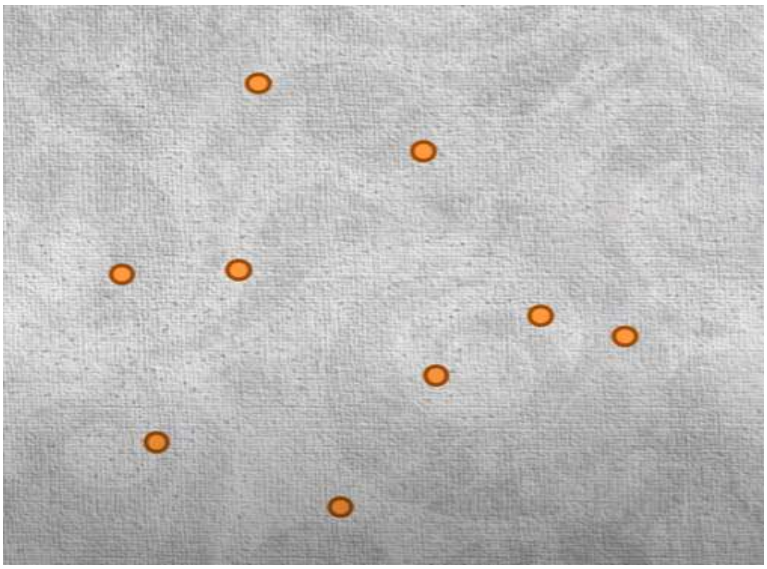
Convex Hull

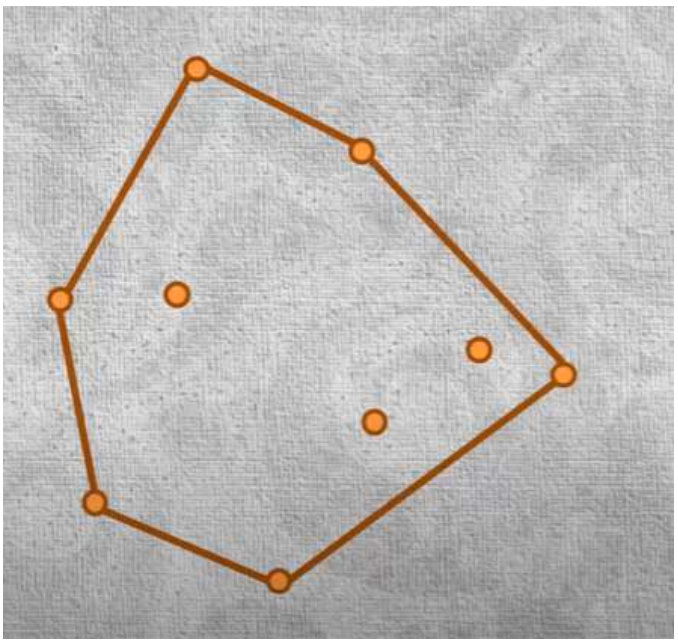
Given a set of points, convex hull is a polygon that encloses all the points.

Points that form the polygon, maximize the area and minimizes the circumference.

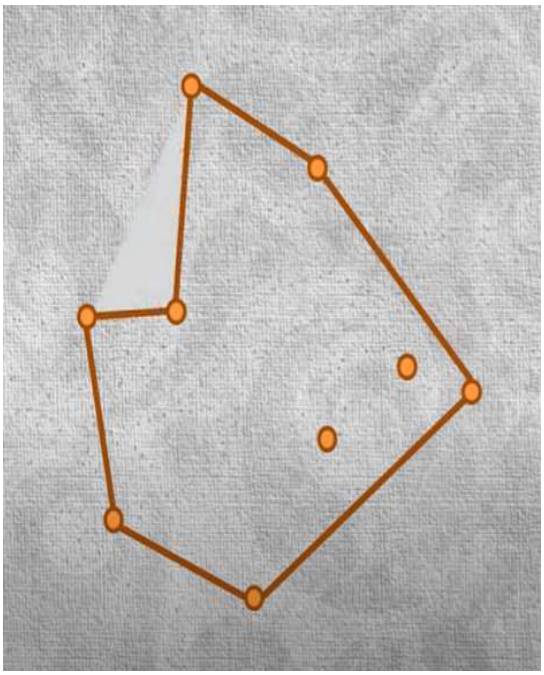
Consider the points given below:

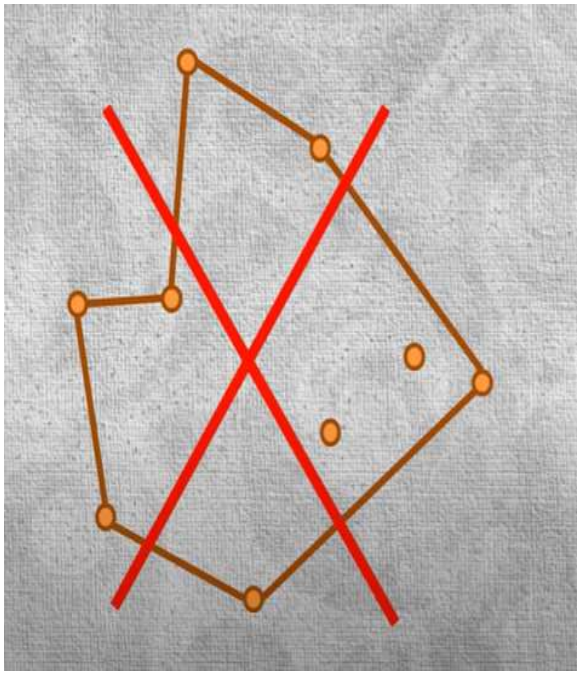


Convex hull of given set of points is as follows:

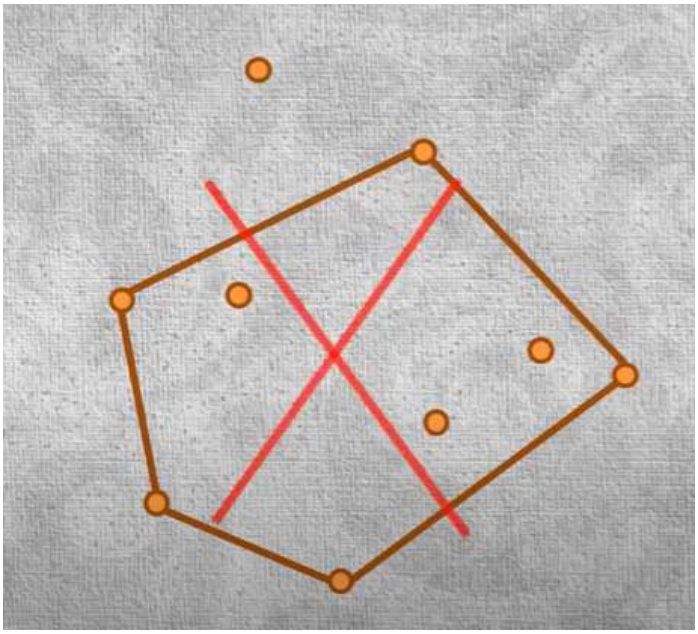


If some additional points are included, then area of polygon gets reduced and circumference gets increased as shown below:





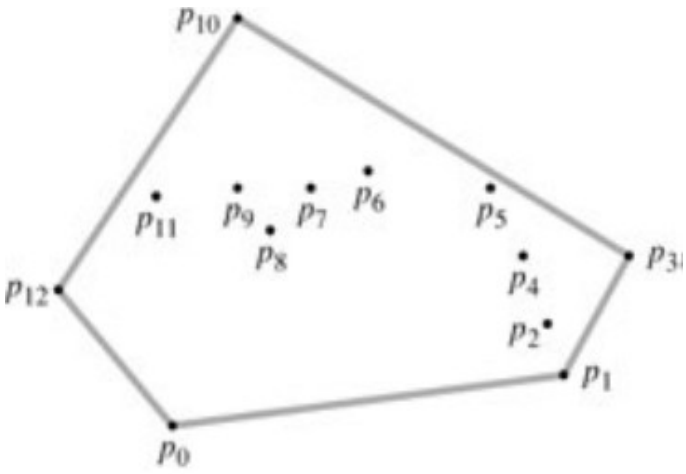
If any of the points are omitted, then the convex hull (resulting polygon) will not cover all the points.



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 its interior.

Example:



Applications:

Convex hulls have wide applications in mathematics, statistics, combinatorial optimization, economics, geometric modelling, and ethology.

- Robotics: Avoiding obstacles
- Graphics and Vision: Image and shape analysis
- Computational geometry: Many applications
  - Finding farthest pair of points in a set (Fact. Farthest pair of points are on convex hull.)

Convex Hull finding algorithms:

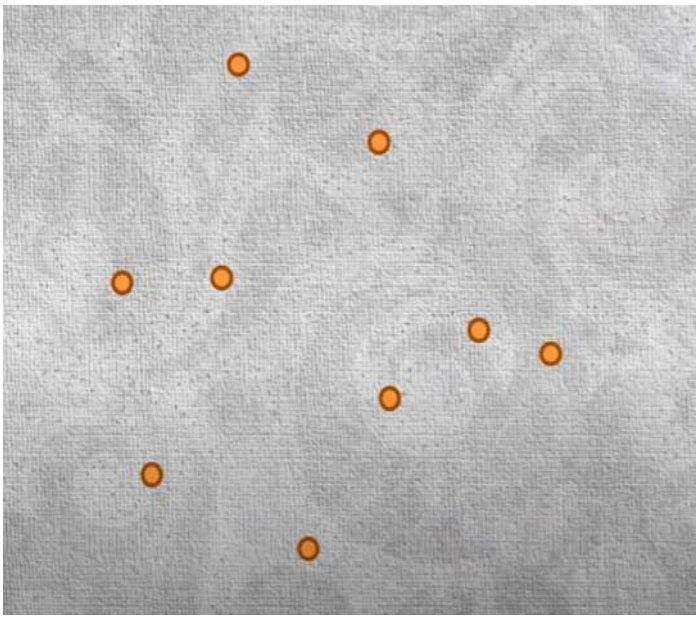
1. Graham's Scan Algorithm
2. Jarvis's March Algorithm (Gift wrapping algorithm)

Graham's Scan Algorithm:

