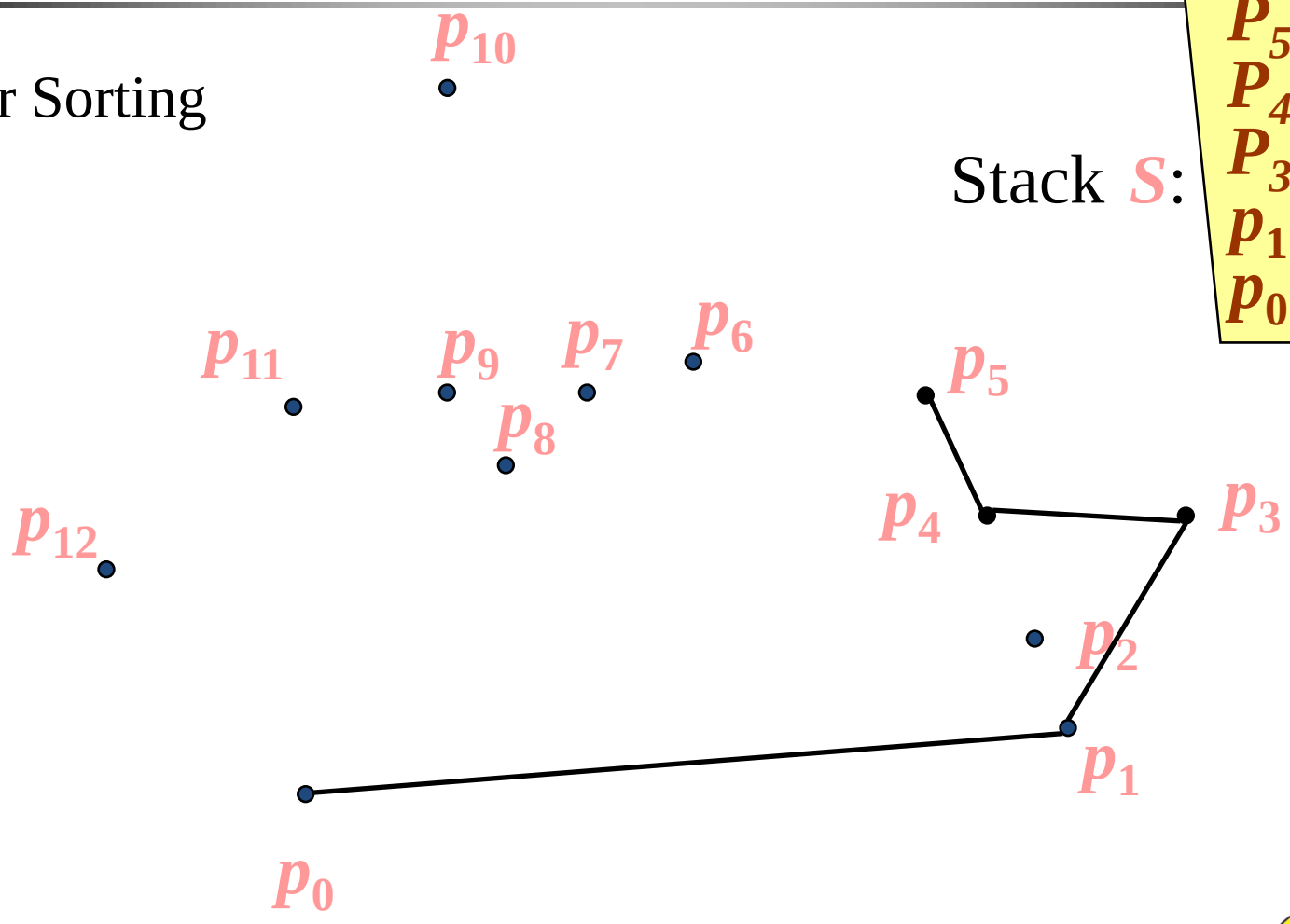


Cross: p1p3 X p1p4 > 0

made by electron & free999

Convex Hull

- Polar Sorting

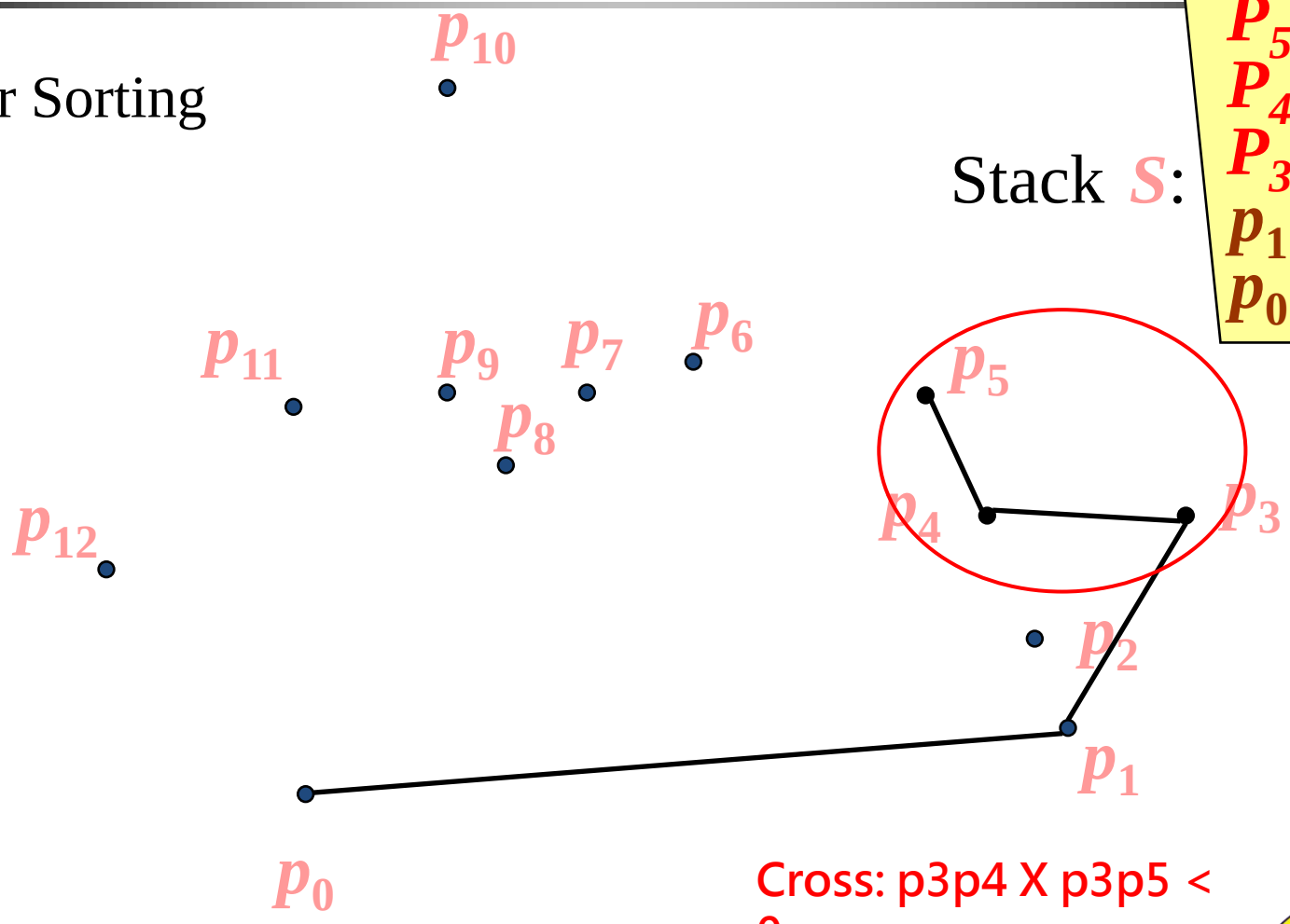


Stack S:

