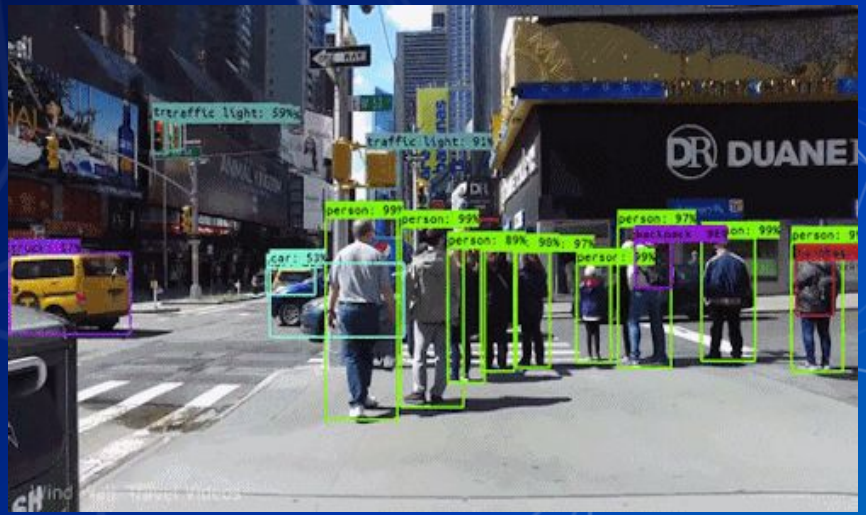
Collision Detection and Avoidance using Convex Hull Algorithms

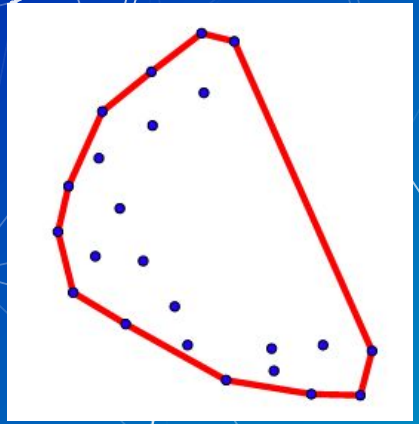
By: Shounak Rangwala, Pratik Mistry, Pranit Ghag, Vikhyat Dhamija

How did we get here?

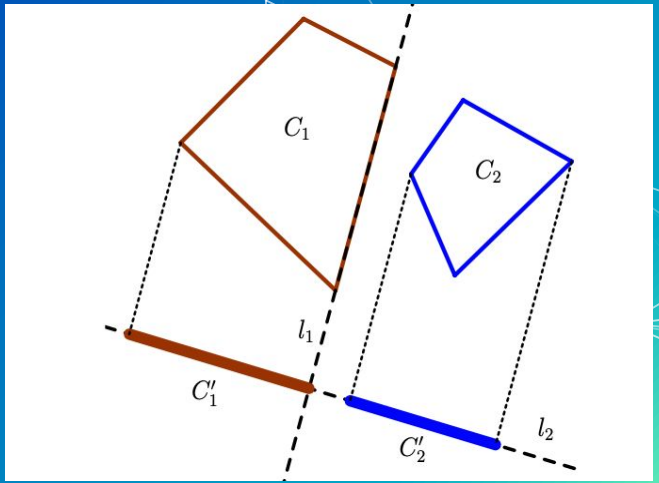
- Collision Detection and Avoidance in Self Driving Cars



How it is done? - Convex Hull



- Convex Hull - It is the smallest convex region which contains all the points from a given set of points
- LiDAR points are used as data for Self Driving Cars
- We find out the approximate outline of the object - cars,humans or other obstacles
- Check for overlap between two different convex hulls - self driving car and obstacle
- If Overlap - Collision Detected!!!

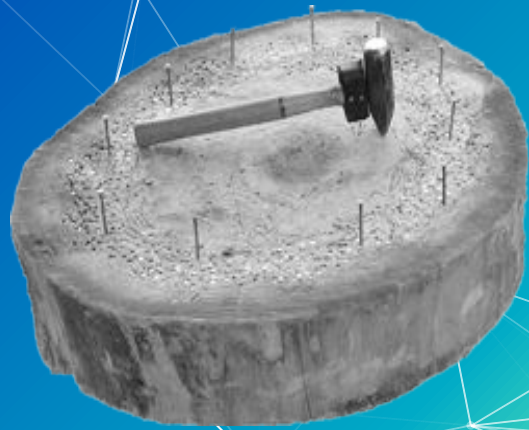


How to find Convex Hull?