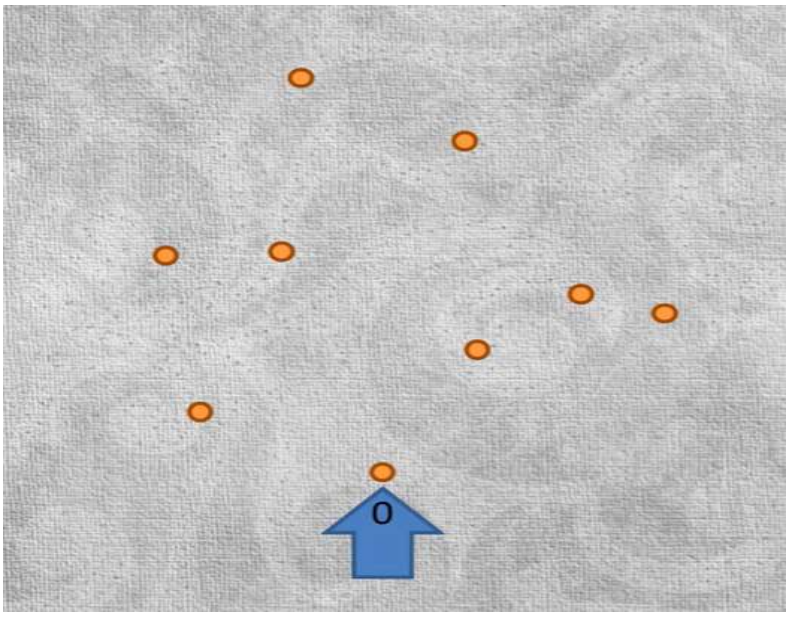
```
GRAHAM-SCAN(Q)
1  let p_0 be the point in Q with the minimum y-coordinate,
       or the leftmost such point in case of a tie
2  let ⟨p_1, p_2, ..., p_m⟩ be the remaining points in Q,
       sorted by polar angle in counterclockwise order around p_0
       (if more than one point has the same angle, remove all but
       the one that is farthest from p_0)
3  PUSH(p_0, S)
4  PUSH(p_1, S)
5  PUSH(p_2, S)
6  for i ← 3 to m
7      do while the angle formed by points NEXT-TO-TOP(S), TOP(S),
                  and p_i makes a nonleft turn
8          do POP(S)
9          PUSH(p_i, S)
10 return S
```

Example:

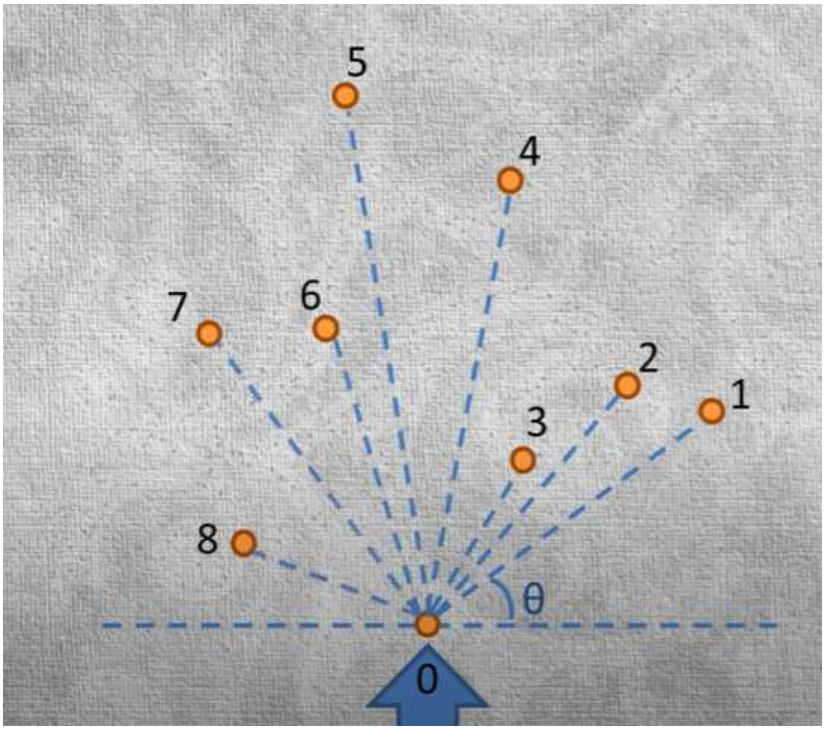
Set of points Q be as follows:



p0 be the point in Q with the minimum y-coordinate, or the leftmost such point in case of a tie



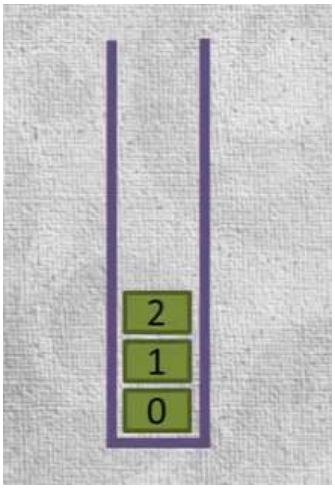
p1, p2, ..., pm be the remaining points in Q, sorted by polar angle in counter clockwise order around p0(if more than one point has the same angle, remove all but the one that is farthest from p0).



PUSH(p0, S)

PUSH(p1, S)

PUSH(p2, S)



For loop begins in the algorithm:

for $i \leftarrow 3$ **to** m
    do while the angle formed by points NEXT-TO-TOP(S), TOP(S), and p_i makes a nonleft turn
        do POP(S)
    PUSH(p_i, S)

m denotes the number of points other than *p*0 that remain.

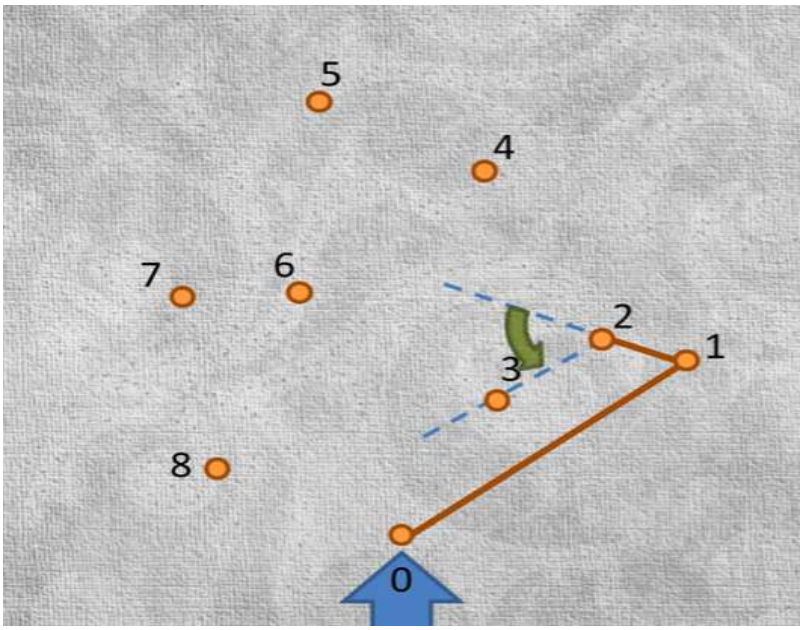
So, i in for loop can take the values from 3 to 8.

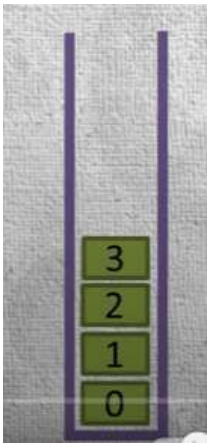
i = 3; so, $p_i = p_3 = 3$

Check while loop condition.

It is left turn. Condition is false. While loop terminates.

So, push 3 into stack.

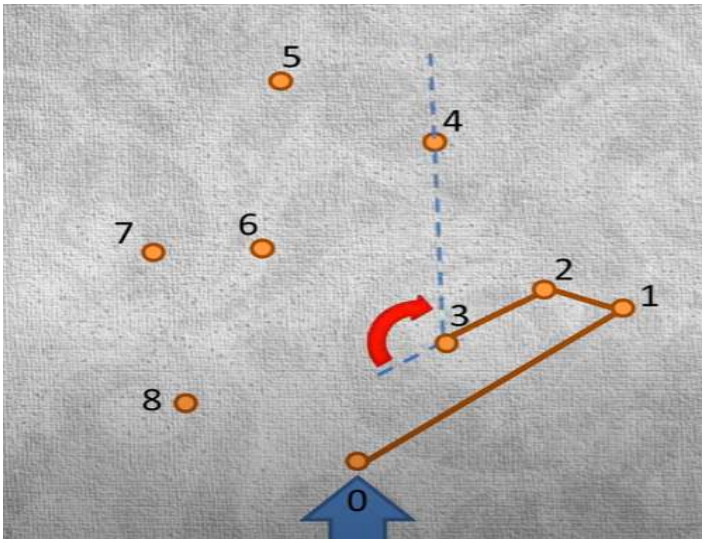


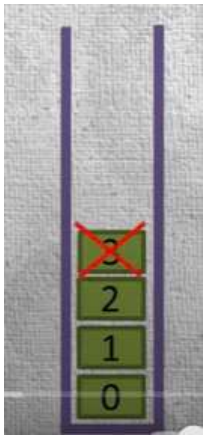


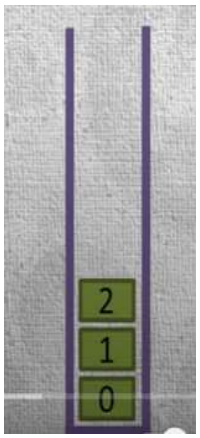
i = 4; so, $p_i = p_4 = 4$

Check while loop condition.

It is right turn (nonleft turn). Condition is true. So, pop from stack.



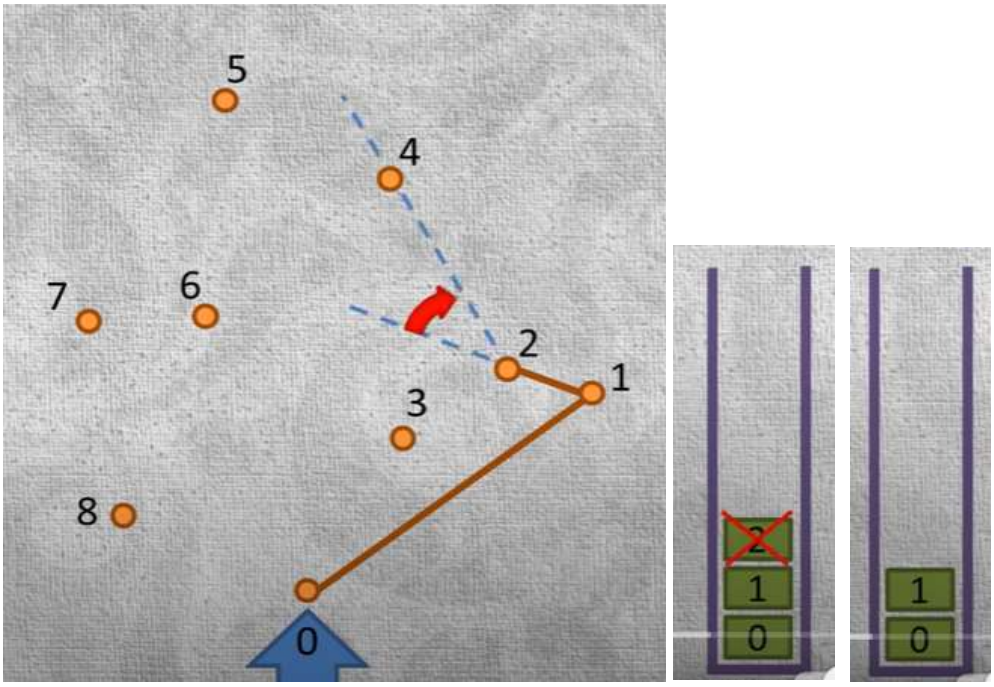




While loop continues.

