CONVEX HULLS of point sets → Find

How fast can you construct this?
Test every point
> Test every point : → is it extreme? → # directions?
Test every point: is it in a triangle? is it extreme? # directions? time?
assuming one test: O(1)
> Test every point: \(\rightarrow\) is it extreme? \(\rightarrow\) # directions?

> Test every edge: is it extreme? time?
> Test every point : is it extreme? → # directions?
> Test every edge : is it extreme? $O(n^3)$

> Is it easy to get the C.H. polygon given unordered edges?
> Test every point : is it in a triangle? $O(n^3)$

> Test every edge : is it extreme? $O(n^3)$

> Is it easy to get the C.A. polygon given unordered edges?
  
  > find a point inside & sort

  \[ \text{how?} \]
Find one C.H. vertex

> easy: find $X_{\max}$ (if ties, find $Y_{\max}$ among them)

Find one C.H. edge

> already saw: could test all pairs (but could do $O(n^3)$ time)
Find one C.H. vertex

> easy: find Xmax (if ties, find Ymax among them)  

Find one C.H. edge

> already saw: could test all pairs (but could do $O(n^3)$ time)

> Start w/ one C.H. vertex
Find one C.H. vertex

> easy: find $X_{\text{max}}$ (if ties, find $Y_{\text{max}}$ among them) $O(n)$

Find one C.H. edge

> already saw: could test all pairs (but could do $O(n^3)$ time)
> Start w/ one C.H. vertex
  $\Rightarrow$ Then find 2nd endpoint of shared edge

Rotate line

Find min angle $O(n)$
Find one C.H. vertex

> easy: find $X_{\text{max}}$ (if ties, find $Y_{\text{max}}$ among them) \(O(n)\)

Find one C.H. edge

> already saw: could test all pairs (but could do \(O(n^3)\) time)
> Start w/ one C.H. vertex

\(\Rightarrow\) Then find 2nd endpoint of shared edge

\[ \text{Rotate line} \]

\[ \text{Find min angle } O(n) \]

\[ \text{AND REPEAT } \ldots \text{ } O(n^3) \text{ for C.H.} \]
GIFT WRAPPING ALGORITHM

\[ \ni \]

JARVIS MARCH

\[ O(n^2) \quad \ldots \text{actually} \quad O(n \cdot h) \]
Graham Scan

- Start w/ an extreme point
Graham Scan

- Start w/ an extreme point
- Sort all others radially
Graham Scan

- Start w/ an extreme point
- Sort all others radially
- Identify extreme in sorting
- Go through sorted list
  - if 5 advance
  - if 6 PoP & backtrack
- go through sorted list
  - if 5 advance
  - if < pop & backtrack

012 5
- go through sorted list
  - if 5 advance
  - if 0 pop & backtrack

012 5
123 0 pop 2 ... gone forever
- go through sorted list
- if \( S \):
  - advance
- if \( C \):
  - pop & backtrack

\[
0, 1, 2, 5, 3, 9, 1, 2, 3, 0, 1, 3
\]

...gone forever
- go through sorted list
  - if 5 advance
  - if C: Pop & backtrack

012
123: Pop 2 ...gone forever
013
134
go through sorted list

if 5 advance

if < Pop & backtrack

012
123
013
134
345

Pop 2

...gone forever

Pop 4
- go through sorted list

- if 5 advance
- if C Pop & backtrack

012 (G) Pop 2 ...gone forever

123 (G)

013 (G)

134 (G) Pop 4

345 (G)

135 (G)
- go through sorted list
  \[ \rightarrow \text{if } \mathbf{5} \text{ advance} \]
  \[ \rightarrow \text{if } \mathbf{c} \text{ Pop & backtrack} \]

012 \[ \circ \]

123 \[ \circ \] \text{ Pop 2 ...gone forever}

013

134 \[ \circ \] \text{ Pop 4}

345 \[ \circ \]

135 \[ \circ \] \text{ Pop 5}

356 \[ \circ \]
- go through sorted list
  - if 5 advance
  - if C POP & backtrack

012
123
013
134
345
135
356
146

...gone forever

Pop 2
Pop 4
Pop 5
- go through sorted list
  \[ \rightarrow \text{if } 5\ \text{advance} \]
  \[ \rightarrow \text{if } C\ \text{Pop & backtrack} \]

012 \[ \rightarrow \]
123 \[ \rightarrow \text{Pop 2} \]
013 \[ \rightarrow \]
134 \[ \rightarrow \]
345 \[ \rightarrow \text{Pop 4} \]
135 \[ \rightarrow \]
356 \[ \rightarrow \text{Pop 5} \]
146 \[ \rightarrow \]
467 \[ \rightarrow \text{Pop 6} \]

... gone forever
- go through sorted list
  \[ \text{if } \sigma \text{ advance} \]
  \[ \text{if } \sigma \text{ Pop & backtrack} \]