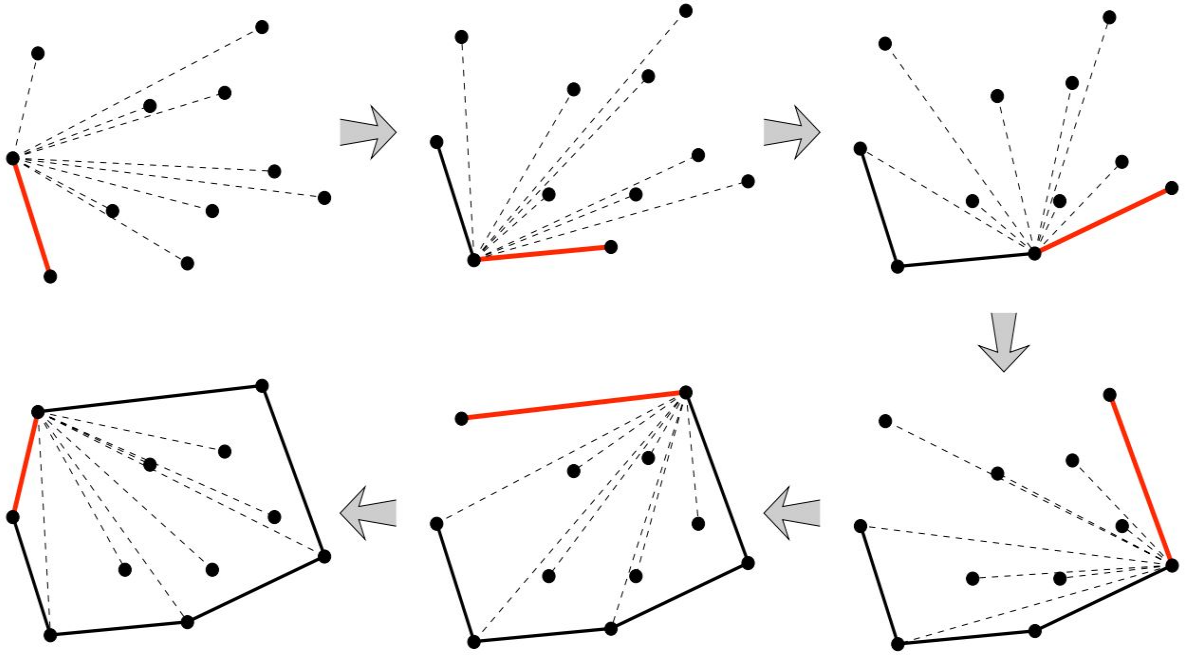
- Computing the convex hull is a problem in computational geometry
- Convex Hull Algorithms:
  - Jarvis March
  - Quickhull
  - Graham Scan
  - Monotone Chain

Jarvis March Algorithm

- Published by R. A. Jarvis [1973]
- Popularly known as Gift Wrapping Algorithm
- Simplest algorithm - simulates the act of wrapping a string around the nails on a piece of wood. (Imagine Hammerschlagen)
- Jarvis March algorithm has the potential to be the cheapest and the most expensive algorithm when seen from a performance point of view
- It is output-dependent (output word used very loosely)



Working procedure of Jarvis March



The execution of Jarvis's March.

Complexity Analysis

- Average case: O(hN)
  - For each added point to the hull, the time complexity is O(N)
  - h is the number of points in the hull

- Worst Case: $O(N^2)$
  - All the N points lie in a circle - makes the entire process happen N times