Check while loop condition.

It is right turn (nonleft turn). Condition is true. So, pop from stack.

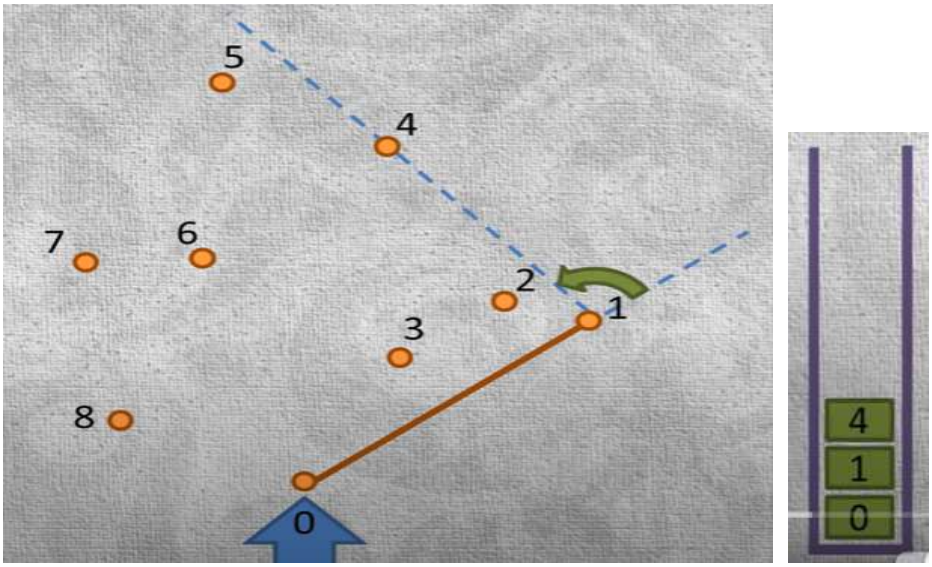


While loop continues.

Check while loop condition.

It is left turn. Condition is false. While loop terminates.

So, push 4 into stack.

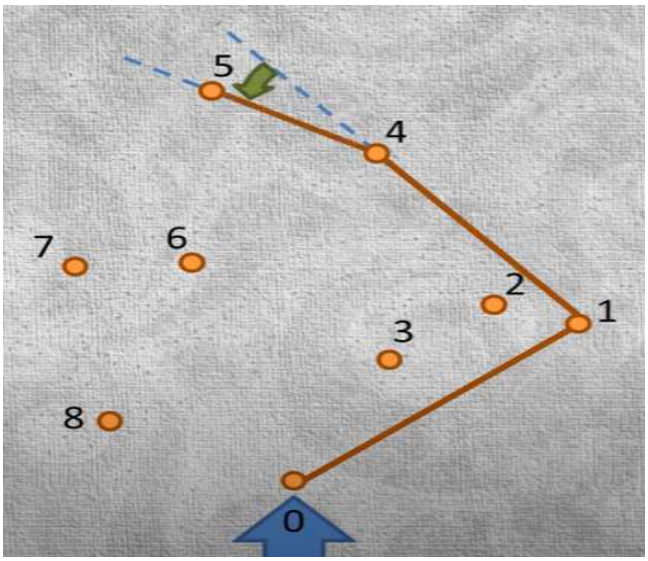


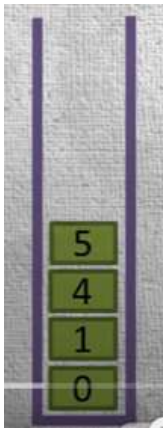
i = 5; so, $p_i = p_5 = 5$

Check while loop condition.

It is left turn. Condition is false. While loop terminates.

So, push 5 into stack.



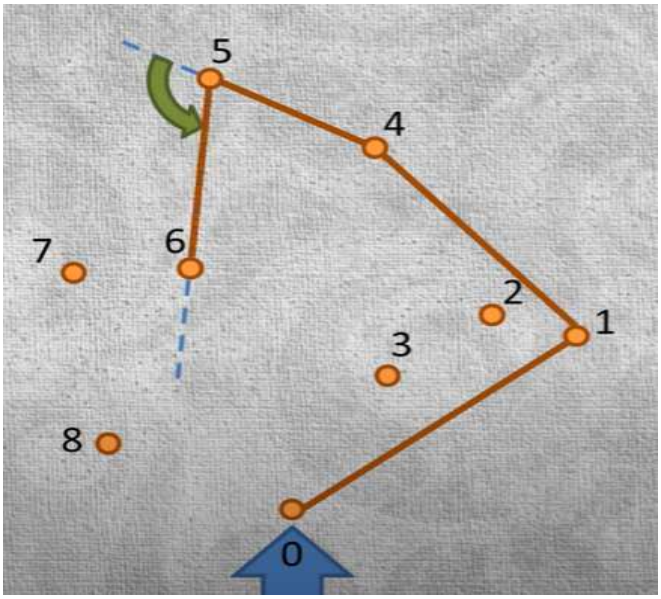


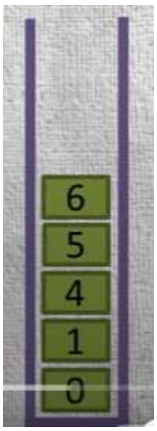
i = 6; so, $p_i = p_6 = 6$

Check while loop condition.

It is left turn. Condition is false. While loop terminates.

So, push 6 into stack.

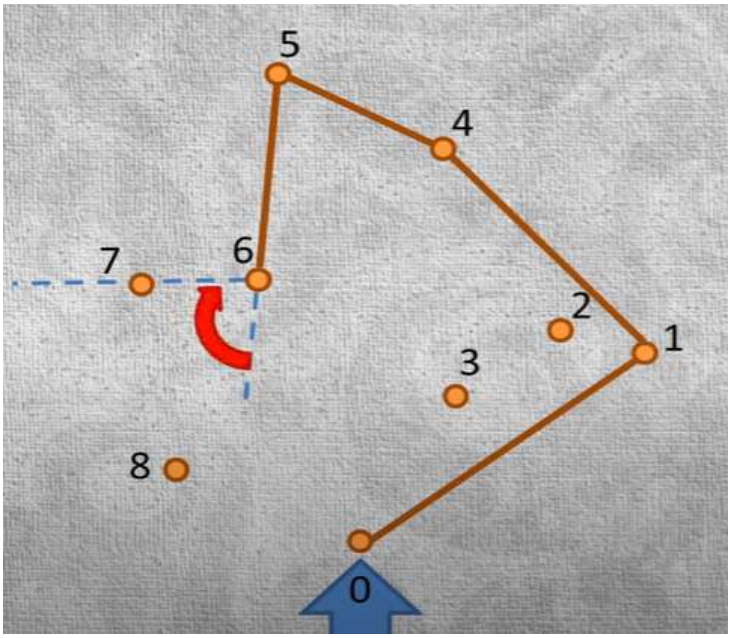


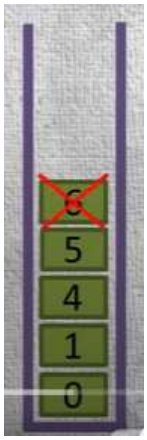


i = 7; so, $p_i = p_7 = 7$

Check while loop condition.

It is right turn (nonleft turn). Condition is true. So, pop from stack.



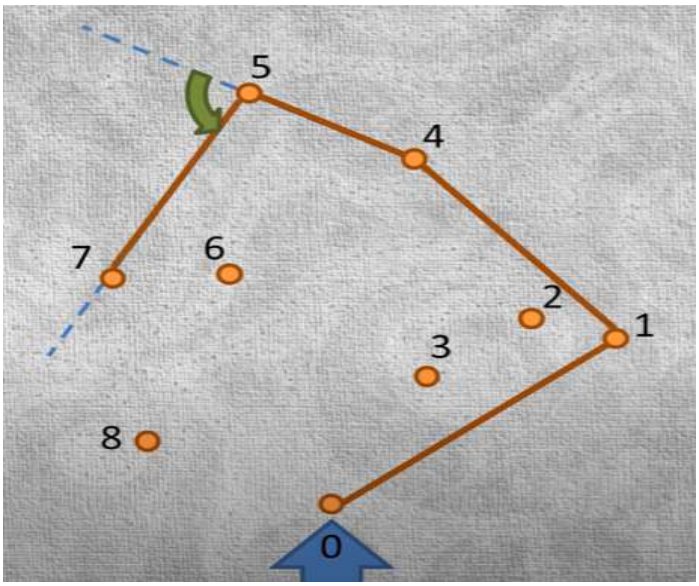


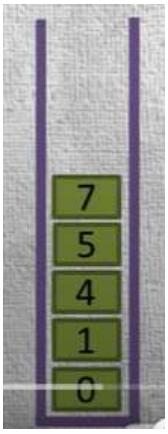
While loop continues.

Check while loop condition.

It is left turn. Condition is false. While loop terminates.

So, push 7 into stack.



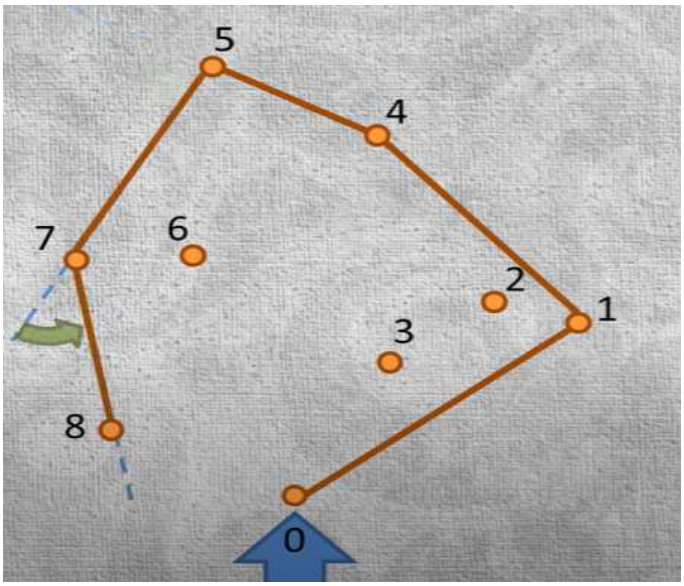


i = 8; so, $p_i = p_8 = 8$

Check while loop condition.

