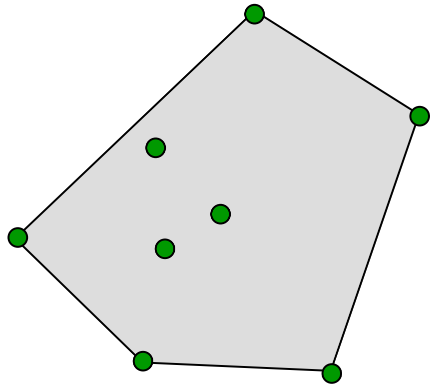
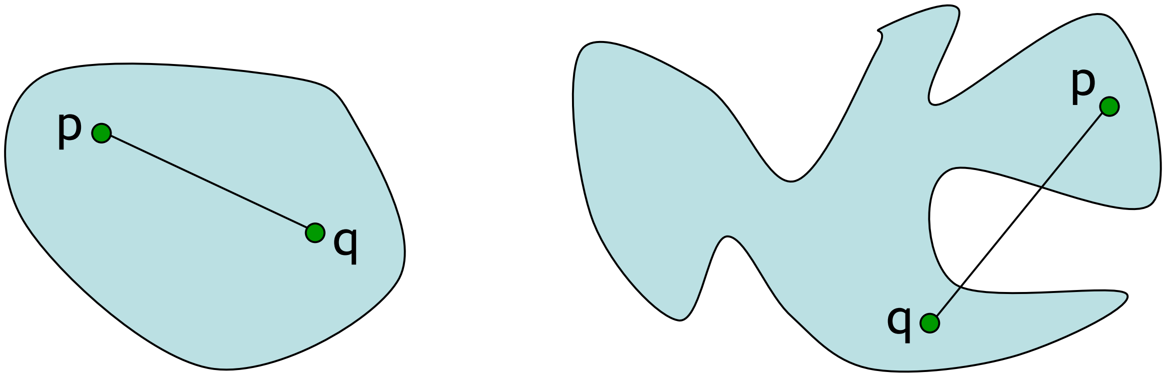
Convex hulls and the sweep line technique



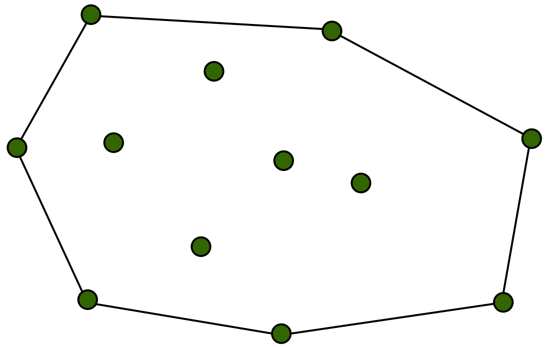
Convex hull

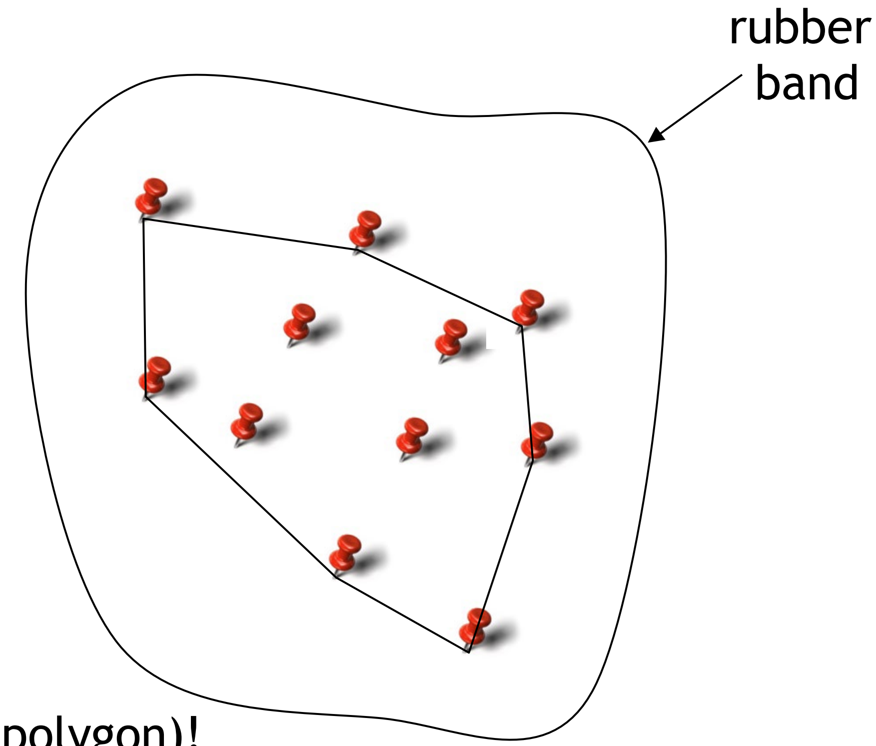
A subset S of the plane is convex if for every pair of points p,q in S the straight line segment pq is completely contained in S.



Convex hull

The convex hull of a point set S is the smallest convex set containing S.





We only want to find the hull (simple polygon)!

CH algorithm 1

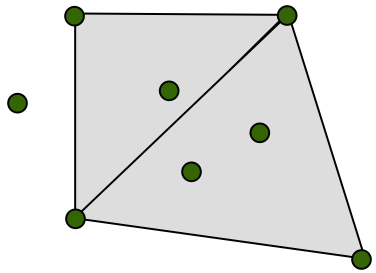
Definition:

The convex set of a set of point S in d dimensions is the union of all convex combinations of (d+1) points of S.

d=2: Convex combination of 3 points $\Rightarrow$ a triangle!

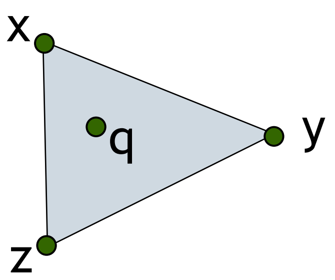
Definition implies an algorithm:

A point that does not lie in the interior of any triangle of S is a CH vertex. Why?



THE UNIVERSITY OF SYDNEY

CH algorithm 1



Algorithm CH1(S)

1. for every possible triple of points x,y,z in S do
2.   for every point q in S do
3.   if q lies within the triangle (x,y,z) then
4.   discard q from S
5. Build CH by sorting points radially around a point on the CH

Time complexity?

Step 1 is performed $O(n^3)$ times
Step 2 is performed n times/iteration
Steps 3 & 4 cost O(1)/iteration
Step 5 costs O(n log n)

Total time: $O(n^4)$

CH algorithm 1: running time

Assumption: 10^9 instructions per second

Input size: 1 million points = 10^6 points

$\Rightarrow$ running time ~ $n^4/10^9 = 10^{15}$ seconds ~ 32 million years

CH in 1 second: 180 points

CH algorithm 2

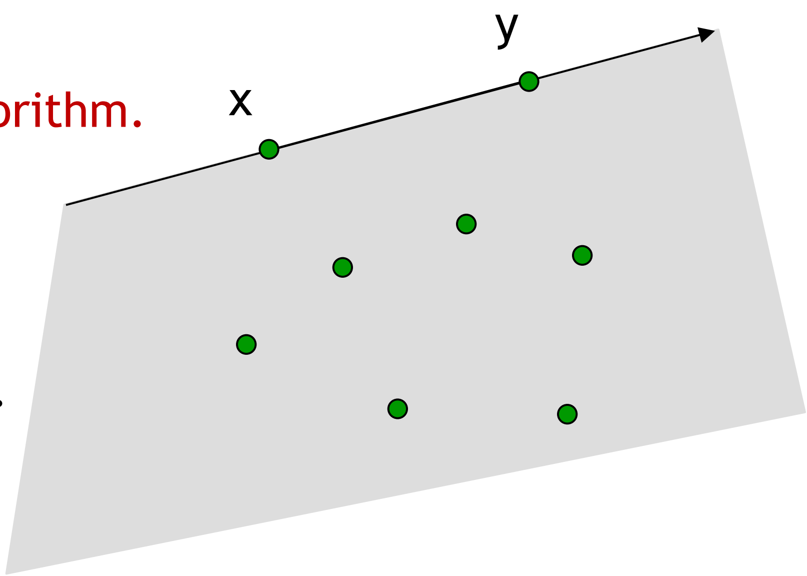
Definition:
The CH of S is the intersection of all halfspaces that contain S.