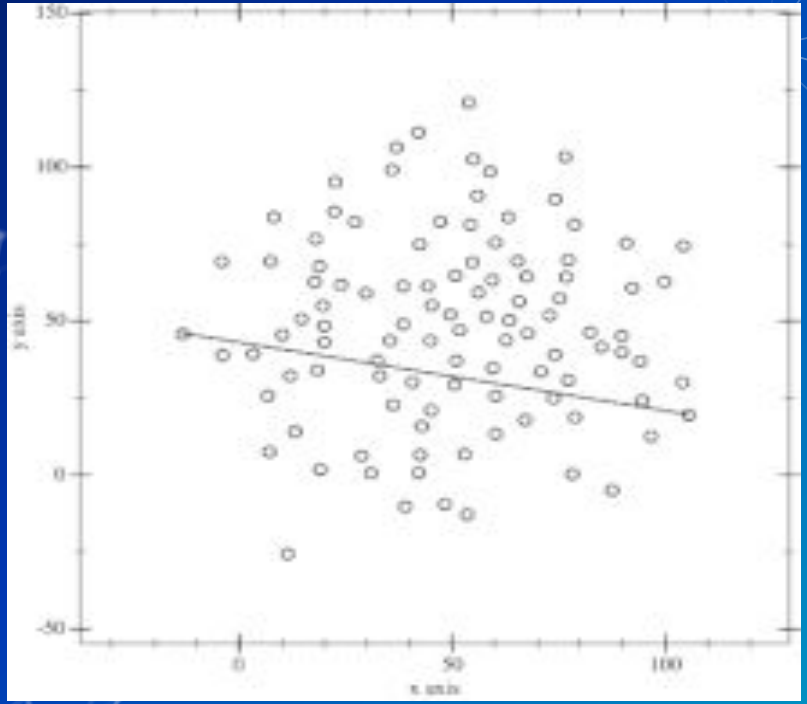
- In practice, h turns out in the order of O(logN) thus resulting complexity is O(NLogN)

The QuickHull Algorithm

- Published by C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpää [1996] - Modelled after Quick Sort Algorithm

- Working Procedure
  - Recursively dividing up the points set into 2 halves divided by the line joining the 2 most extreme points
  - A point from either side is then selected which is at the maximum distance from the dividing ridge and added to the convex hull. A triangle is made using these 3 points. All points inside the triangle are removed from consideration for the convex hull.
  - The above step is recursively carried forth till there exist no points that can be added to the convex hull

Working procedure for Quickhull



Complexity Analysis

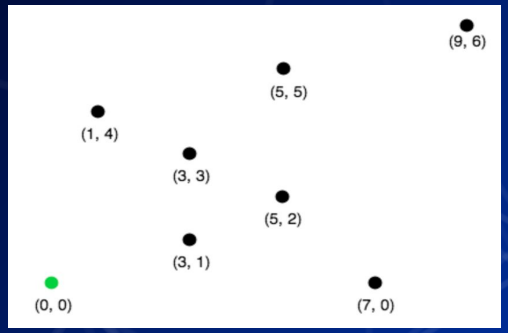
- Average case: O(NlogN)
  - Depends on the starting 2 points of the Quickhull
  - If we have the leftmost and the rightmost points to start with, the recursive division happens like quicksort

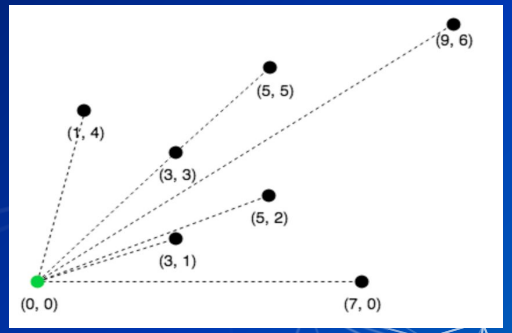
- Worst Case: $O(N^2)$
  - After each division, the number of points reduced are none.i.e the points are arranged in a circle.
  - Run the entire algorithm N times

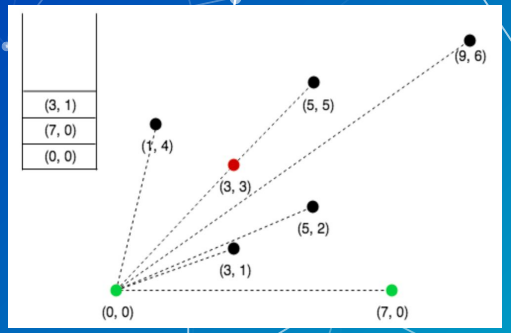
Graham’s Scan

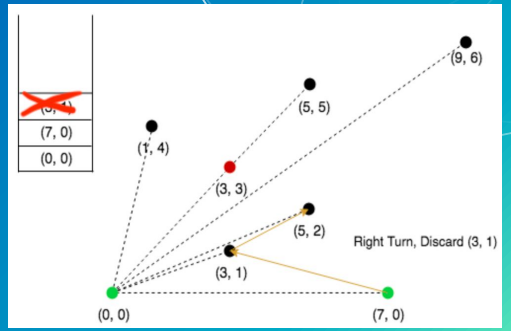
- Published by Ronald Graham - 1972
- One of the first algorithms of computational geometry
- Very popular and widely used algorithm
- Two important working step:
  - Sort the set of points according to their polar angle
  - Scan the points to find the convex hull vertices based on its rotation