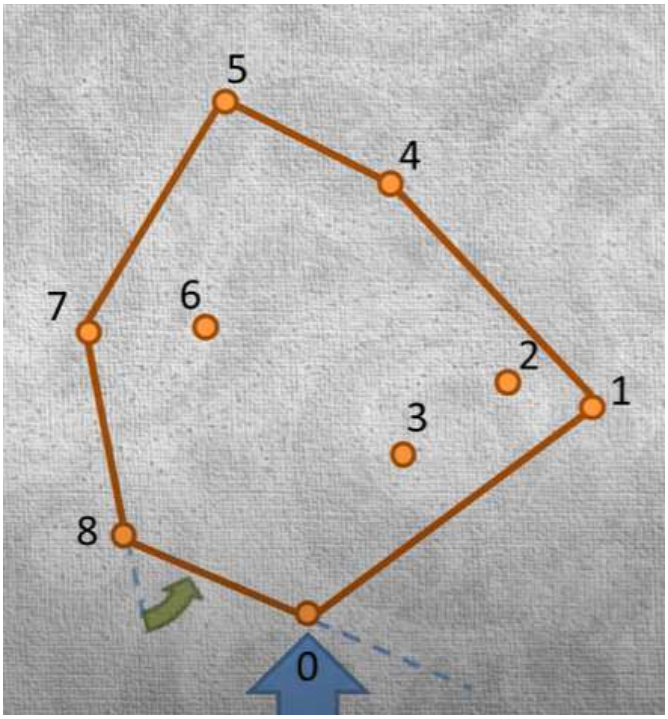
It is left turn. Condition is false. While loop terminates.

So, push 8 into stack.



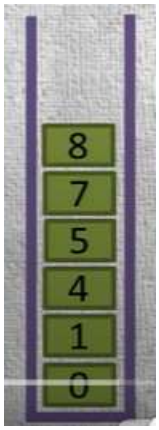
For loop terminates.

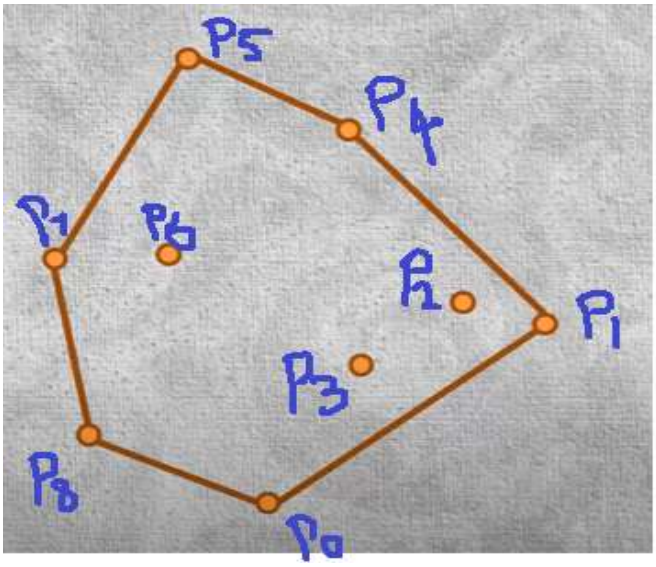
With this, we reach the starting point p_0.



Return S

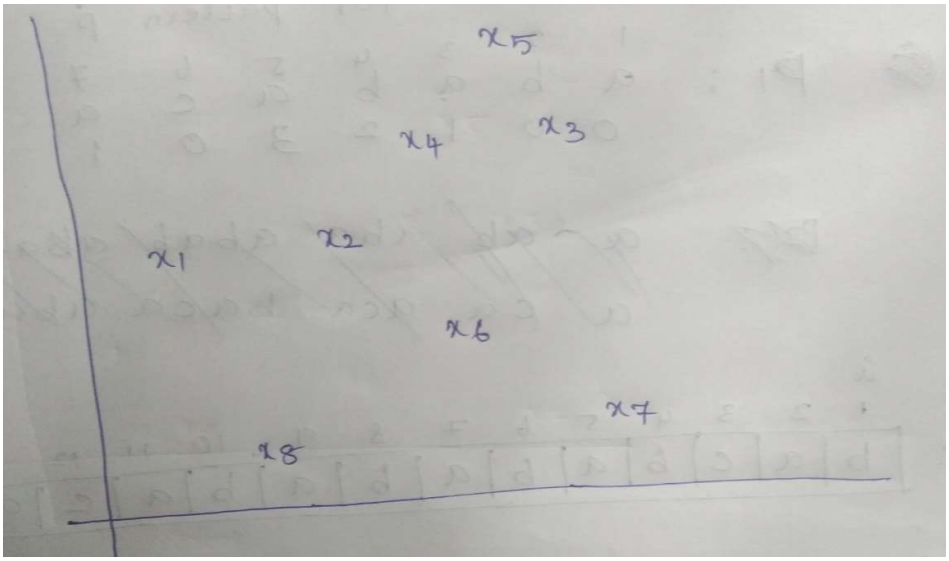
Solution – P_0, P_1, P_4, P_5, P_7, P_8





Jarvis's March Algorithm:

Consider the set of points given below:



We start with leftmost point or minimum y-coordinate point in case of a tie.

Here, the leftmost point is x1. So, we select x1.

Add x1 to the result

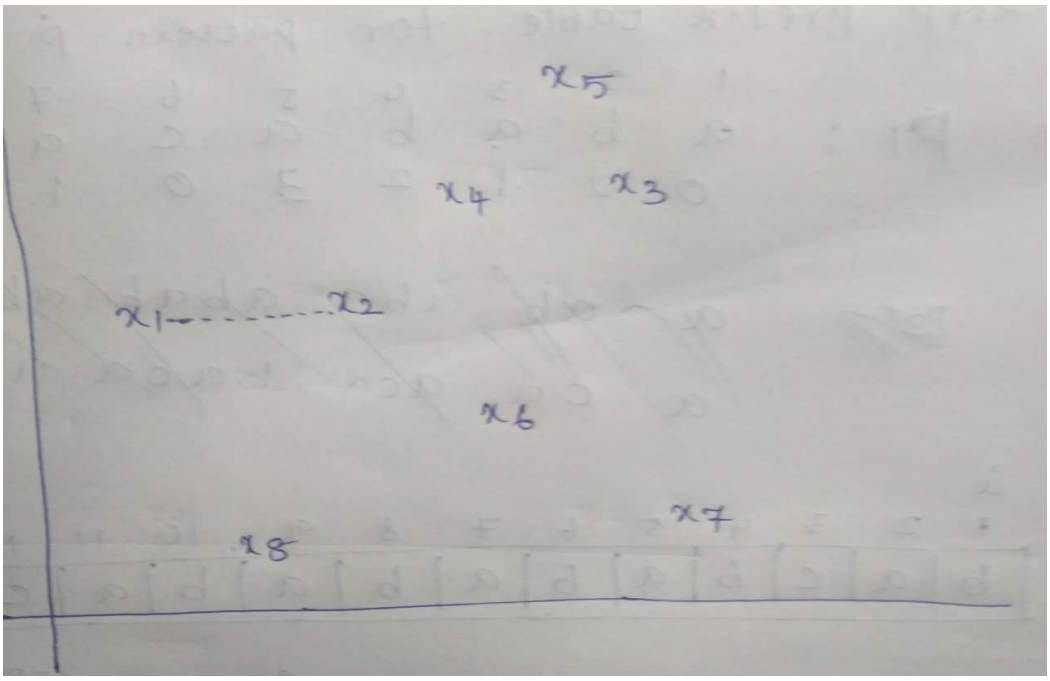
Result = x1

Point x1 is on the boundary of polygon.

From x1, we have to pick next point such that the next point must be the leftmost point from the x1's perspective.

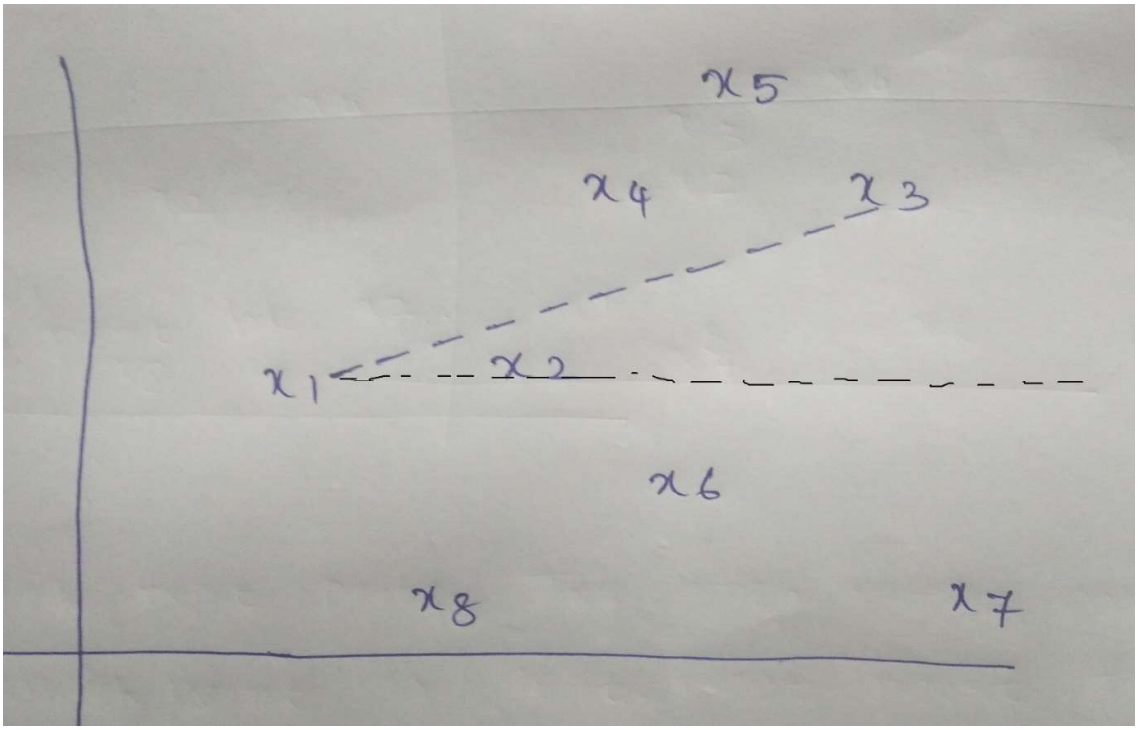
To determine the next point, try with every possibility (pick every other point i.e. brute force approach) and see which point is the leftmost point from x1.

Let us pick x2.