Why is the intersection of two convex sets a convex set? Union?

This definition implies a second algorithm.

Consider an edge xy of CH(S).

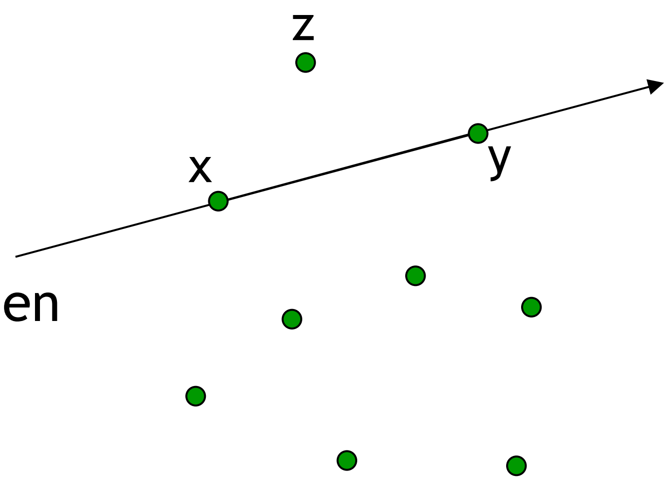
All points of S must lie to the right of the directed line through x and y.



CH algorithm 2

Algorithm CH2(S)

1. for every ordered pair x,y in S do
2. valid $\leftarrow$ true
3. for every point z in S-{x,y} do
4. if z lies to the left of xy then
5. valid $\leftarrow$ false
6. if valid then
7. add xy to CH
8. Sort the edges in CH



Time complexity?

Steps 1-2, 6-7 : $O(n^2)$ times
Steps 3-5 : (n-2) times/iteration
Step 8 : O(n log n)

Total time: $O(n^3)$

CH algorithm 2: running time

Assumption: 10^9 instructions per second

Input size: 1 million points = 10^6 points

$\Rightarrow$ running time ~ $n^3/10^9 = 10^9$ seconds ~ 32 years

CH in 1 second: 1000 points

THE UNIVERSITY OF SYDNEY

Check left turn a primitive?

How can we check if a point r lies to the left of a line pq?

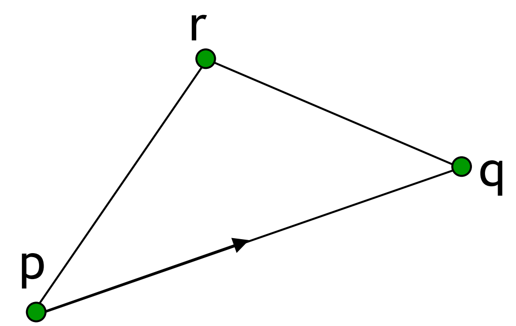
$\Rightarrow$ Triangle $\Delta(p,q,r)$ is oriented counter-clockwise.

$p=(p_x,p_y)$, $q=(q_x,q_y)$ and $r=(r_x,r_y)$

$$CCW(p,q,r) = \begin{vmatrix} p_x & q_x & r_x \\ p_y & q_y & r_y \\ 1 & 1 & 1 \end{vmatrix}$$

$$= (q_x - p_x)(r_y - p_y) - (r_x - p_x)(q_y - p_y)$$

[2 multiplications,
5 subtractions]

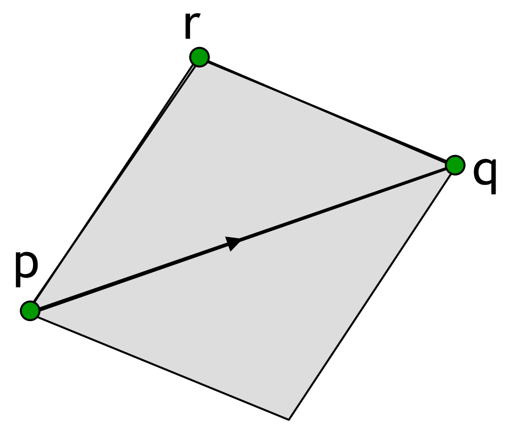


$\Delta(p,q,r)$ is oriented counter-clockwise iff $CCW(p,q,r) > 0$.

Check left turn a primitive?

What is CCW(p,q,r)?

|CCW(p,q,r)| =



(*)

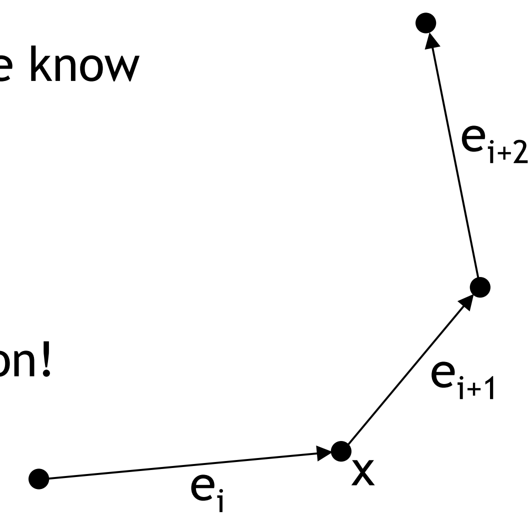
If pqr is a left turn then CCW(p,q,r) > 0

If pqr is a right turn then CCW(p,q,r) < 0

(*) For proof see
https://people.richland.edu/james/lecture/m116/matrices/area.html

CH algorithm 3 (Gift Wrapping)

Can we compute the CH faster? Is there anything we know about the CH that we haven’t used?

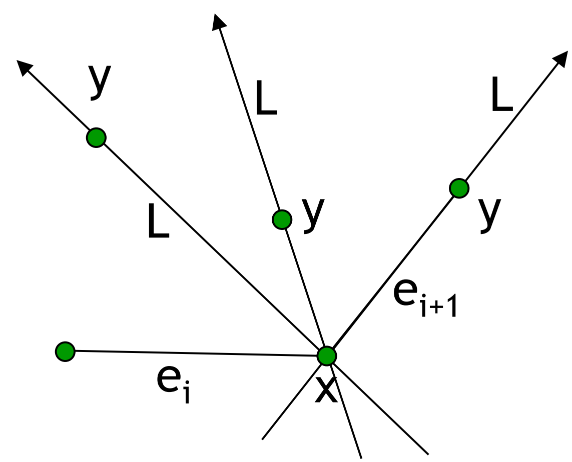


The edges in the CH are linked into a convex polygon!

If we found an edge on the CH with endpoint at x then the next edge must start at x.

Idea:
Draw a line L through x and a point y. Are there any points to the right of L?
If not (x,y) is an edge of CH.