P_5
P_4
P_3
p_1
p_0

Convex Hull

- Polar Sorting

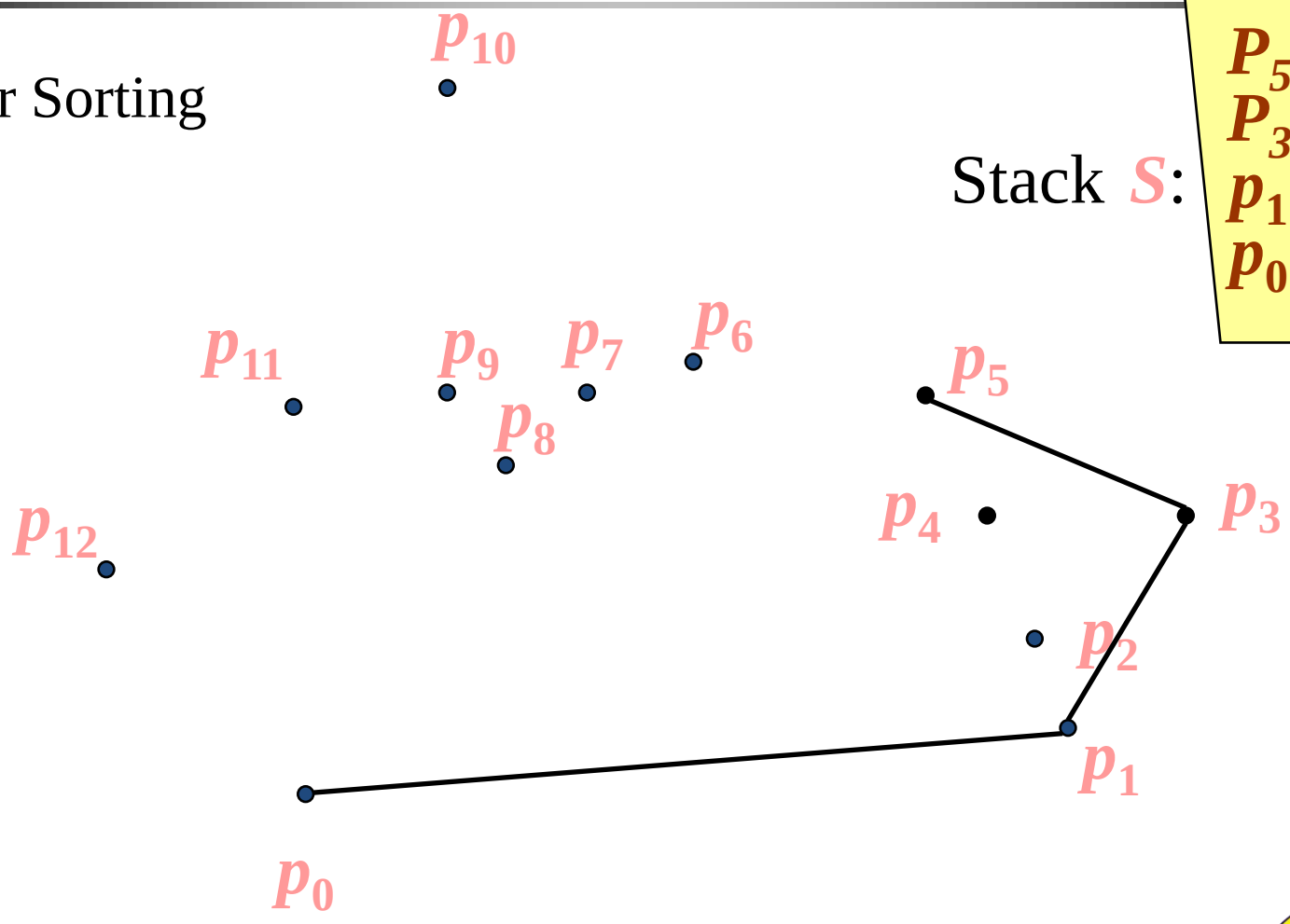


Cross: p3p4 X p3p5 < 0

made by electron & free999

Convex Hull

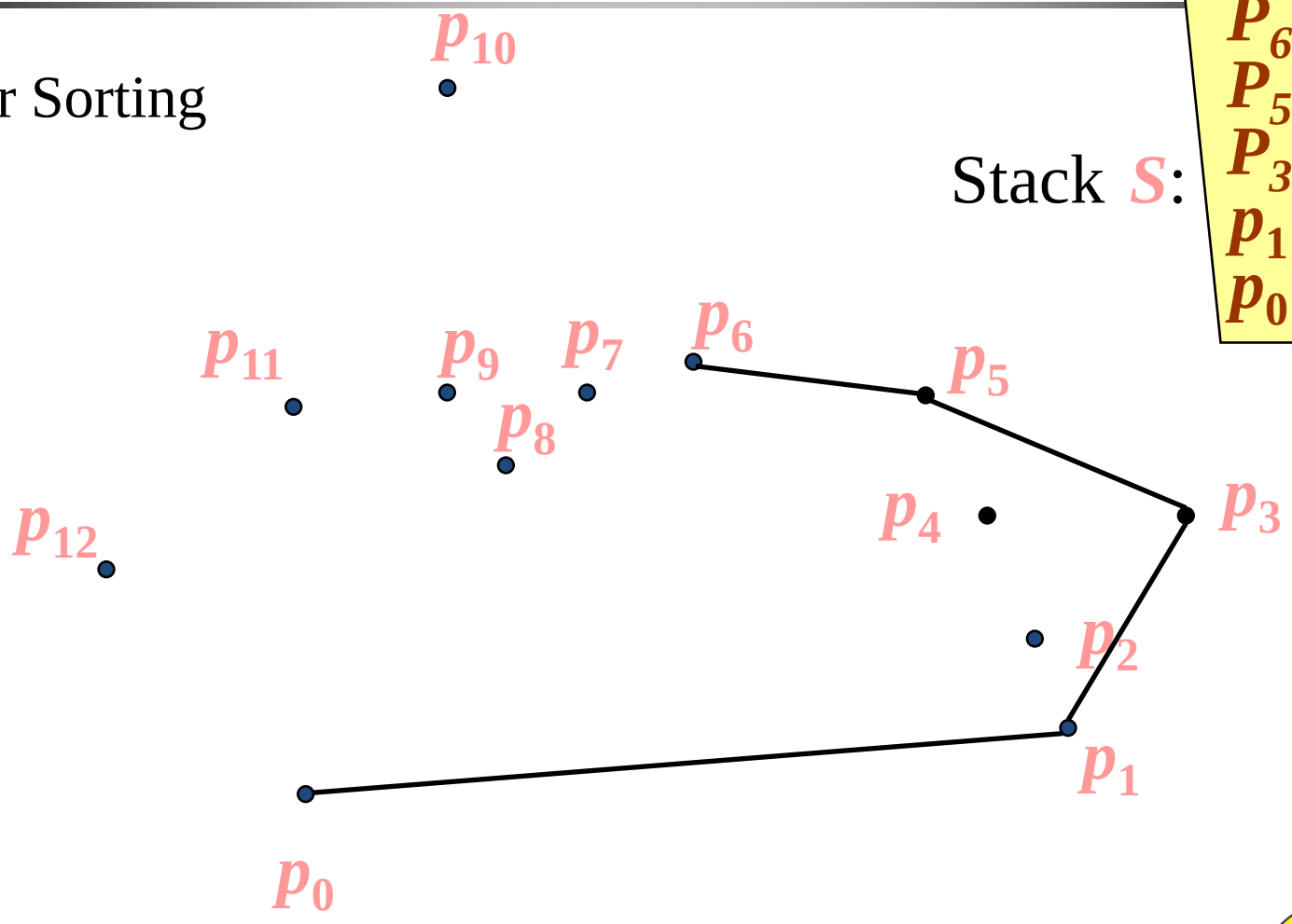
- Polar Sorting



made by electron & free999

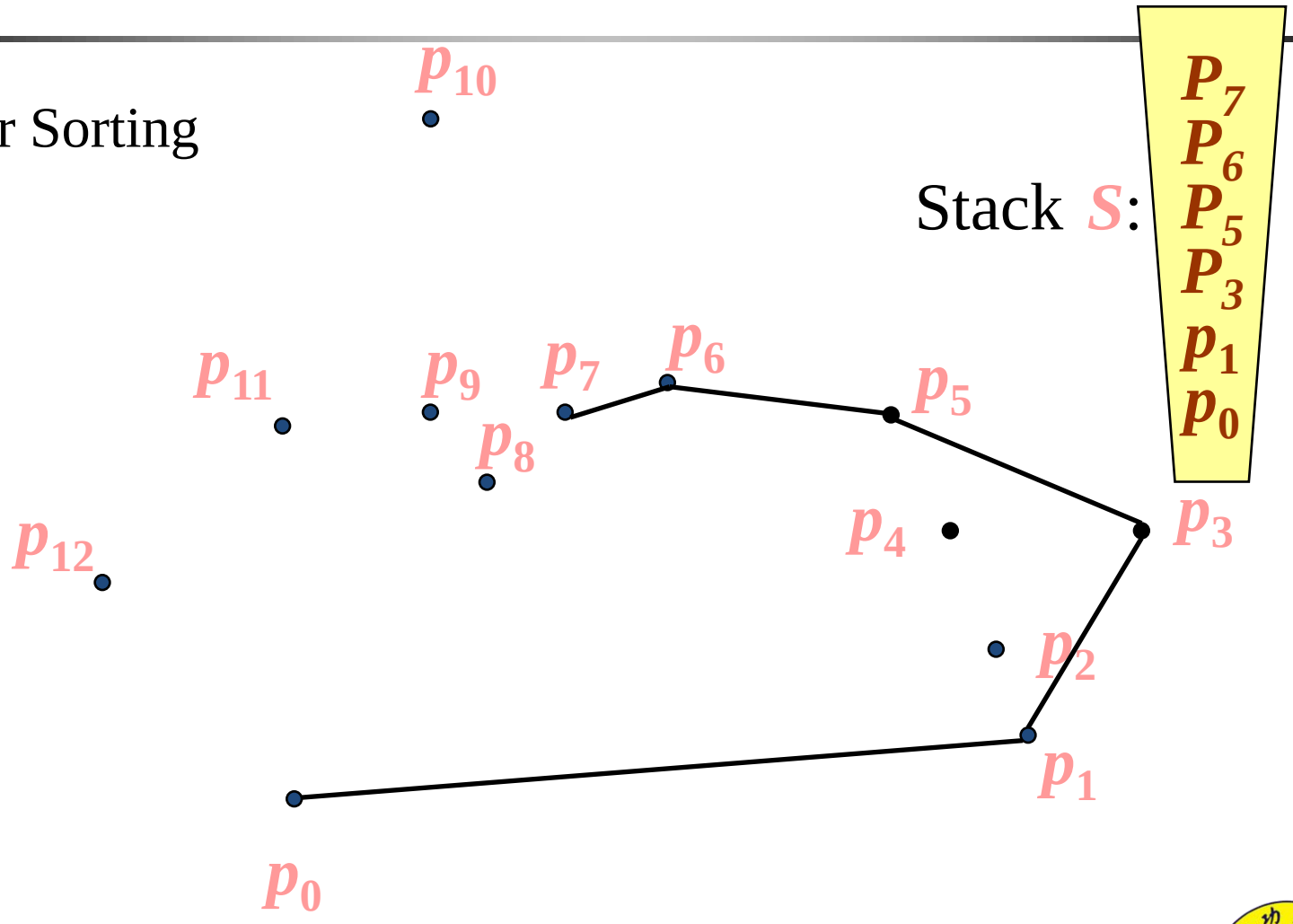
Convex Hull

- Polar Sorting



Convex Hull

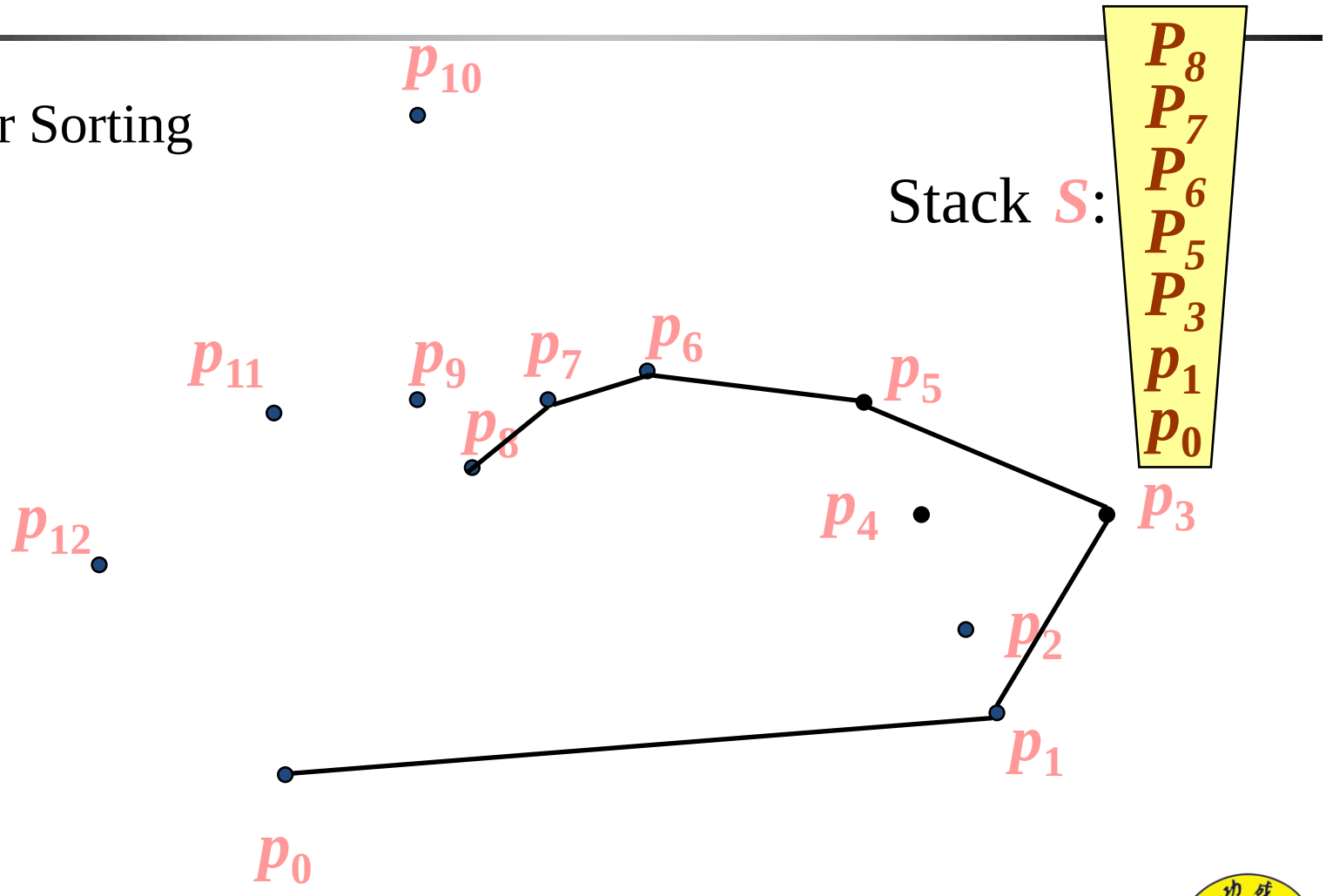
- Polar Sorting



made by electron & free999

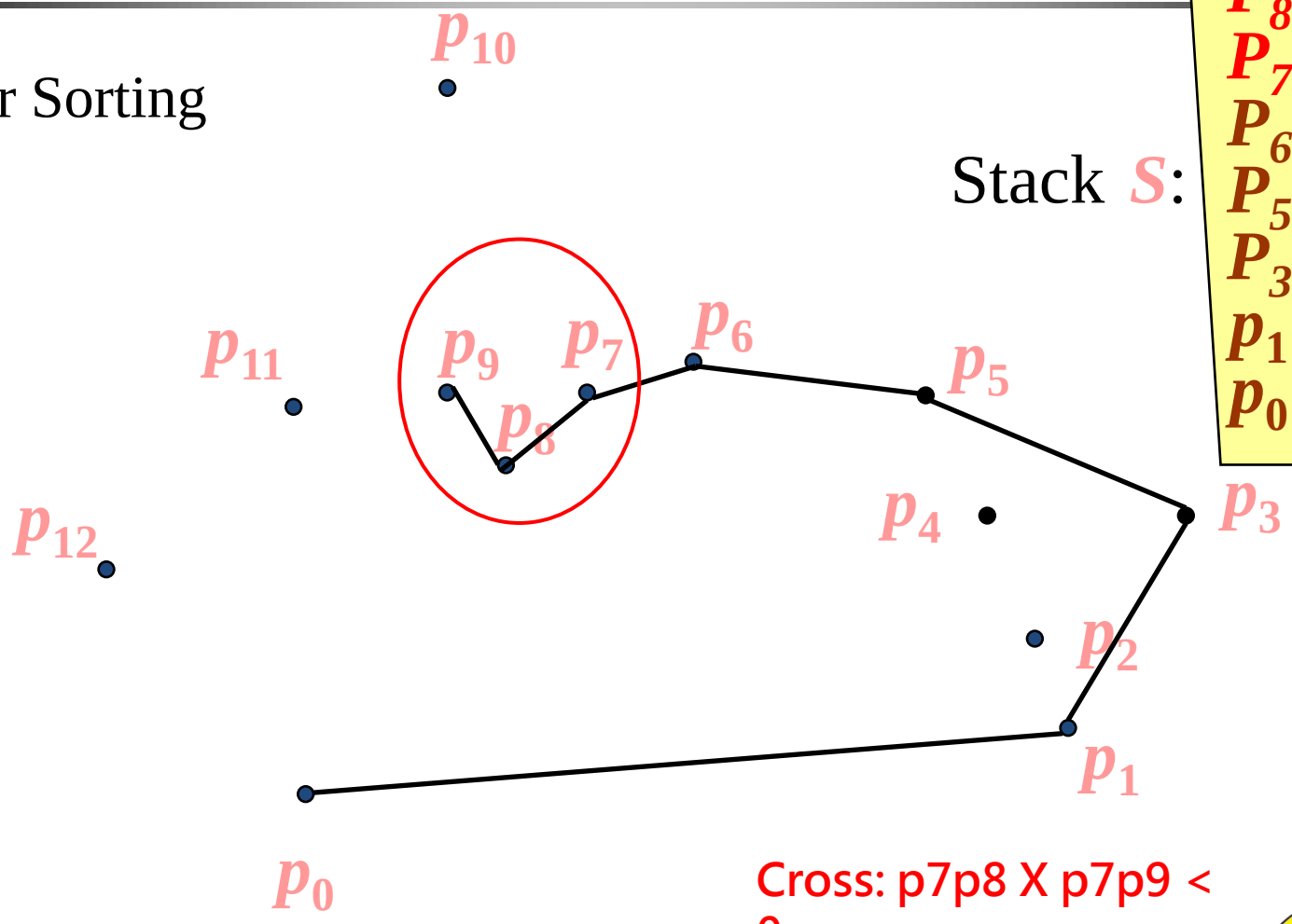
Convex Hull

- Polar Sorting



Convex Hull

- Polar Sorting

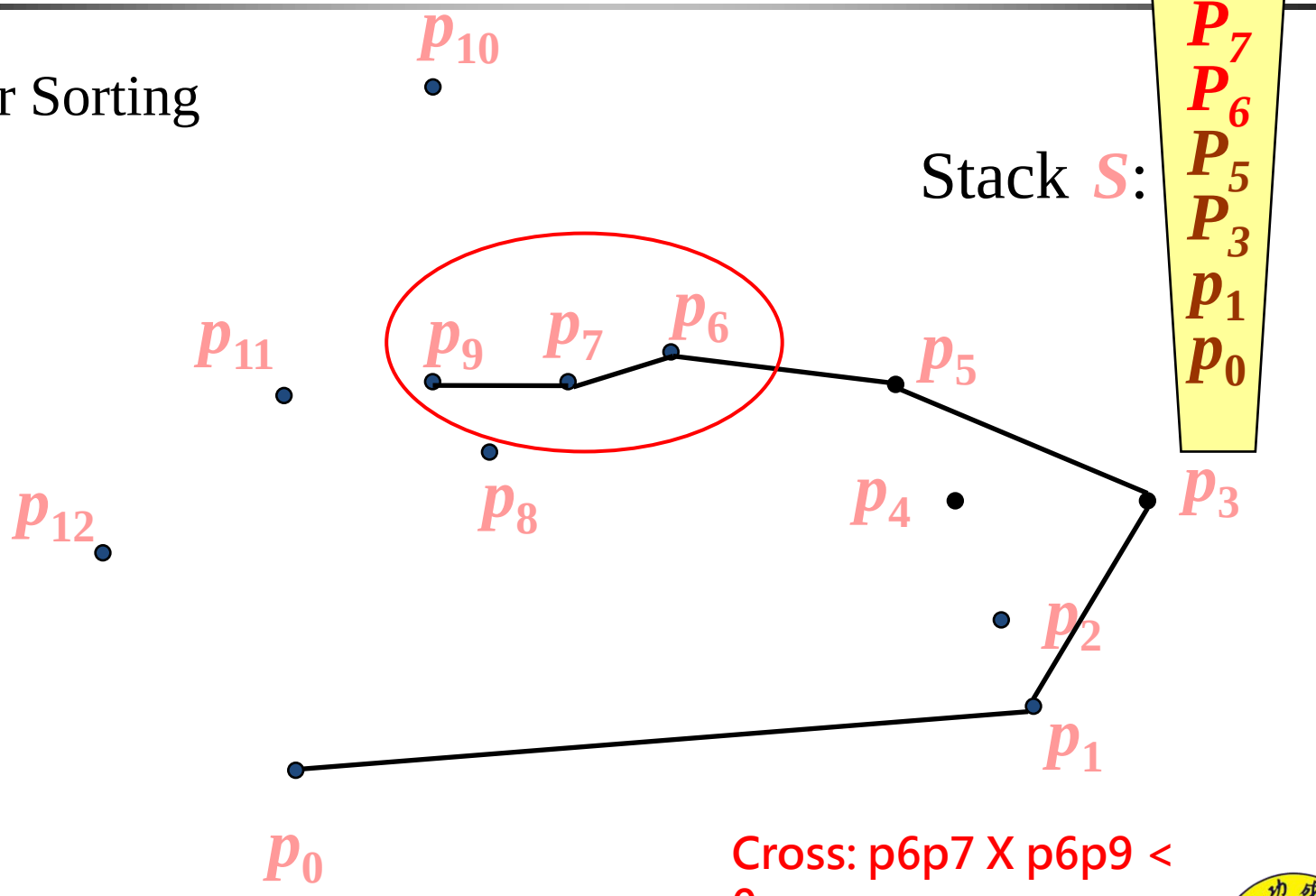


Cross: p7p8 X p7p9 < 0

made by electron & free999

Convex Hull

- Polar Sorting

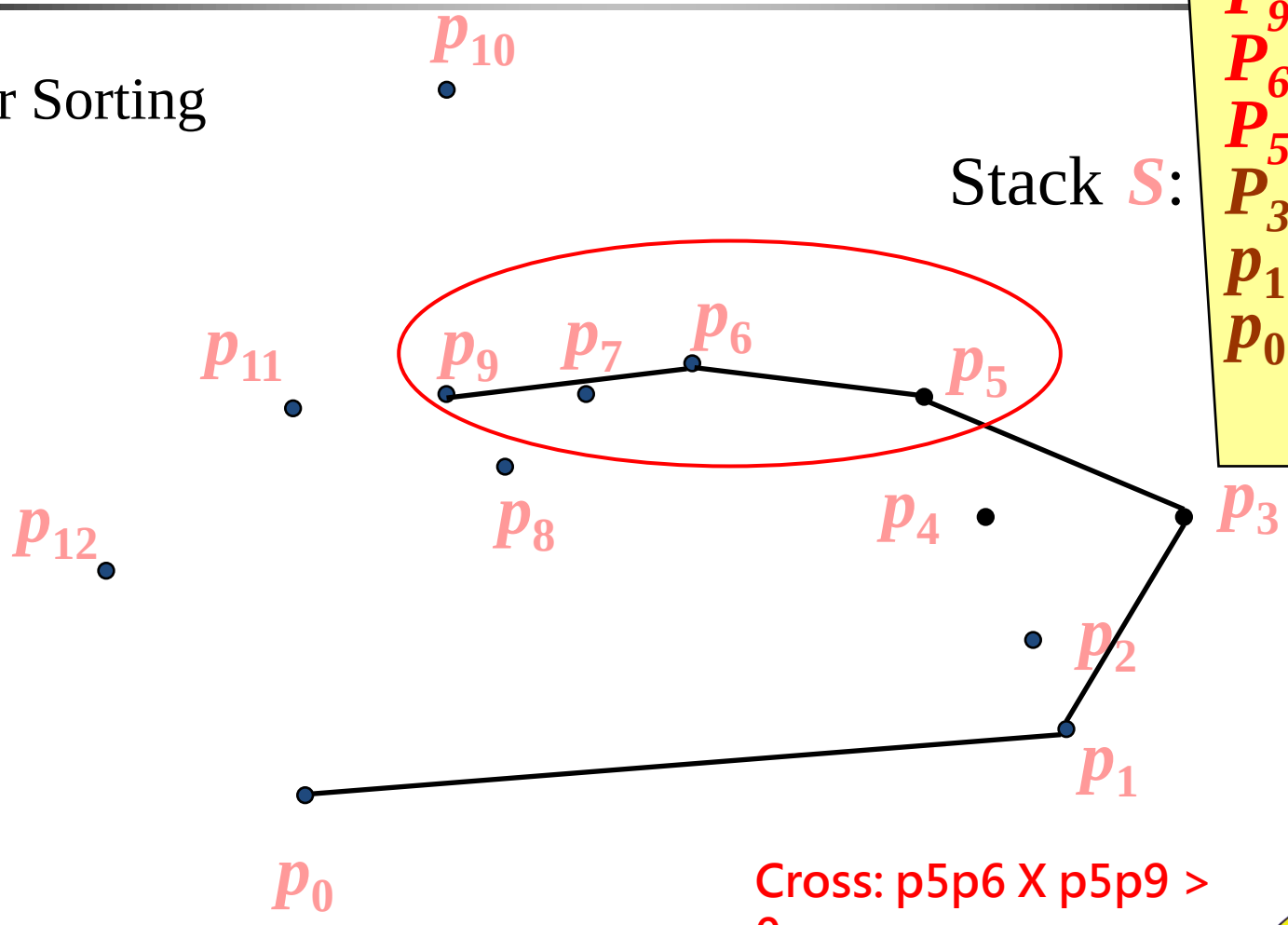


Cross: p6p7 X p6p9 < 0

made by electron & free999

Convex Hull

- Polar Sorting

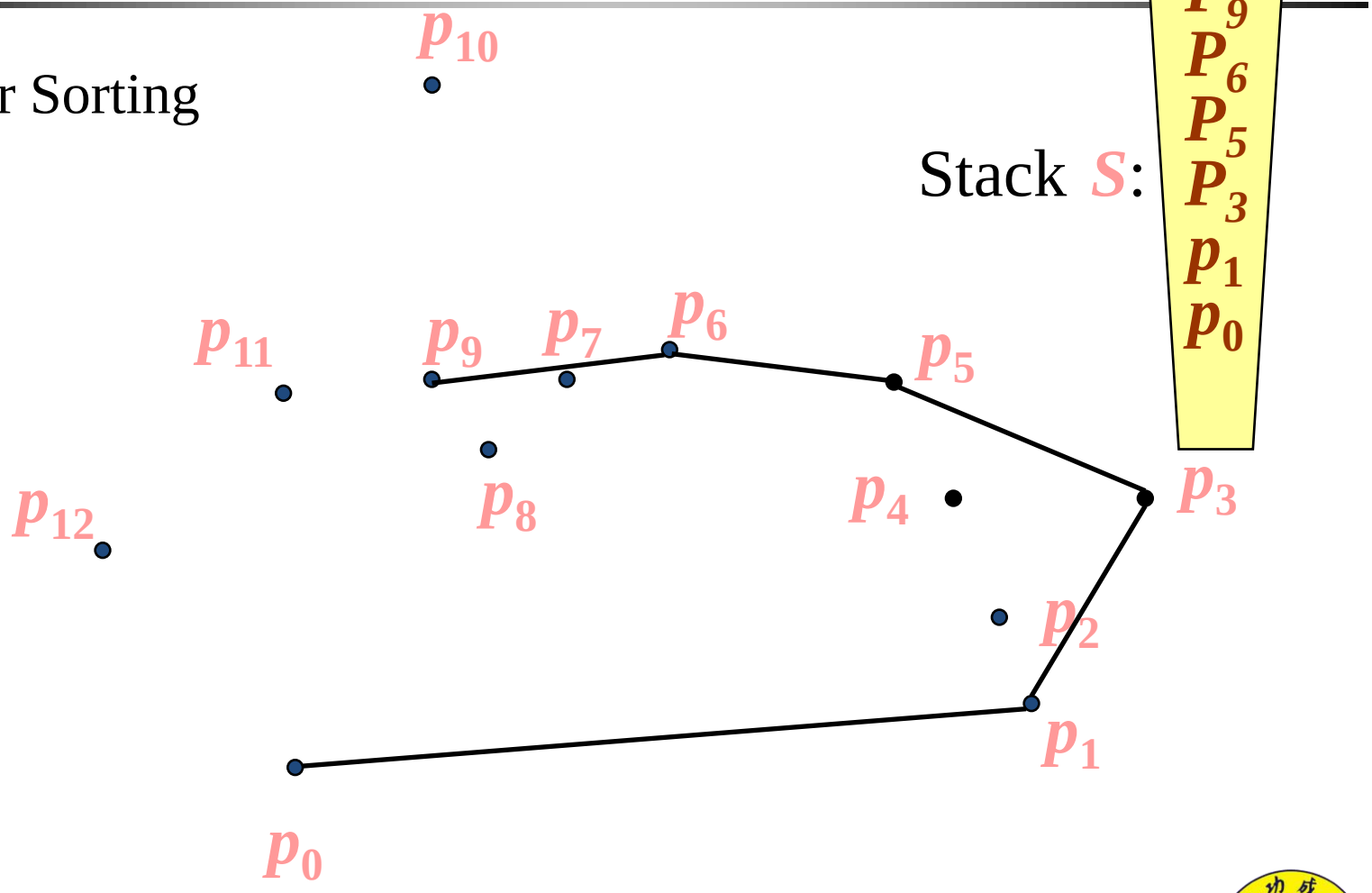


Cross: p5p6 X p5p9 > 0

made by electron & free999

Convex Hull

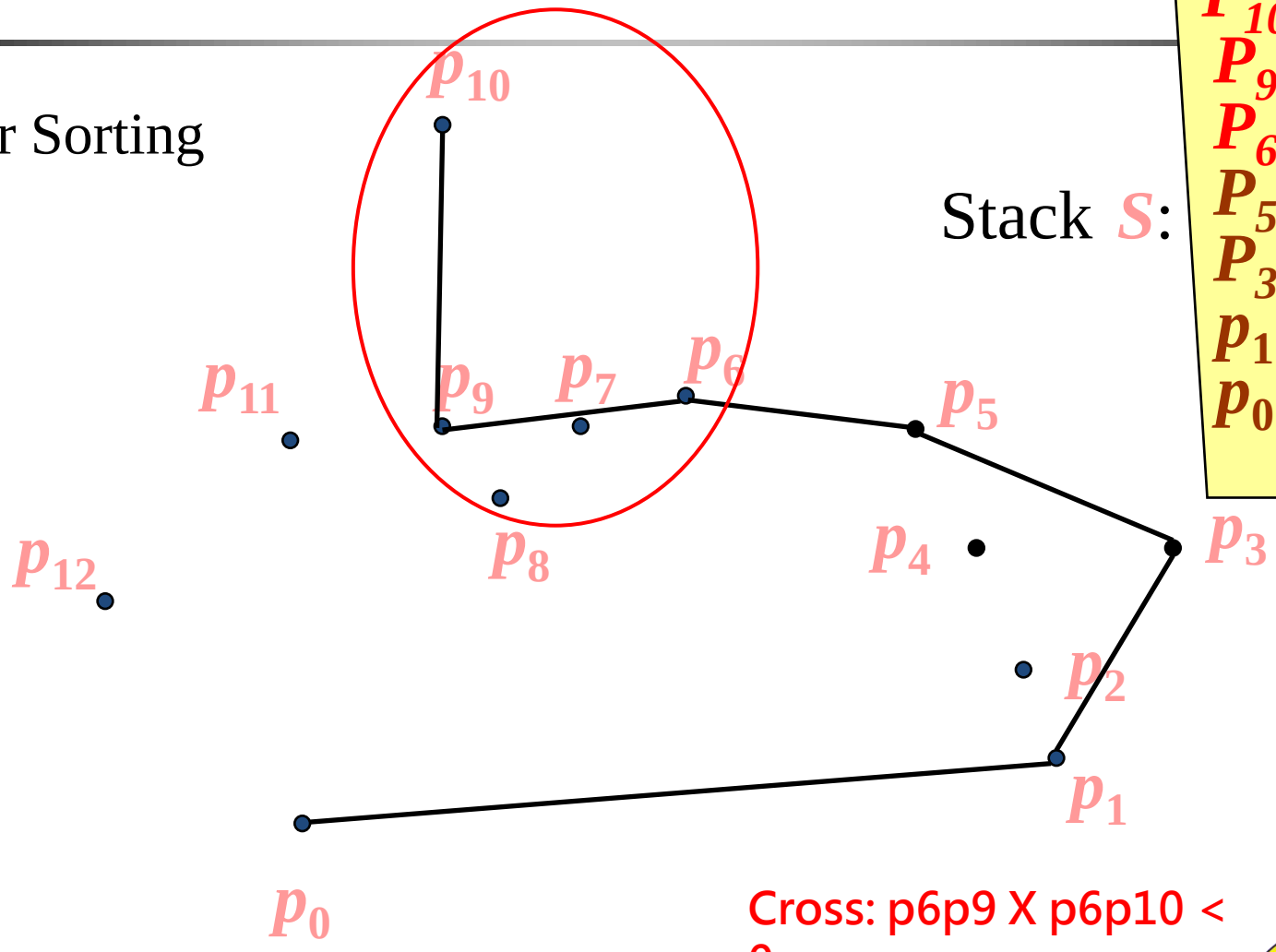
- Polar Sorting



made by electron & free999

Convex Hull

- Polar Sorting



Stack S:

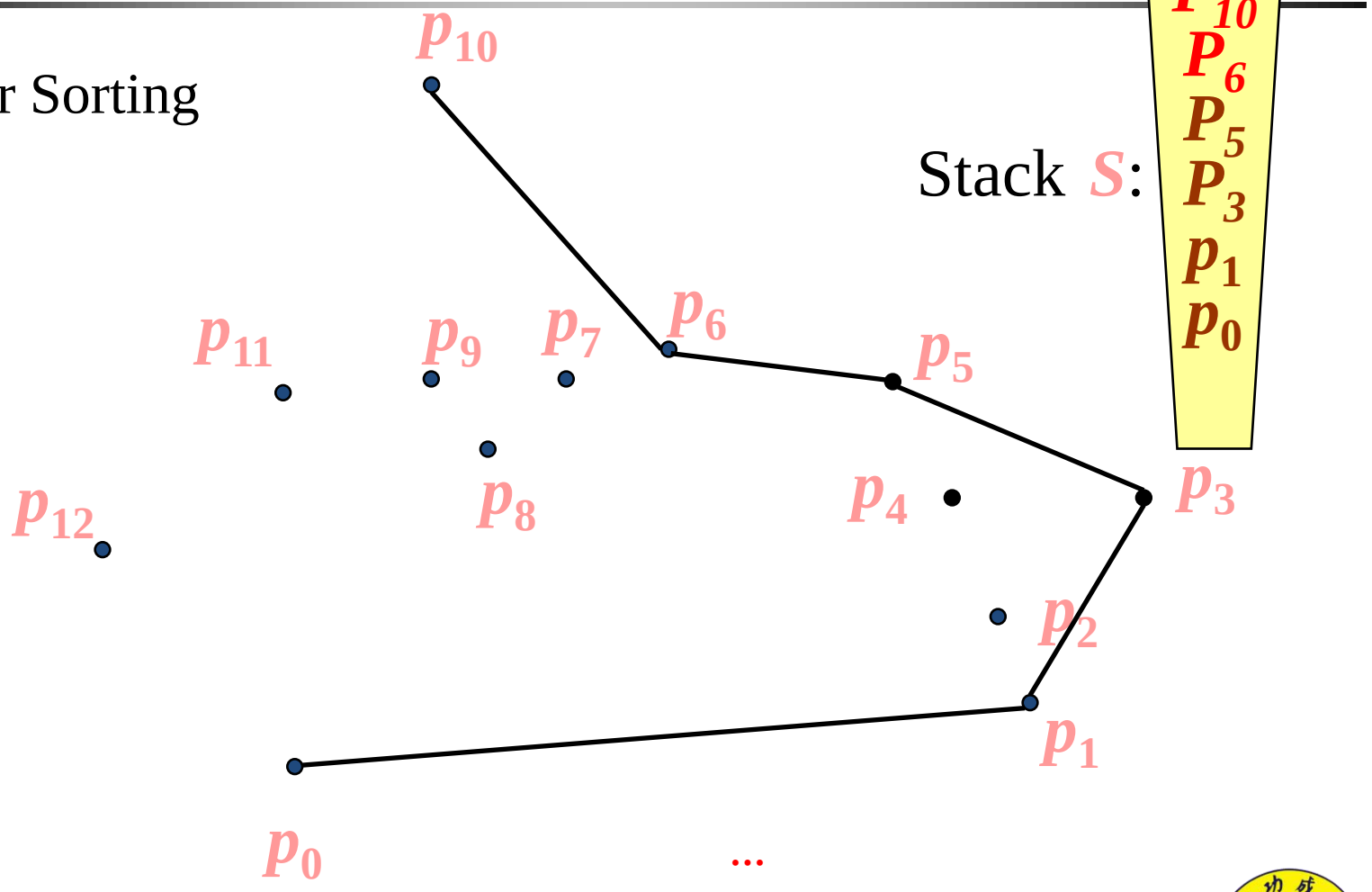
P_{10}
P_9
P_6
P_5
P_3
p_1
p_0

Cross: p6p9 X p6p10 < 0

made by electron & free999

Convex Hull

- Polar Sorting



made by electron & free999

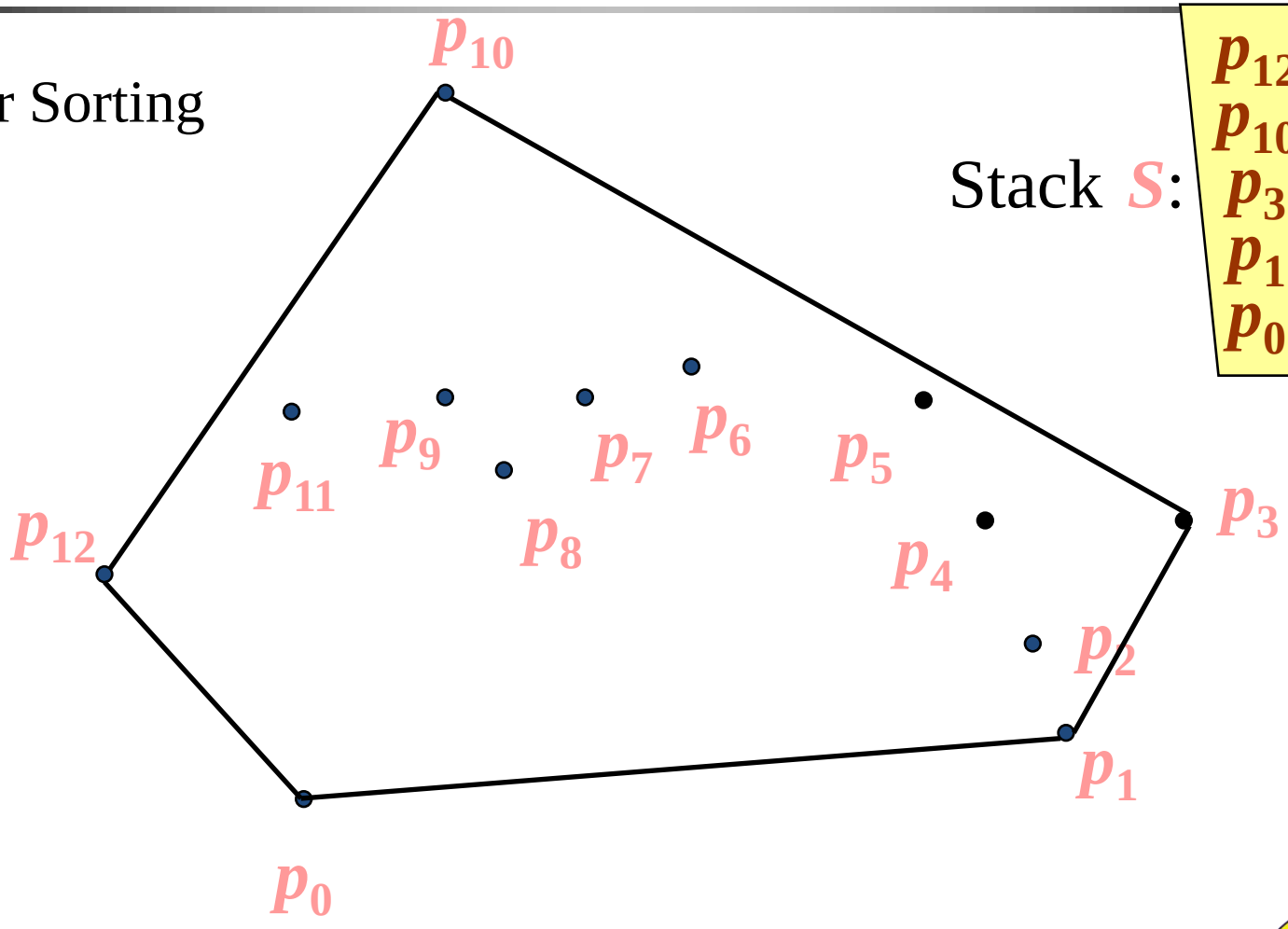
Convex Hull

. . .

