GEOMETRY ALGO

Anu yadav

- Geometry Algo termed first coined by Marvin Minsky in his book perceptron, related to pattern recognition, to describe the algorithm for manipulating curve and surfaces in solid modelling.
- Applications
  - CAD – computer aided design
  - Computer vision
  - Image processing
  - Robotics
  - GIS – geographical information system
  - Computer Graphics
  - VLSI design
  - Statistics

- It is the study of algorithm to solve the problems in terms of geometry.
- Goal : to provide basic geometric tools needed from which application area can then build their programs.
- PRO:
  - Development of geometric tools
  - Efficiency proved
  - Correctness/Robustness
  - Linkage to discrete math's combo with geometry algo
- Limitations
  - Discrete geometry
  - Flat objects
  - Low dimensional spaces

Polygons

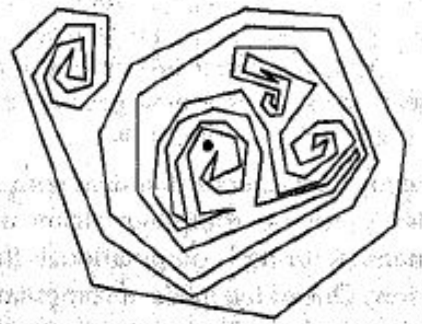
A polygon is just a collection of line segments, forming a cycle, and not crossing each other. We can represent it as a sequence of points, each of which is just a pair of coordinates. For instance the points (0, 0), (0, 1), (1, 1), (1, 0) form a square. The line segments of the polygon connect adjacent points in the list, together with one additional segment connecting the first and last point.

Not all sequences of points form a polygon ; for instance the points (0, −1), (0,1), (1,0), (0, −1) would result in two segments that cross each other. Sometimes we use the phrase simple polygon to emphasize the requirement that no two segments cross.