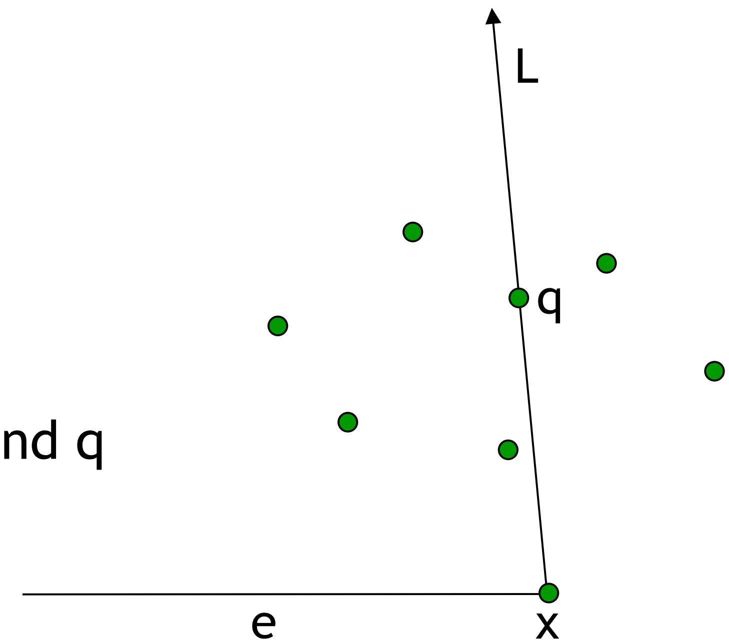
Start point?



CH algorithm 3

Algorithm CH3(S)

```
find lowest point p in S
e ← ( (-∞,p_y),p)
x ← p
repeat
    valid ← true
    for every point q in S-{x} do
        L ← directed line through x and q
        for every point r in S-{x,q} do
            if r to the right of L then
                valid ← false
    if valid then
        add xq to CH
        x ← q
until x == p
```



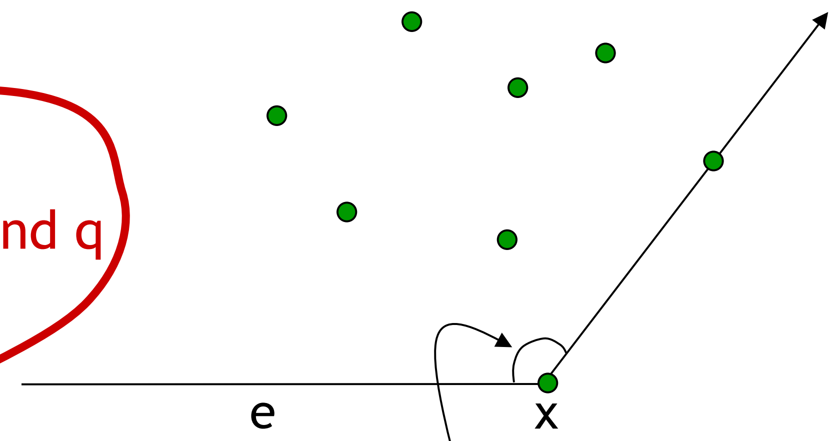
Time complexity: $O(n^3)$

CH algorithm 3

Algorithm CH3(S)

```
find lowest point p in S
e ← ( (-∞,p_y),p)
x ← p
repeat
    valid ← true
    for every point q in S-{x} do
        L ← directed line through x and q
        for every point r in S-{x,q} do
            if r to the right of L then
                valid ← false
    if valid then
        add xq to CH
        x ← q
until x == p
```

Can this be done faster?

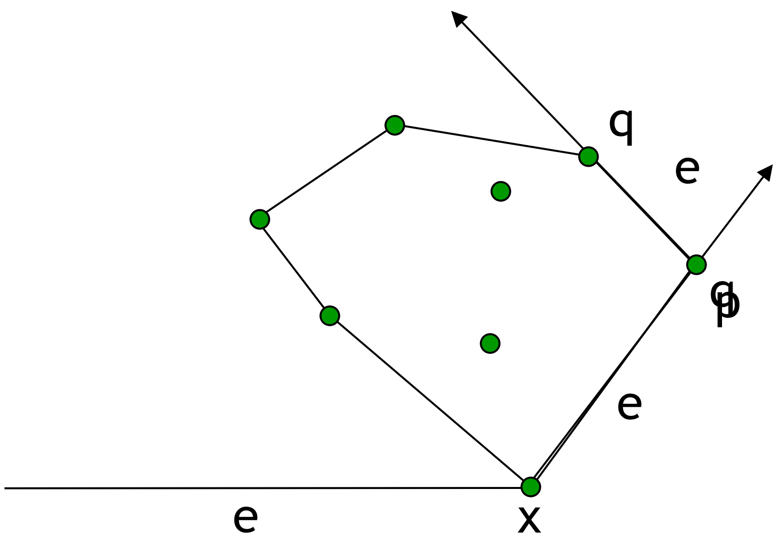


Looking for the point with largest angle!

CH algorithm 4

Algorithm CH4(S)

1. find lowest point p in S
2. x $\leftarrow$ $(-\infty, p_y)$
3. e $\leftarrow$ (x,p)
4. repeat
5.   maxAngle $\leftarrow$ 0
6.   for every point q in S do
7.   if $\angle$(e,(p,q)) > maxAngle then
8.   nextPoint $\leftarrow$ q
9.   maxAngle $\leftarrow$ $\angle$(e,(p,q))
10.   e $\leftarrow$ (p,nextPoint)
11.   p $\leftarrow$ q
12. until x == p

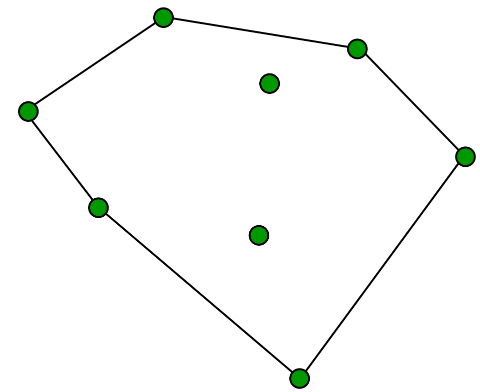


Time complexity: $O(n^2)$
What if the number of points on the CH is small?

CH algorithm 4

Algorithm CH4(S)

```
find lowest point p in S
x ← (-∞,p_y)
e ← (x,p)
repeat
    maxAngle ← 0
    for every point q in S do
        if ∠(e,(p,q)) > maxAngle then
            nextPoint ← q
            maxAngle ← ∠(e,(p,q))
    e ← (p,nextPoint)
    p ← q
until x == p
```



Time complexity: $O(n^2)$
What if the number of points on the CH is small? $O(hn)$