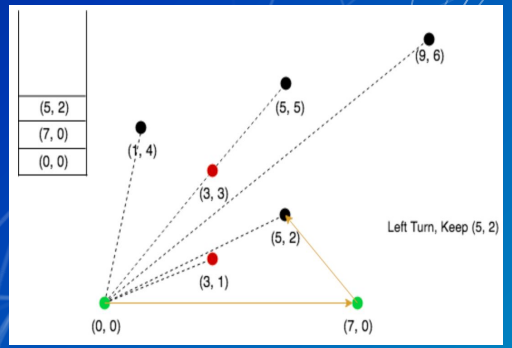
Working Procedure for Graham Scan (1)

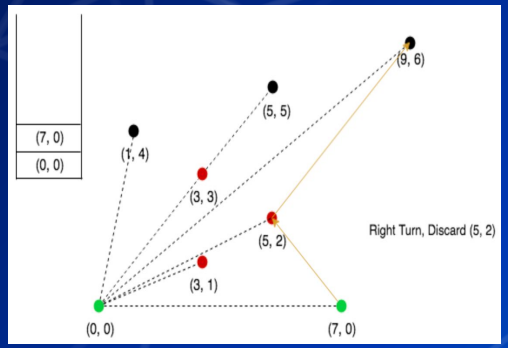




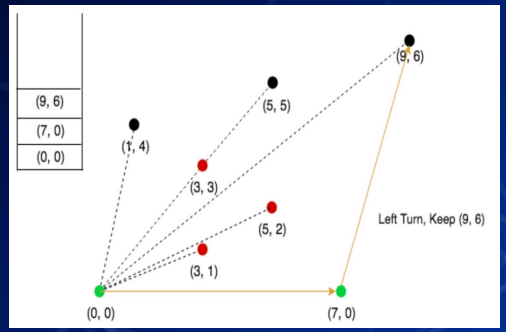


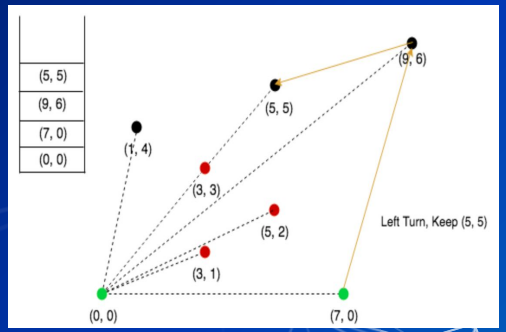


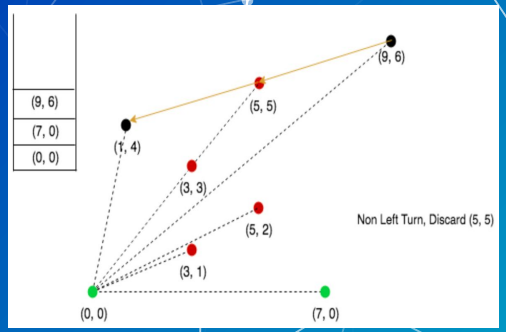


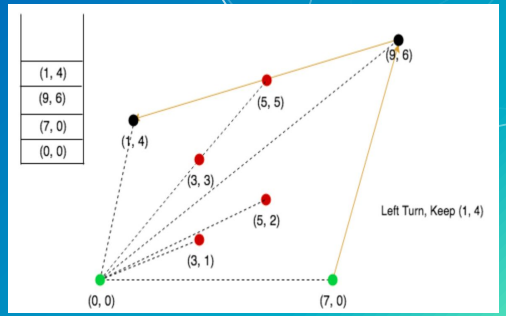


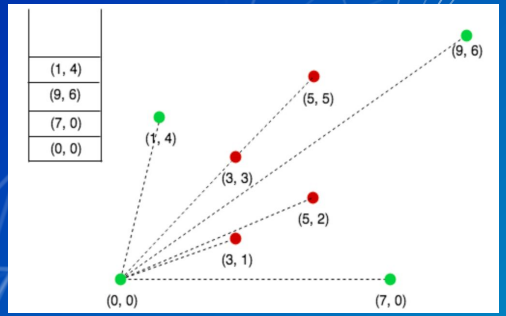
Working Procedure for Graham Scan (2)