made by electron & free999

Convex Hull

- Polar Sorting



Stack S:

p_{12}
p_{10}
p_3
p_1
p_0

made by electron & free999

Algorithm

Graham-Scan(*Q*)

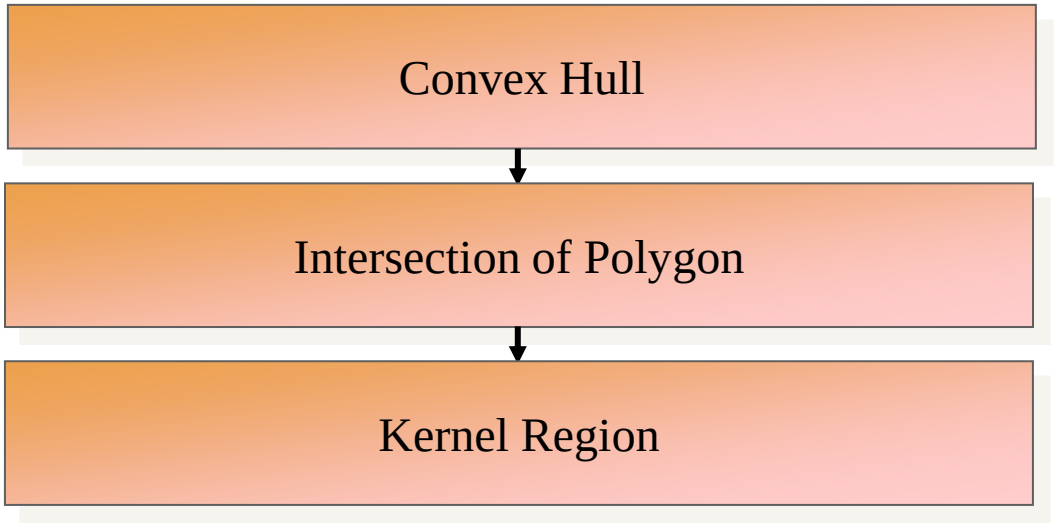
```
1. p0 = the lowest-left point of Q
2. Sort by count-clockwise direction ( p0pa X p0pb > 0  )
3. Assign p0, p1, p2 to Stack
4. For (i=3; i<=N; i++)
   {
        while( p[top-1]p[top] X p[top-1]p[i] <=0 )
             stack.pop();
        stack.push( p[i] )
   }
```

Practice

PKU: 2007

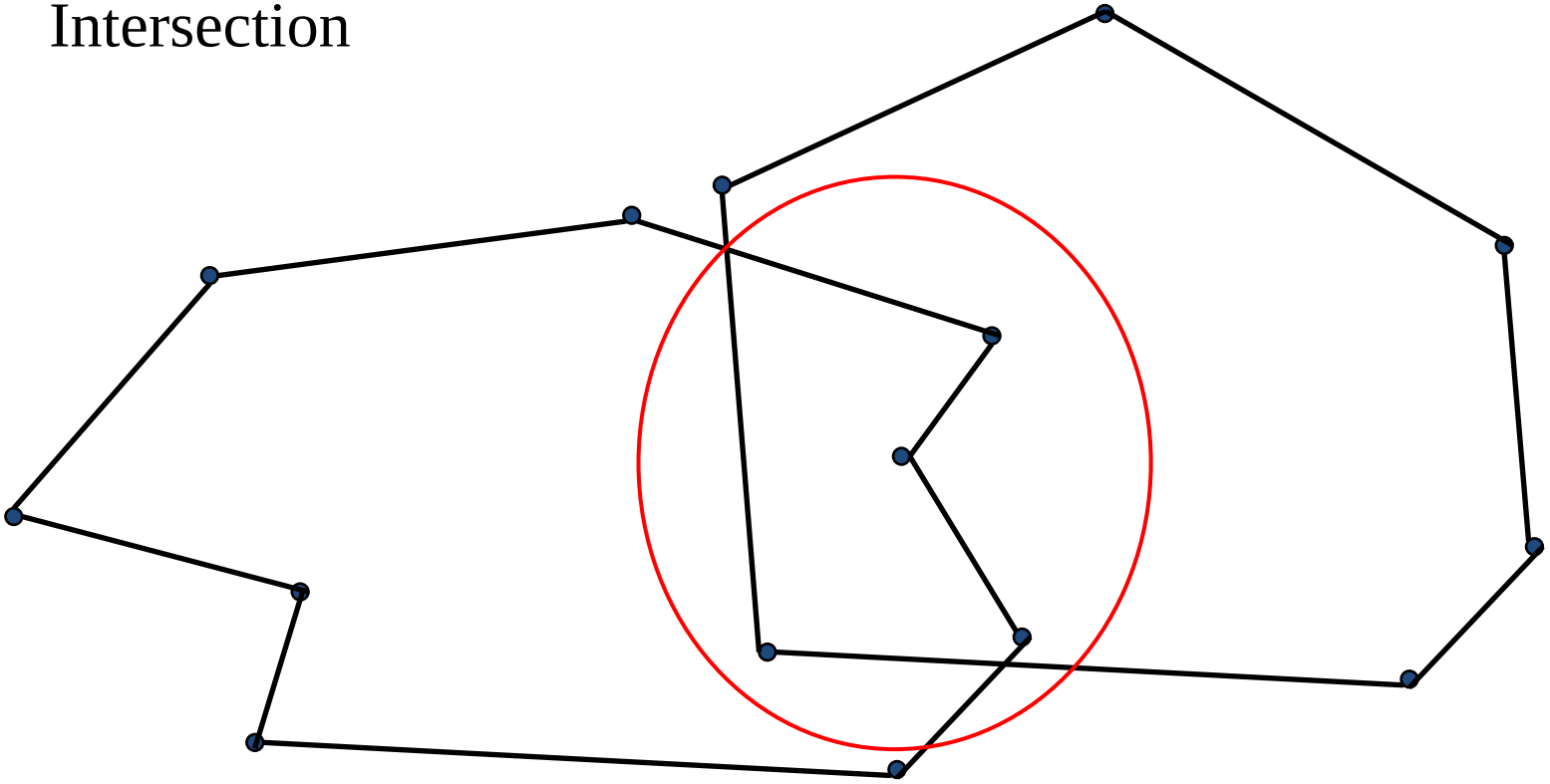
PKU: 1113

Outline



Intersection

- Intersection



made by electron & free999

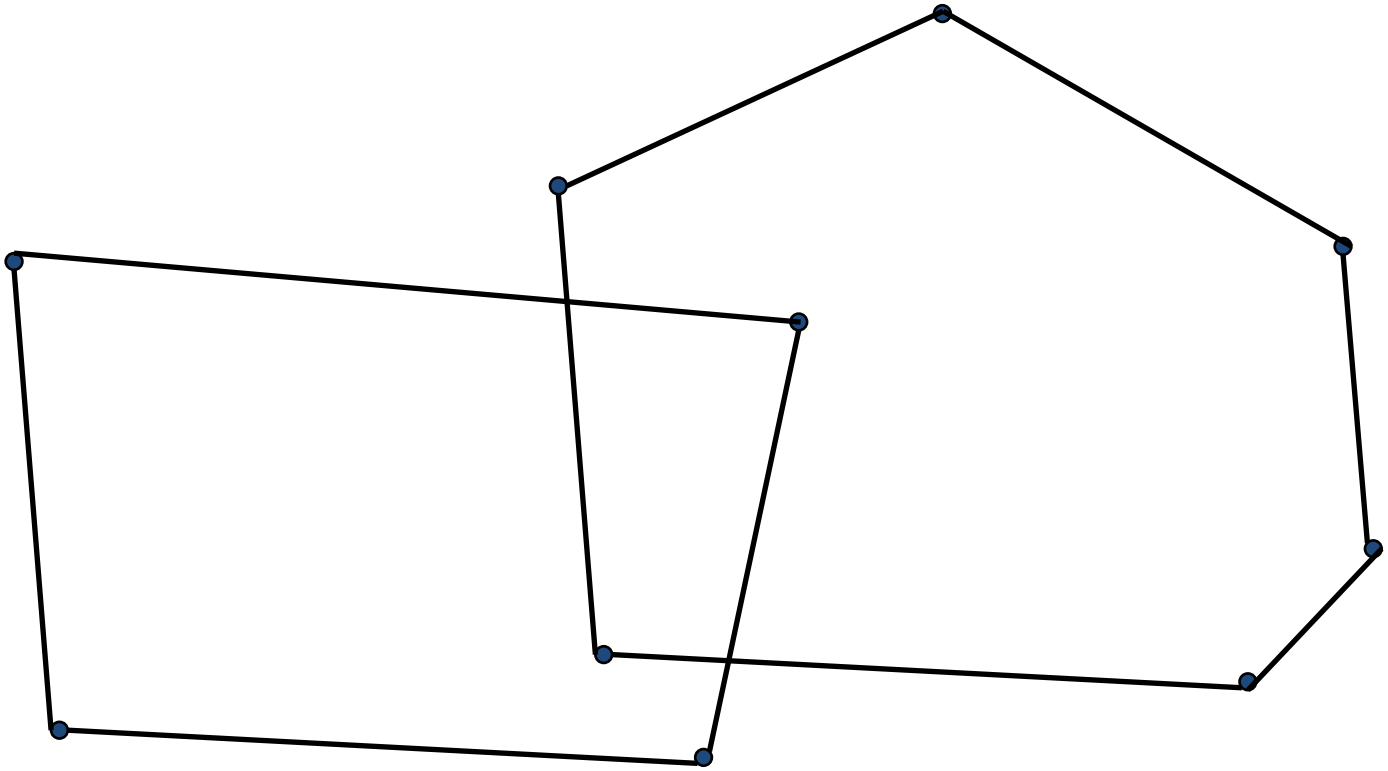
Intersection

- Review of line intersection

```
//找出兩條"線"的交點
POINT CheckTwoLine(LINE line1, LINE line2)                    //給定 2 個line
{
        POINT all[4];

        double a1, a2, b1, b2, c1, c2;
        double d, dx, dy;

        all[0].x = line1.p1.x, all[0].y = line1.p1.y;
        all[1].x = line1.p2.x, all[1].y = line1.p2.y;

        all[2].x = line2.p1.x, all[2].y = line2.p1.y;
        all[3].x = line2.p2.x, all[3].y = line2.p2.y;

        a1=all[0].y-all[1].y, a2=all[2].y-all[3].y;
        b1=all[1].x-all[0].x, b2=all[3].x-all[2].x;
        c1=all[1].x*all[0].y-all[1].y*all[0].x;
        c2=all[3].x*all[2].y-all[3].y*all[2].x;
        d=a1*b2-a2*b1;
        dx=c1*b2-c2*b1,             dy=a1*c2-a2*c1;
        POINT inter;
        inter.x = dx/d, inter.y = dy/d;
        return inter;
}
```