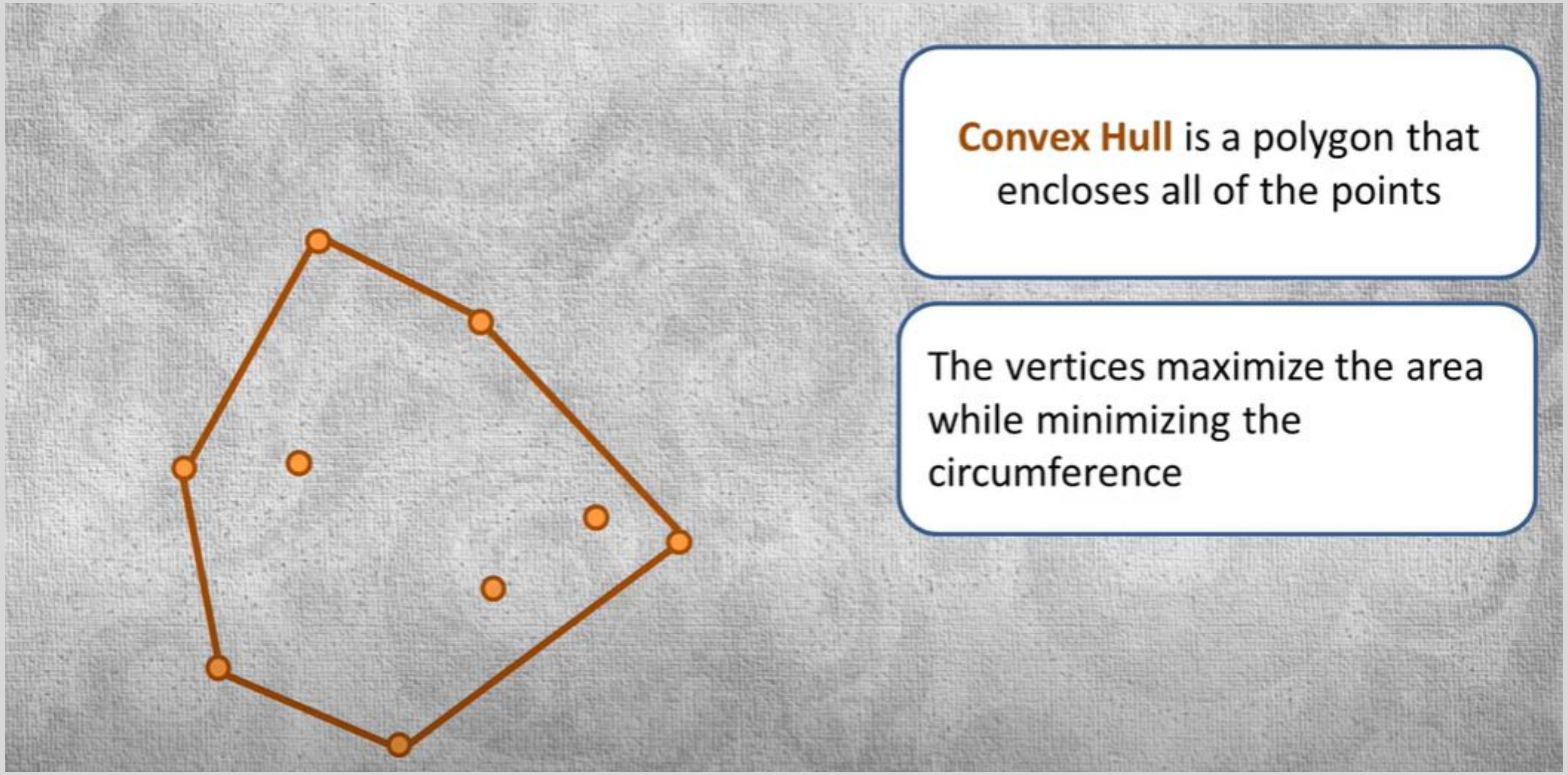
Testing if a point in a polygon

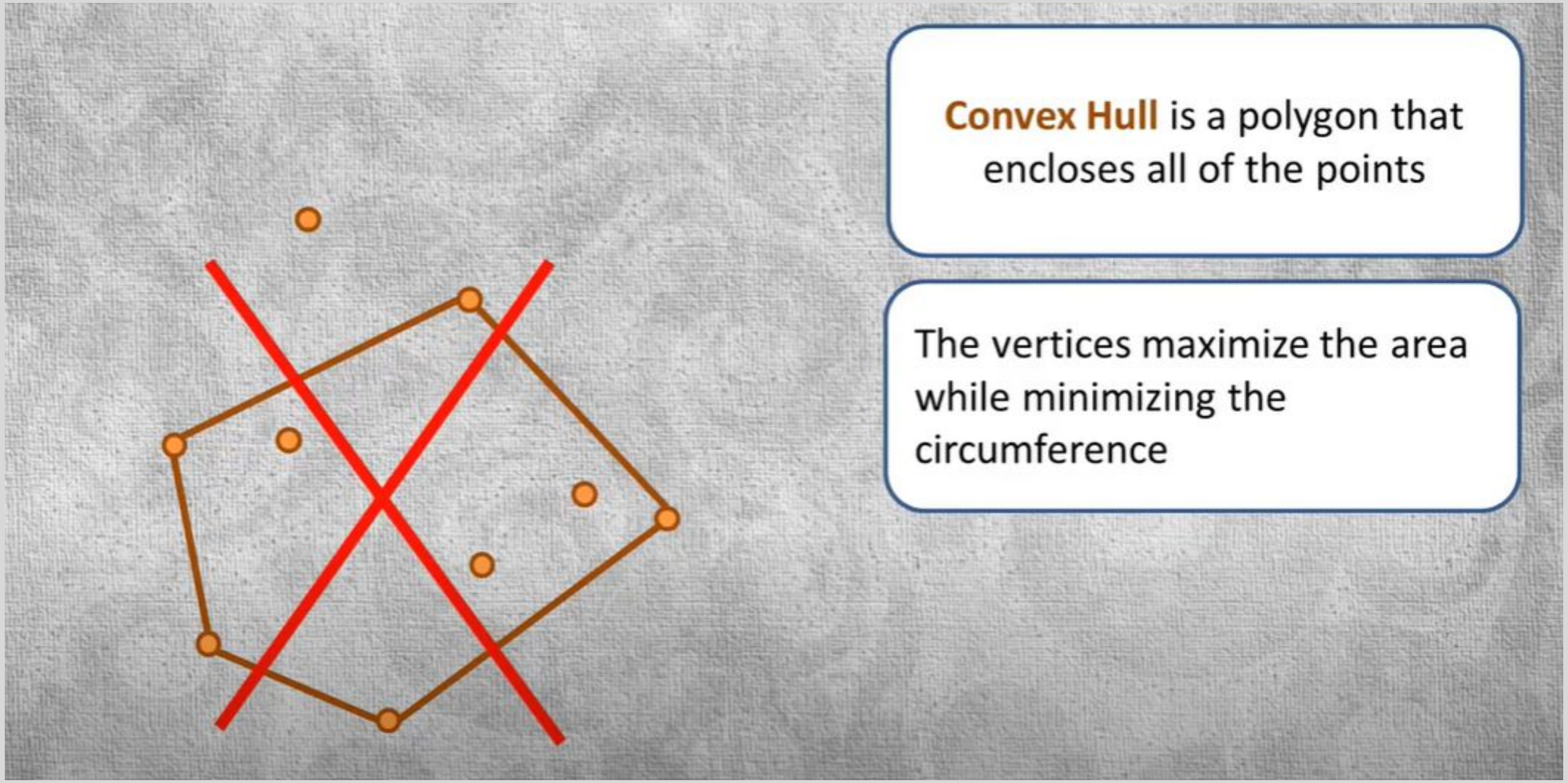
It is a famous theorem (the **Jordan curve theorem**) that any polygon cuts the plane into exactly two connected pieces : the *inside* and the *outside*. (The inside always has some finite size, while the outside contains points arbitrarily far from the polygon.) Actually, the Jordan curve theorem is more generally true of certain curves in the plane, not just shapes formed by straight line segments ; the more general fact is often proved by approximating these curves by polygons.

For uncomplicated enough polygons, it's easy to see by eyes, which parts of the place are inside and which are outside. But this is not always easy. For instance is the marked point inside the following polygon ?