012 0  
123 0  
134 0  
345 1  
135 1  
356 1  
146 1  
467 1  
147 1  
...gone forever
- go through sorted list
- if \( 5 \) advance
- if \( C \) pop & backtrack

012
123
013
134
345
135
356
146
467
147
478
- go through sorted list
  - if 5 advance
  - if C Pop & backtrack

012 [go] 5
123 [go] 2

013 [go]
134 [go]

345 [go]
135 [go]

135 [go]
356 [go]
146 [go]

467 [go]
147 [go]

147 6 7 8 [go]
789 [go]

...gone forever

Pop 2
Pop 4
Pop 5
Pop 6
Pop 8
- go through sorted list
  \[ \text{if } 5 \text{ advance} \]
  \[ \text{if } C \text{ Pop & backtrack} \]

[Diagram with nodes and edges labeled with numbers 0 to 9, arrows indicating connections and actions.]
> all points checked
> can a C.H. vertex get popped?
> can a non-C.H. vertex survive?
> all points checked
> can a C.H. vertex get popped?
> can a non-C.H. vertex survive?

Time complexity?
- Could backtrack on positions between advancing
- Obviously there could be \( \mathcal{O} \) advances.
> all points checked
> can a C.H. vertex get popped?
> can a non-C.H. vertex survive?

Time complexity?
- Could backtrack on positions between advancing
- Obviously there could be \( nn \) advances.
- But you can only backtrack once on each point (app)

\[ O(n) \] after sorting : \( O(n \log n) \) total
Can we do better?
Can we do better?

Say you have $n$ numbers to sort: $x_1, x_2, x_3, x_4, \ldots, x_n$

*Augment* them to 2D data: $(x_1, x_1^2), (x_2, x_2^2), (x_3, x_3^2), \ldots, (x_n, x_n^2)$
Can we do better?

Say you have n numbers to sort: \( x_1, x_2, x_3, \ldots, x_n \)

Augment them to 2D data: \((x_1, x_1^2), (x_2, x_2^2), (x_3, x_3^2), \ldots, (x_n, x_n^2)\)

The new points are, by definition, a convex set.

Construct C.H. and then read in order

\[ \text{sort: } \mathcal{O}(n) + \text{C.H.} = \Omega(n \log n) \]

\[ \text{C.H.}(n) = \Omega(n \log n) \]
So if C.H. is $\Omega(n \log n)$

WHY is Jarvis March $O(nh)$?  \( \text{sometimes } n.h < n \log n \)

$O(nh)$ is just a more sensitive bound. & worst-case is $O(n^2)$

The knowledge of $h$ is important.

Clearly if I told you $h=3$ you wouldn’t sort anything. ... $O(n)$

We could try to get a lower bound in terms of $n$ and $h$

$\Rightarrow$ and also a better upper bound
Quick Hull

- Maintain a convex polygon, for which:
  - you know every vertex is on C.H.
  - all points inside are discarded forever

1) Find $X_{min}$ $X_{max}$ $Y_{min}$ $Y_{max}$
   Discard all points inside triangle/quad.
Quick Hull

- Maintain a convex polygon, for which
  - you know every vertex is on C.H.
  - all points inside are discarded forever

1) Find $X_{min}, X_{max}, Y_{min}, Y_{max}$
   Discard all points inside triangle/quad.

2) Find extreme pts orthogonal to current polygon edges
Quick Hull

- Maintain a convex polygon, for which
  - you know every vertex is on C.H.
  - all points inside are discarded forever

1) Find $X_{\min}$, $X_{\max}$, $Y_{\min}$, $Y_{\max}$
   Discard all points inside triangle/quad.

2) Find extreme pts orthogonal to current polygon edges

3) Form new triangles, discard.

repeat
1) Find $X_{min}$, $X_{max}$, $Y_{min}$, $Y_{max}$.
   Discard all points inside triangle/quad.

2) Find extreme pts orthogonal to current polygon edges.

3) Discard, repeat.

- Every discard involves a quad, so $O(n)$.
- Number of regions to check doubles each time ... but ends up on $n$, so total = $O(n)$.
- Worst case: you never discard .... $O(n^2)$ total.

$O(h \log n)$ expected.
QuickHull in a little more detail

- What is the worst case?

Form zero deletions
QuickHull in a little more detail

- What is the worst case?

Search extremes
Find only one
Zero deletions
QuickHull in a little more detail

- What is the worst case?

  - Search extremes
  - Find only one
  - Zero deletions
QuickHull in a little more detail

- What is the worst case?

Search extremes
Find only one
Zero deletions
Etc
... $O(n^2)$
QuickHull in a little more detail

Also output-sensitive? \( f(n,h) \)

Notice that every new search happens when we find a convex hull vertex \( O(n \cdot h) \) (iff)