Working Procedure for Graham Scan (3)

1. Find the point with lowest y coordinate (Choose point with smaller x coordinate in case of tie) - P0

2. Sort other points by polar angles wrt P0 (Consider nearest distance in case of tie)

3. Create an empty stack S and push points[0] and points[1] to S

4. Process remaining points as:
   a. Keep removing points from stack while orientation of 3 points is not counterclockwise (or they don't make a left turn)
   b. Else Push point to S if orientation is counterclockwise

5. Orientation is determined by comparing slopes of lines *P1P2* and *P1P3*

   (x2-x1)(y3-y1) - (y2-y1)(x3-x1)

   0 = Collinear, -ve = Right turn or clockwise, +ve = Left turn or counterclockwise

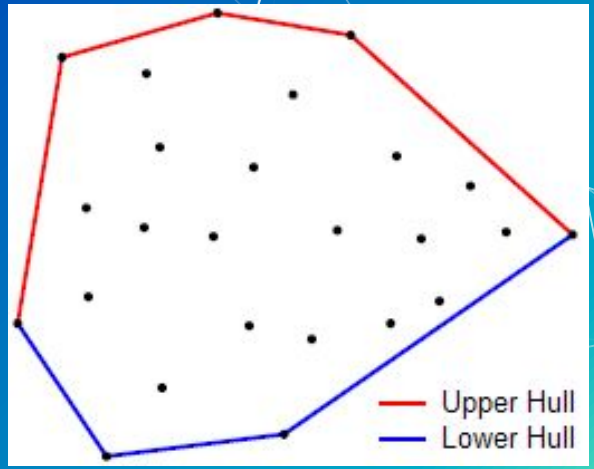
Complexity Analysis

- Total running time: O(NlogN)

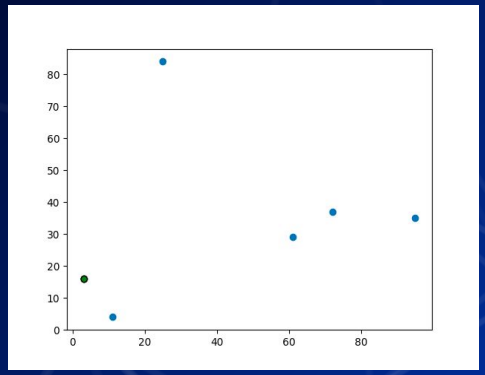
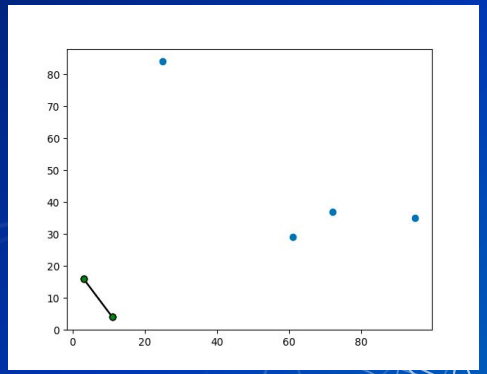
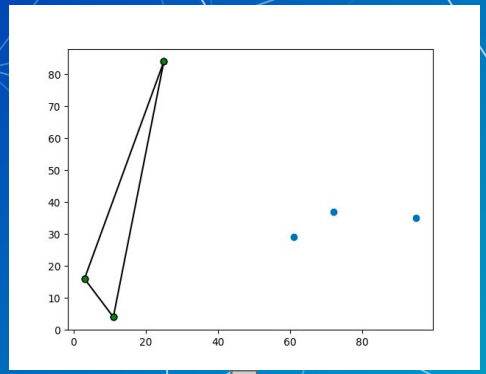
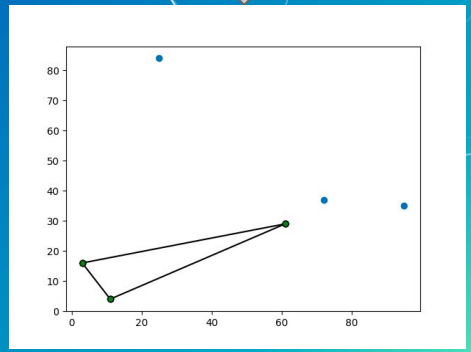
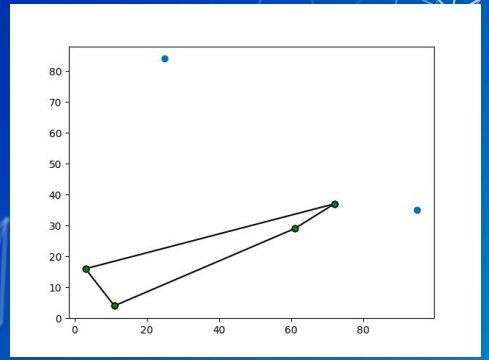
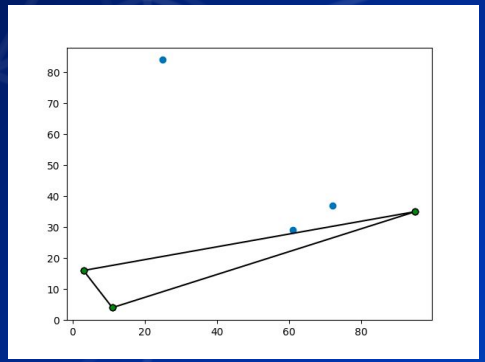
- Runtime of the Graham Scan
  - O(n) for finding anchor point
  - O(nlogn) for sorting the points
  - O(1) constant time for pushing items into the stack
  - O(n) each point gets pushed once within the for loop
  - O(n) for popping within the loop, each point gets popped once at most