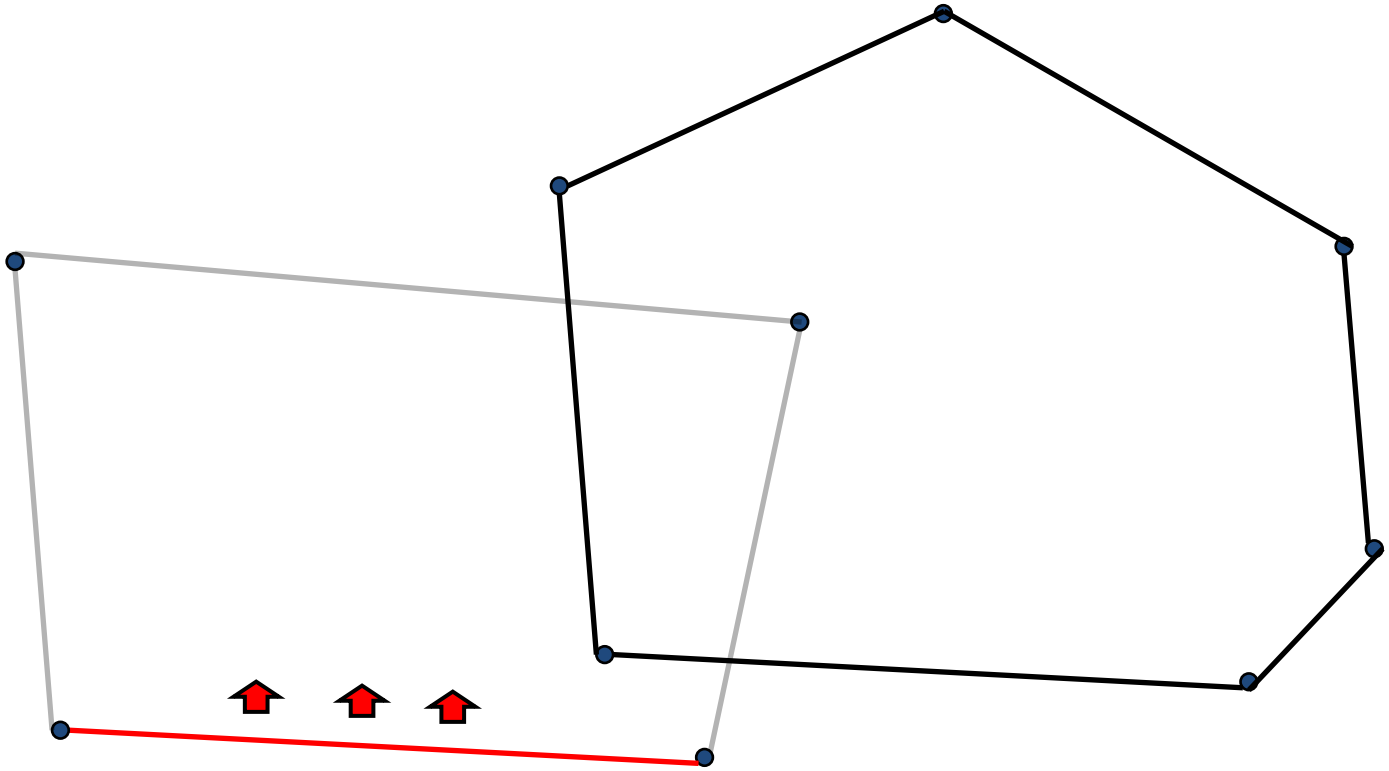

made by electron & free999

Intersection



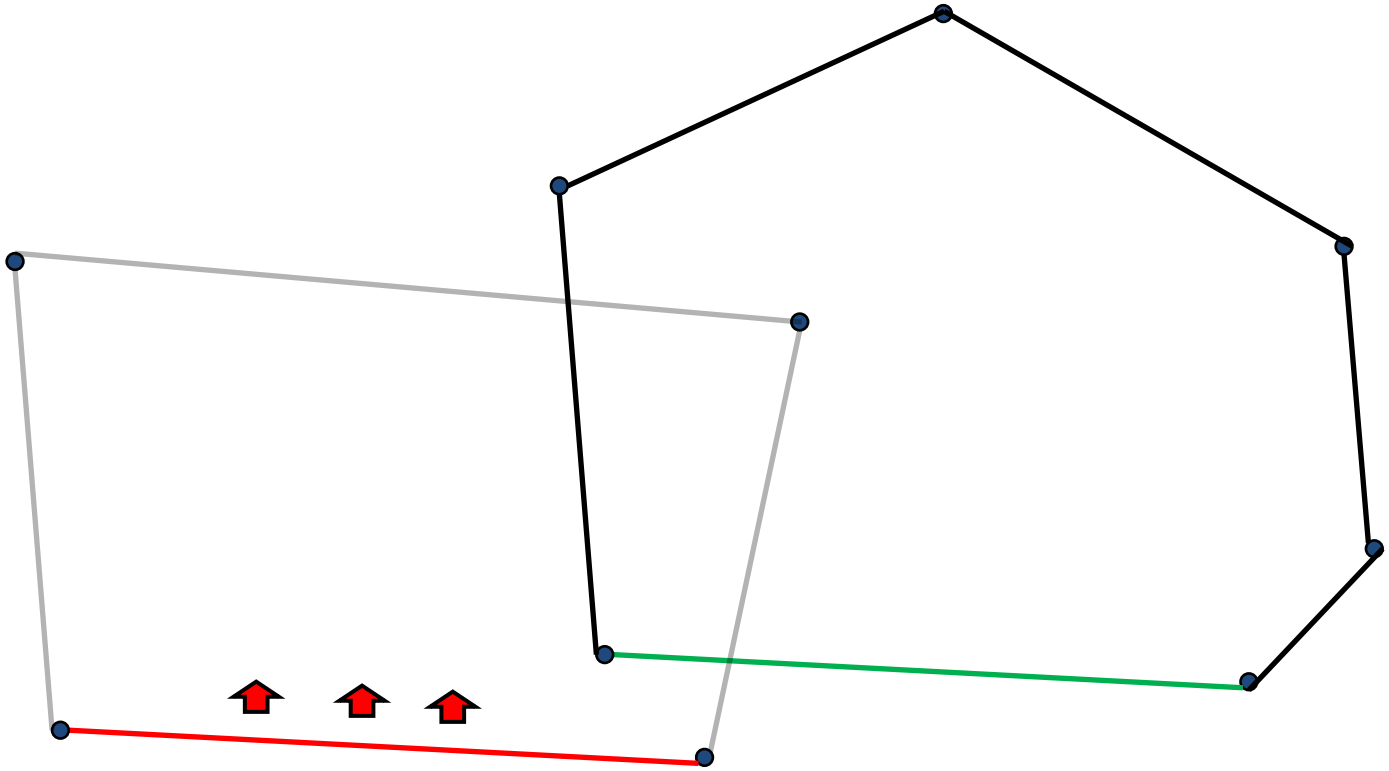
made by electron & free999

Intersection



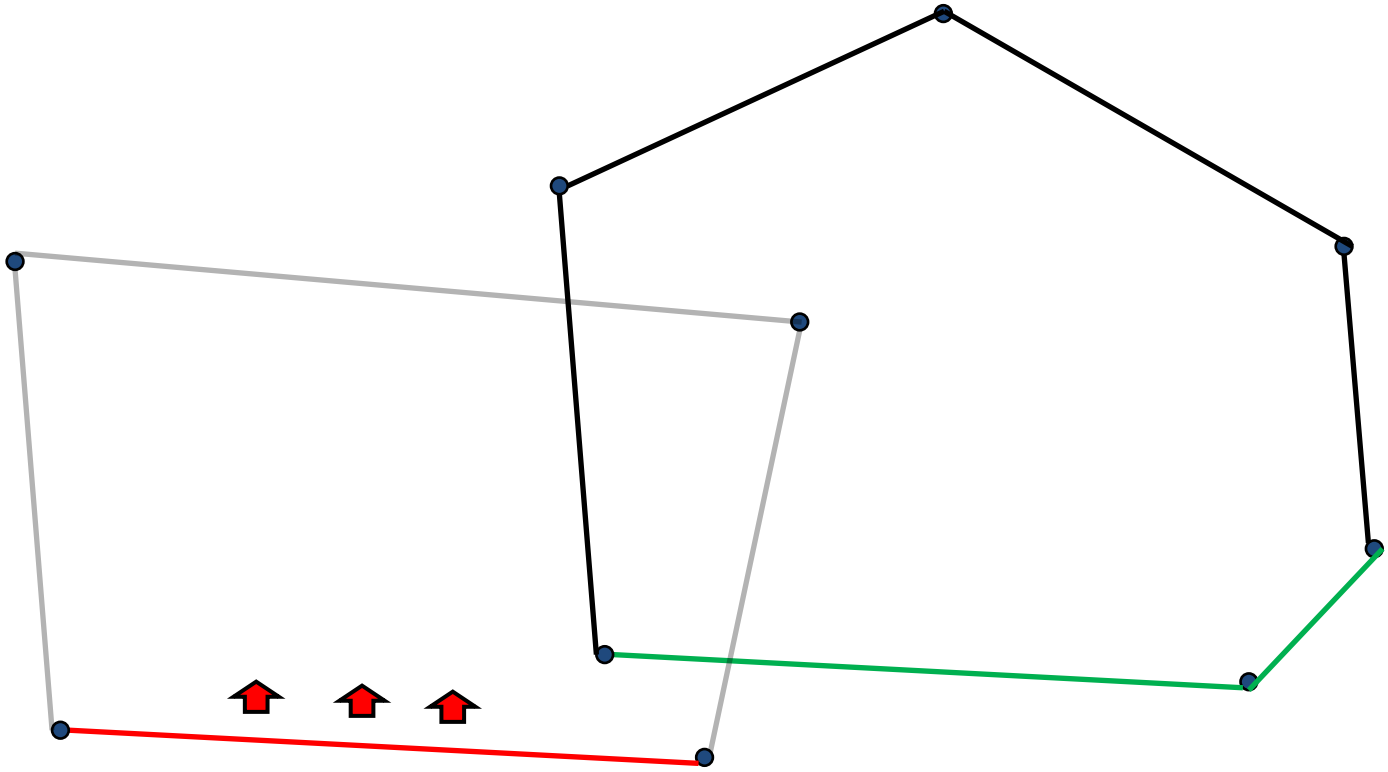
made by electron & free999

Intersection



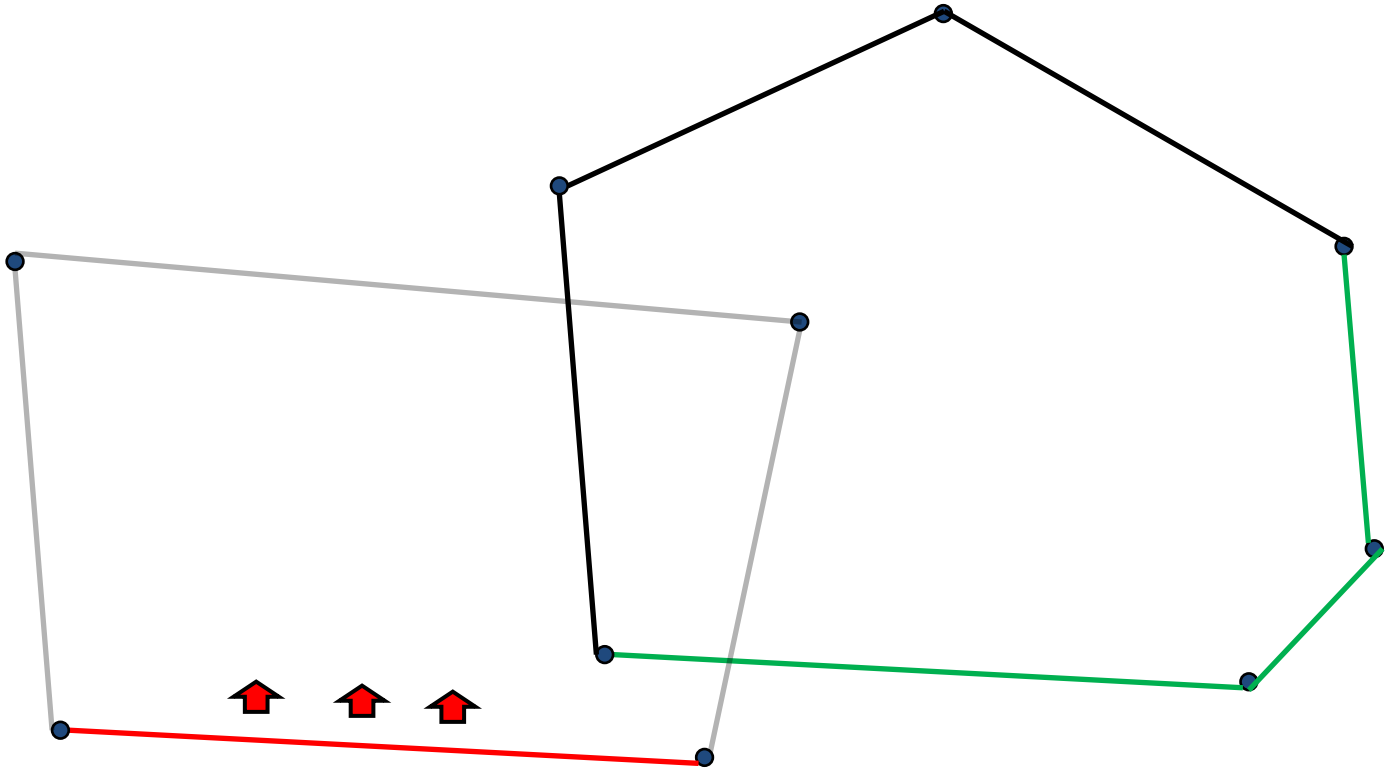
made by electron & free999

Intersection



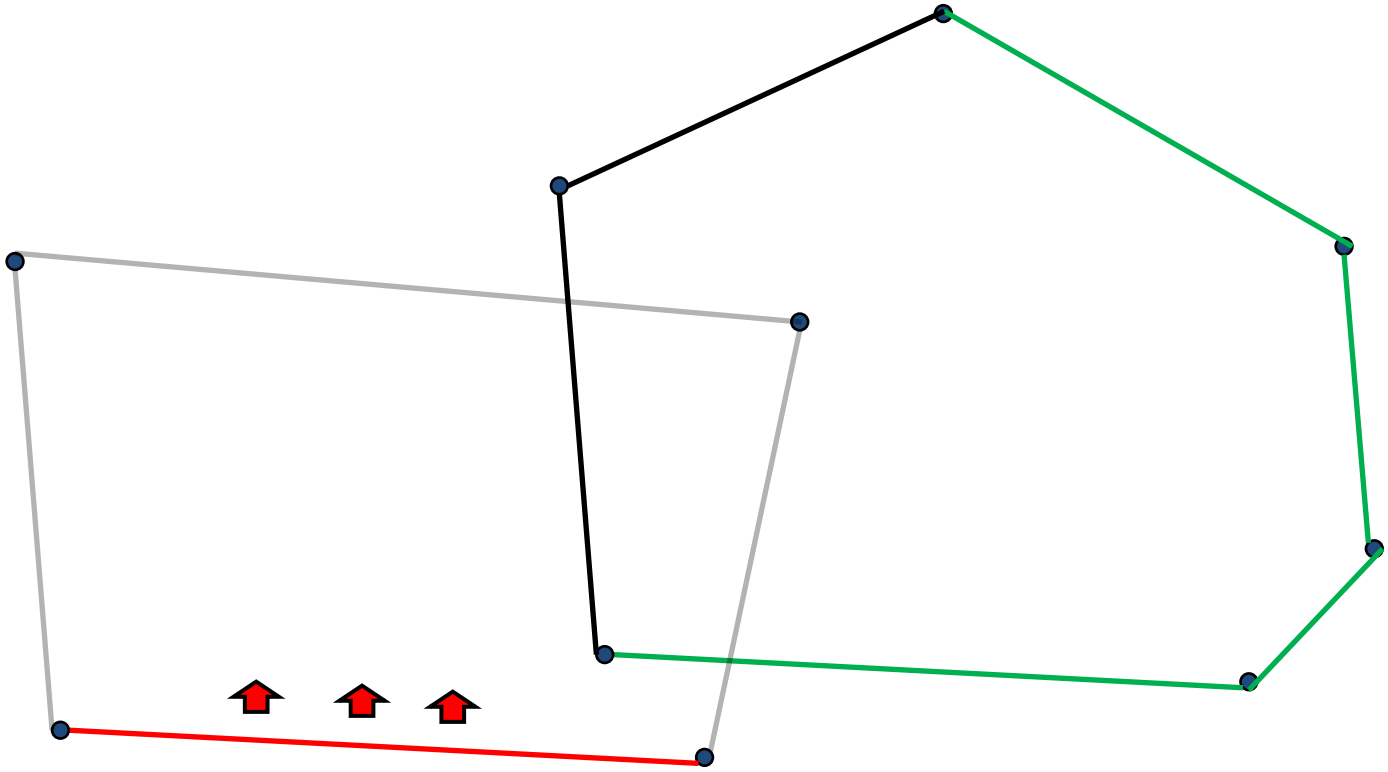
made by electron & free999

Intersection



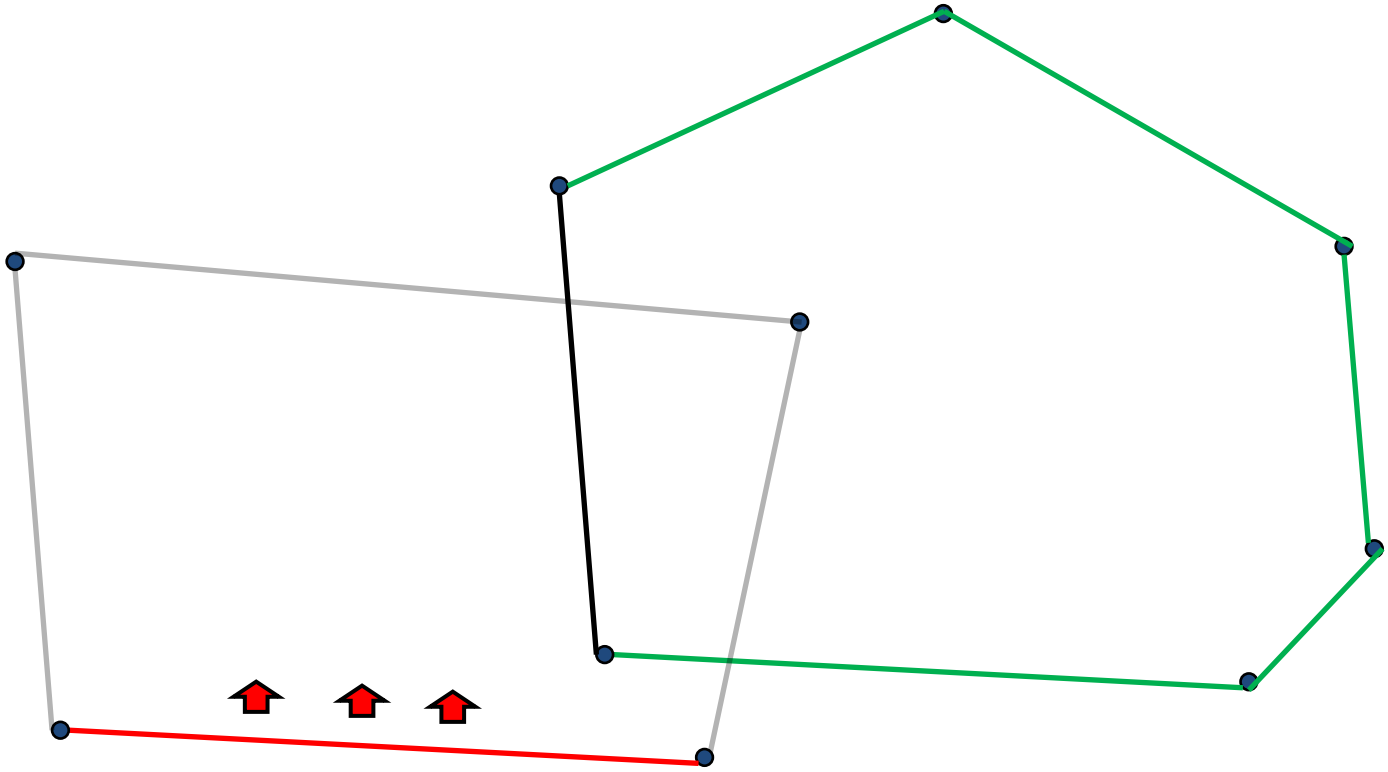
made by electron & free999

Intersection



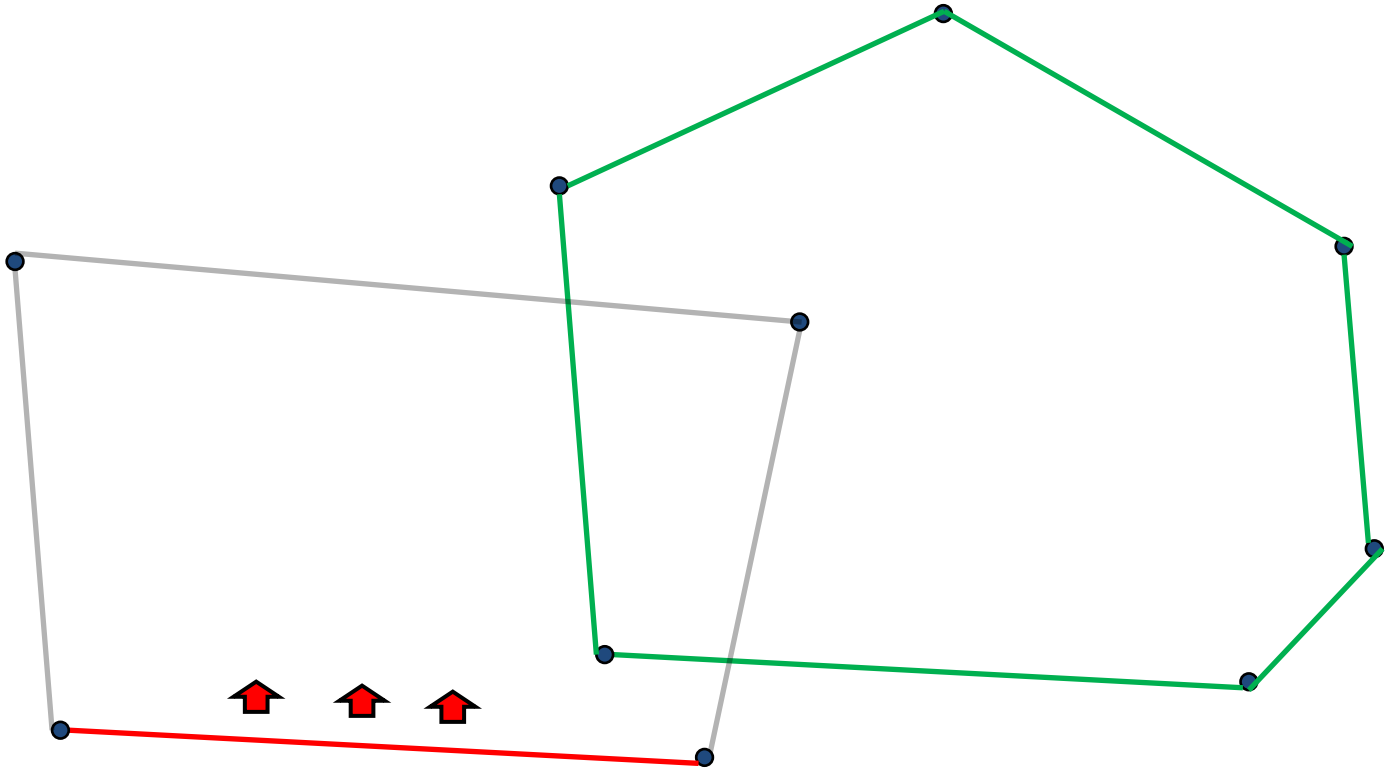
made by electron & free999

Intersection



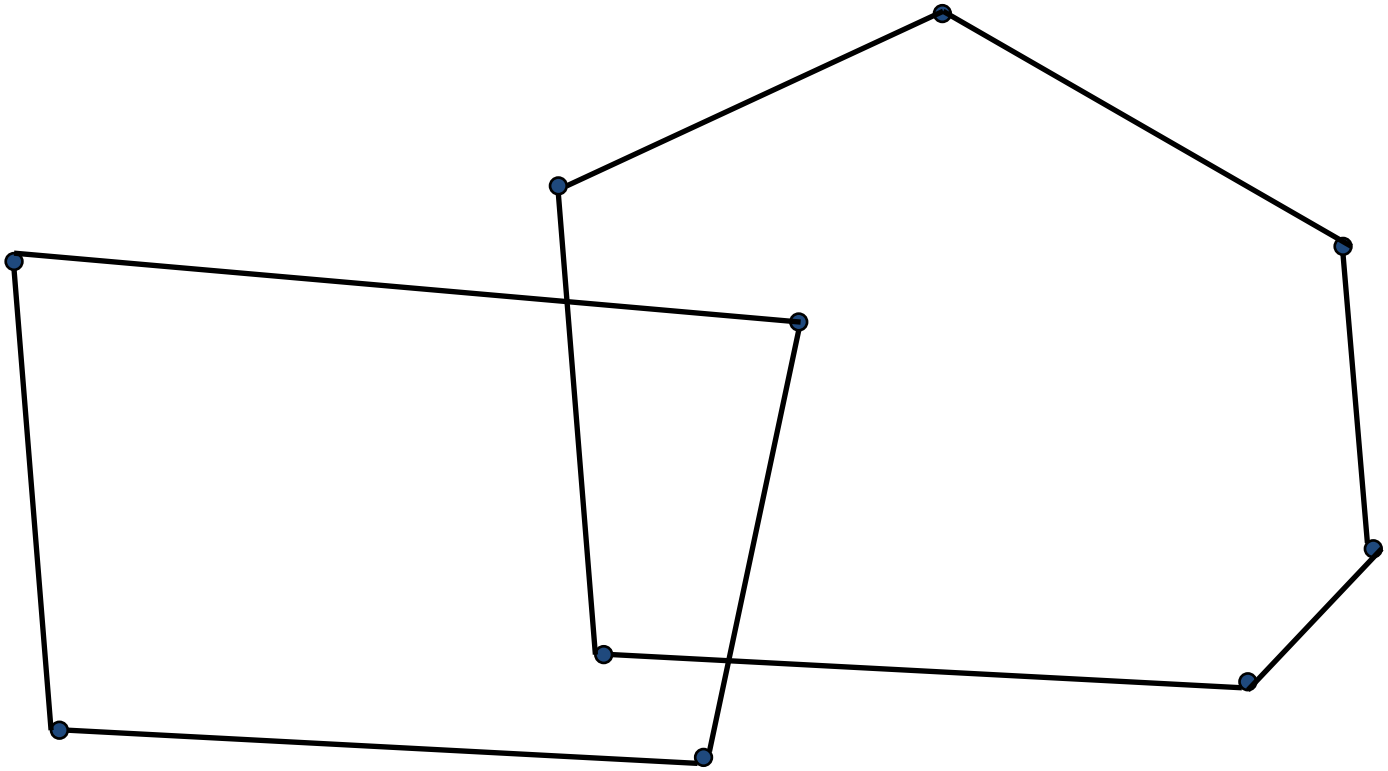
made by electron & free999

Intersection



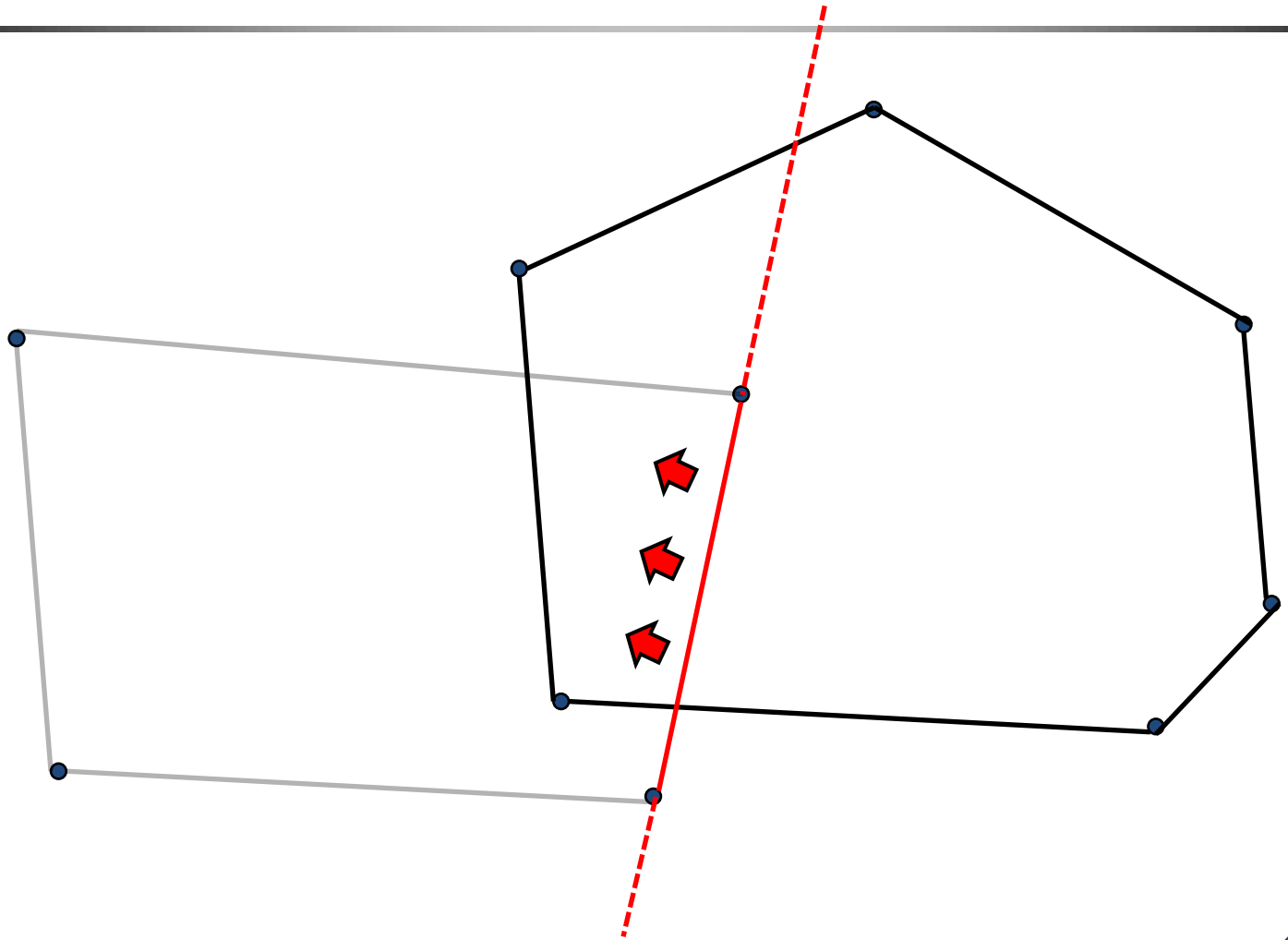
made by electron & free999

Intersection



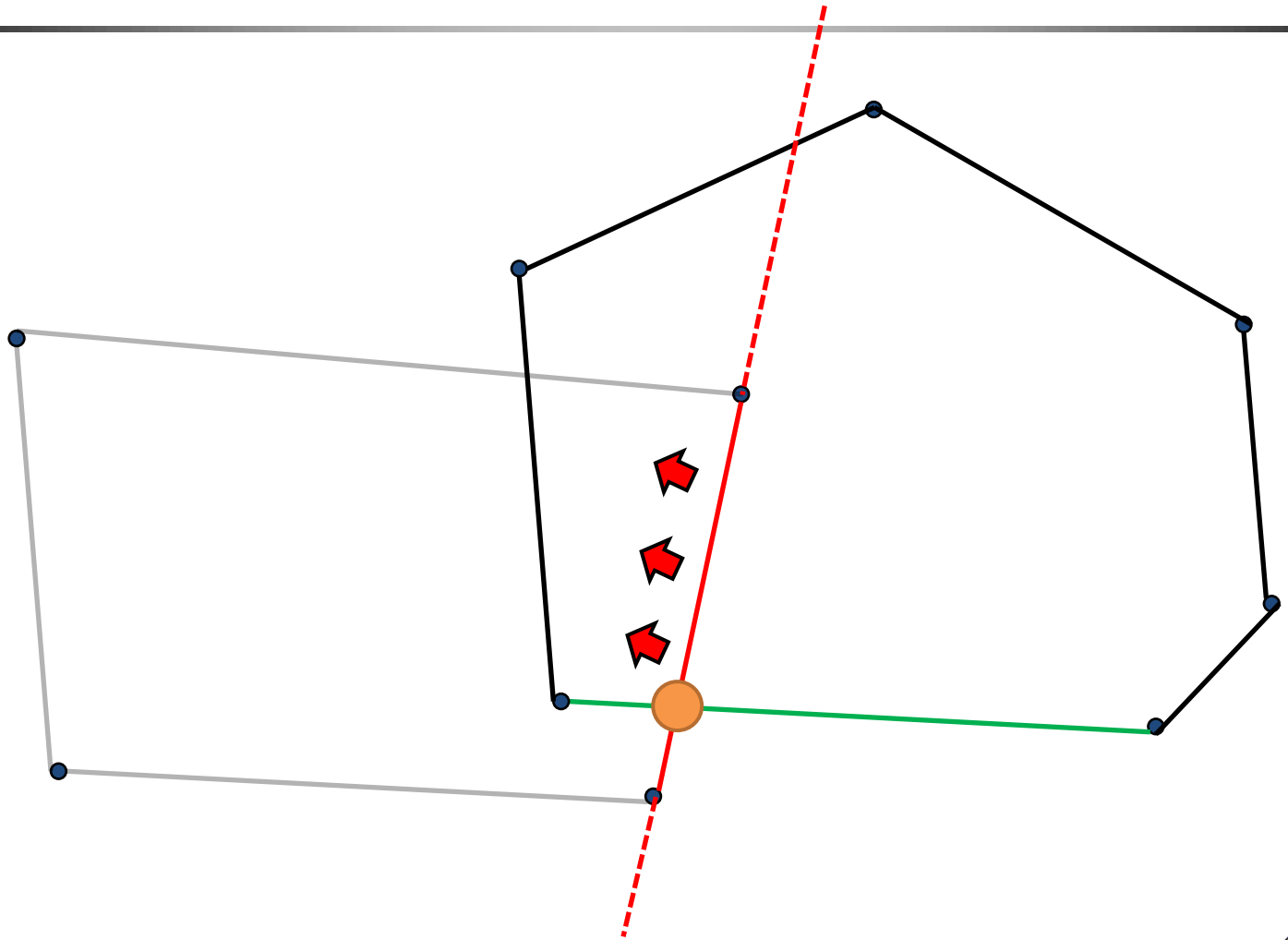
made by electron & free999

Intersection