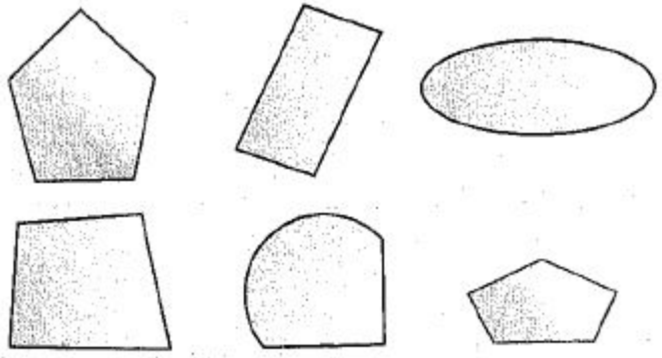


Convex hull


Convex Hull is a polygon that encloses all of the points
The vertices maximize the area while minimizing the circumference


Convex Hull is a polygon that encloses all of the points
The vertices maximize the area while minimizing the circumference

The following figures are convex.



ure 37.3

The following figures are concave.

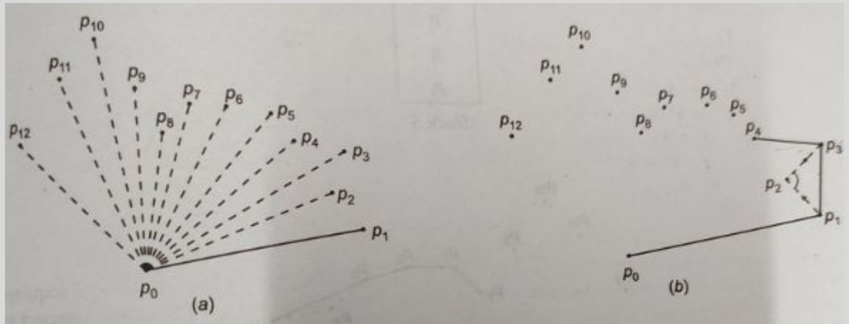
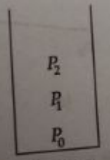
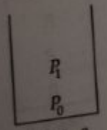


Figure 37.6



Stack S

$\angle P_1, P_2, P_3$ is clockwise turn *i.e.*, non-left turn. So pop p_2, now



Stack S

Now $\angle P_0, P_1, P_3$ is left turn, then push P_3 in the stack. Now the elements in the stack are

Stack S

$\angle P_1, P_3, P_4$ is counter-clock wise *i.e.*, left turn. So push p_4 in the stack and the elements in the stack are

Stack S

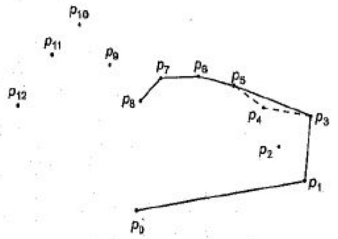


Figure 37.6

(c)

$\angle P_3\ P_4\ P_5$ clockwise turn *i.e.*, non left turn. So pop the stack. Now the elements in the stack

$\angle P_1 P_3 P_5$ is **left** turn. So push P_5 in the stack, now stack S contains

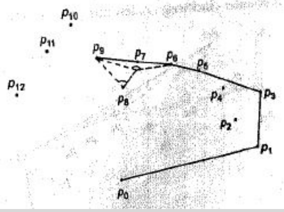
Stack S

$\angle P_3 P_5 P_6$ is also left turn so push P_6 in the stack.

$\angle P_5 P_6 P_7$ is also left turn push P_7 in the stack and $\angle P_6 P_7 P_8$ is also left turn push P_8 in the stack and now, the stack S contains

Stack S

$\angle P_7 P_8 P_9$ is clockwise turn *i.e.*, non left turn. So pop the stack then $\angle P_6 P_7 P_9$ is also non left turn, so again pop the stack. Now, the stack S contains.



P_6
P_5
P_3
P_1
P_0

Stack S

$\angle P_5\ P_6\ P_9$ is left turn so push P_9 in the stack and now the elements of stack S are

P_9
P_6
P_5
P_3
P_1
P_0

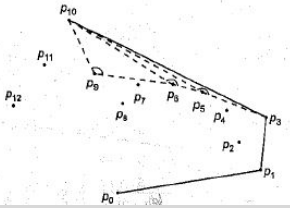
Stack S

Now $\angle P_6\ P_9\ P_{10}$ is non left turn so pop P_9 and $\angle P_5\ P_6\ P_{10}$ is also non left turn and $\angle P_3\ P_5\ P_{10}$ is non left turn also, thus pop P_6 and P_5 respectively and the stack S contains.

P_3
P_1
P_0

Stack S

Now, $\angle P_1, P_3, P_{10}$ is counter clockwise *i.e.*, left turn. So push P_{10} into the stack



Now the stack contains.

P_{10}
P_3
P_1
P_0

Stack S

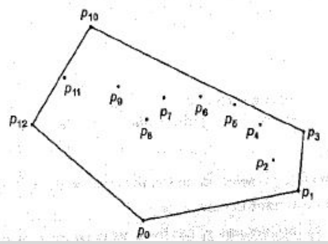
$\angle P_3\ P_{10}\ P_{11}$ is counter clock wise that is left turn so push P_{11} into the stack and stack contains.