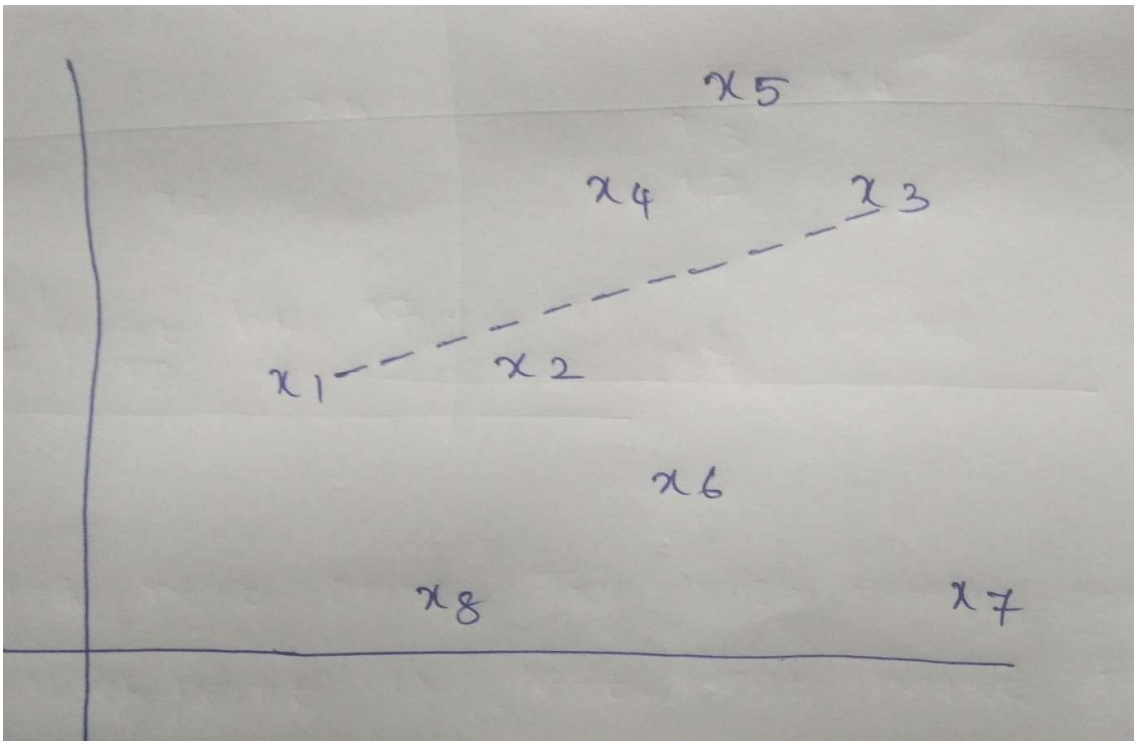


Next pick x3 and ask yourself is x3 is to the left of the line from x1 to x2.

Yes, it is in the left side.

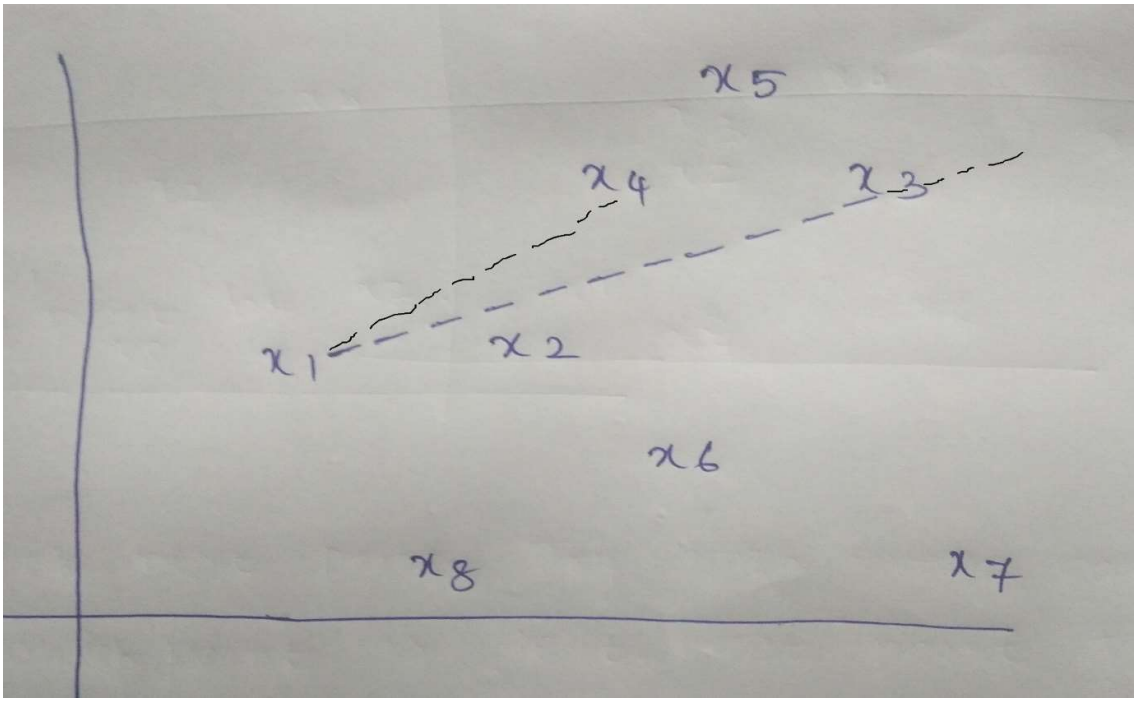


So, next point is not x2. It is x3.

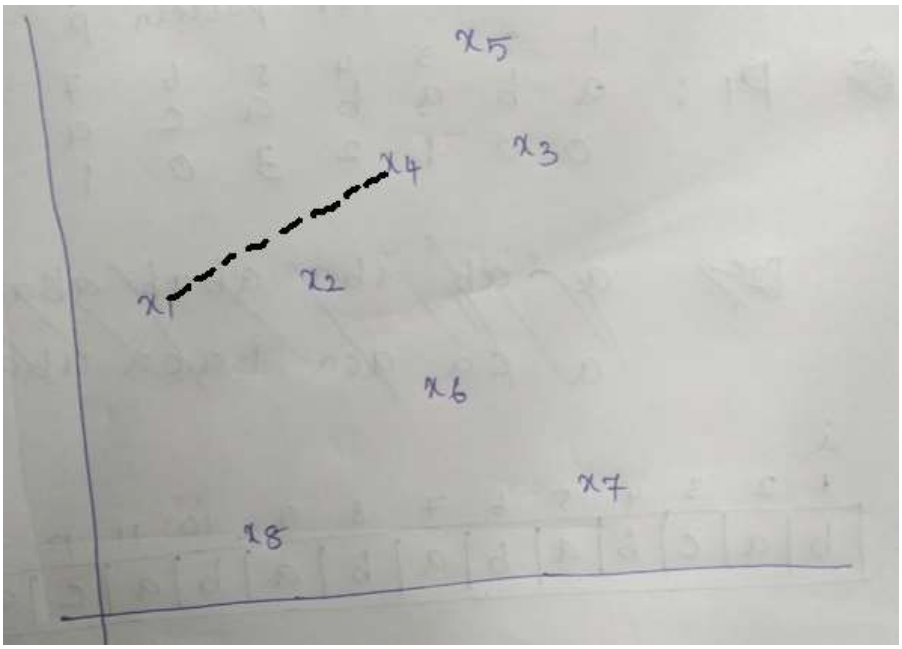


Next pick x4 and ask yourself is x4 is to the left of the line from x1 to x3.

Yes, it is in the left side.

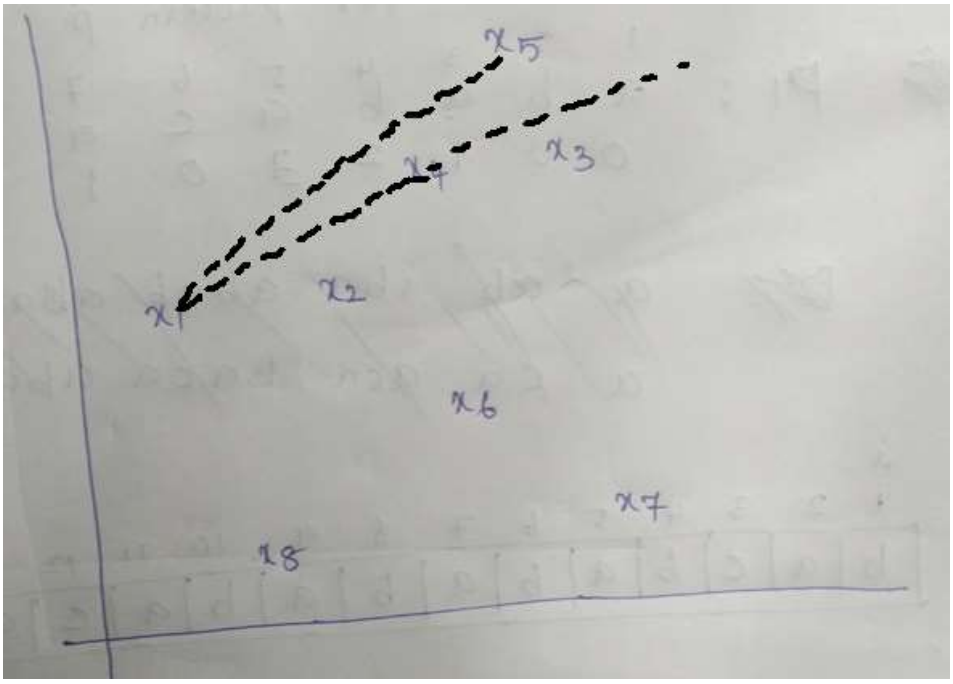


So next point is not x3. It is x4.

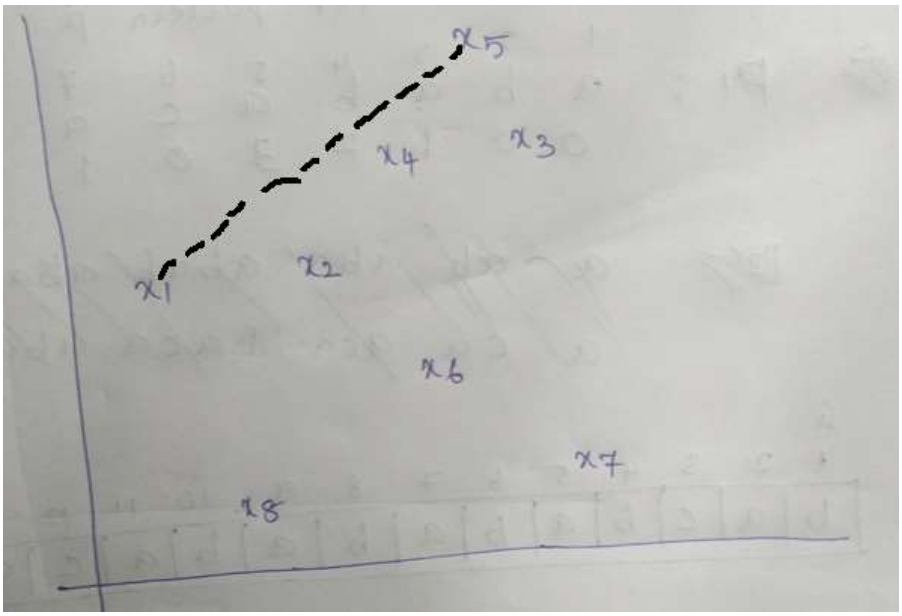


Next pick x5 and ask yourself is x5 is to the left of the line from x1 to x4.