Monotone Chain

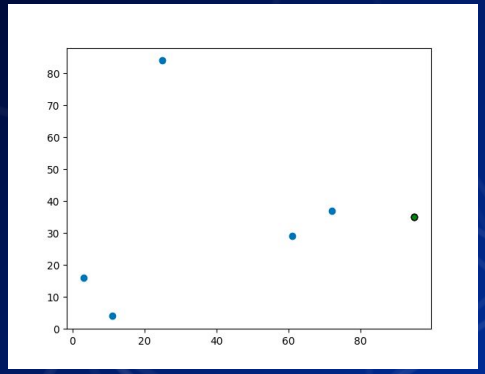
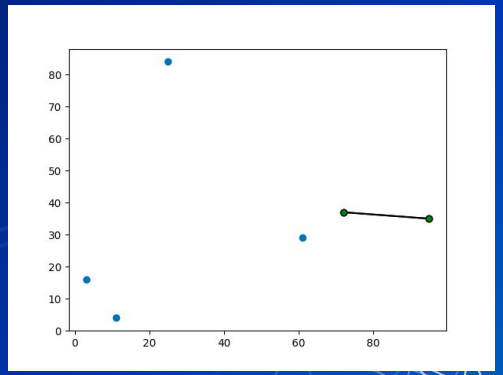
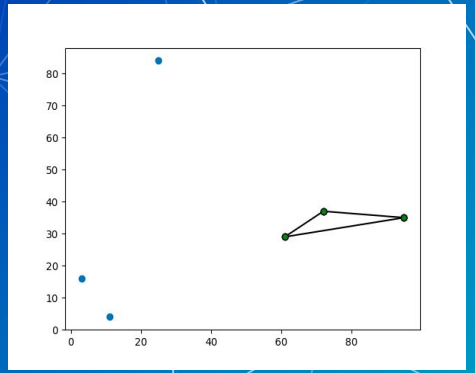
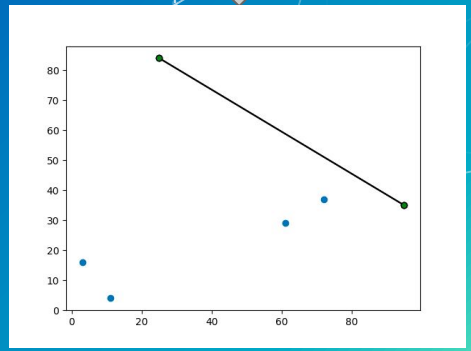
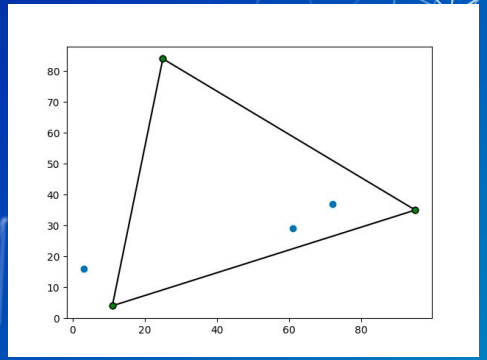
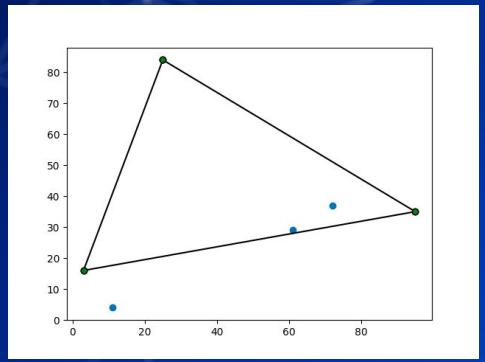
- Published by A. M. Andrew - 1979
- Similar to Graham’s Scan
- Important working steps:
  - Sort the set of points lexicographically
  - Scan points from left to right to find lower hull
  - Scan points from right to left to find upper hull
  - Merge lower and upper hull points