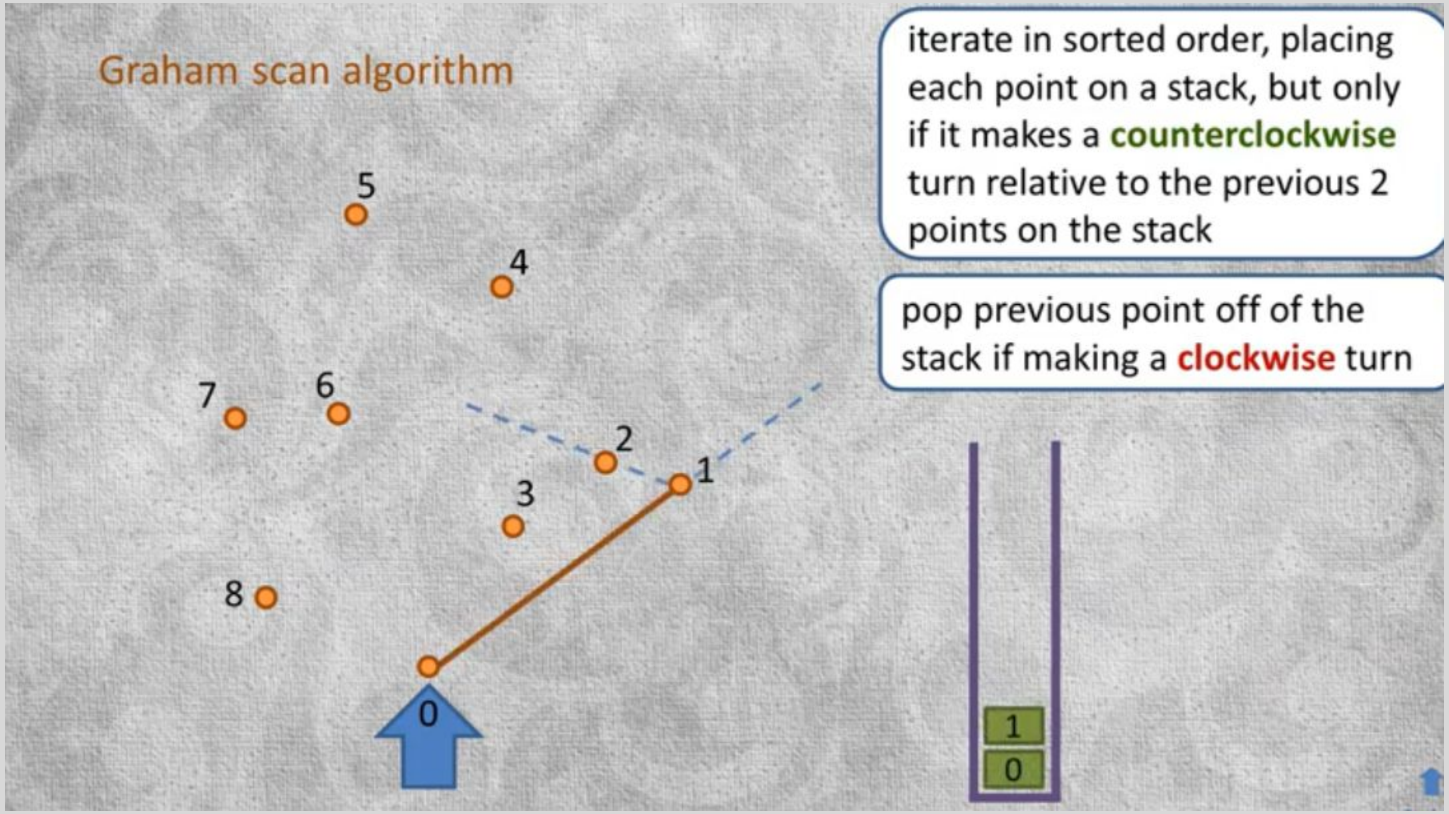
P_{11}
P_{10}
P_3
P_1
P_0

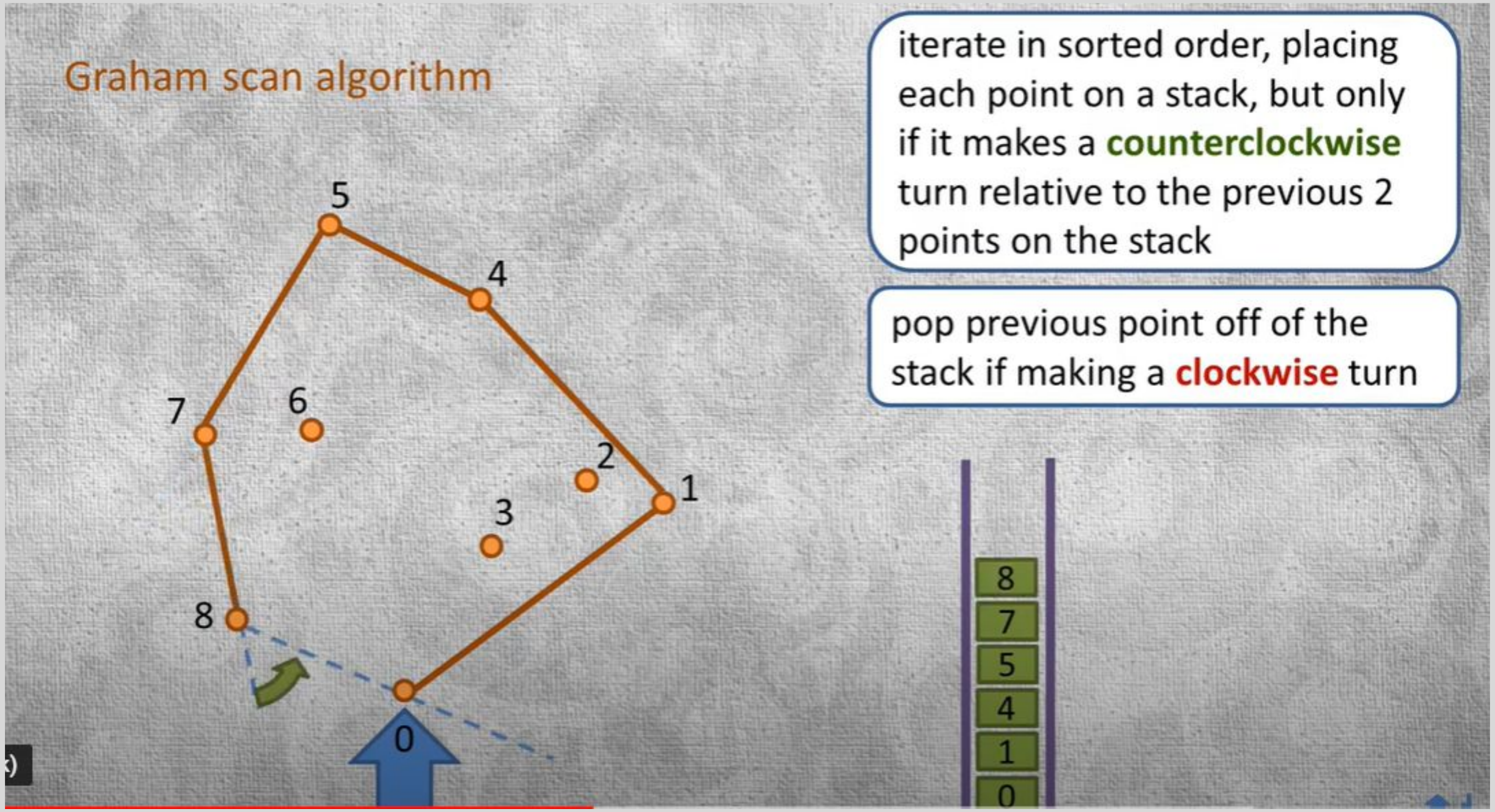
Stack S

$\angle P_{10}\ P_{11}\ P_{12}$ is clock wise that is not left turn so pop the element