Yes, it is in the left side.

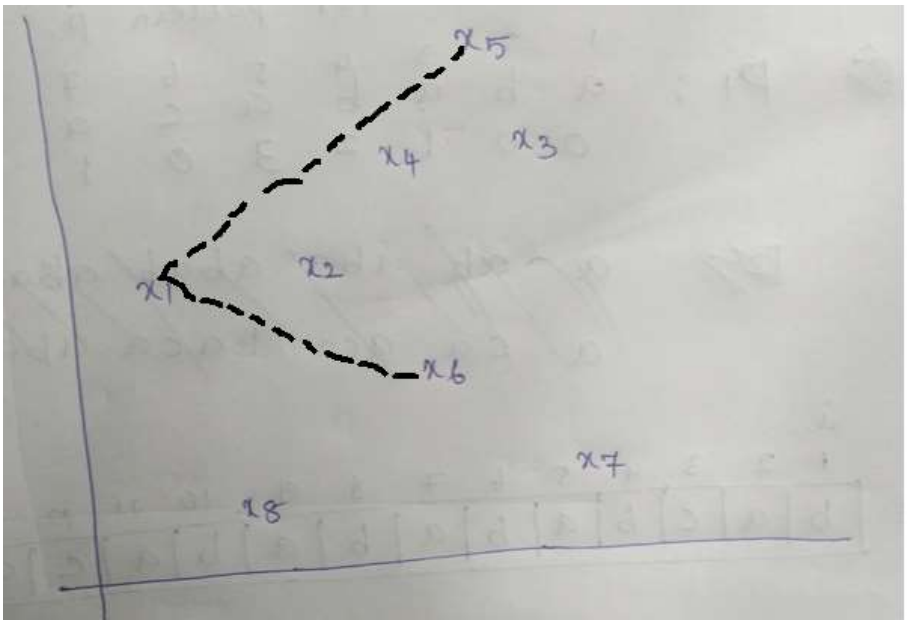


So next point is not x4. It is x5.



Next pick x6 and ask yourself is x6 is to the left of the line from x1 to x5.

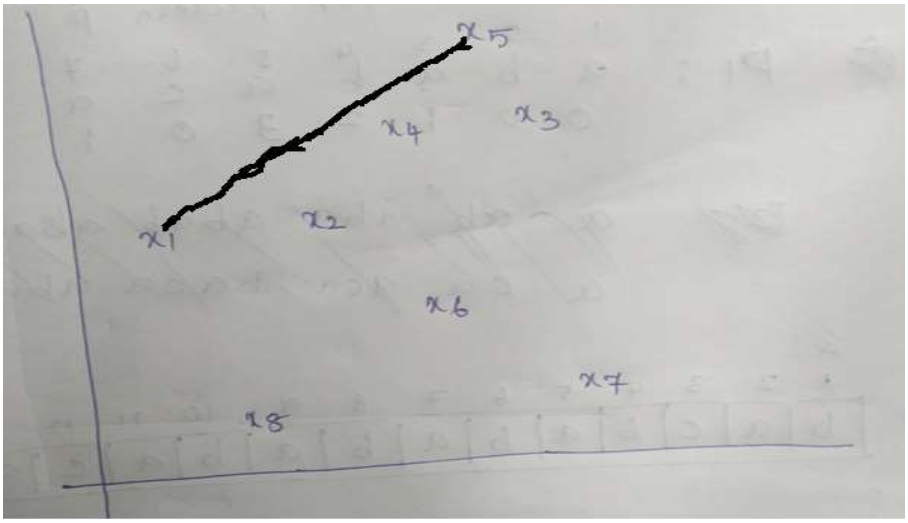
No, it is in the right side.



So, we don't consider x6.

Similarly, x7 and x8 are in the right side of line from x1 to x5. So, we don't consider.

So, after exploring all points, we choose our next point to be x5 (we make the line permanent).

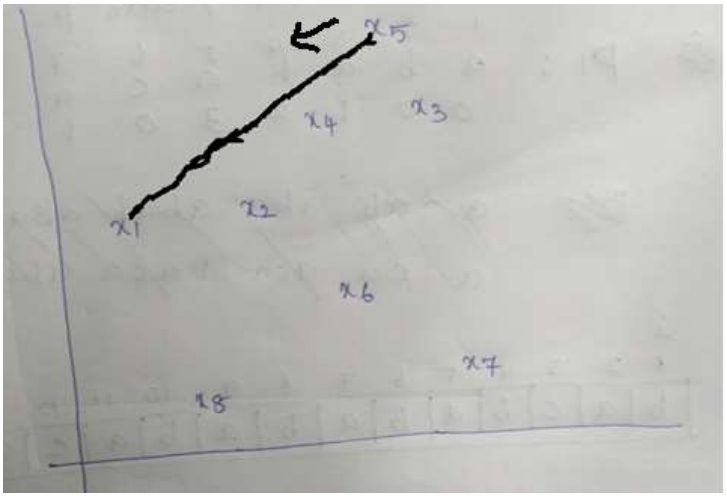


Add x5 to the result

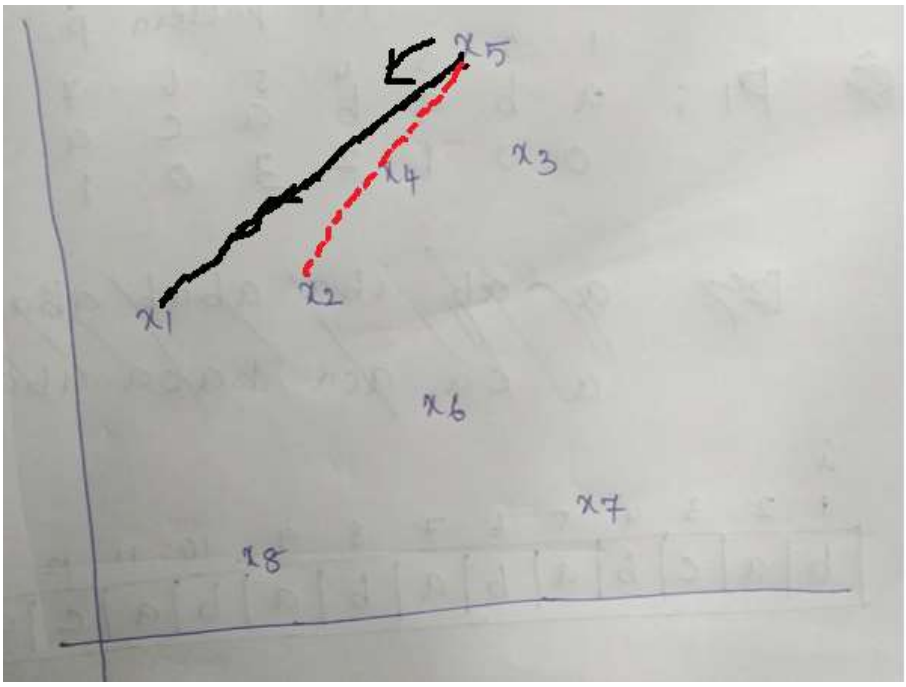
Result = x1, x5

Point x5 is on the boundary of the polygon.

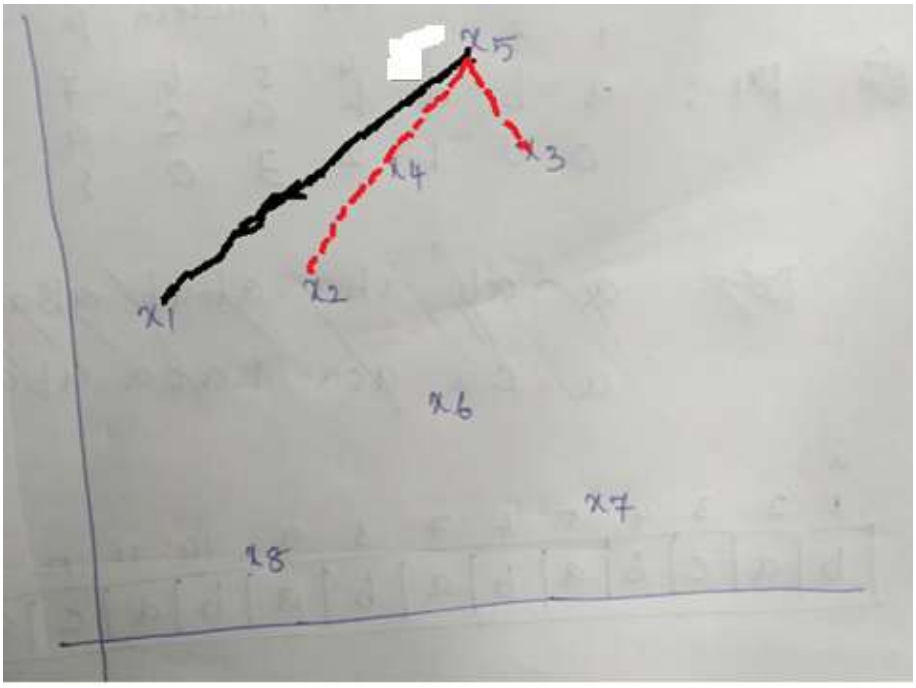
Now, repeat same process from x5's perspective. From x5 face x1 (First point chosen) and explore all other points.