Working Procedure for Monotone Chain (1)

Working Procedure for Monotone Chain (2)

Working Procedure for Monotone Chain (3)

1. Sort other points lexicographically (first by x-coordinate, and in case of a tie, by y-coordinate)

2. Lower Hull: *Procedure for lower hull is same as Graham Scan*

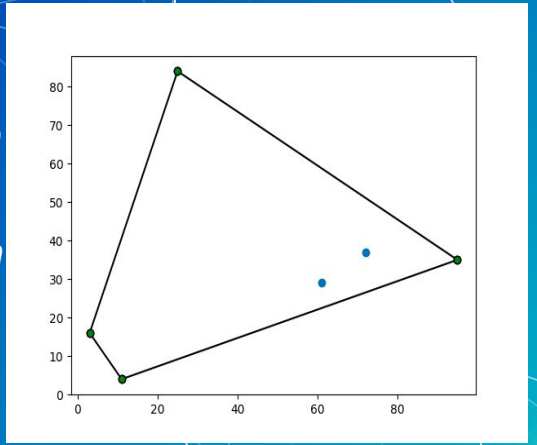
3. Upper Hull: *Procedure for lower hull is same as Graham Scan (but reverse the list of points)*

4. *Merge Lower and Upper Hull points to get Convex Hull*

5. Orientation is determined by comparing slopes of lines *P1P2* and *P1P3*

   (x2-x1)(y3-y1) - (y2-y1)(x3-x1)

   0 = Collinear, -ve = Right turn or clockwise, +ve = Left turn or counterclockwise