CH algorithm 4: running time

Assumption: 10^9 instructions per second

Input size: 1 million points = 10^6 points

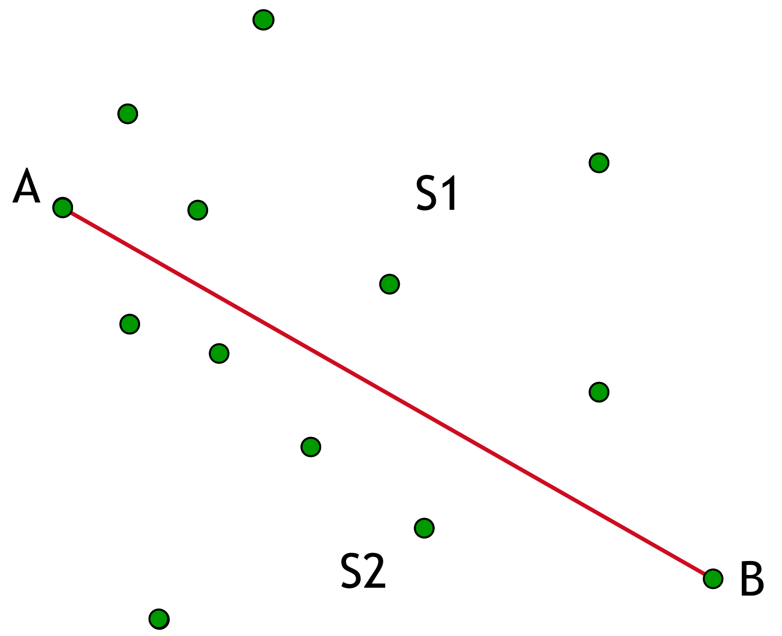
$\Rightarrow$ running time ~ $n^2/10^9 = 10^3$ seconds ~ 17 minutes

CH in 1 second: 30,000 points

Divide-and-Conquer approach

QuickHull(S)

1. A = leftmost point of S
2. B = rightmost point of S
3. S1 = {points in S above AB}
4. S2 = {points in S below AB}
5. FindHull(S1,A,B)
6. FindHull(S2,B,A)

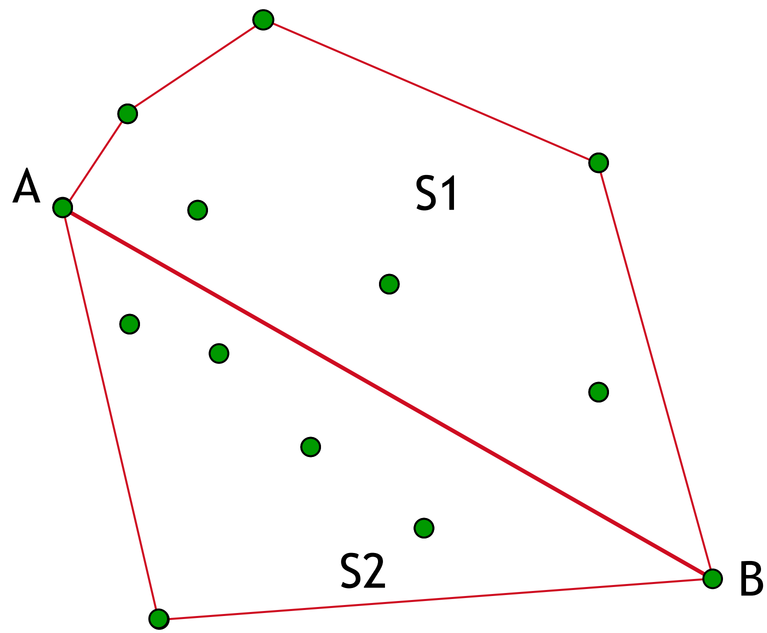


CH algorithm 5 - QuickHull

Divide-and-Conquer approach

QuickHull(S)

1. A = leftmost point of S
2. B = rightmost point of S
3. S1 = {points in S above AB}
4. S2 = {points in S below AB}
5. FindHull(S1,A,B)
6. FindHull(S2,B,A)

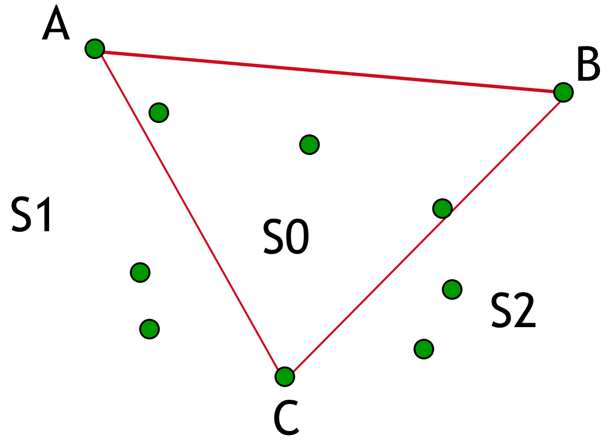


CH algorithm 5 - QuickHull

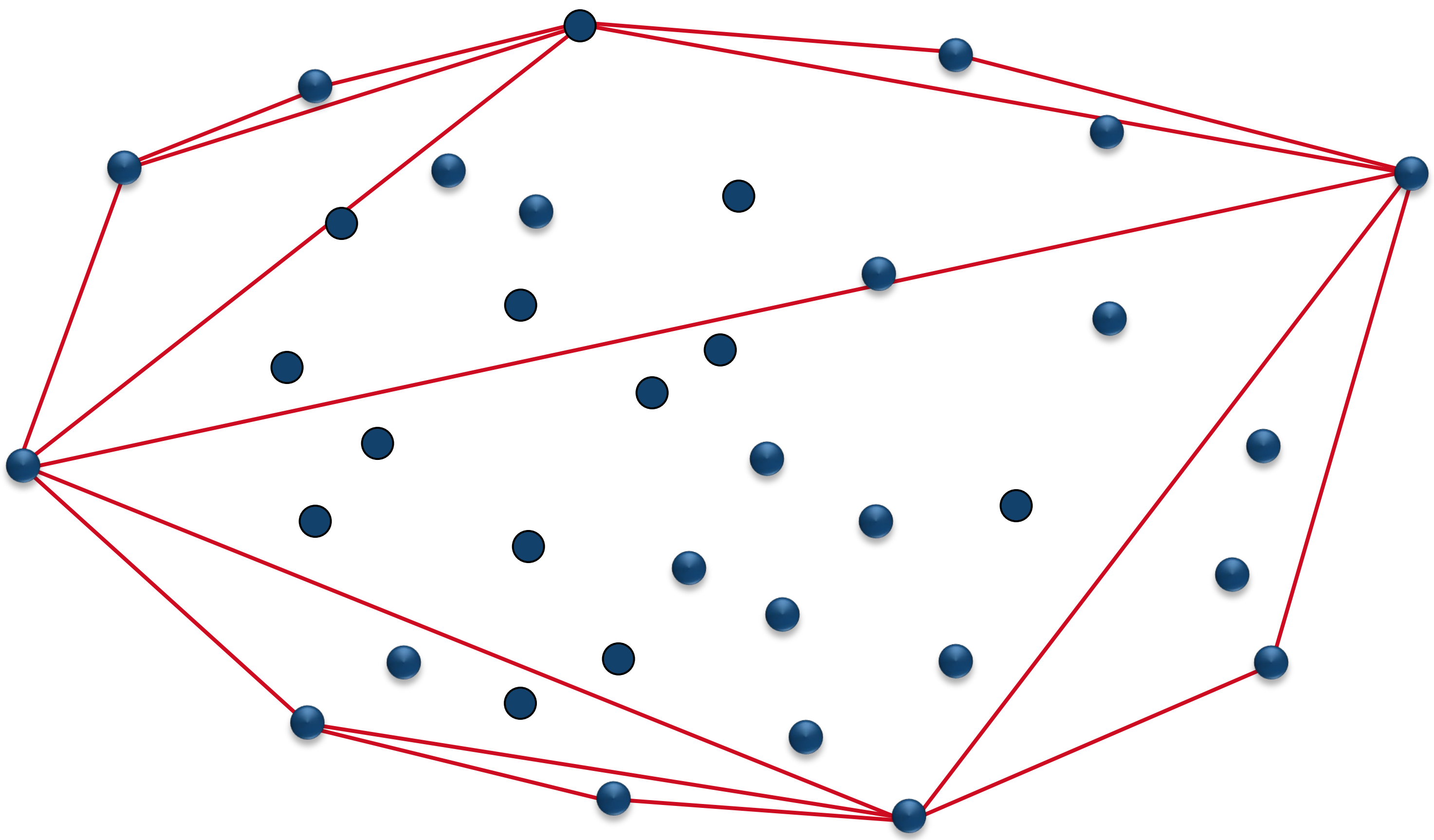
FindHull(S, A, B)