P_{11} and $\angle P_3\ P_{10}\ P_{12}$ is anticlockwise.

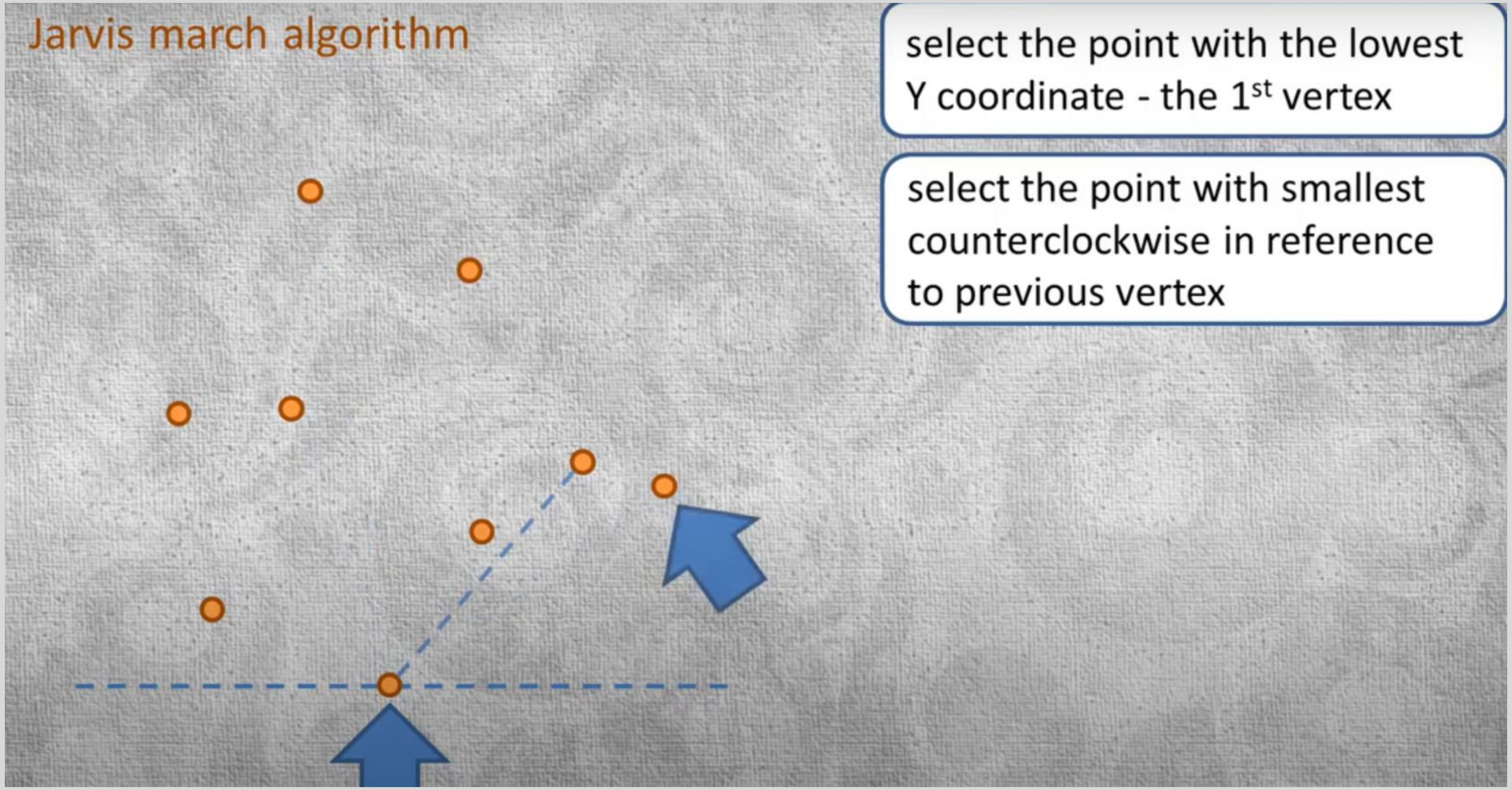
Hence push this into stack S and the final convex hull returned by the produce is show in the figure.