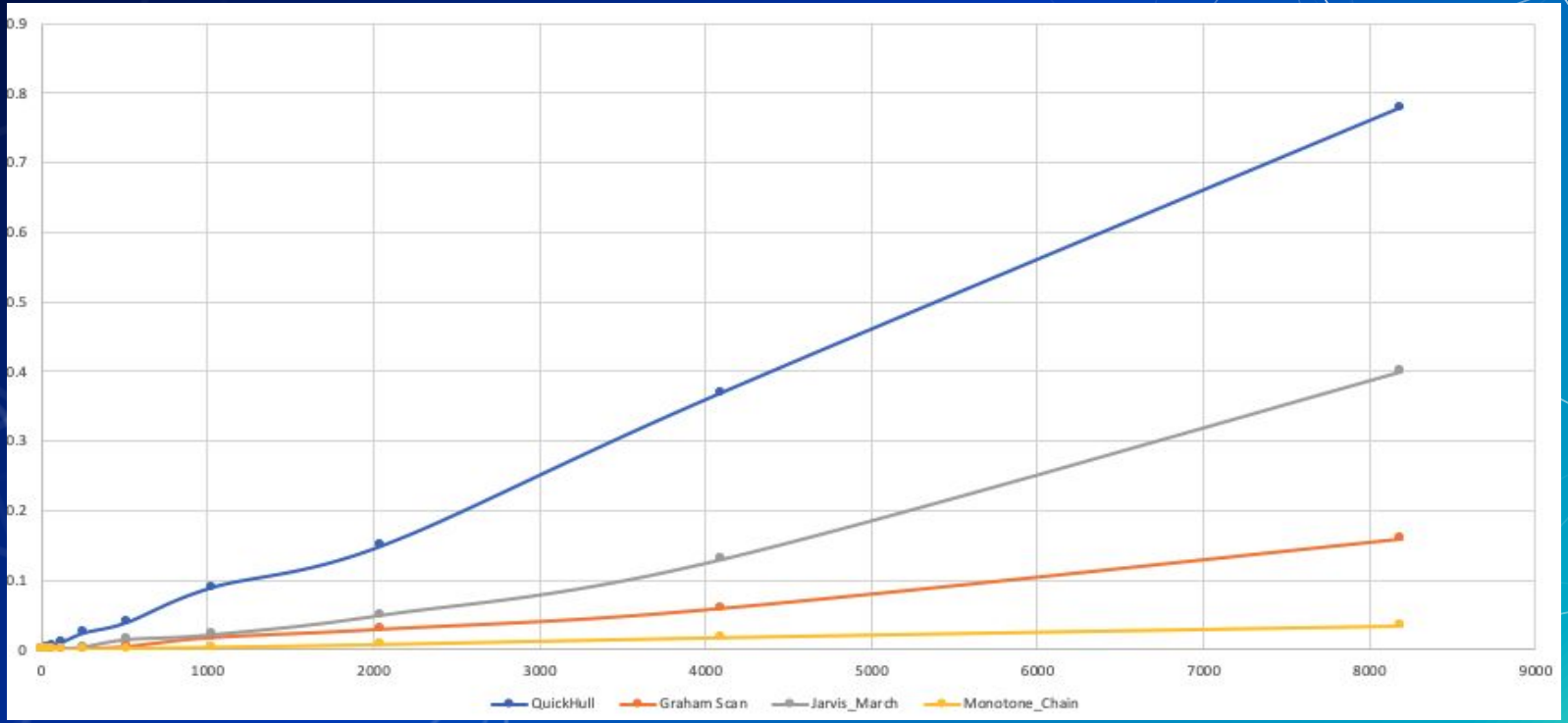


Complexity Analysis

- Total running time: O(NlogN)

- Runtime of the Monotone Chain
  - O(nlogn) for sorting
  - Combined complexity for lower and upper hull:
    - 2*O(1) constant time for pushing items into stack
    - 2*O(n) each point gets pushed once within the for loop
    - 2*O(n) for popping within the loop, each point gets popped once at most
  - O(n) to merge lower and upper convex hull

Complexity Analysis Comparison



Further scope and applications

1. Convex hulls are useful for fingerprint matching in security systems

2. They can be used to identify the deep fake images from real ones

3. Other applications are pattern recognition, image processing, statistics, geographic information system, game theory, construction of phase diagrams, and static code analysis by abstract interpretation

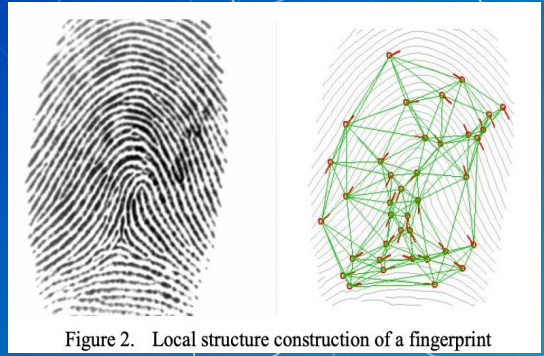
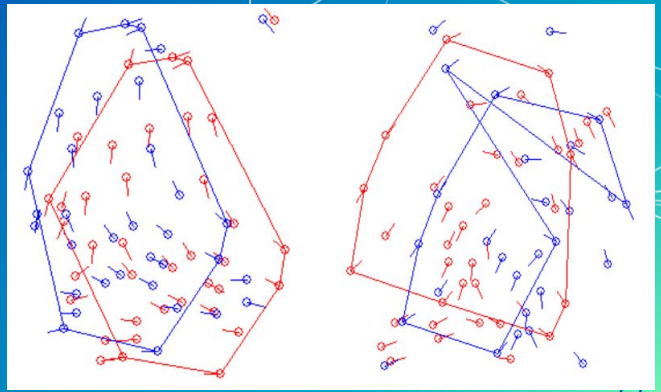


Figure 2. Local structure construction of a fingerprint



Thank You