If S not empty then

1. Find farthest point C in S from AB
2. Add C to convex hull between A and B
3. S0={points inside ABC}
4. S1={points to the right of AC}
5. S2={points to the right of CB}
6. FindHull(S1, A, C)
7. FindHull(S2, C, B)



CH algorithm 5 - QuickHull



CH algorithm 5 - QuickHull

QuickHull

- Compute A and B — O(n) time
- FindHull(S_1,A,B) — T(|S1|) time
- FindHull(S_2,B,A) — T(|S2|) time

Worst Case:

$$
\begin{aligned}
T(n) &= T(n-2) + O(n) \\
&= T(n-3) + O(n) + O(n) \\
&= \ldots = O(n^2)
\end{aligned}
$$

CH algorithm 5 - QuickHull

QuickHull

- Compute A and B — O(n) time
- FindHull(S_1,A,B) — T(|S1|) time
- FindHull(S_2,B,A) — T(|S2|) time

Worst Case:

$$T(n) = T(n-2) + O(n)$$
$$= T(n-3) + O(n) + O(n)$$
$$= \ldots = O(n^2)$$

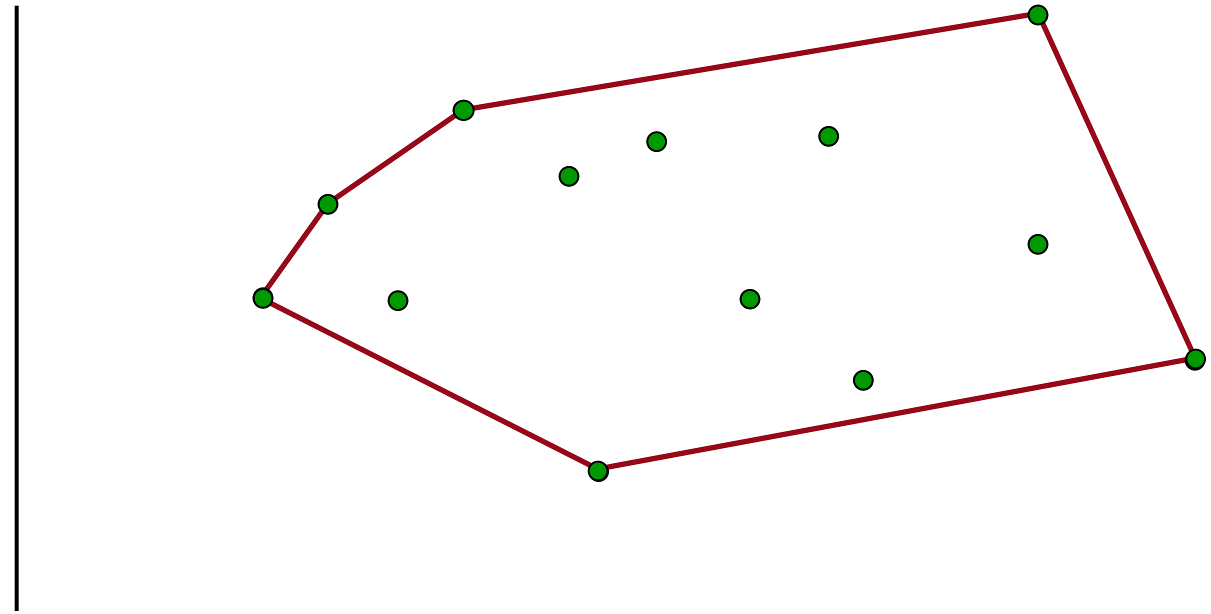
What if points are "nicely" distributed?

$$T(n) = O(T(n/2)) + O(T(n/2)) + O(n)$$
$$= O(n \log n) \qquad \text{Why?}$$

CH algorithm 6 – sweep line approach

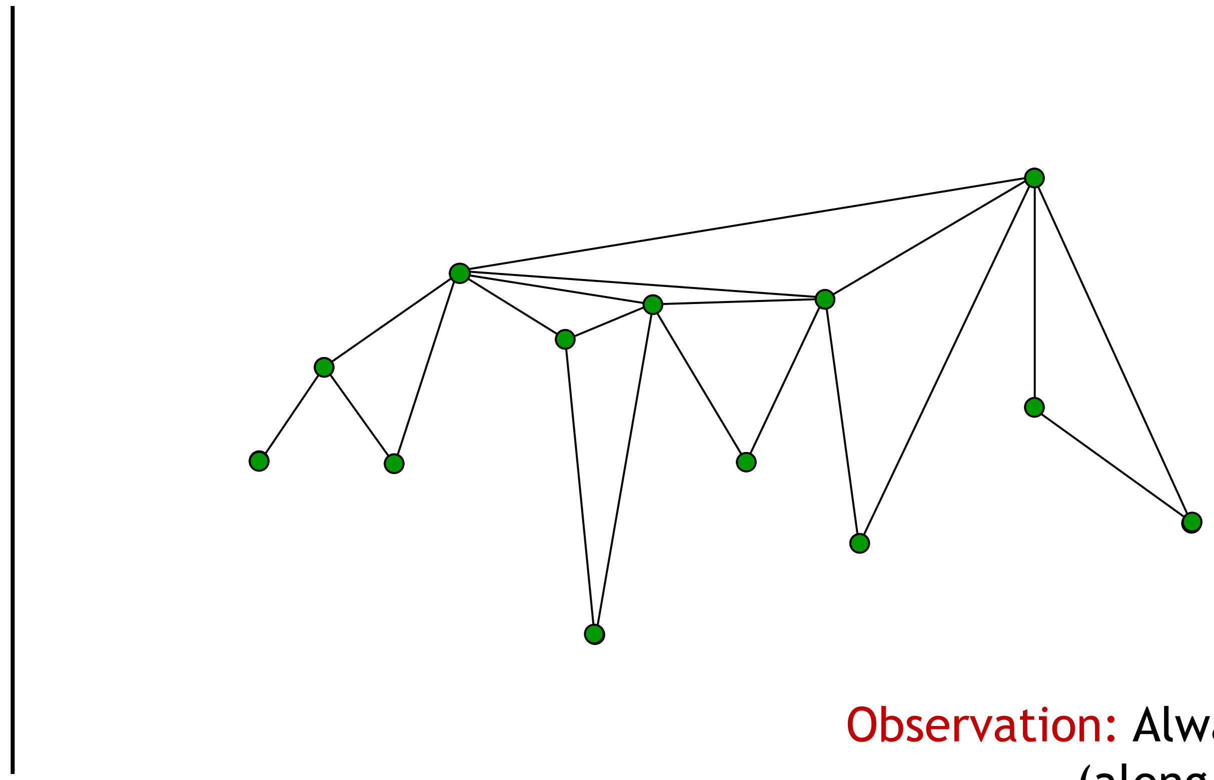
Idea: Maintain hull while adding the points one by one, from left to right ⇔ sweep the point from left to right

Build the upper and lower part of the hull separately, and then merge them at the end.