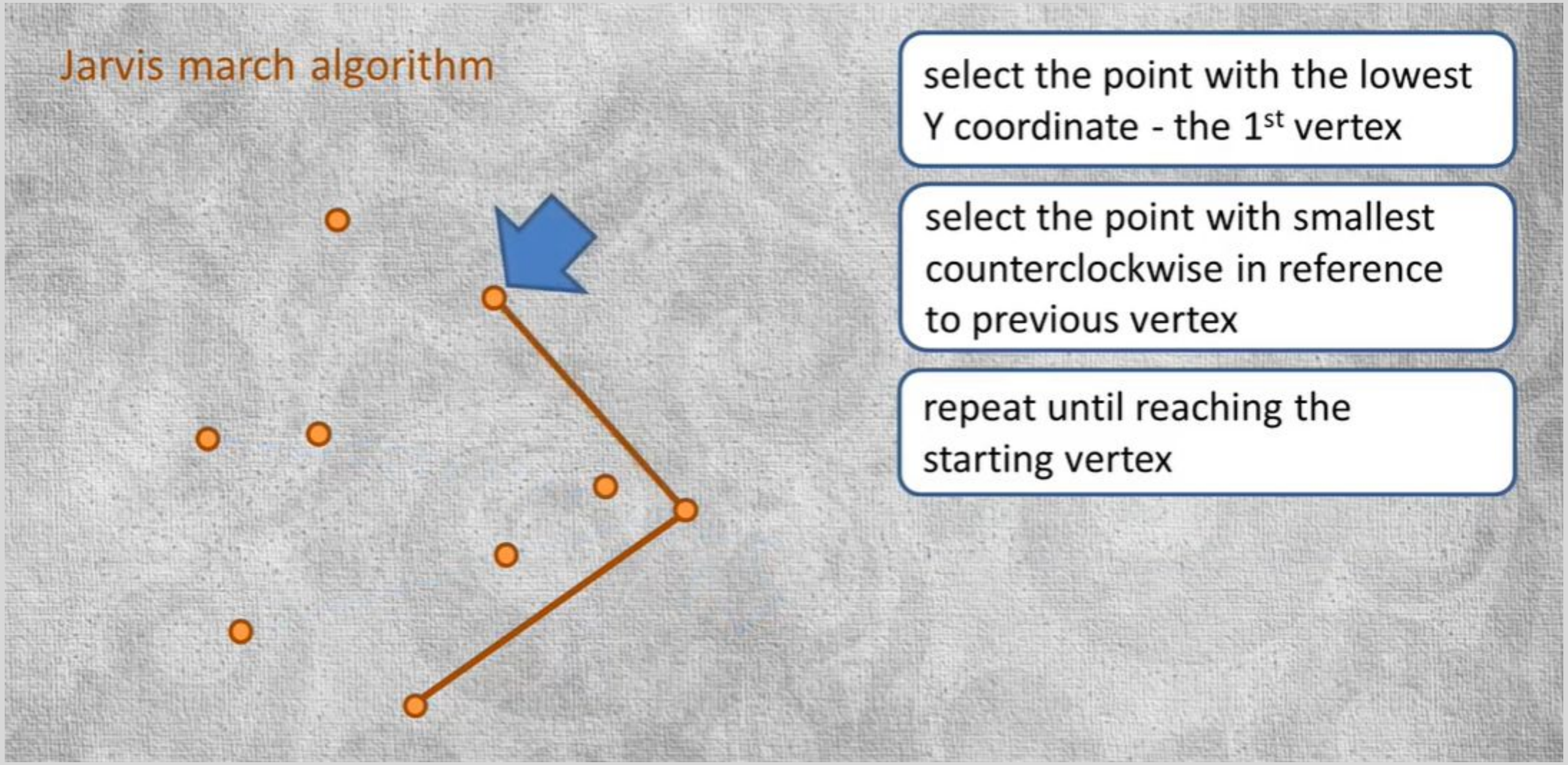

Graham scan algorithm
iterate in sorted order, placing each point on a stack, but only if it makes a counterclockwise turn relative to the previous 2 points on the stack
pop previous point off of the stack if making a clockwise turn
5
4
7
6
2
1
3
8
0
1
0


Graham scan algorithm
iterate in sorted order, placing each point on a stack, but only if it makes a counterclockwise turn relative to the previous 2 points on the stack
pop previous point off of the stack if making a clockwise turn
5
4
7
6
2
1
3
8
0
8
7
5
4
1
0

Jarvis march algo

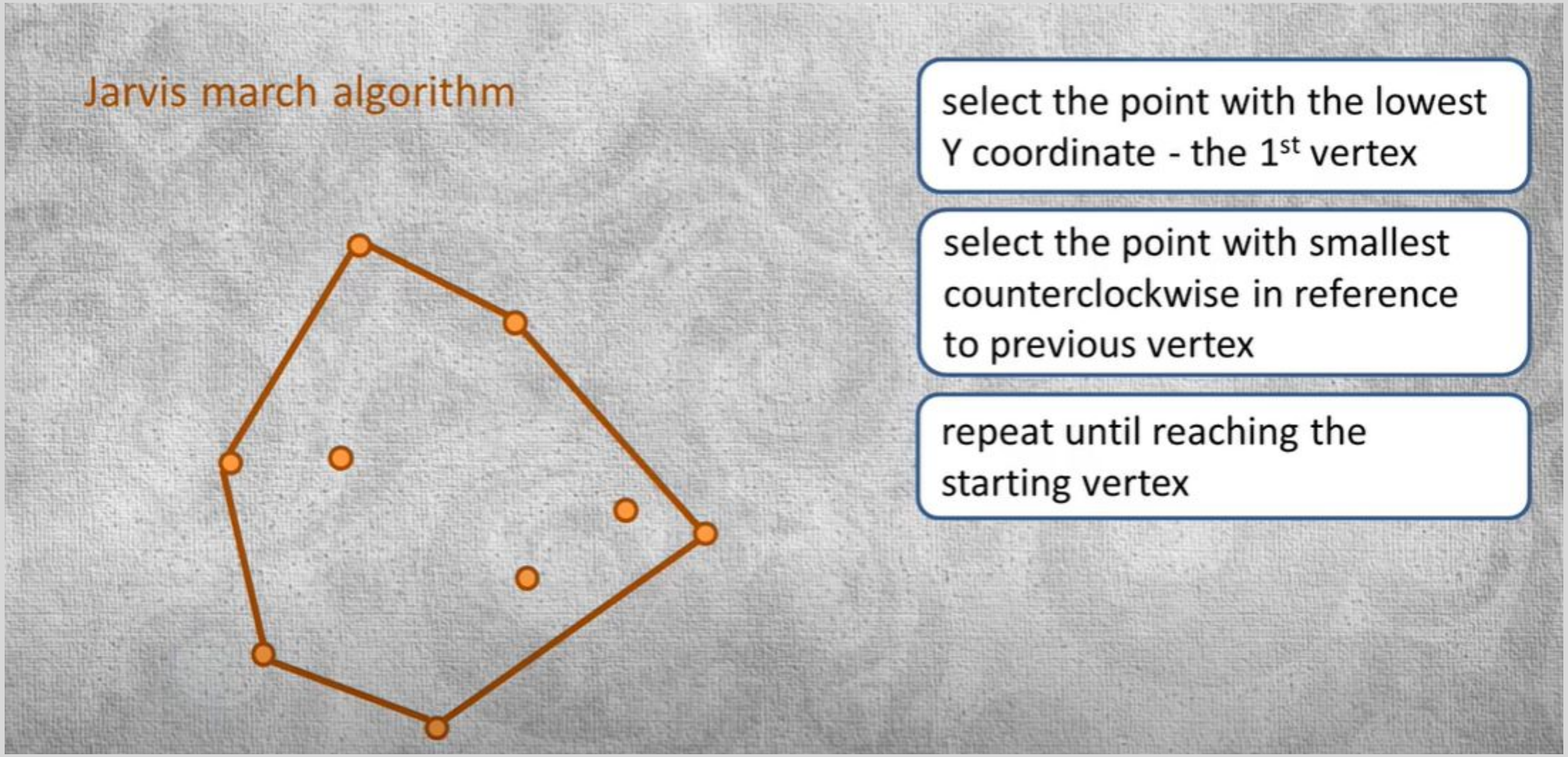


Cont.



