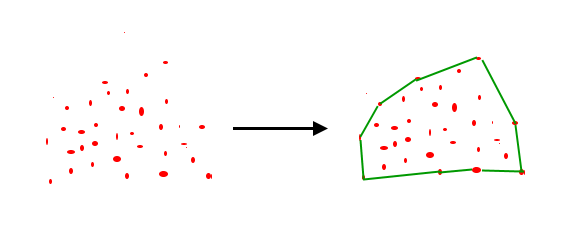
# Convex Hull (Jarvis’s Algorithm or Wrapping)

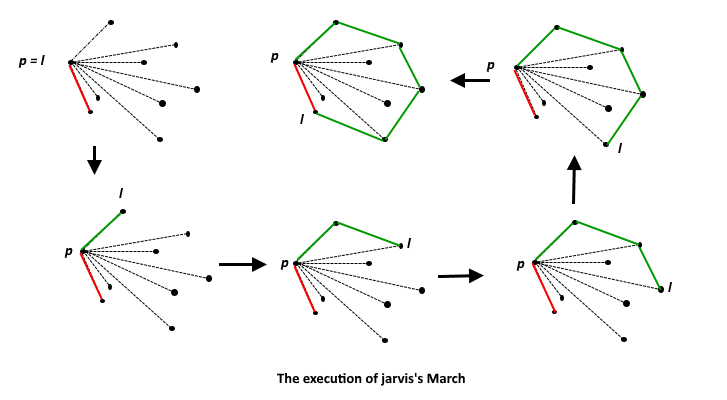
Given a set of points in the plane. the convex hull of the set is the smallest convex polygon that contains all the points of it.



[How to check if two given line segments intersect?](https://www.geeksforgeeks.org/check-if-two-given-line-segments-intersect/)

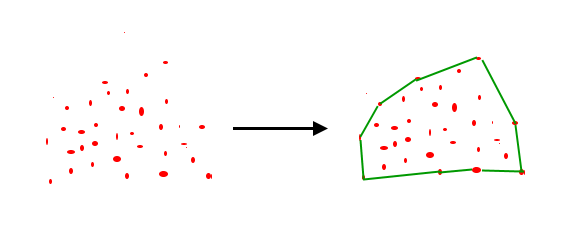
The idea of Jarvis’s Algorithm is simple, we start from the leftmost point (or point with minimum x coordinate value) and we keep wrapping points in counterclockwise direction. The big question is, given a point p as current point, how to find the next point in output? The idea is to use [orientation()](https://www.geeksforgeeks.org/orientation-3-ordered-points/) here. Next point is selected as the point that beats all other points at counterclockwise orientation, i.e., next point is q if for any other point r, we have “orientation(p, q, r) = counterclockwise”. Following is the detailed algorithm.

**1)** Initialize p as leftmost point.  
**2)** Do following while we don’t 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).



# Convex Hull (Graham Scan Algorithm)

Given a set of points in the plane. the convex hull of the set is the smallest convex polygon that contains all the points of it.



[**How to check if two given line segments intersect?**](https://www.geeksforgeeks.org/check-if-two-given-line-segments-intersect/)

We have discussed [Jarvis’s Algorithm](https://www.geeksforgeeks.org/convex-hull-set-1-jarviss-algorithm-or-wrapping/) for Convex Hull. The worst case time complexity of Jarvis’s Algorithm is O(n^2). Using Graham’s scan algorithm, we can find Convex Hull in O(nLogn) time. Following is Graham’s algorithm

Let points[0..n-1] be the input array.

**1)** Find the bottom-most point by comparing y coordinate of all points. If there are two points with the same y value, then the point with smaller x coordinate value is considered. Let the bottom-most point be P0. Put P0 at first position in output hull.

**2)**Consider the remaining n-1 points and sort them by polar angle in counterclockwise order around points[0]. If the polar angle of two points is the same, then put the nearest point first.

**3** After sorting, check if two or more points have the same angle. If two more points have the same angle, then remove all same angle points except the point farthest from P0. Let the size of the new array be m.

**4)** If m is less than 3, return (Convex Hull not possible)

**5)** Create an empty stack ‘S’ and push points[0], points[1] and points[2] to S.

**6)**Process remaining m-3 points one by one. Do following for every point ‘points[i]’

**4.1)**Keep removing points from stack while [orientation](https://www.geeksforgeeks.org/orientation-3-ordered-points/)of following 3 points is not counterclockwise (or they don’t make a left turn).  
            a) Point next to top in stack  
            b) Point at the top of stack  
            c) points[i]  
         **4.2)** Push points[i] to S

**5)** Print contents of S

The above algorithm can be divided into two phases.

**Phase 1 (Sort points):** We first find the bottom-most point. The idea is to pre-process points be sorting them with respect to the bottom-most point. Once the points are sorted, they form a simple closed path (See the following diagram).  
  
What should be the sorting criteria? computation of actual angles would be inefficient since trigonometric functions are not simple to evaluate. The idea is to use the orientation to compare angles without actually computing them (See the compare() function below)

**Phase 2 (Accept or Reject Points):** Once we have the closed path, the next step is to traverse the path and remove concave points on this path. How to decide which point to remove and which to keep? Again, [orientation](https://www.geeksforgeeks.org/orientation-3-ordered-points/)helps here. The first two points in sorted array are always part of Convex Hull. For remaining points, we keep track of recent three points, and find the angle formed by them. Let the three points be prev(p), curr(c) and next(n). If orientation of these points (considering them in same order) is not counterclockwise, we discard c, otherwise we keep it. Following diagram shows step by step process of this phase

**Time Complexity:** Let n be the number of input points. The algorithm takes O(nLogn) time if we use a O(nLogn) sorting algorithm.  
The first step (finding the bottom-most point) takes O(n) time. The second step (sorting points) takes O(nLogn) time. The third step takes O(n) time. In the third step, every element is pushed and popped at most one time. So the sixth step to process points one by one takes O(n) time, assuming that the stack operations take O(1) time. Overall complexity is O(n) + O(nLogn) + O(n) + O(n) which is O(nLogn)

**RELEVANT READING MATERIAL AND REFERENCES:**

**Source Notes:**

1. <https://www.geeksforgeeks.org/convex-hull-set-1-jarviss-algorithm-or-wrapping/>
2. https://www.geeksforgeeks.org/convex-hull-set-2-graham-scan/

**Lecture Video:**

1. <https://youtu.be/ZFxFKABnXN0>
2. https://youtu.be/B2AJoQSZf4M

**Online Notes:**

1. <http://vssut.ac.in/lecture_notes/lecture1428551222.pdf>

**Text Book Reading:**

1. Cormen, Leiserson, Rivest, Stein, “*Introduction to Algorithms*”, Prentice Hall of India, 3rd edition 2012. problem, Graph coloring.

**In addition: PPT can be also be given.**