CH algorithm 6 – sweep line approach

Idea: Maintain hull while adding the points one by one, from left to right ⇔ sweep the point from left to right



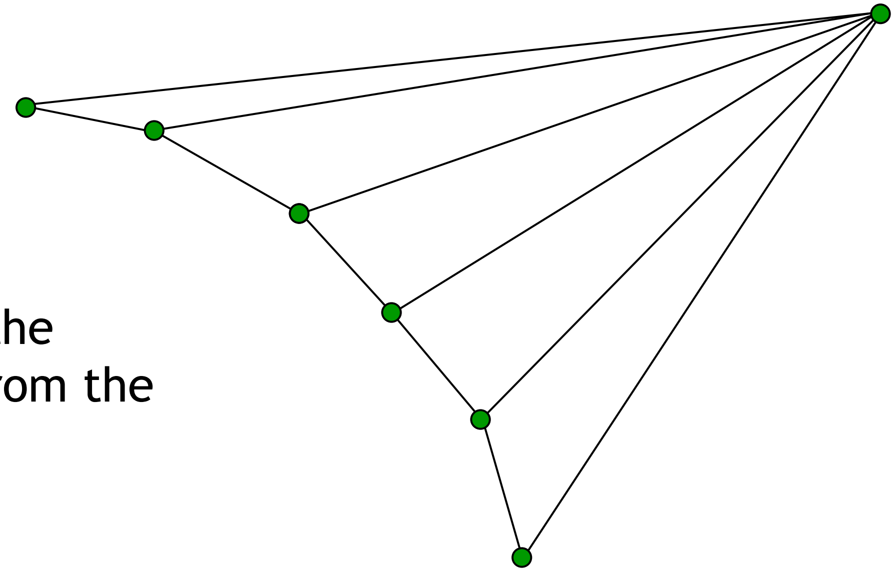
Observation: Always right-turns!
(along the upper hull)

CH algorithm 6 - sweep line approach

Running time?

O(n) per insertion $\Rightarrow$ O(n^2) in total

Can it be that bad?



A point is only added to the hull once and removed from the hull once!

CH algorithm 6 - sweep line approach

Algorithm CH6(S)

1. sort the points in S from left to right $\langle p_1, p_2, \ldots, p_n \rangle$
2. $L_{upper} \leftarrow \langle p_1, p_2 \rangle$
3. for $i \leftarrow 3$ to n do
4. append p_i to L_{upper}
5. while $|L_{upper}| > 2$ and the last three points (q_1, q_2, q_3) turn left do
6. Delete q2 from L_{upper}
7. $L_{lower} \leftarrow \langle p_1, p_2 \rangle$

…

13. $L \leftarrow$ join(L_{upper}, L_{lower})
14. return L

Time complexity: O(n log n)

CH algorithm 6: running time

Assumption: 10^9 instructions per second

Input size: 1 million points = 10^6 points
$\Rightarrow$ running time ~ $n \log n / 10^9$ = 0.006 seconds

CH in 1 second: 100,000,000 points

Lower bound

Can we do better than O(n log n)?

Prove a lower bound! Use a reduction from Sorting.

Sorting = Ω(n log n) in the algebraic decision tree model

Input to sorting
n real values
$X = (x_1, x_2, \ldots, x_n)$ → **Sort** → The set X sorted in increasing order

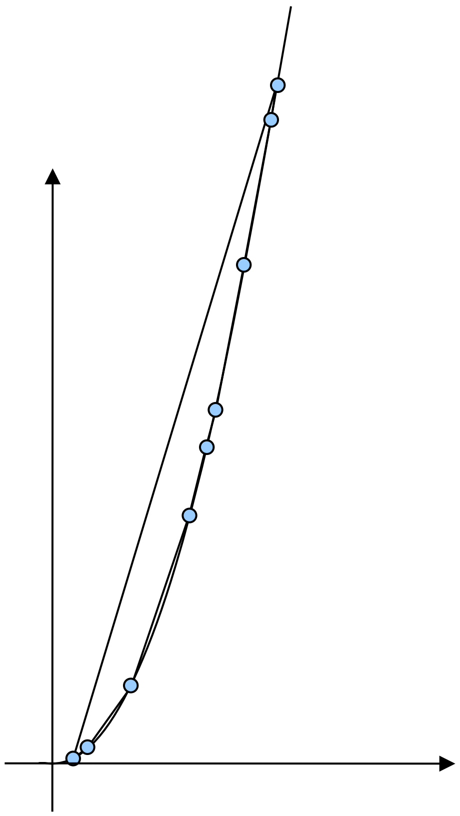
Ω(n log n)

Lower bound

For each value x_i in X construct a point

$$p_i = (x_i, x_i^2)$$

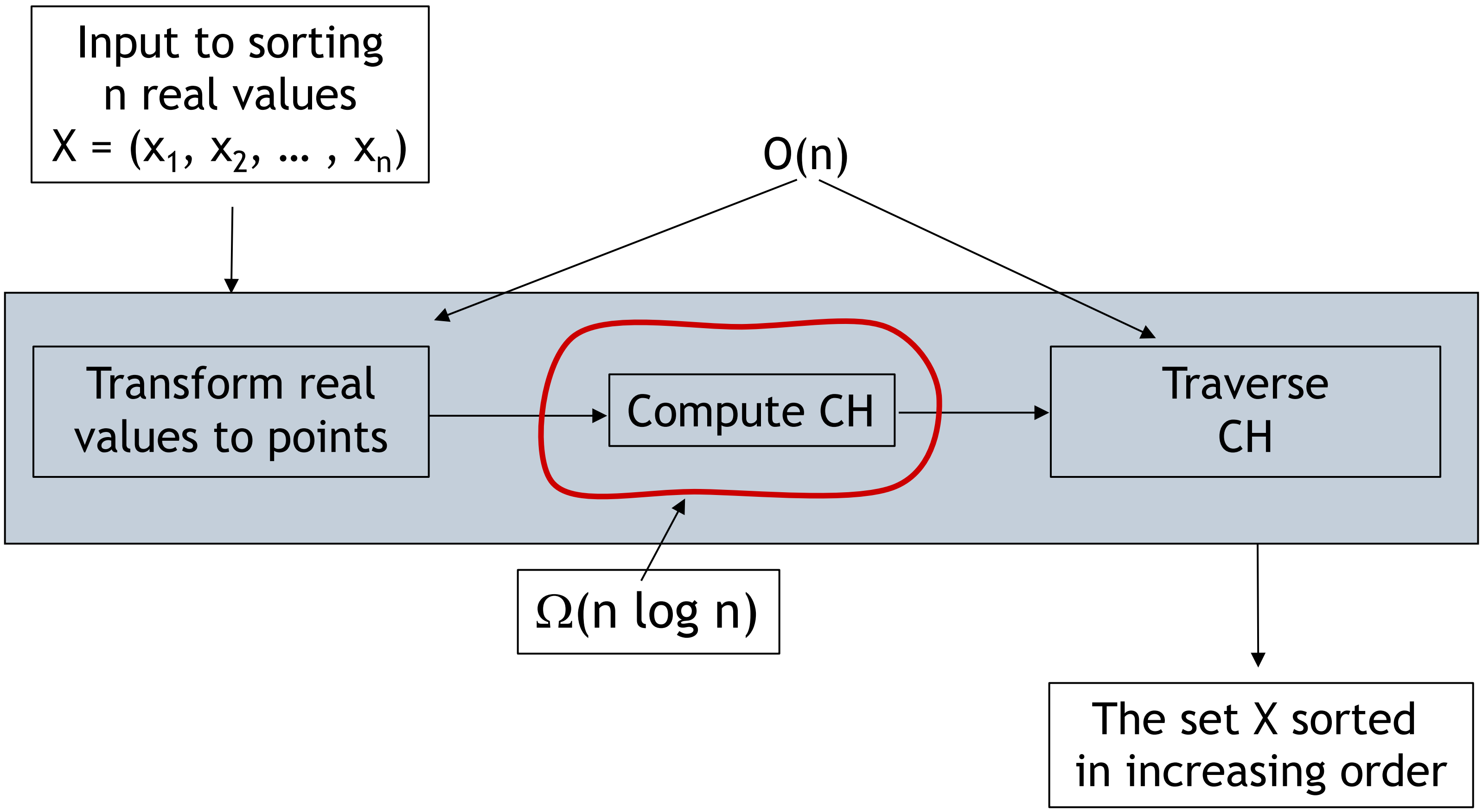
$P = (p_1, p_2, \ldots, p_n)$



Compute CH of P

Find the leftmost point p in the CH.
Traverse the CH counter-clockwise from p and output the vertices in the order they are encountered $\rightarrow$ Points in sorted order!