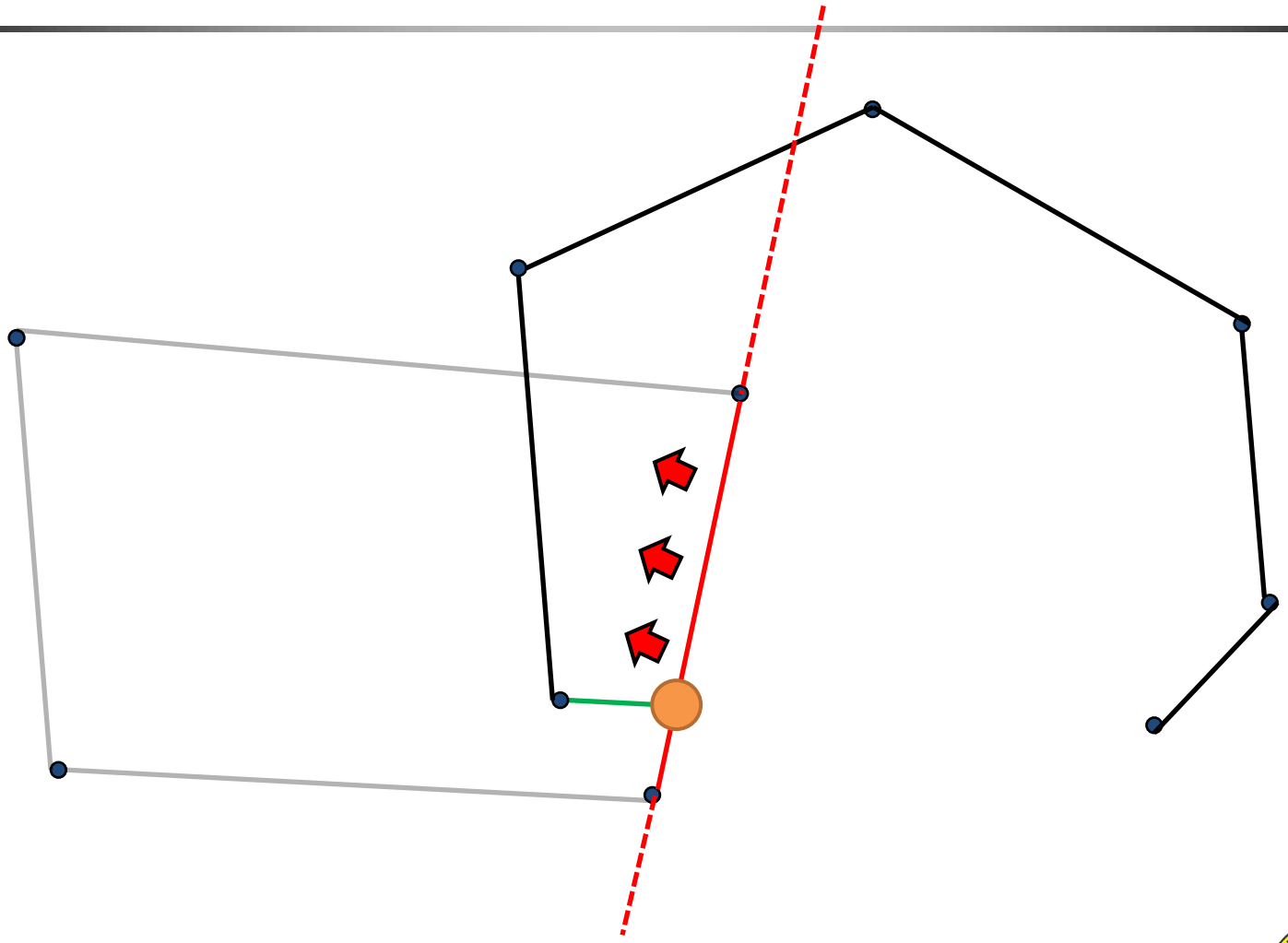
made by electron & free999

Intersection



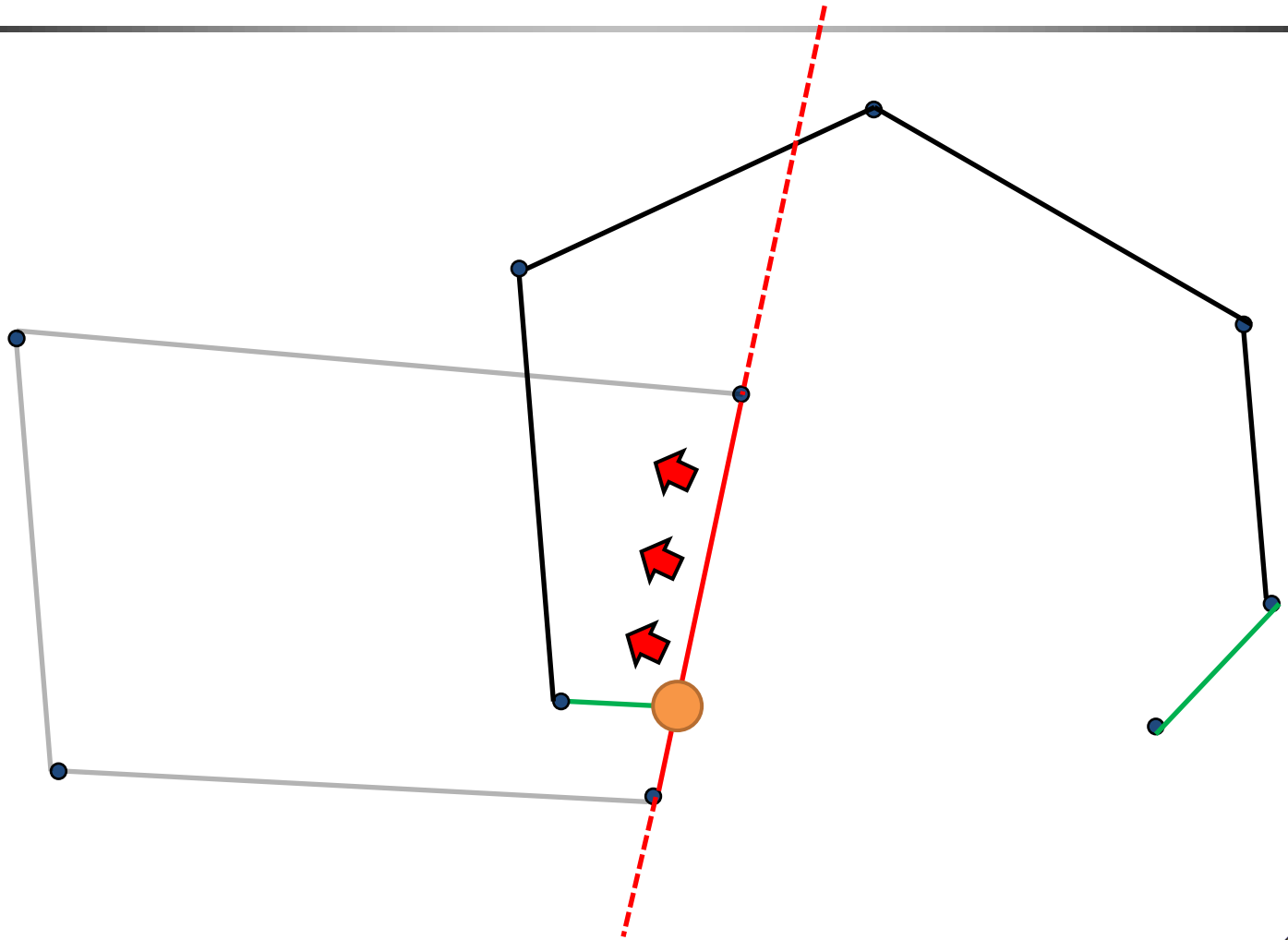
made by electron & free999

Intersection



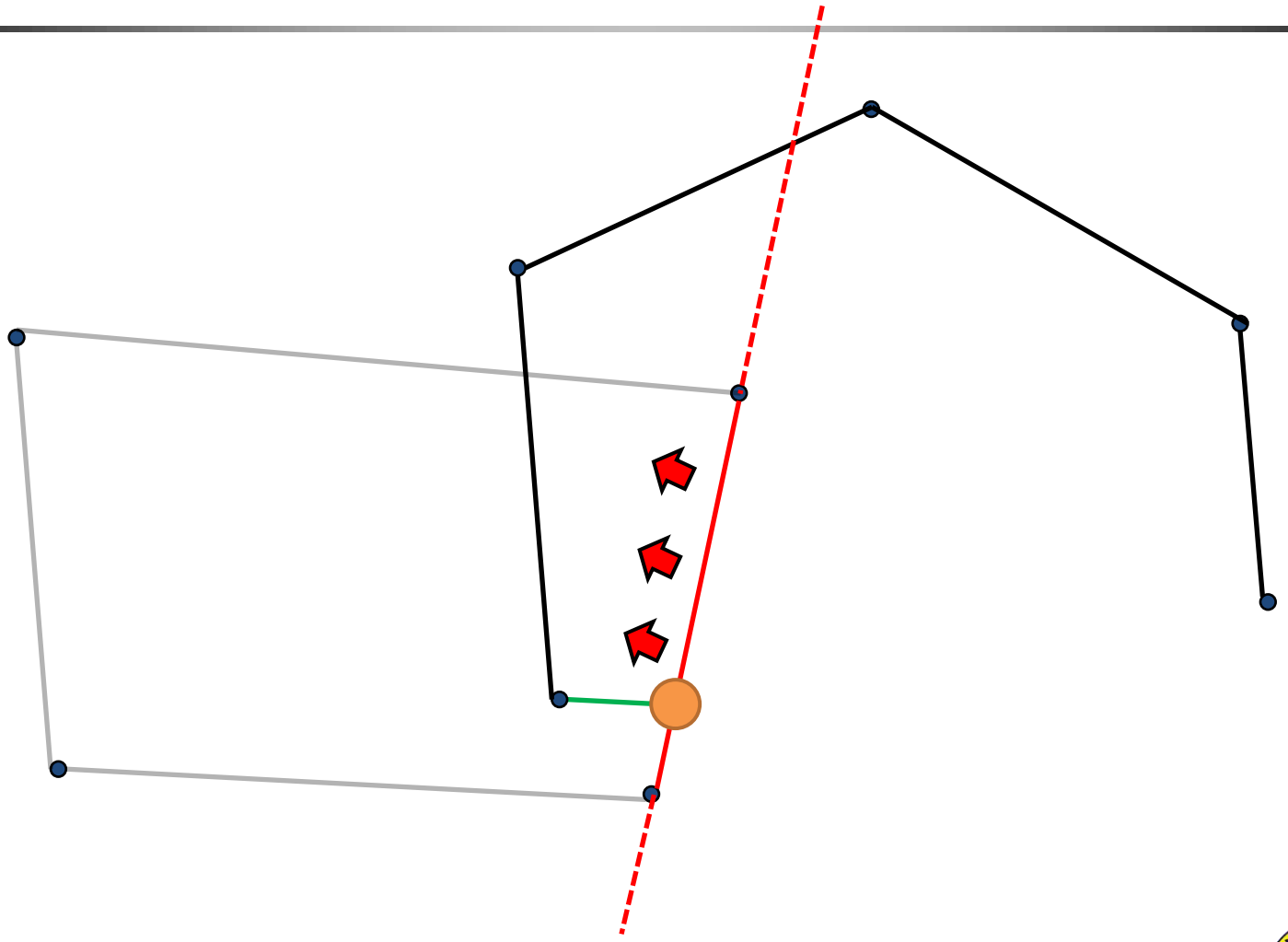
made by electron & free999

Intersection



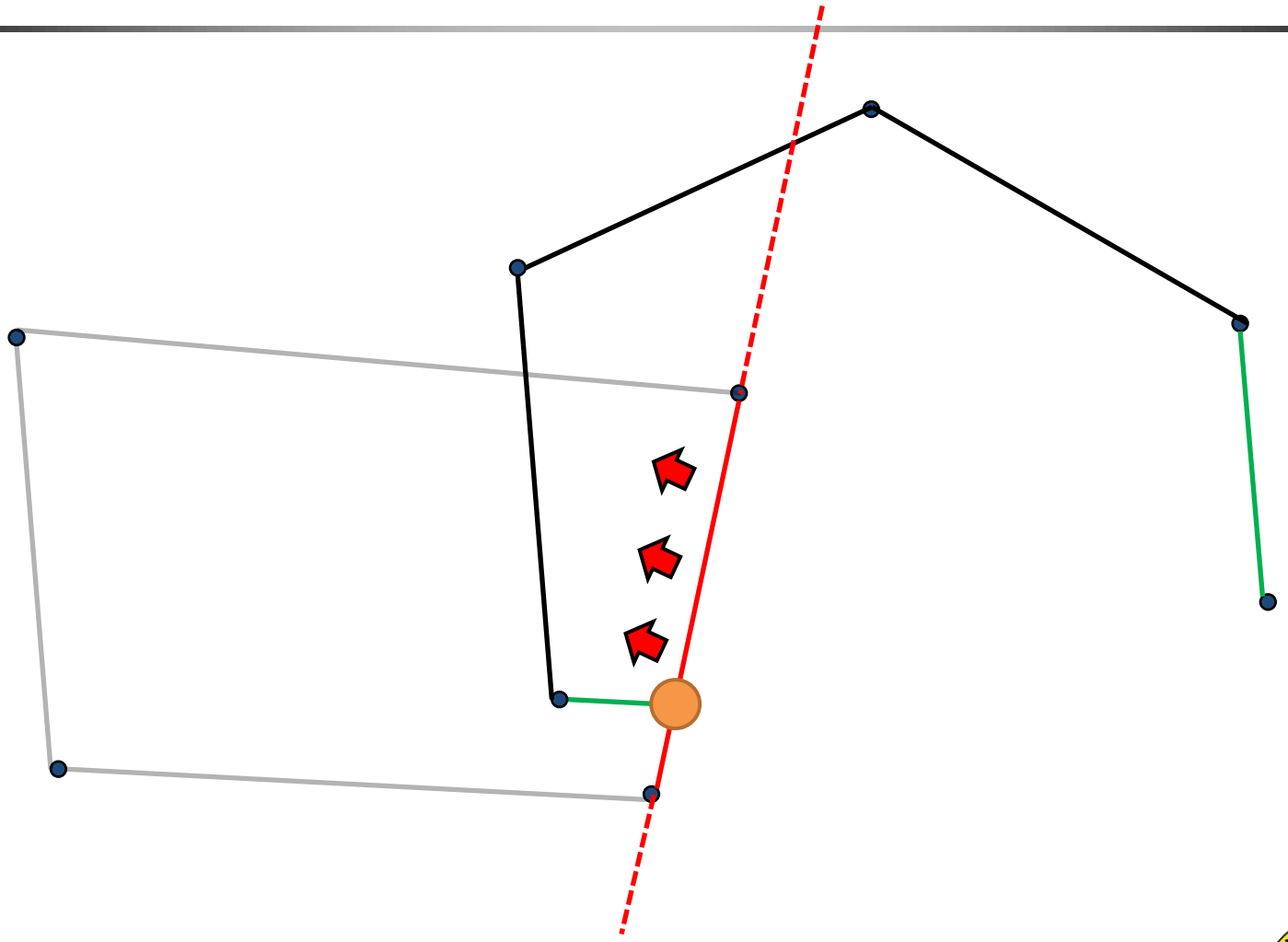
made by electron & free999

Intersection



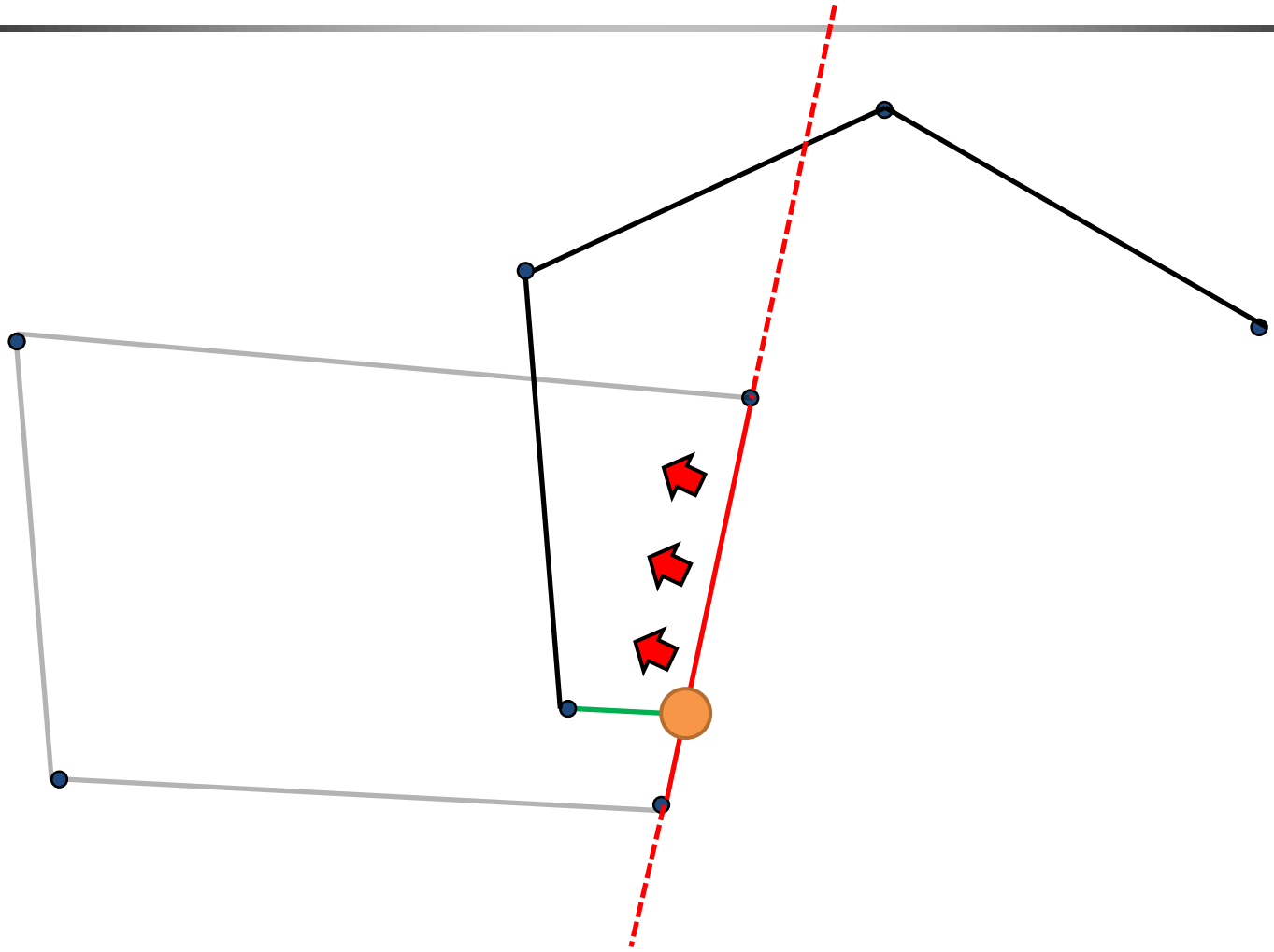
made by electron & free999

Intersection



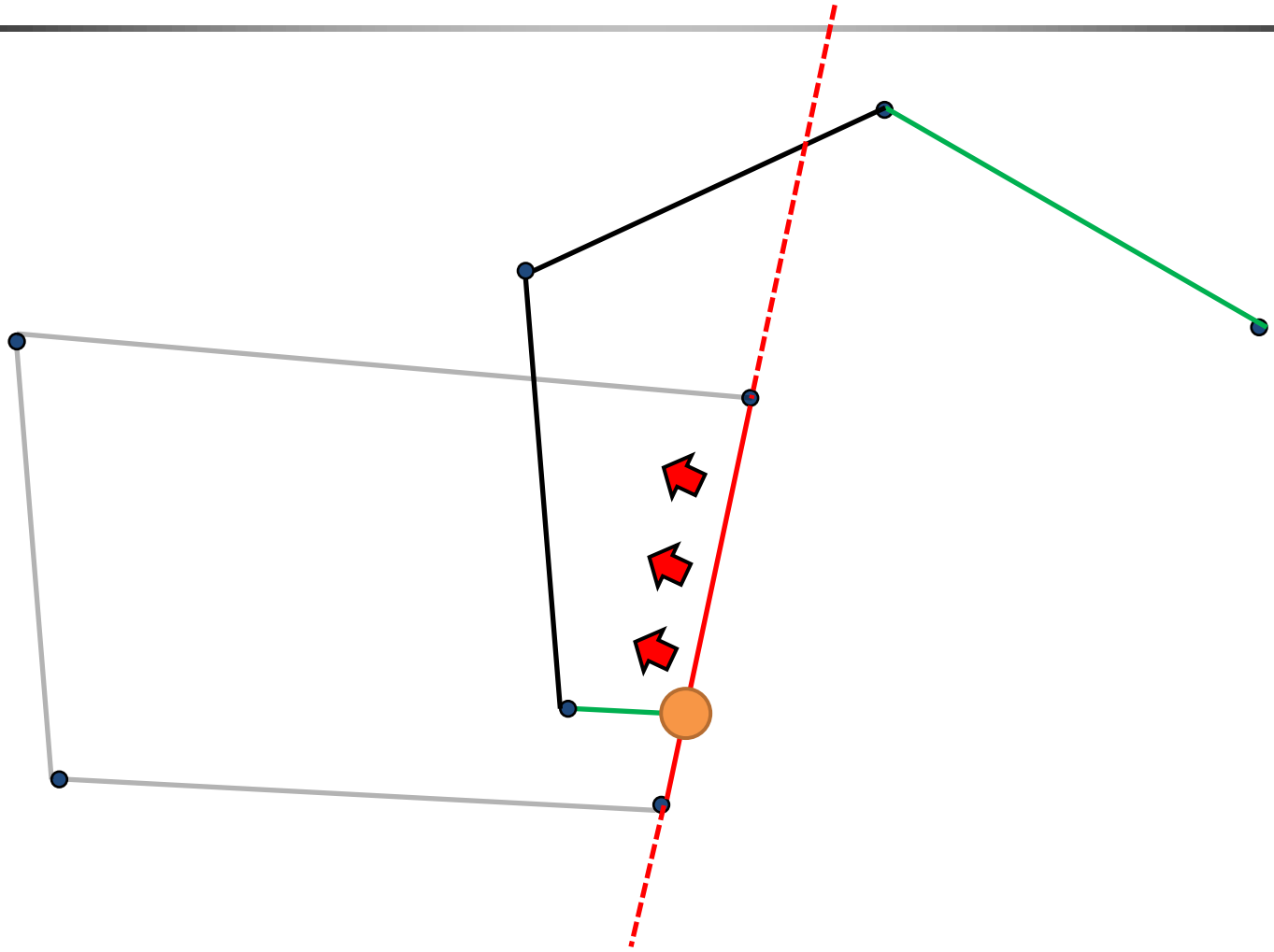
made by electron & free999

Intersection



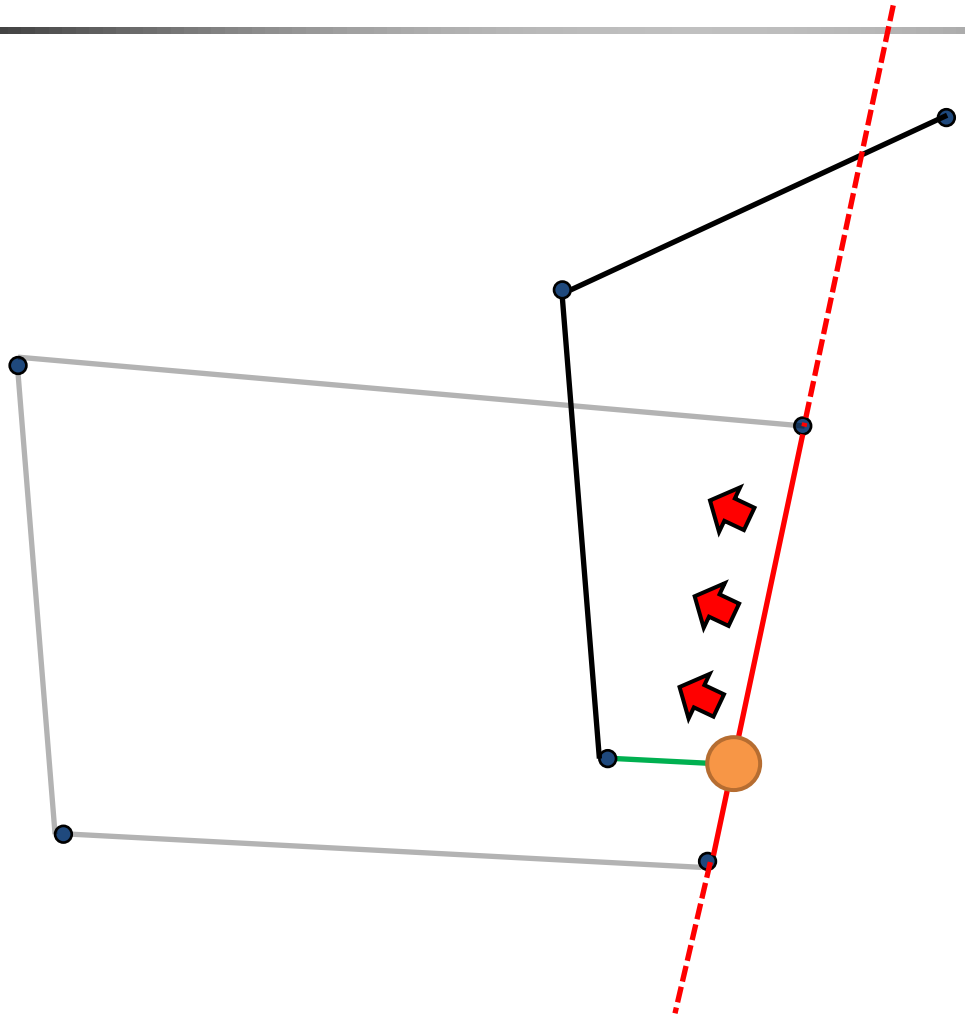
made by electron & free999

Intersection



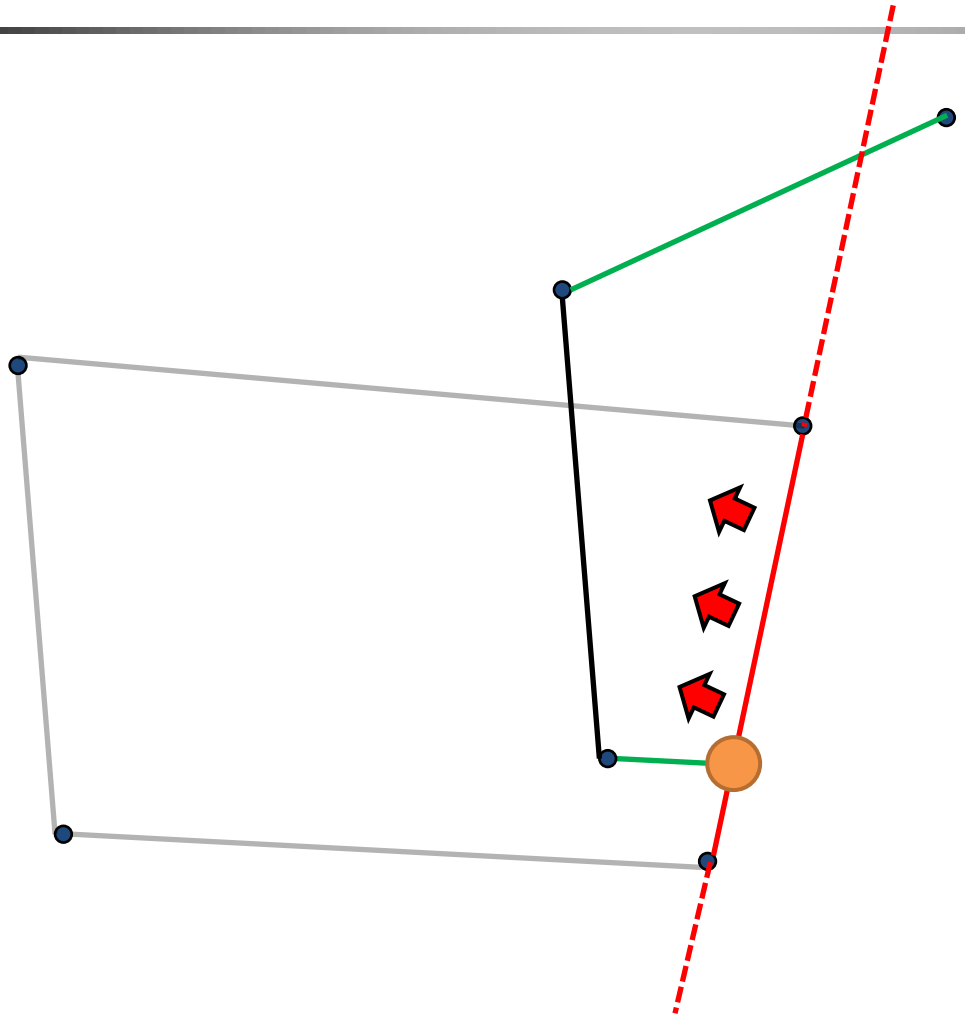
made by electron & free999

Intersection



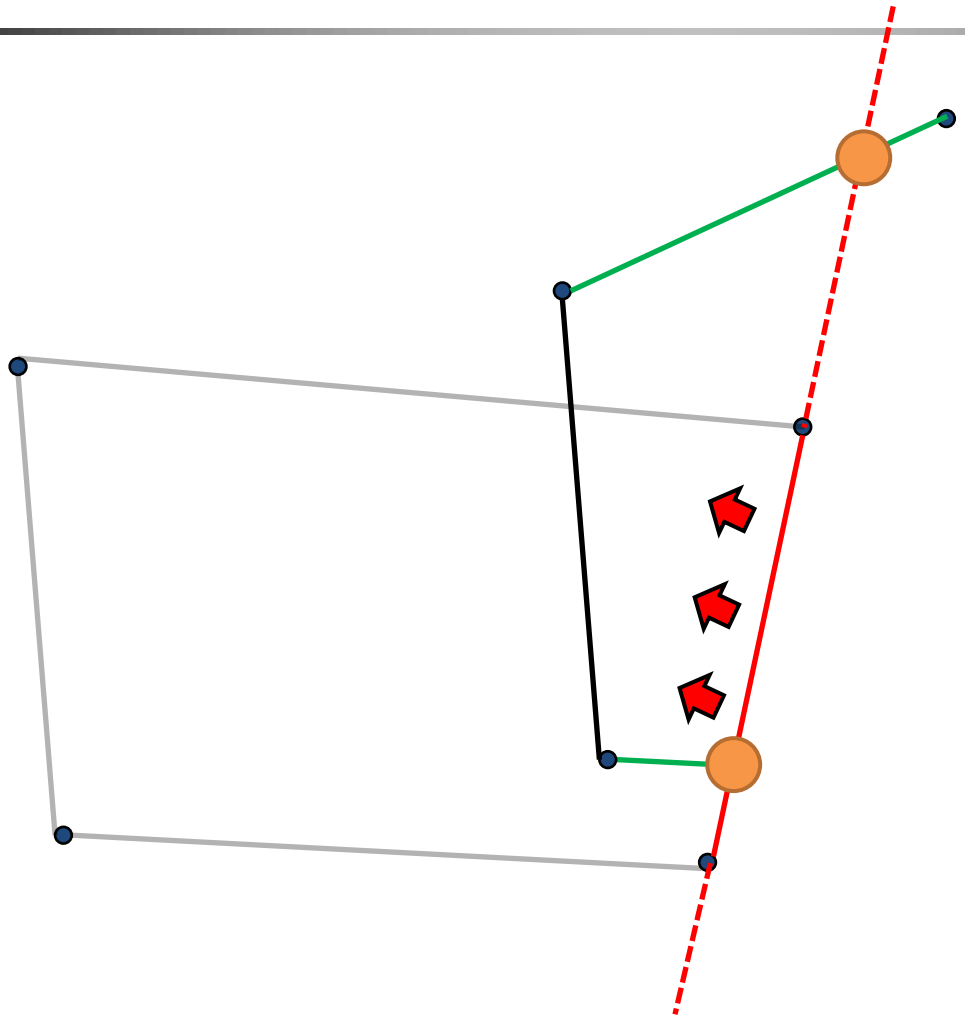
made by electron & free999

Intersection



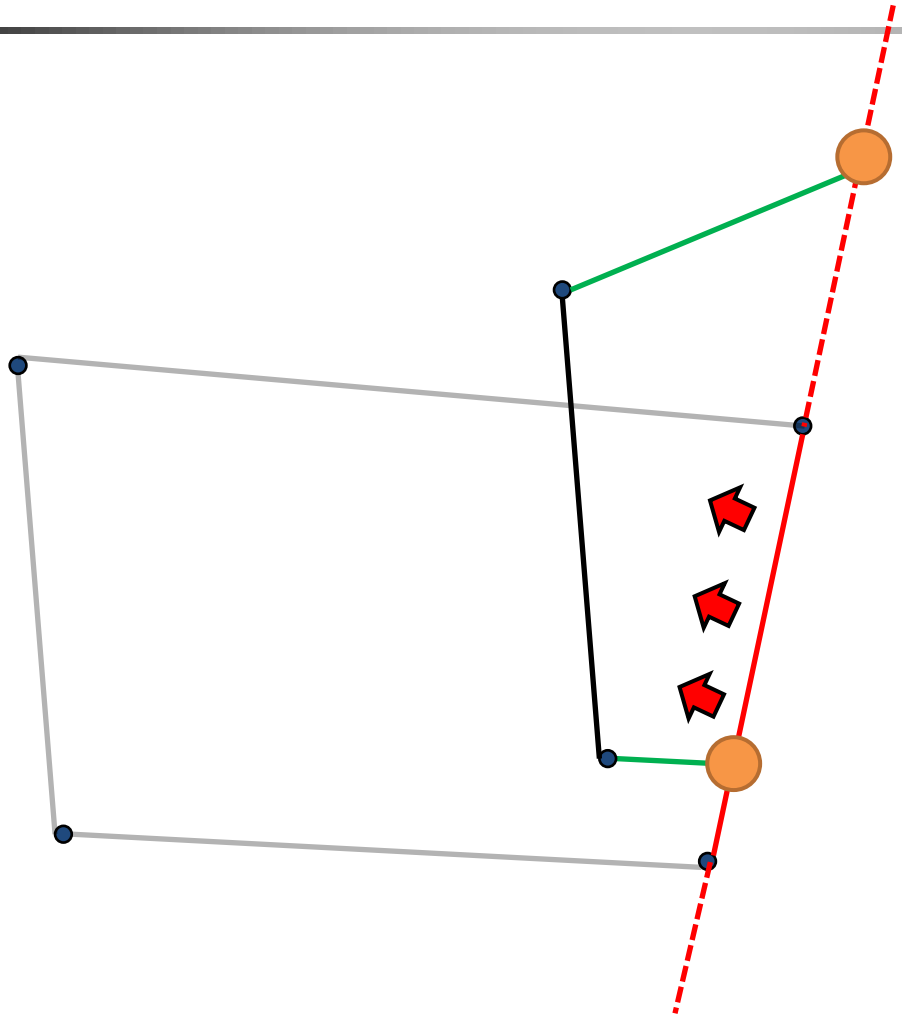
made by electron & free999