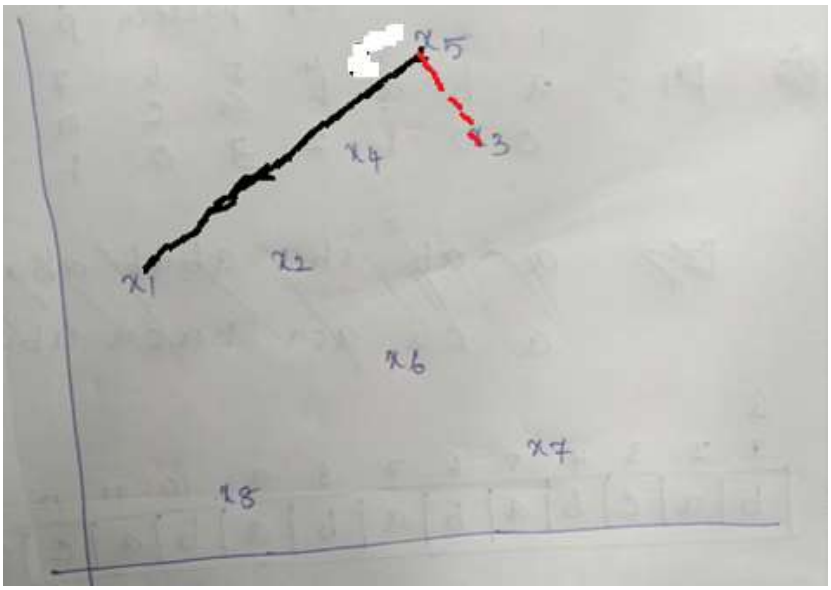
Let us pick x2. It is in the left of (x5, x1).



Next pick x3. It is in the left of (x5, x2).

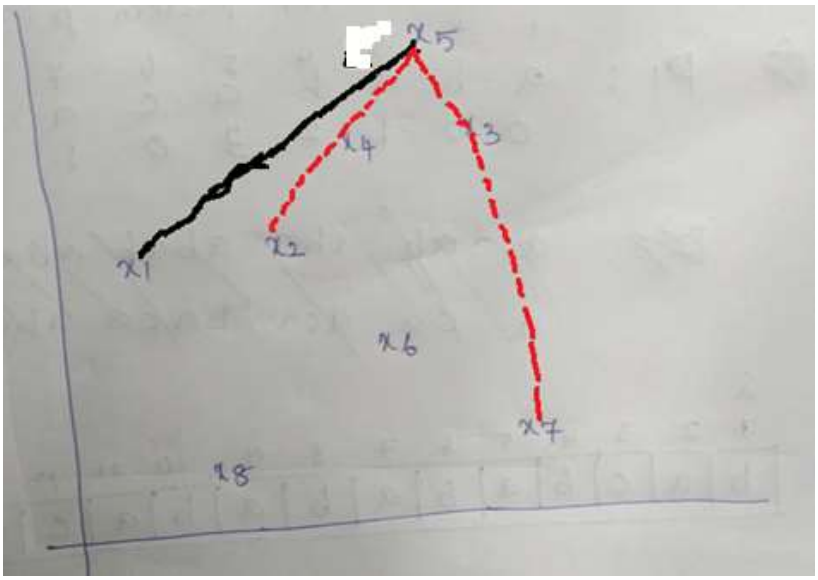


So, next point is not x2. It is x3.



Next, points x4 and x6. They are to the right of (x5, x3). So, ignore them.

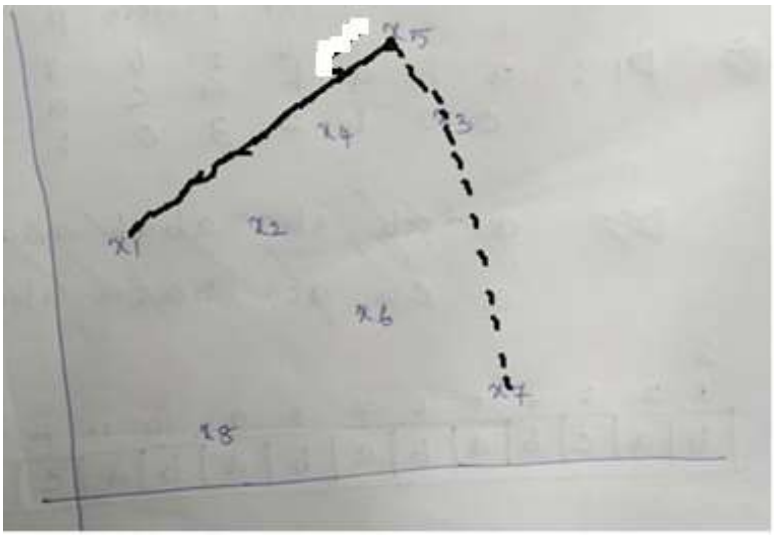
Next x7. Point x7 is on the exact same line of (x5, x3).



Points which are on the same line are called colinear points.

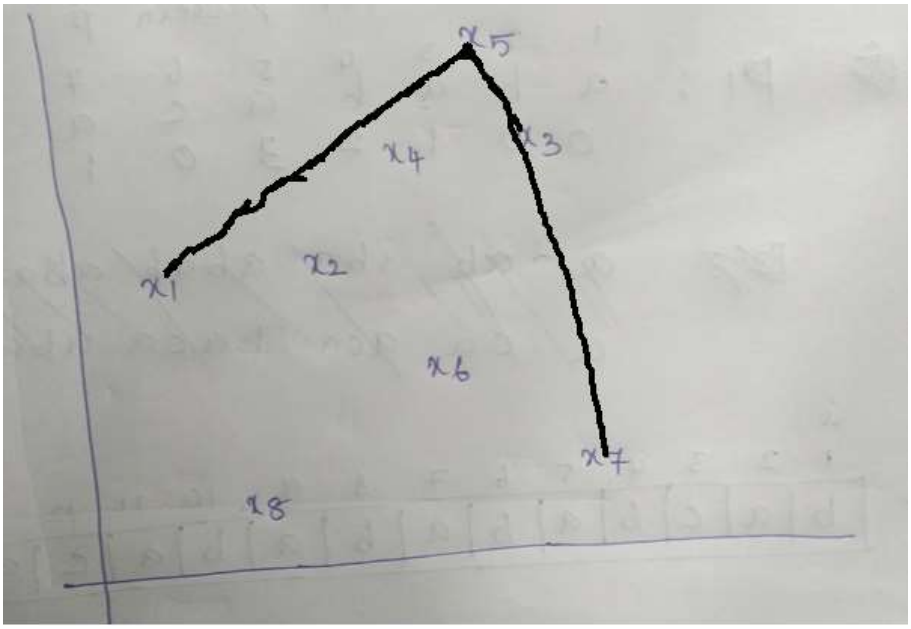
Farthest point from x5 is x7.

So, our next point is not x3. It is x7. Point x3 becomes colinear point.



Next pick x8. It is to the right of (x5, x7). So, ignore.

So, after exploring all points, we choose our next point to be x7 (we make the line permanent).

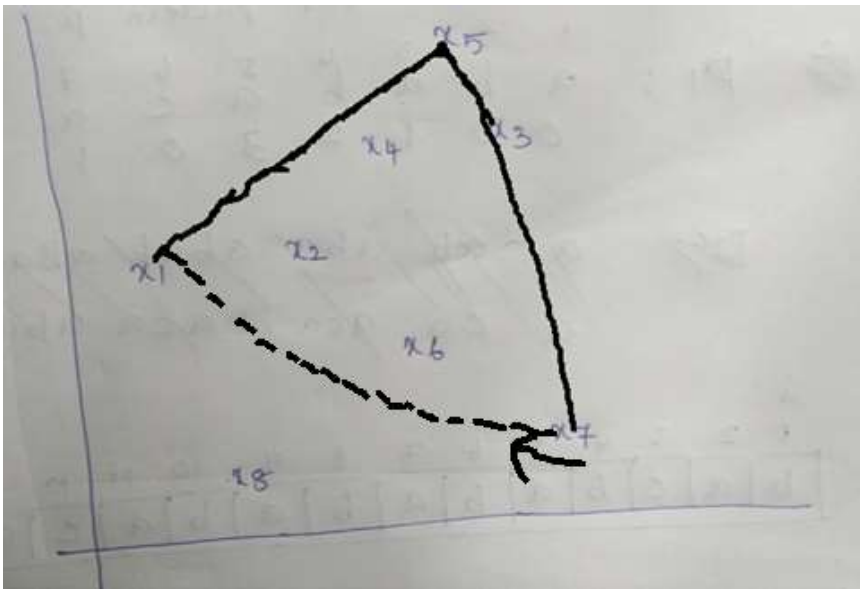


Add x7 to the result. Also, add colinear point x3 to the result.

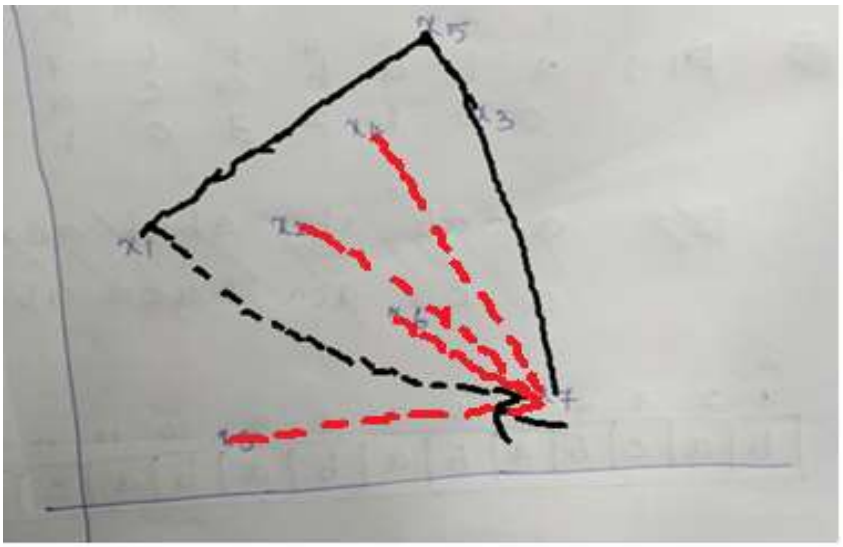
Result = x1, x5, x3, x7

Points x3 and x7 is on the boundary of the polygon

Now, repeat same process from x7's perspective. From x7 face x1 (First point chosen).

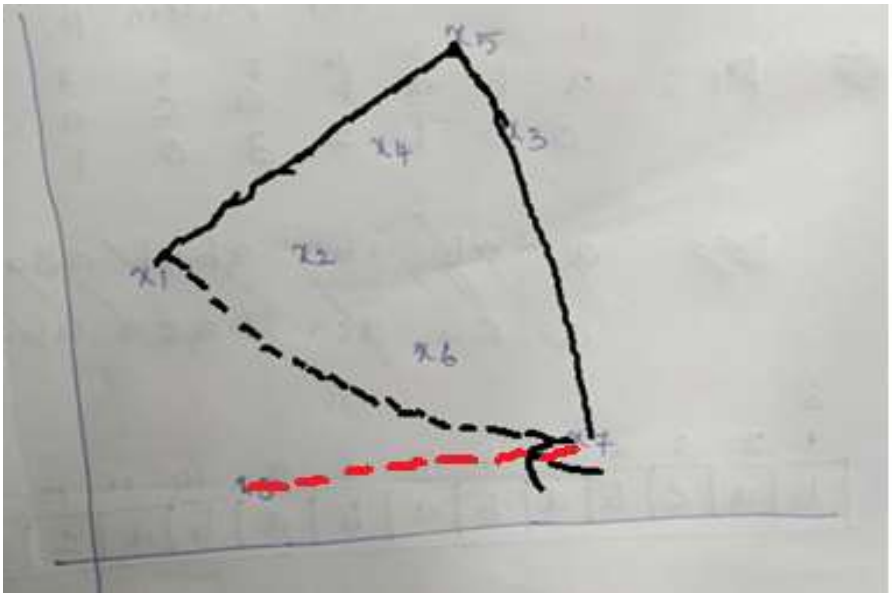


From x7 face x1 (First point chosen), explore all other points.