Lower bound



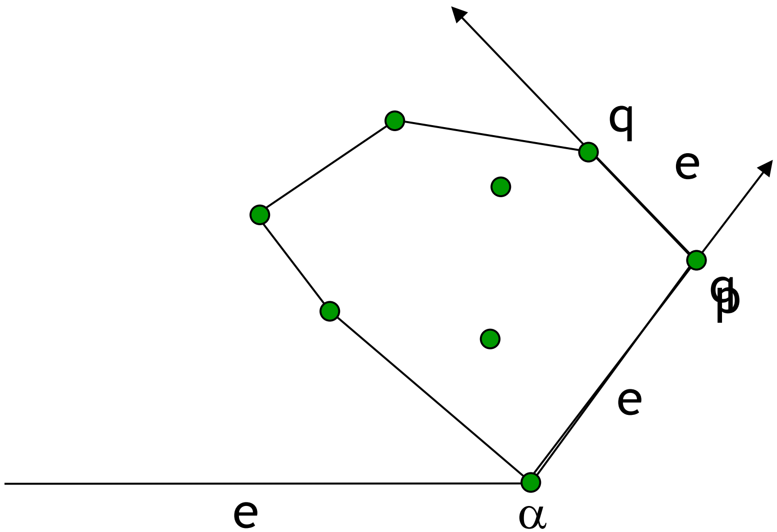
Input to sorting
n real values
$X = (x_1, x_2, \dots, x_n)$
O(n)
Transform real values to points
Compute CH
Traverse CH
$\Omega(n \log n)$
The set X sorted in increasing order

CH algorithm 7: Chen's shattering algorithm

Recall CH algorithm 4

Algorithm CH4(S)

```
find lowest point p in S
α ← (-∞,p_y)
e ← (α,p)
repeat
   maxAngle ← 0
   for every point q in S do
      if ∠(e,(p,q)) > maxAngle then
         nextPoint ← q
         maxAngle ← ∠(e,(p,q))
   e ← (p,nextPoint)
   p ← q
until α == p
```

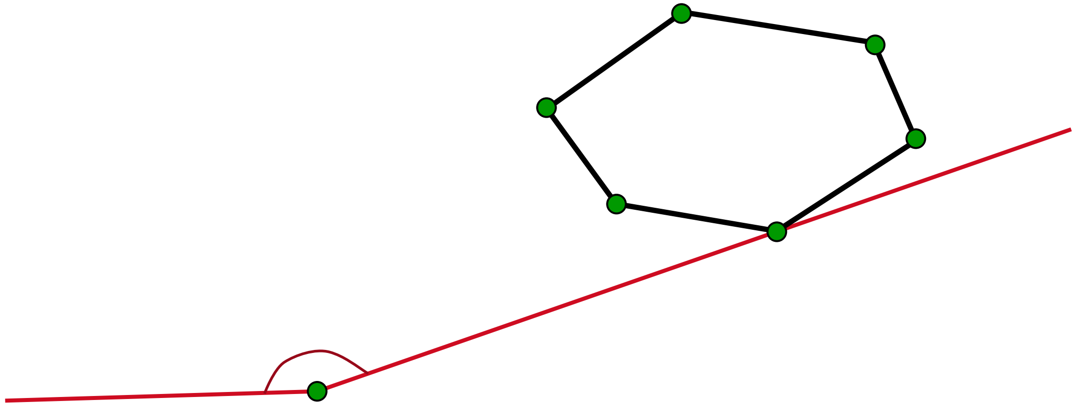


Time complexity: $O(hn)$, where h is the number of points on the boundary of the CH.

What if we could compute the next point faster?

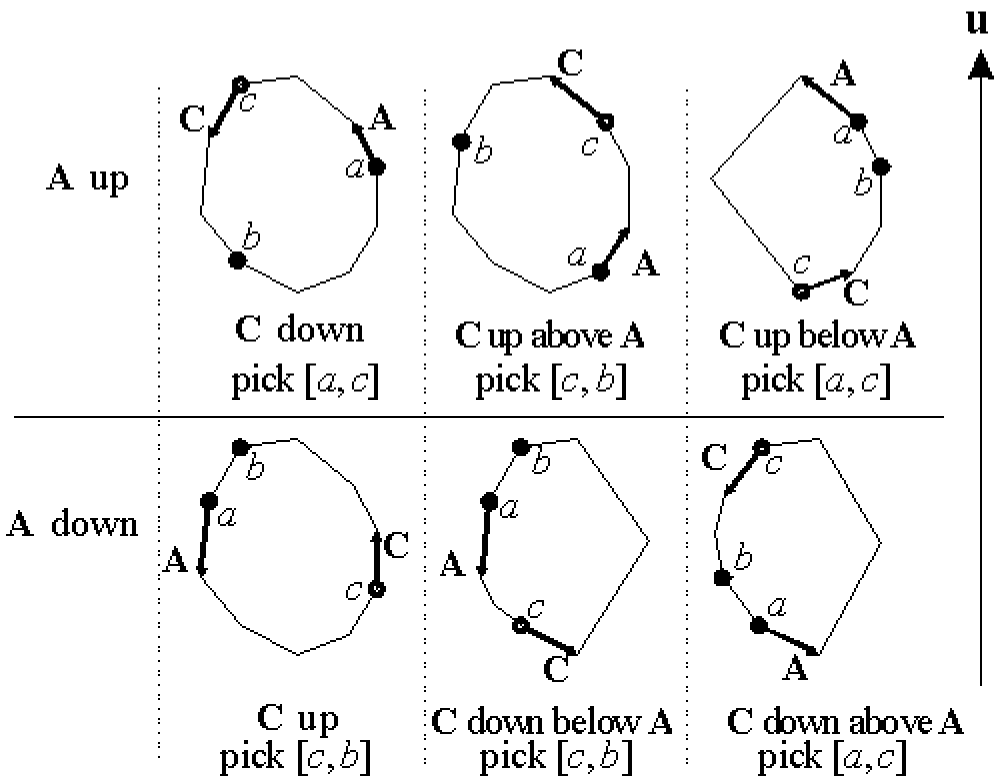
CH algorithm 7: Chen's shattering algorithm

Question: Given a convex polygon with n vertices, how fast can we find an extreme point (maxAngle)?



CH algorithm 7: Chen's shattering algorithm

Question: Given a convex polygon with n vertices, how fast can we find an extreme point (maxAngle)?