Intersection



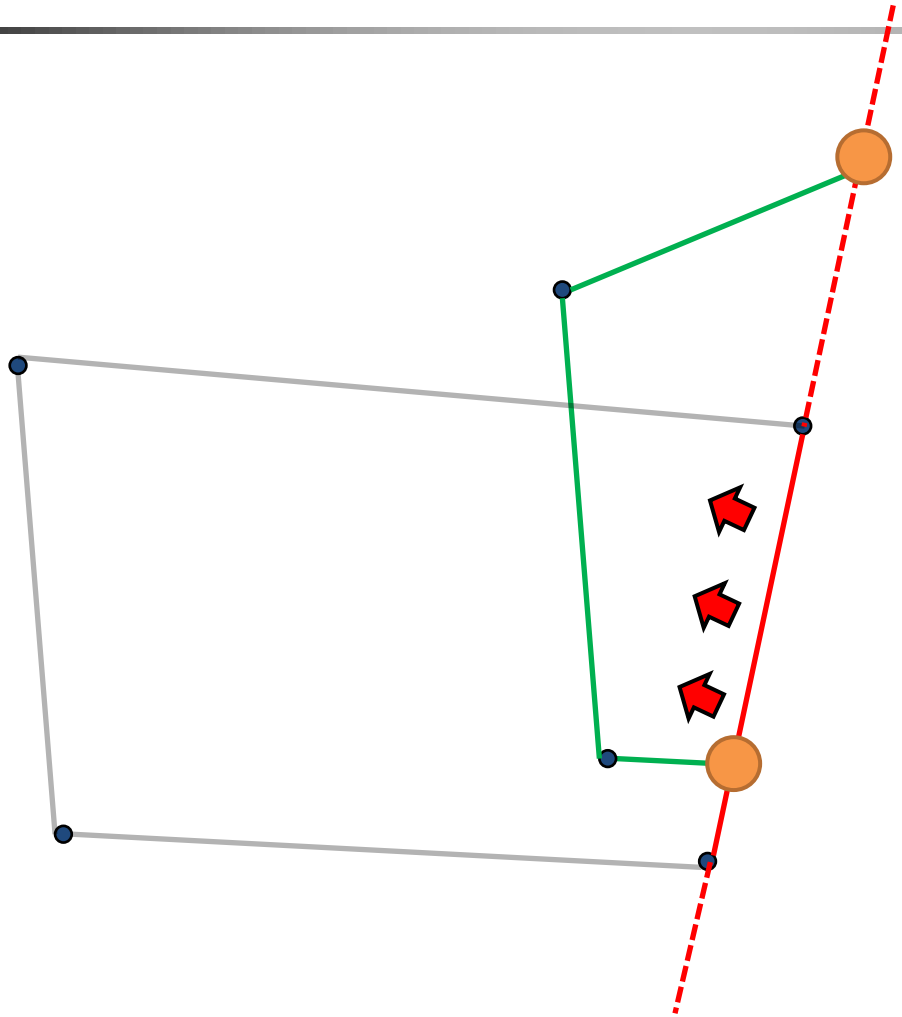
made by electron & free999

Intersection



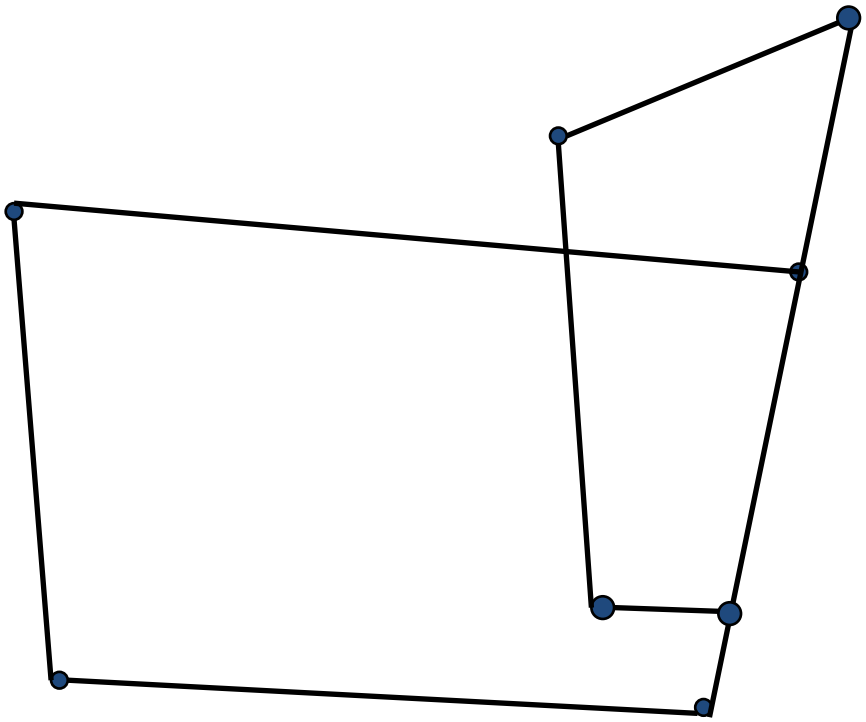
made by electron & free999

Intersection



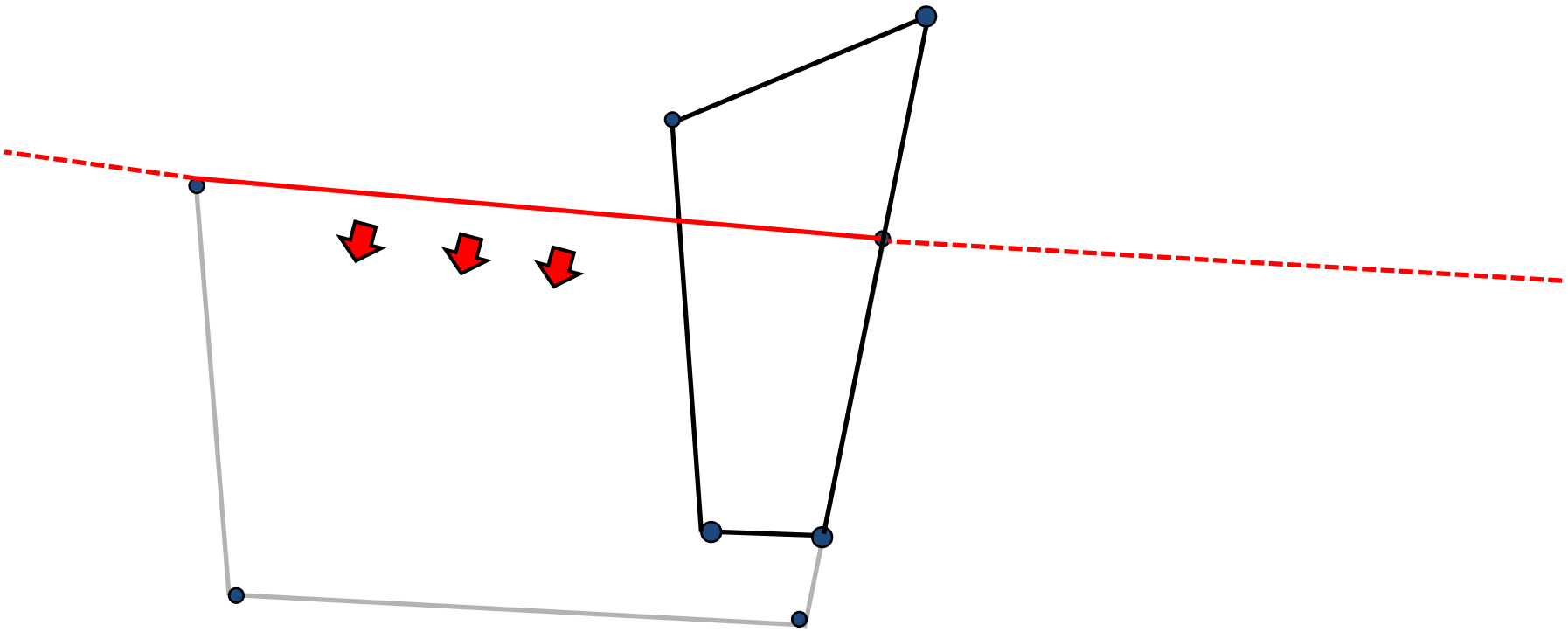
made by electron & free999

Intersection



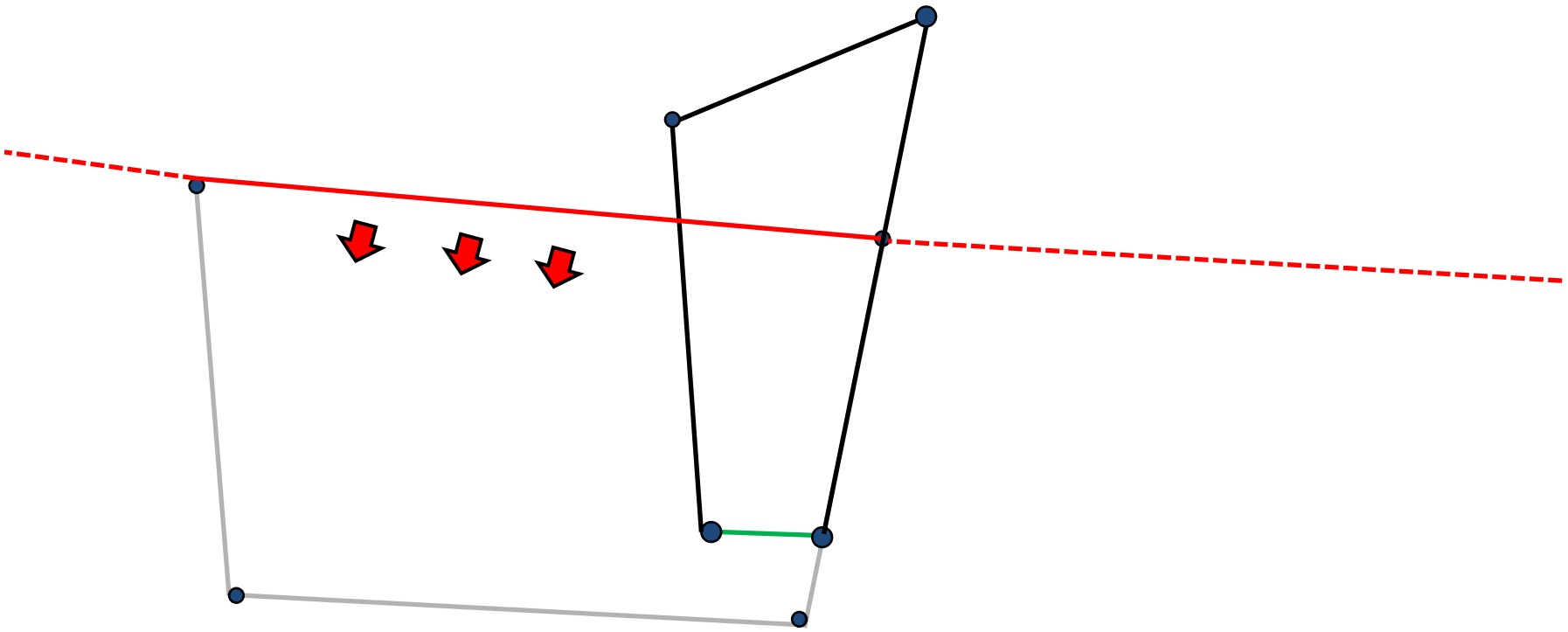
made by electron & free999

Intersection



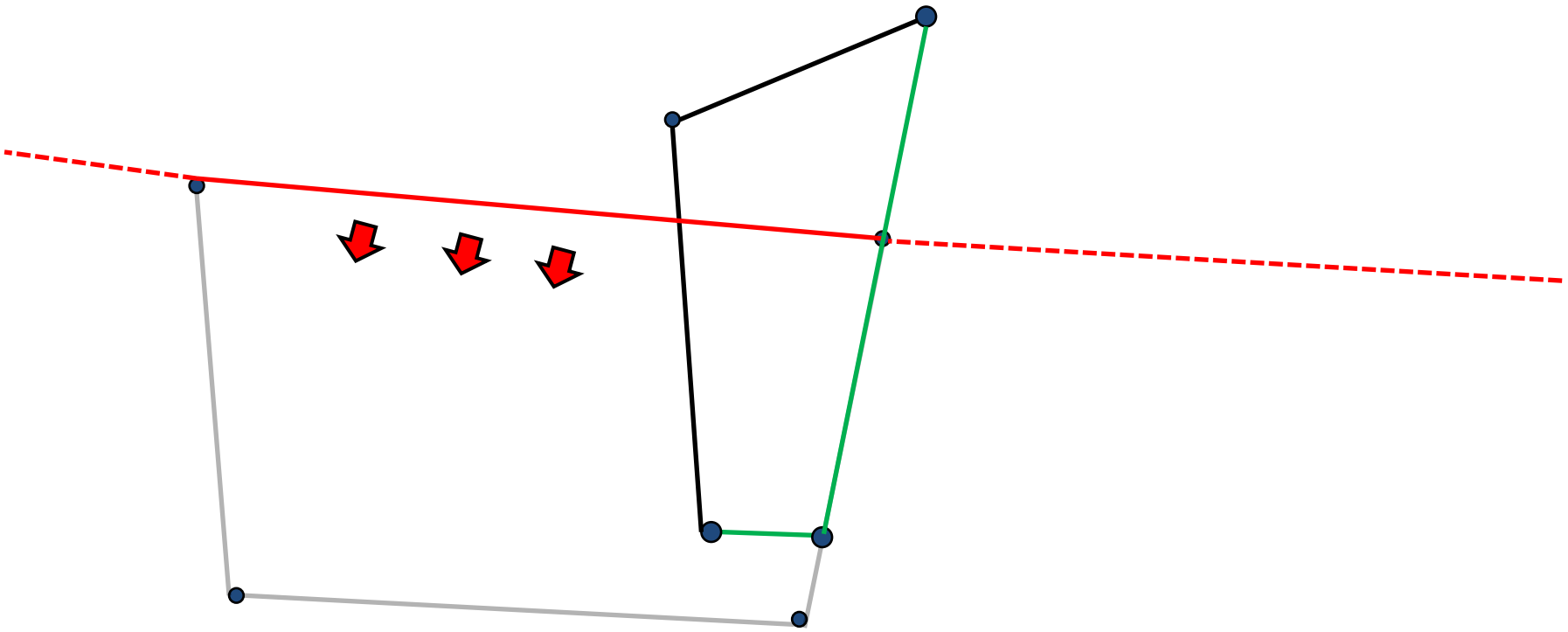
made by electron & free999

Intersection



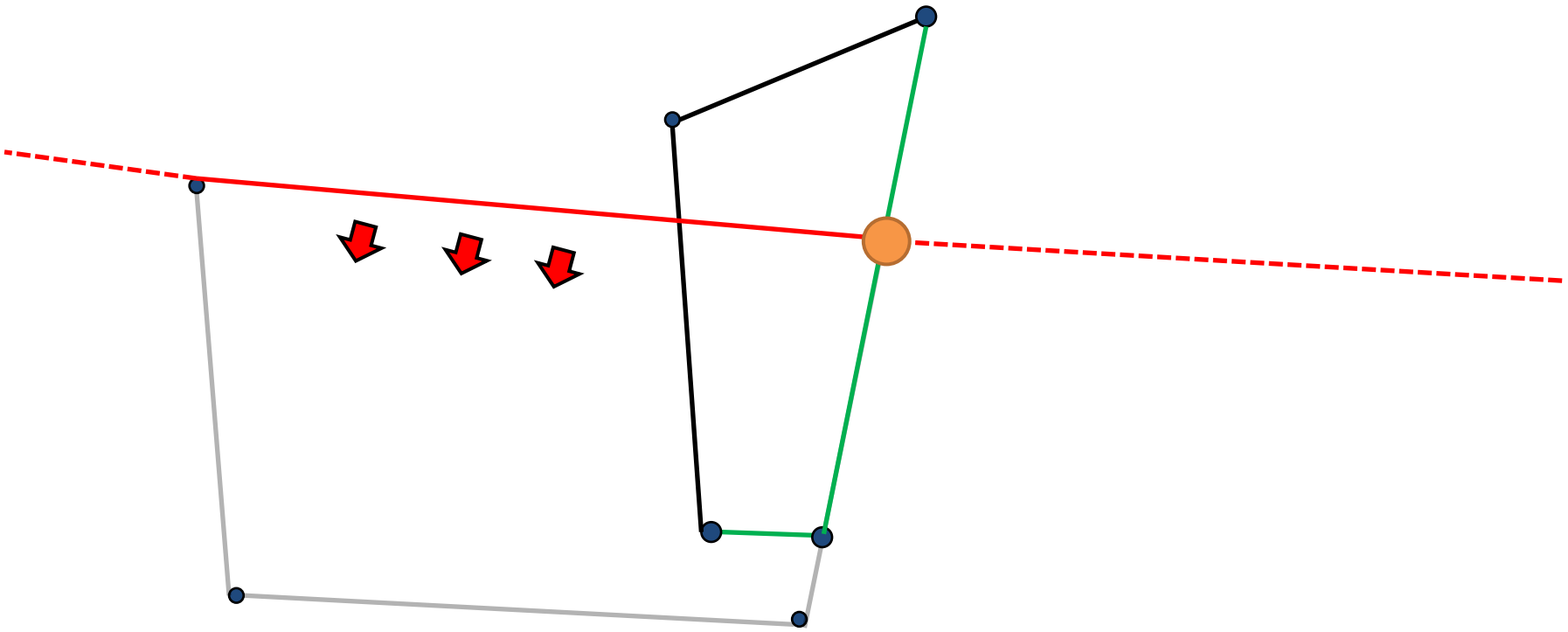
made by electron & free999

Intersection



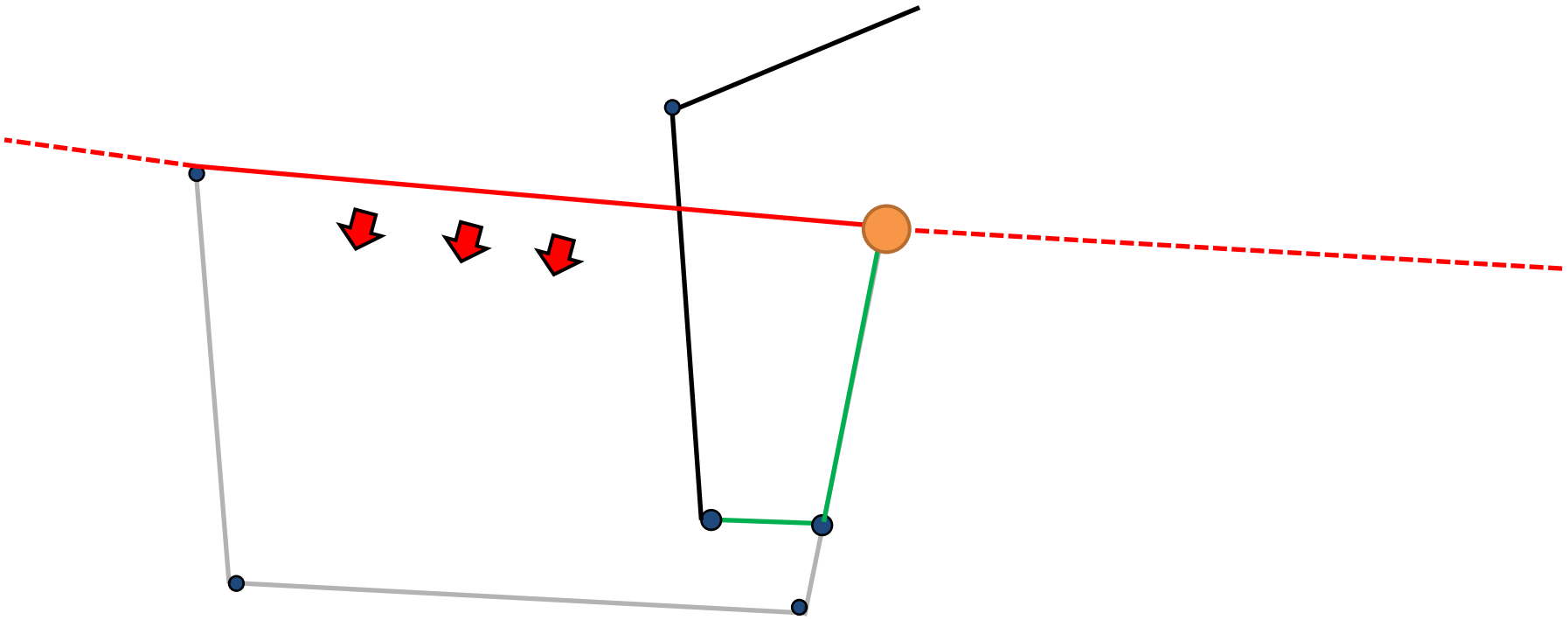
made by electron & free999

Intersection



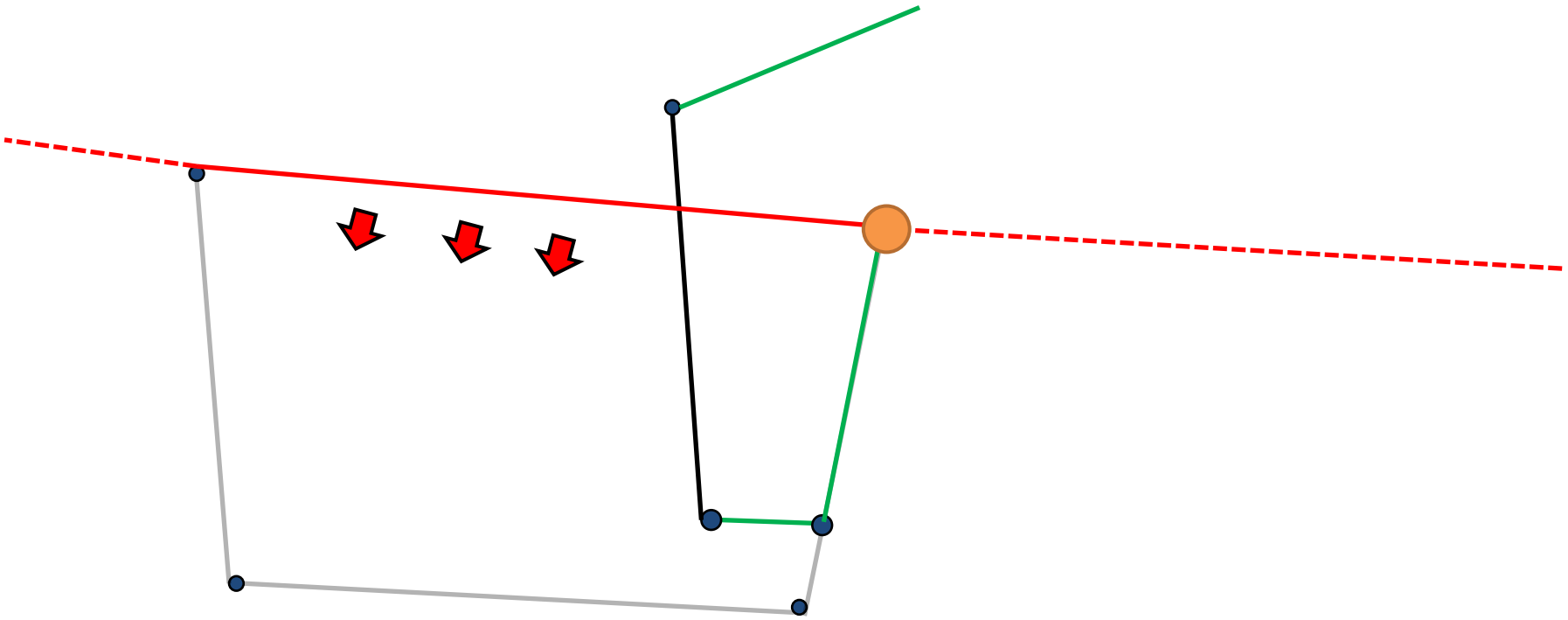
made by electron & free999

Intersection



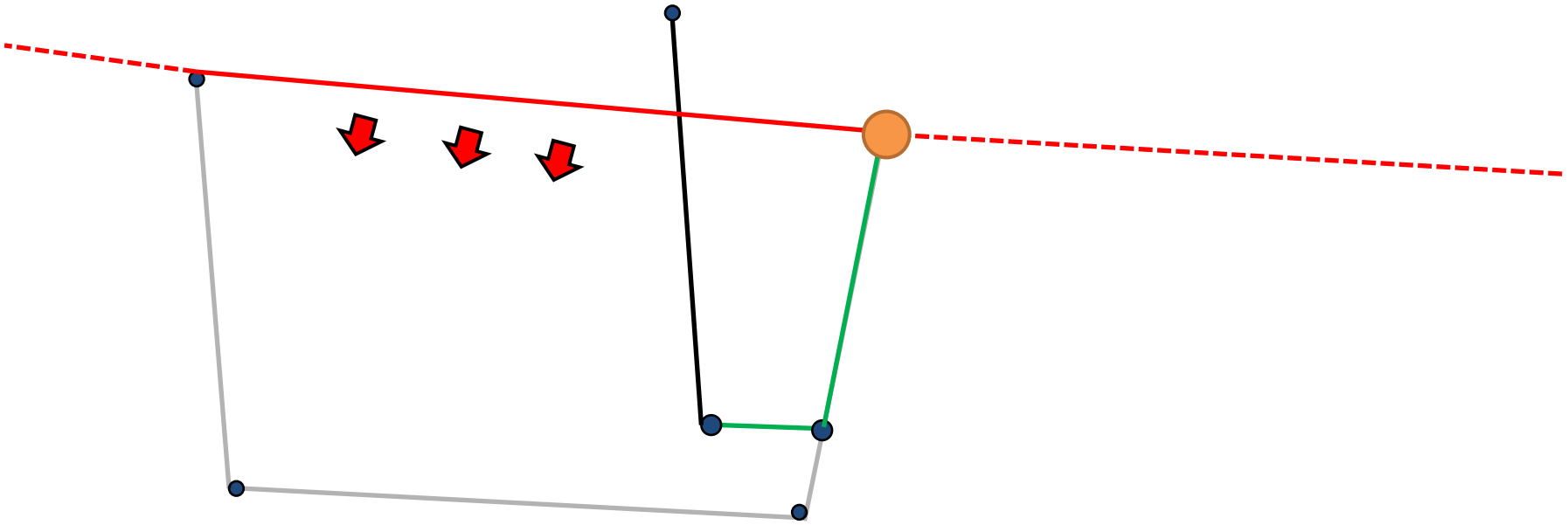
made by electron & free999

Intersection



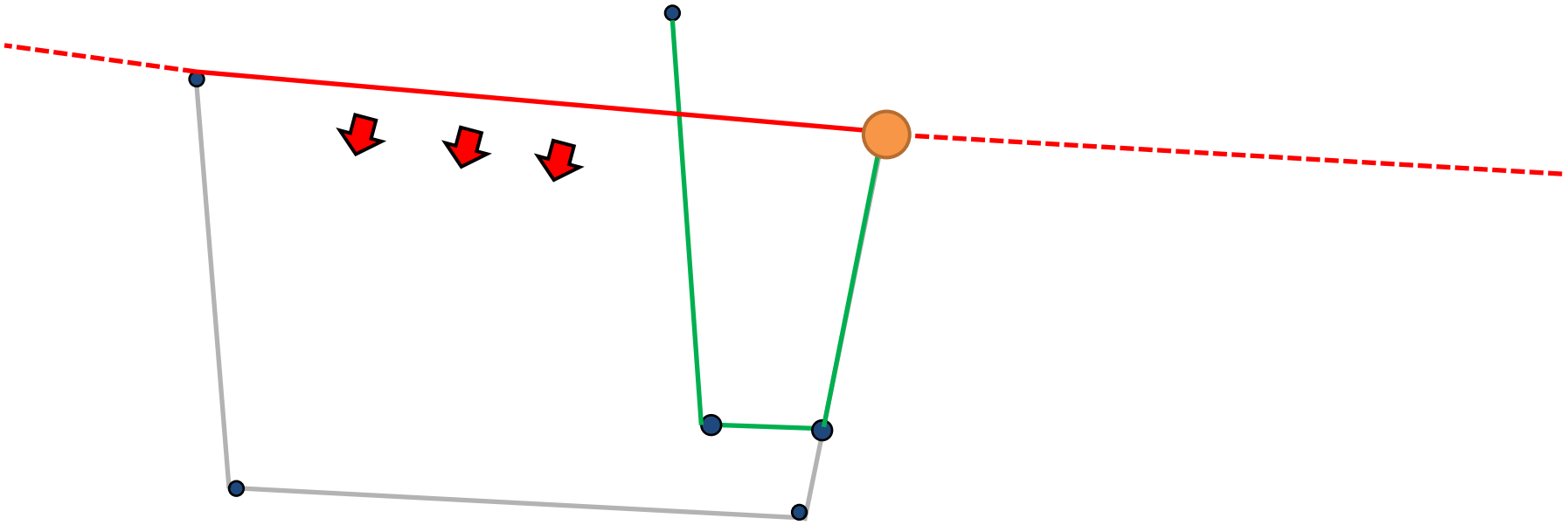
made by electron & free999