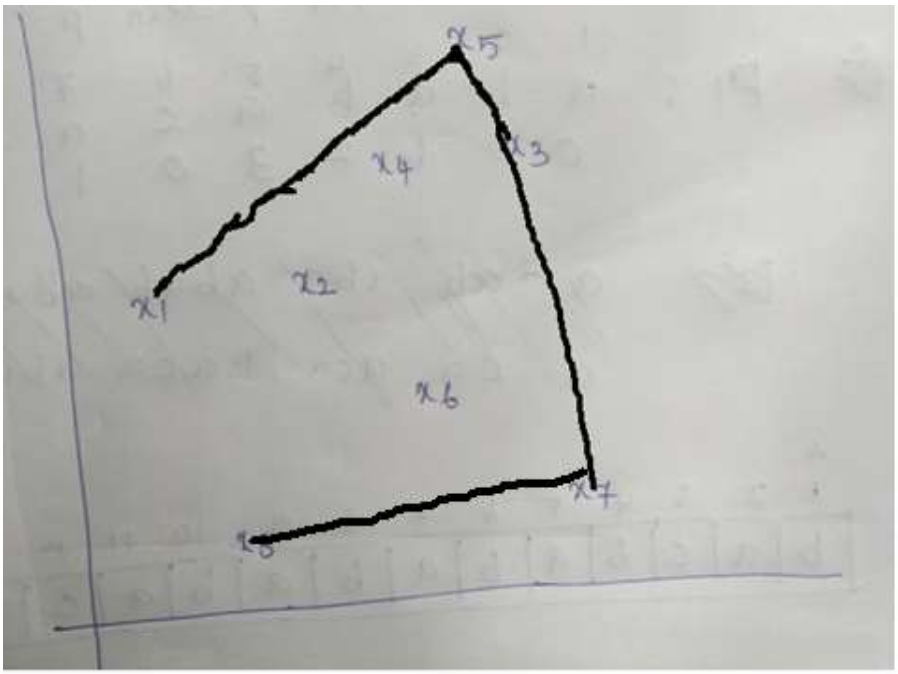


Points x2, x4, x6 are to the right of (x7, x1). So ignore them.

Point x8 is to the left of (x7, x1).



So, after exploring all points, we choose our next point to be x8 (we make the line permanent).

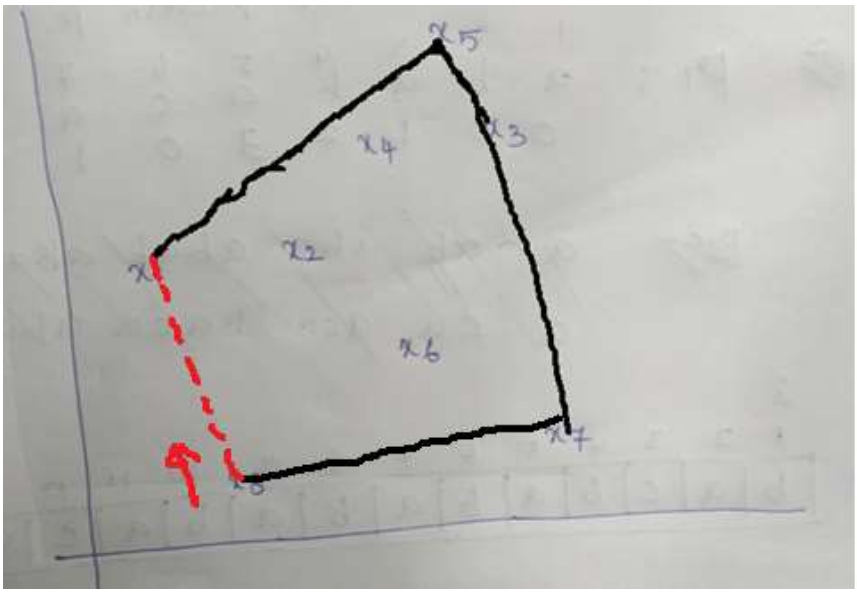


Add x8 to the result.

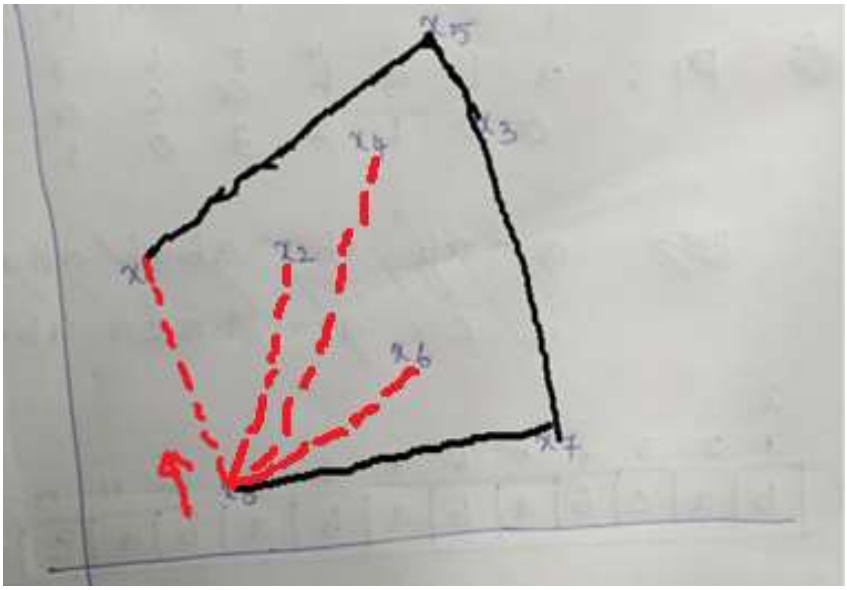
Result = x1, x5, x3, x7, x8

Point x8 is on the boundary of the polygon

Now, repeat same process from x8's perspective. From x8 face x1 (First point chosen).



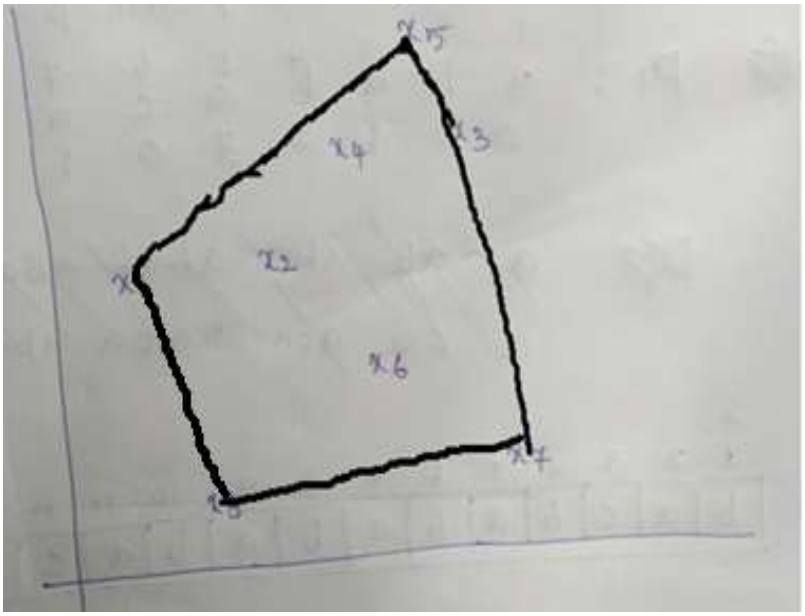
From x8 face x1 (First point chosen) and explore all other points.



Points x2, x4, x6 are to the right of (x8, x1). So ignore them.

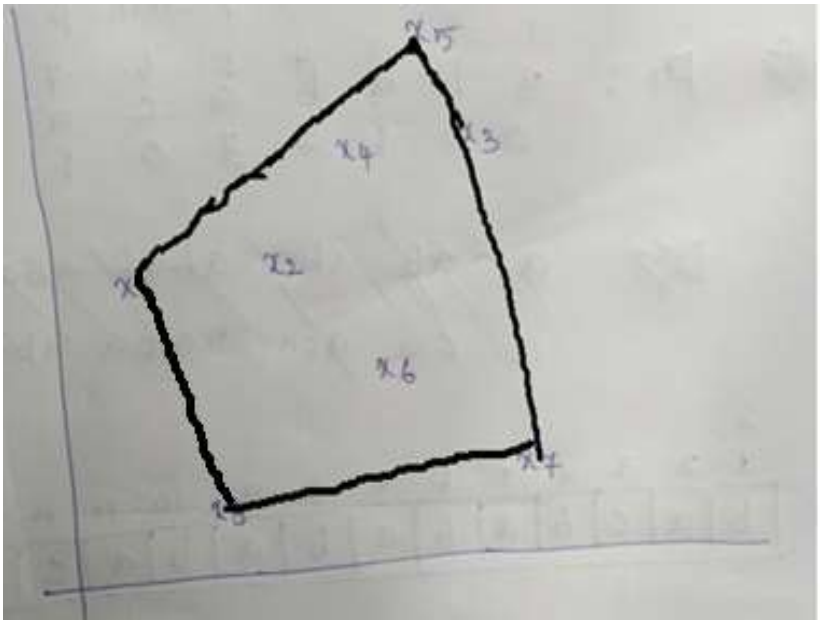
No points lie on the left side.

Now we have reached the starting point x1 (we make the line permanent).



Final Result:

Result = x1, x5, x3, x7, x8



Algorithm:

1) Initialize p as leftmost point.
2) Do following while we do not come back to the first (or leftmost) point.
    a) The next point q is the point such that the triplet (p, q, r) is counterclockwise for any other point r.
    b) next[p] = q (Store q as next of p in the output convex hull).
    c) p = q (Set p as q for next iteration).

Practice Problem:

Given the following set of points ‘P’ in a plane. Construct the smallest polygon Q such that each and every point in ‘P’ is either on the boundary or inside the polygon. Use the algorithm which starts with minimum x coordinate value and keep wrapping the points given below in counter clockwise direction to find the polygon.

(2,1)(3,3)(5,4)(-2,2)(-3,1)(-4,3)(-1,-2)(-3,-4)(2,-3)(4,-4)(1,-4)