Jarvis march algorithm
select the point with the lowest Y coordinate - the 1st vertex
select the point with smallest counterclockwise in reference to previous vertex
repeat until reaching the starting vertex

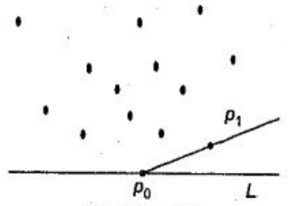
Cont.


Jarvis march algorithm
select the point with the lowest Y coordinate - the 1st vertex
select the point with smallest counterclockwise in reference to previous vertex
repeat until reaching the starting vertex

Jarvis's march computes the convex hull of a set Q of points by a technique known as package wrapping (or gift wrapping). The algorithm runs in time $O(nh)$ where h is the number of vertices of $CH(Q)$. The algorithm operates by considering any one point that is on the hull, say, the lowest point. We then find the "next" edge on the hull in counterclockwise order. Assuming that $p(k)$ and $p(k-1)$ were the last two points added to the hull, compute the point q that maximizes the angle $[p(k-1)\,p(k)\,q]$. Thus, we can find the point q in $O(n)$ time. After repeating this h times, we will return back to the starting point and we are done. Thus, the overall running time is $O(nh)$. Note that if h is $o(\log n)$ (asymptotically smaller than log n) then this is a better method than Graham's algorithm Jarvis's march is asymptotically faster than Graham's scan.

The basic idea is as follows :

- Locating the point p_0 with minimum y-coordinate (or Start at some extreme point, which is guaranteed to be on the hull).
- Let L be the horizontal line through p_0, L is clearly tangent to $CH(S)$ at p_0. We orient L from left to right. Then we perform "wrapping" step to locate point p_1 that forms the smallest counterclockwise angle with L Point p_1 must also be on $CH(S)$ (See Fig).
- Repeat the wrapping step at p_1 with line pp_1 until we return to the point p_0



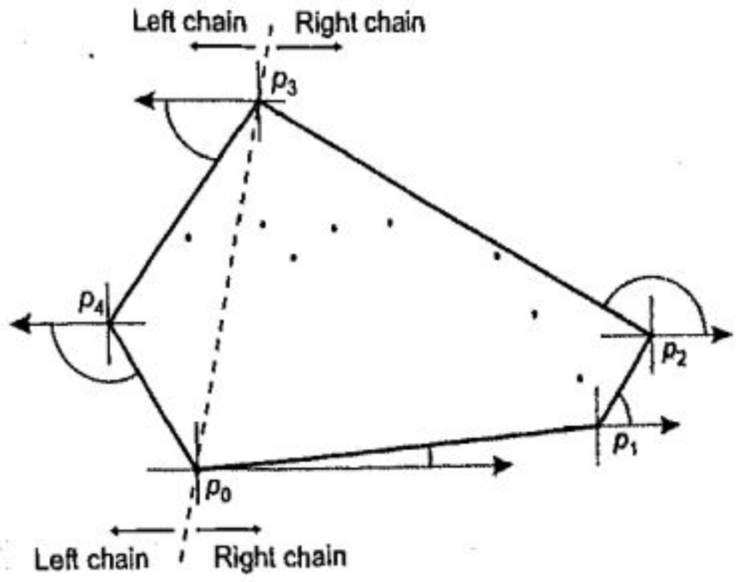
Jarvis' march

Because this process marches around the hull in counter-clockwise order, like a ribbon wrapping itself around the points, this algorithm also called the "gift-wrapping" algorithm.

Algorithm Jarvis March

- First, a base point p_0 is selected, this is the point with the minimum y-coordinate.
  - Select leftmost point in case of tie.
- The next convex hull vertices p_1 has the least polar angle w.r.t. the positive horizontal ray from p_0.
  - Measure in counterclockwise direction.
  - If tie, choose the farthest such point.
- Vertices $p_2, p_3, \ldots, p_k$ are picked similarly until $y_k = y_{max}$

- p_{i+1} has least polar angle w.r.t. positive ray from p_0.
- If tie, choose the farthest such point.



- The sequence $p_0, p_1, \ldots, p_k$ is right chain of $CH(Q)$
- To choose the left chain of $CH(Q)$ start with p_k.
    - Choose p_{k+1} as the point with least polar angle w.r.t. the negative ray from p_k.
    - Again measure counterclockwise direction.
    - If tie occurs, choose the farthest such point.
    - Continue picking $p_{k+1}, p_{k+2}, \ldots, p_t$ in same fashion until obtain $P_t = p_0$.

Complexity of Jarvis March

For each vertex p belongs to $CH(Q)$, it takes

- $O(1)$ time to compare polar angles.
- $O(n)$ time to find minimum polar angel.
- $O(n)$ total time.

If $CH(Q)$ has h vertices, then running time $O(nh)$. If $h = o(\lg n)$, then this algorithm is asymptotically faster than the Graham's scan. If points in set Q are generated by random generator, then we expect $h = c \lg n$ for $c \approx 1$.

In practice, Jarvis march is normally faster than Graham's scan on most application. Worst case occurs when $O(n)$ points lie on the convex hull *i.e.*, all points lie on the circle.