Intersection



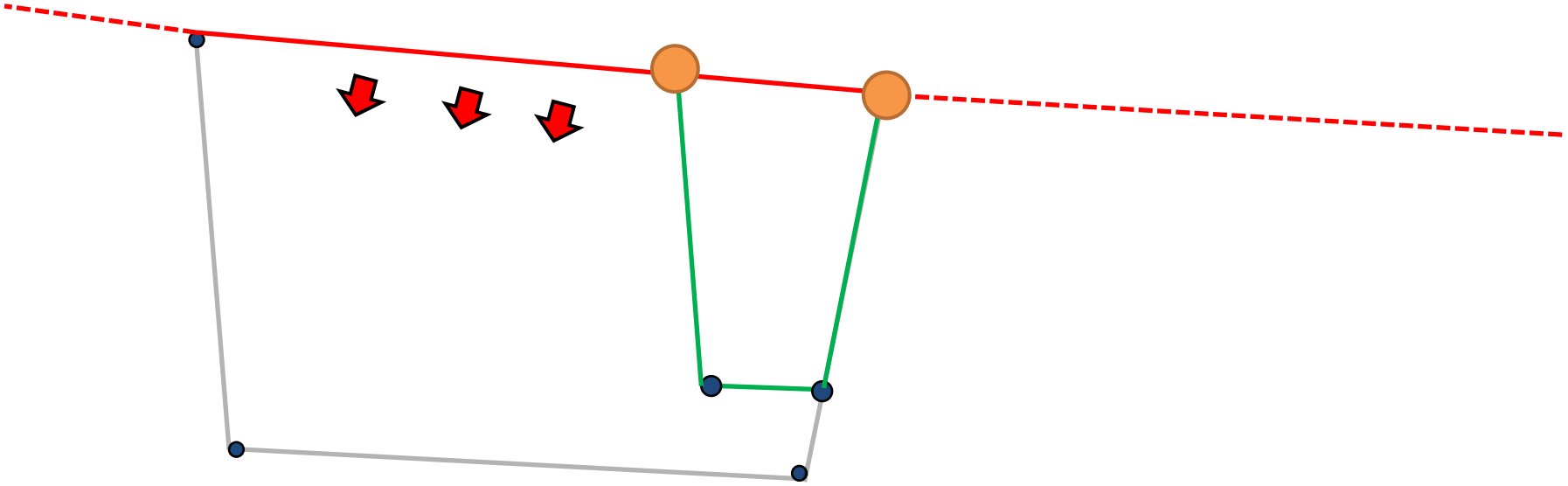
made by electron & free999

Intersection



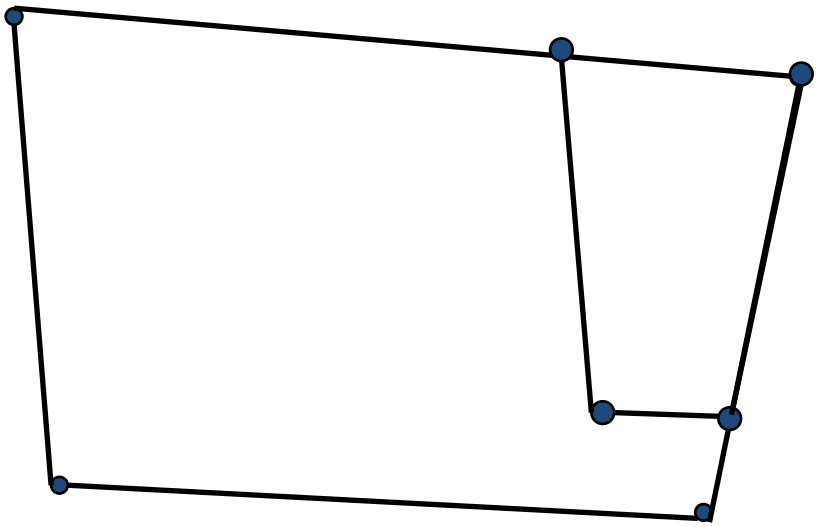
made by electron & free999

Intersection



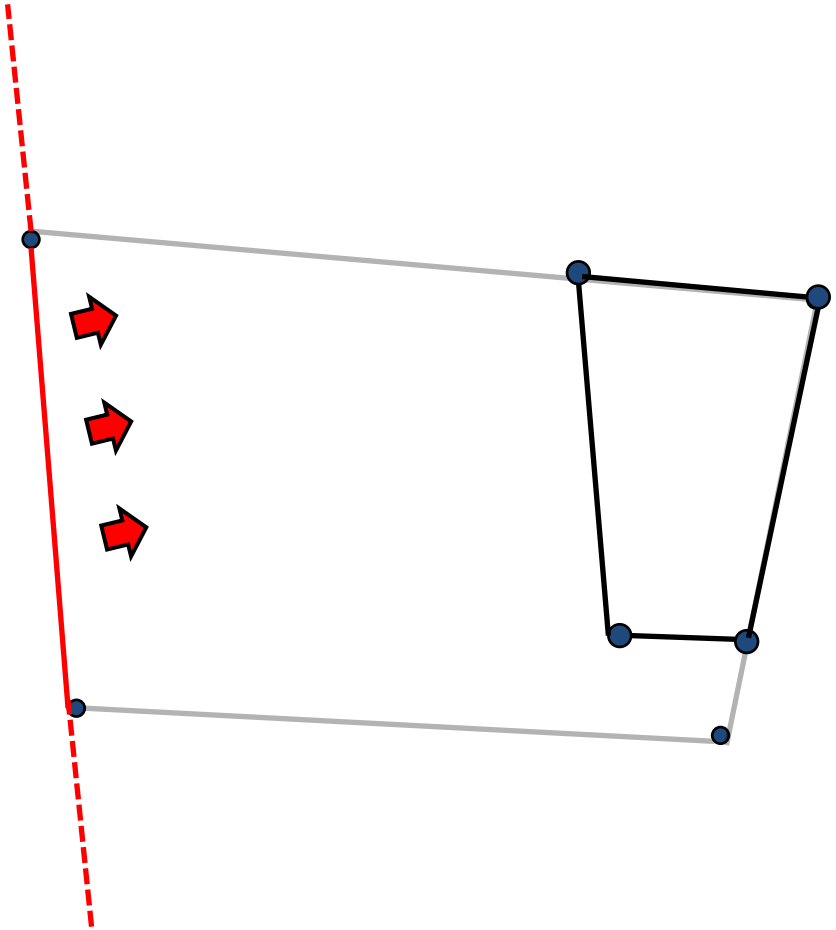
made by electron & free999

Intersection



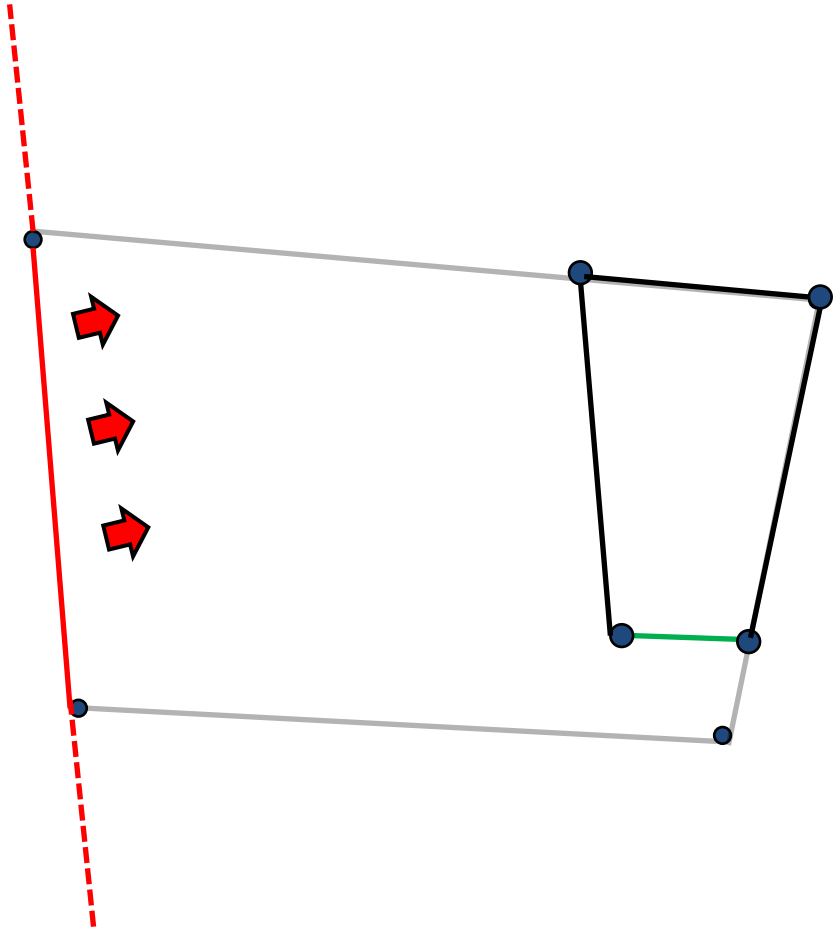
made by electron & free999

Intersection



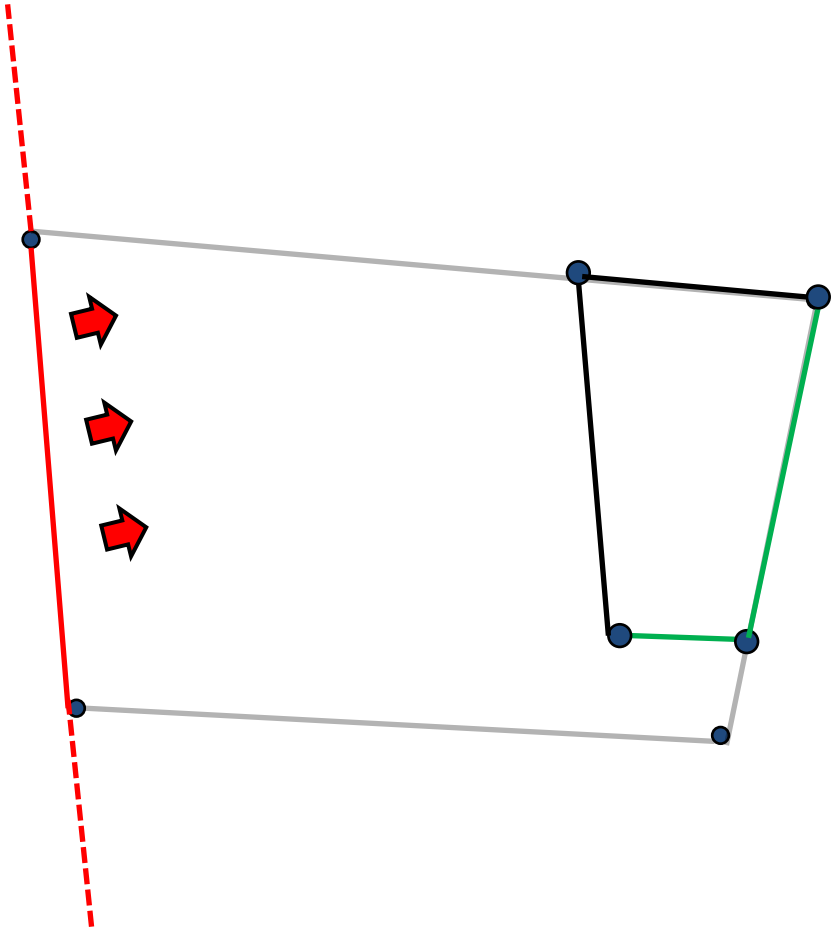
made by electron & free999

Intersection



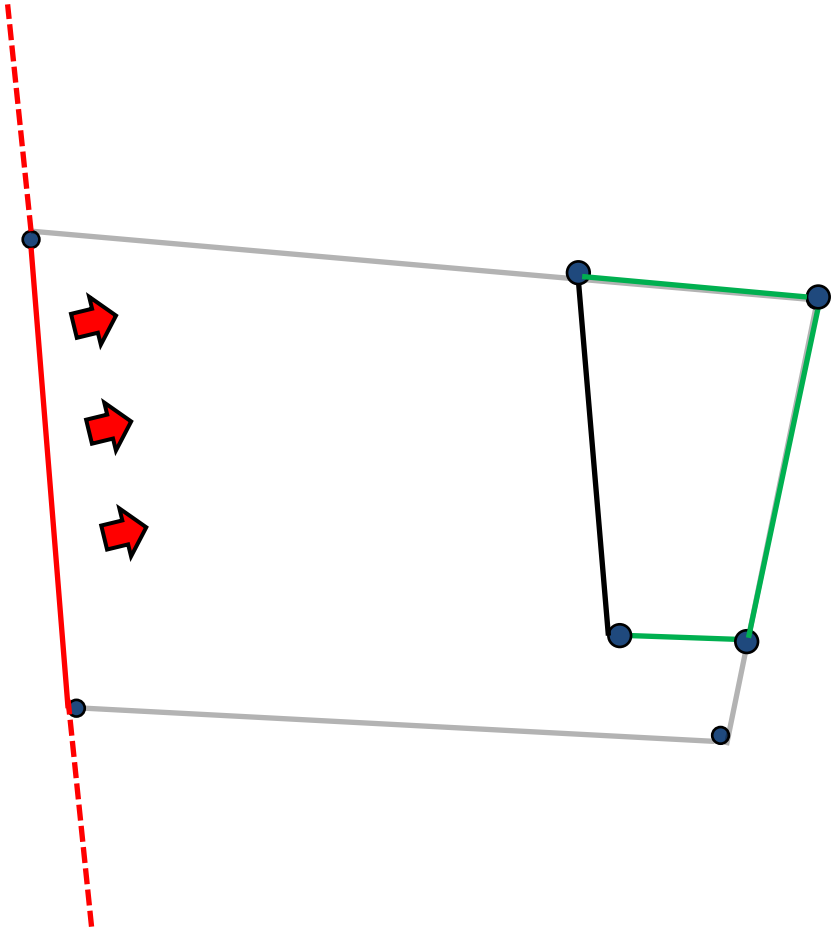
made by electron & free999

Intersection



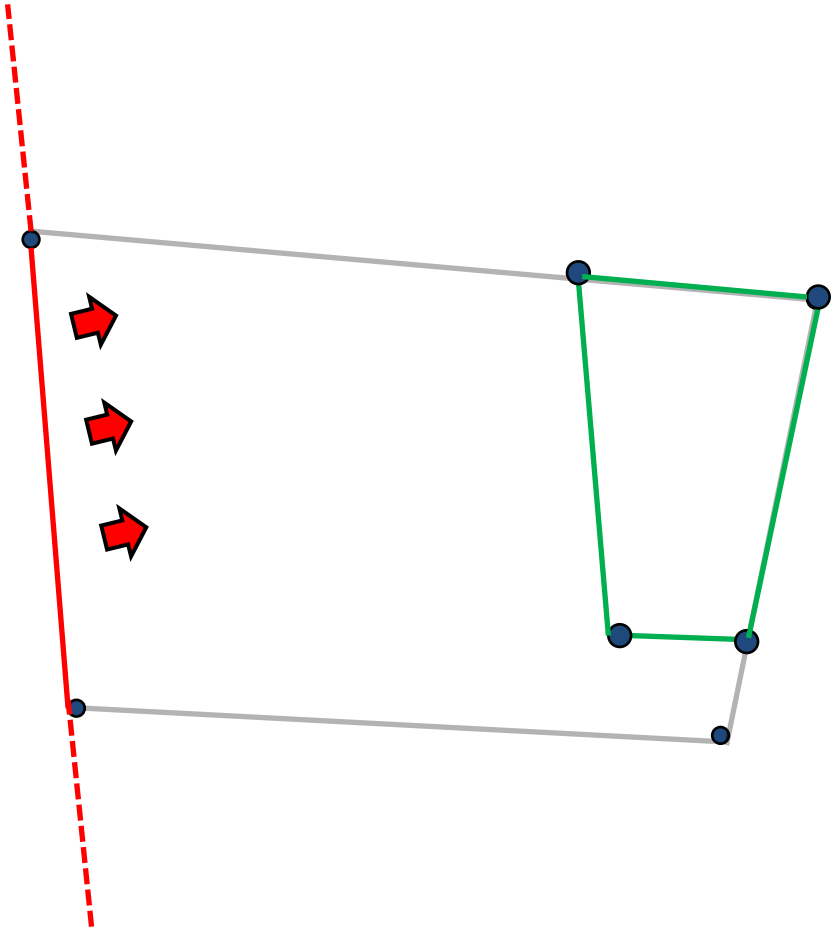
made by electron & free999

Intersection



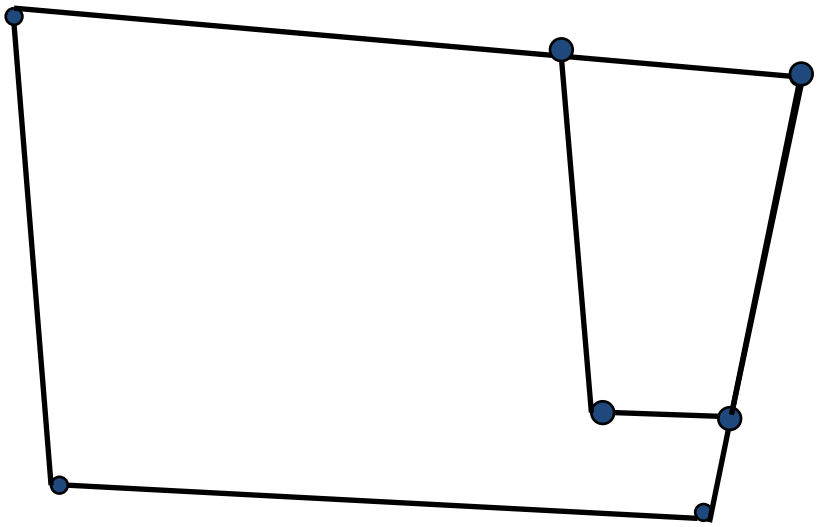
made by electron & free999

Intersection



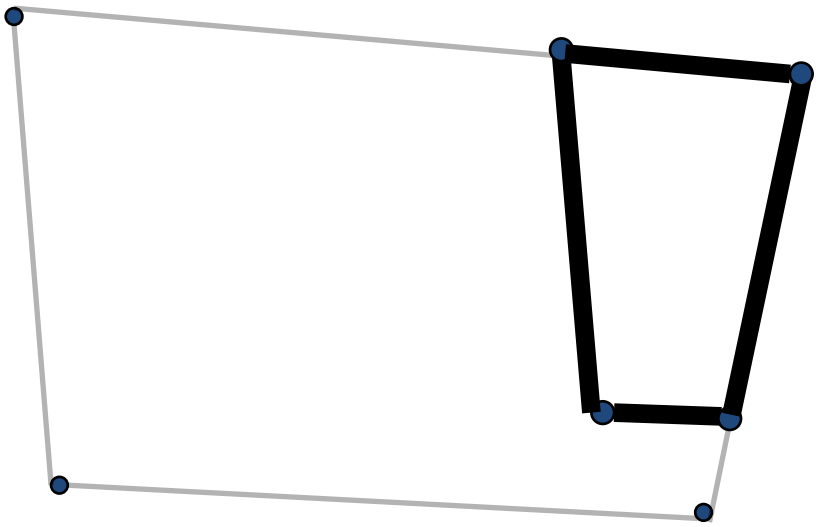
made by electron & free999

Intersection



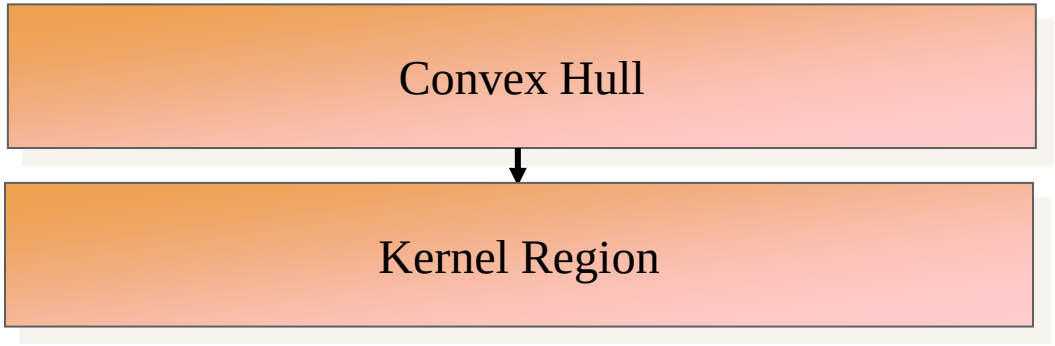
made by electron & free999

