1. [*CLRS*] Devise an algorithm to determine whether any three points in a set of n points are collinear. You may work out algorithms with the following time complexities: $O(n^3)$ and $O(n^2 \lg n)$. Implement both versions of the algorithm and test their correctness.

O(N³)):

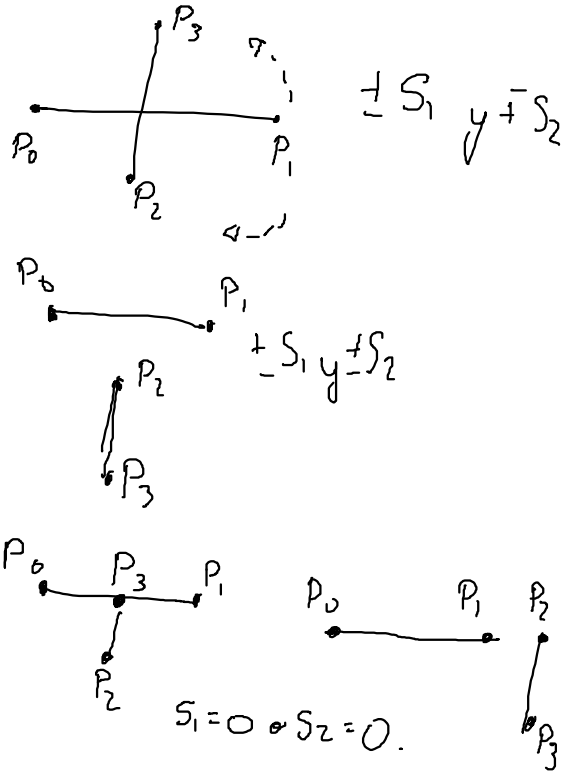
```
for (p,q,r) ∈ P×P×P:
    if p=q OR p=r OR q=r
        continue
    if pq⃗ × pr⃗ = 0
        return True, (p,q,r)
return False, None
```

O(N² log(N)); O(N²) space)

```
slopes = {}
for (p,q) in P×P:                    -> O(N²)
    if p=q:                          -> O(1)
        continue
    m = slope of pq⃗                  -> O(1)
    if m is not a key in slopes:     -> O(log N)
        add m:[pq] to slopes
    else:
        add pq to the list of points with slope m -> O(log N)
for slope in slopes:                 -> O(N²)   (viceversa)
    if pq⃗, qr⃗ ∈ slopes[slope] for some p,q,r ∈ P:   -> O(1)
        return True, [p,q,r]
return False, None
```

$$T(N) \approx N^2 \log(N) O(1) + O(1) N^2 = O(N^2 \log(N))$$

2. [*CLRS*] Given a point $p_0 = (x_0, y_0)$, the *right horizontal ray* from p_0 is the set of points $\{p_i = (x_i, y_i) | x_i \geqslant x_0 \text{ and } y_i = y_0\}$,that is, it is the set of points due right to p_0 along with p_0 itself. Show how to determine whether a given right horizontal ray from p_0 intersects a line segment $\overline{p_1 p_2}$ in $O(1)$ time by reducing the problem to that of determining whether two line segments intersect. Write the pseudocode of the corresponding algorithm.



Input: $P_0 = (x_0, y_0), P_1 = (x_1, y_1), P_2 = (x_2, y_2)$

$X = \max\{x_1, x_2\}$
if $x < x_0$:
  return False
$p = (x, y_0)$
return Intersection($\overrightarrow{pp_0}, \overrightarrow{pp_2}$)

Intersection($\overrightarrow{p_0p_1}, \overrightarrow{p_2p_3}$):

$S_1 = (p_1 - p_0) \times (p_3 - p_1)$

$S_2 = (p_1 - p_0) \times (p_2 - p_1)$

if S_1 and S_2 have diferent sign:
  return True

if $S_1 = 0$ and $x_3 \in (\min\{x_0, x_1\}, \max\{x_0, x_1\})$ and $y_3 \in (\min\{y_0, y_1\}, \max\{y_0, y_1\})$:
  return True

if $S_2 = 0$ and $x_2 \in (\min\{x_0, x_1\}, \max\{x_0, x_1\})$ and $y_2 \in (\min\{y_0, y_1\}, \max\{y_0, y_1\})$:
  return True

return False

3. [*Point in polygon.*] Consider a *convex polygon* and a point defined in the plane (two dimensions). One way to determine whether a point p_0 is in the interior of a simple, convex, polygon P is to look at any ray from p_0 and check that the ray intersects the boundary of P an odd number of times but that p_0 itself is not on the boundary of P. Show how to compute in $\Theta(n)$ time whether a point p_0 is in the interior of an n-vertex polygon P. You may also implement your algorithm and test it thoroughly. (Hint: Use the previous exercise. Make sure your algorithm is correct when the ray intersects the polygon boundary at a vertex and when the ray overlaps a side of the polygon.)

Input: P a set with the boundary points of the polygon sorted, p the point to check

$P = \{P_0, P_1, \ldots, P_n\}$, $p = (x, y)$

```
def Inside(P, p):
    for all four r rays from p: -> Θ(1)
        r_valid = False
        for i = 0 to n-1: -> O(N)
            if p ∈ P_i P_{i+1}: -> O(1)
                return False
            if r intersects with P_i P_{i+1}: O(1)
                r_valid = True
        if r_valid is False:
            return False
    return True
```

$\Theta(N)$.