Intersection



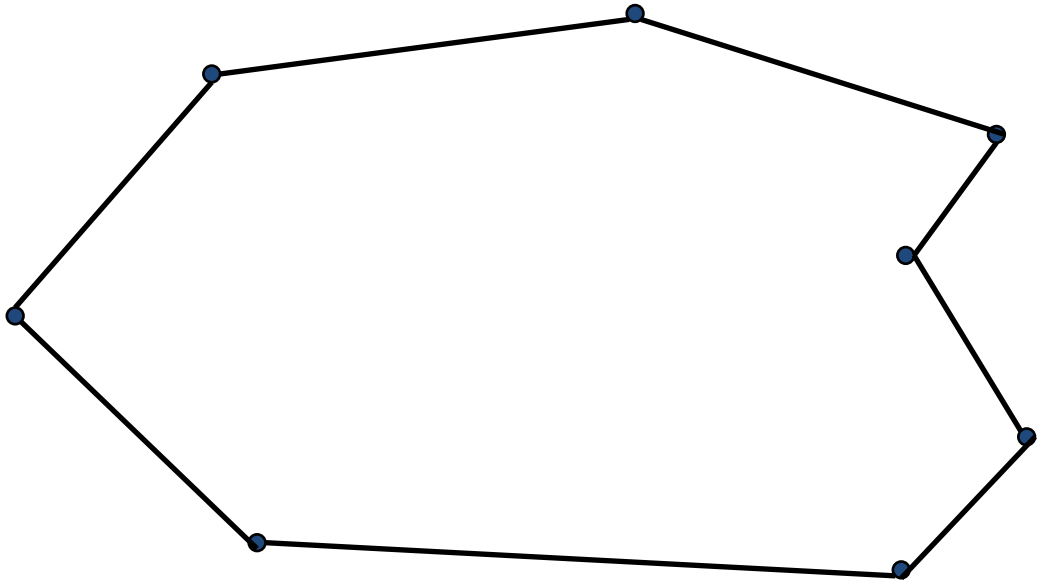
made by electron & free999

Outline



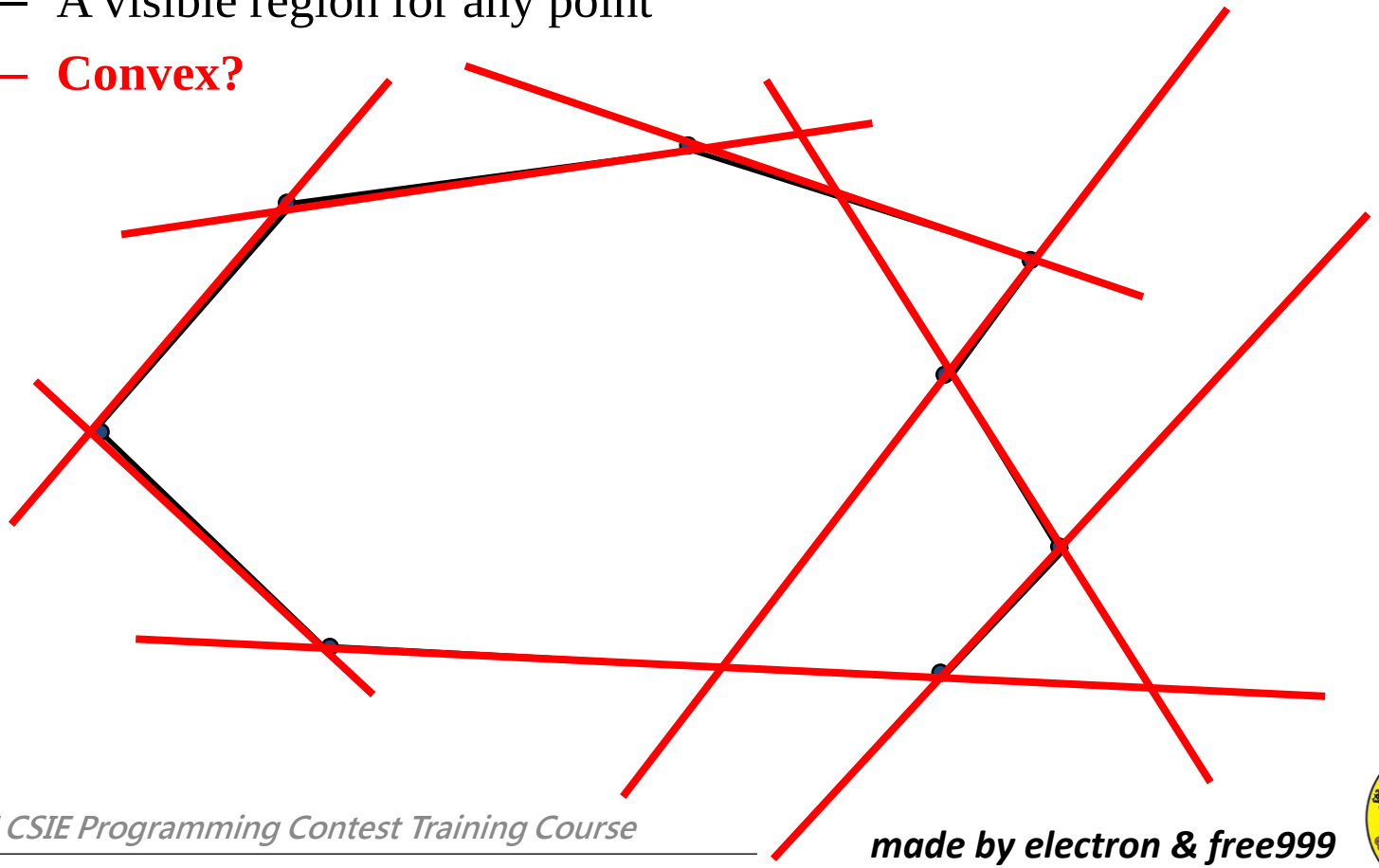
Kernel Region

- Definition
  - A visible region for any point
  - **Convex?**



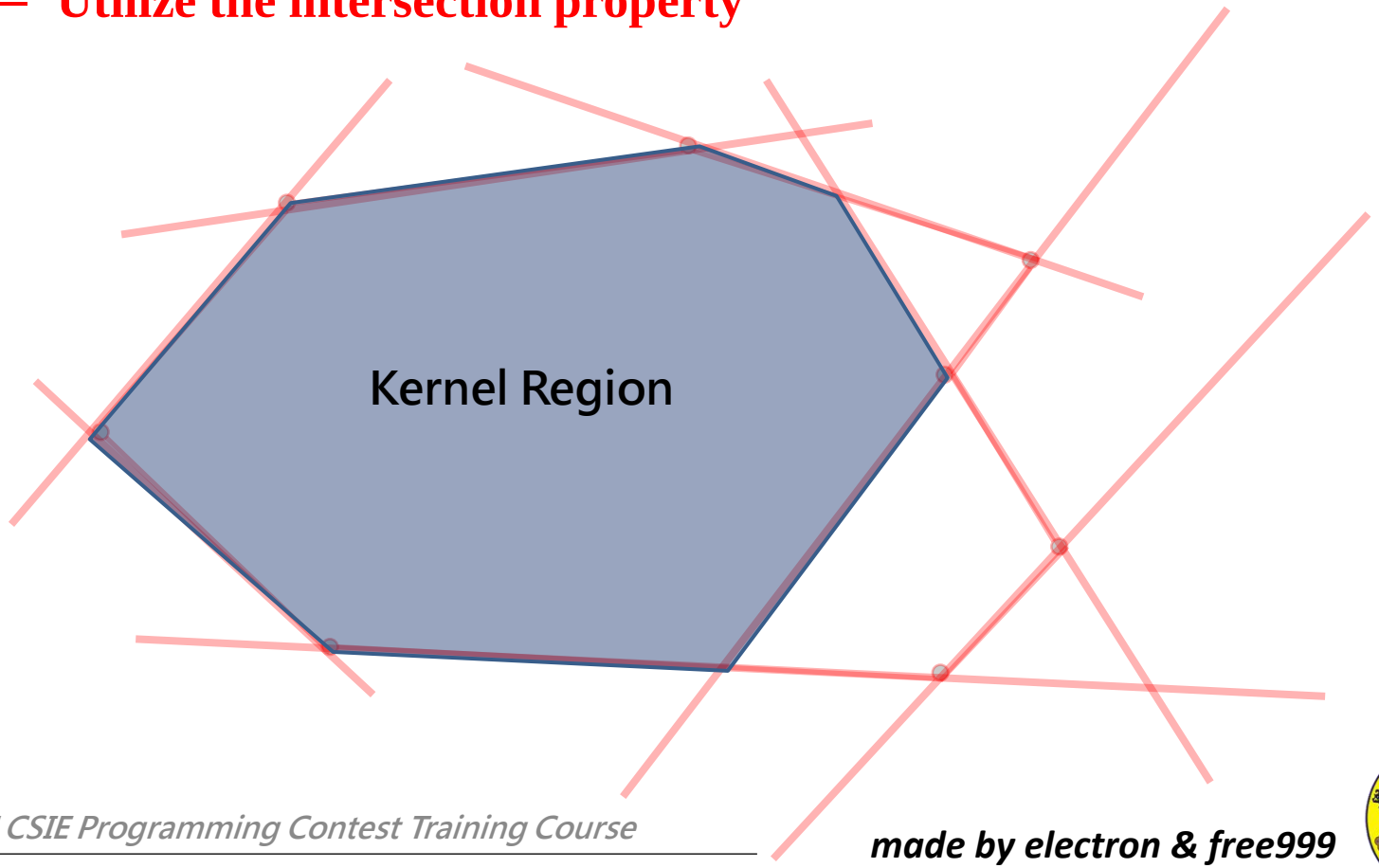
Kernel Region

- Definition
  - A visible region for any point
  - **Convex?**



Kernel Region

- Algorithm
  - **Utilize the intersection property**



made by electron & free999

Practice

PKU: 1279

PKU: 1474

Homework 18

PKU: 1113
PKU: 2007
PKU: 1873
PKU: 1228
PKU: 3348
PKU: 1654
PKU: 1265
PKU: 3335
PKU: 3130
PKU: 1474

PKU: 1279
PKU: 3525
PKU: 3384
PKU: 1755
PKU: 2540
PKU: 1654
PKU: 1265
PKU: 2954

ICPC: 4450

Uva
109, 132, 218
361, 675, 681
811, 819, 802
10002, 10065
10078, 10135
10173, 10256
10625, 11168
11626, 10089
10084, 10117
11265, 11989

made by electron & free999