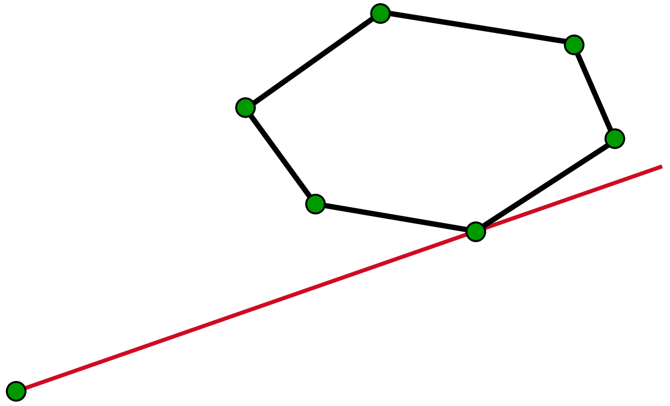


Finding an extreme point with respect to direction u.

Binary search [a,b] with c=(a+b)/2.

CH algorithm 7: Chen's shattering algorithm

Question: Given a convex polygon with n vertices, how fast can we find an extreme point (maxAngle)?



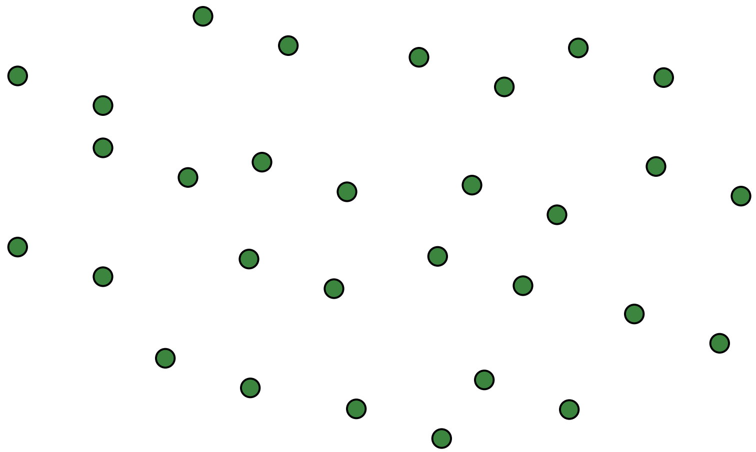
Binary type search

Time: O(log n)

CH algorithm 7: Chen's shattering algorithm

Assume we know the value h [to be discussed later]

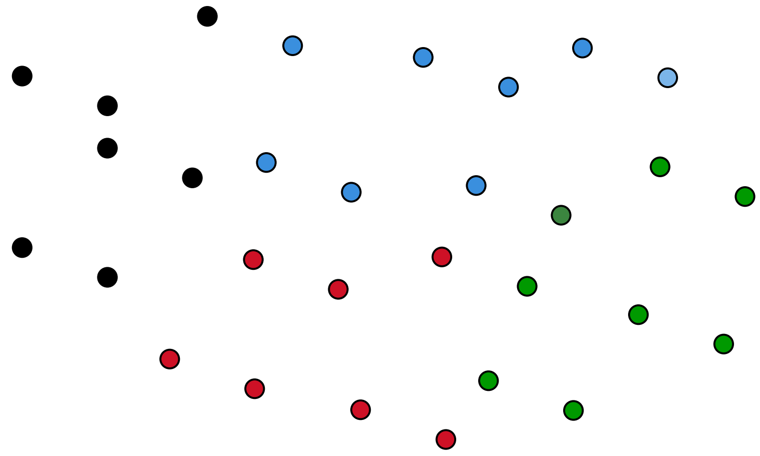
1. Partition the point set into n/h subsets of size h [shattering]



CH algorithm 7: Chen's shattering algorithm

Assume we know the value h [to be discussed later]

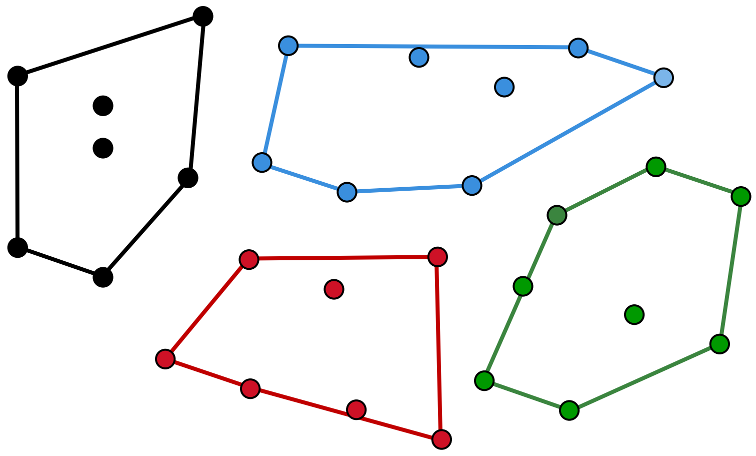
1. Partition the point set into n/h subsets of size h [shattering]



CH algorithm 7: Chen's shattering algorithm

Assume we know the value h [to be discussed later]

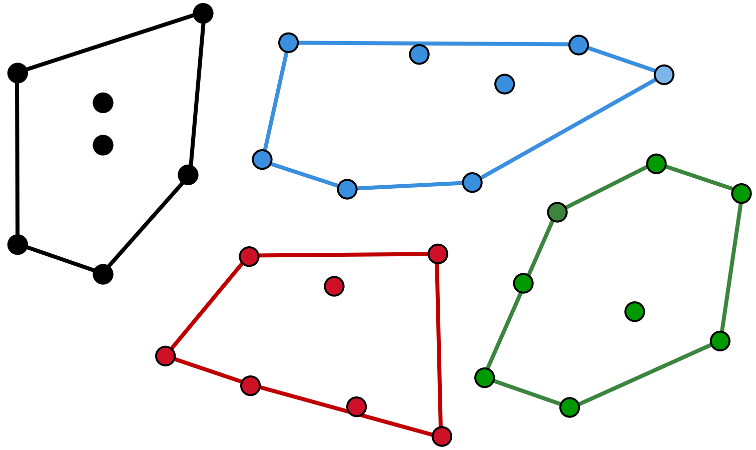
1. Partition the point set into n/h subsets of size h [shattering]
2. Compute the CH of each subset



CH algorithm 7: Chen's shattering algorithm

Assume we know the value h [to be discussed later]

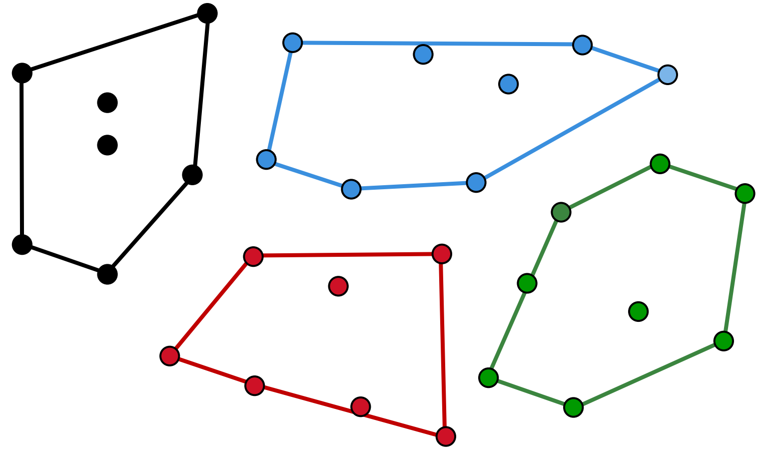
1. Partition the point set into n/h subsets of size h [shattering]
2. Compute the CH of each subset



CH algorithm 7: Chen's shattering algorithm

Assume we know the value h [to be discussed later]

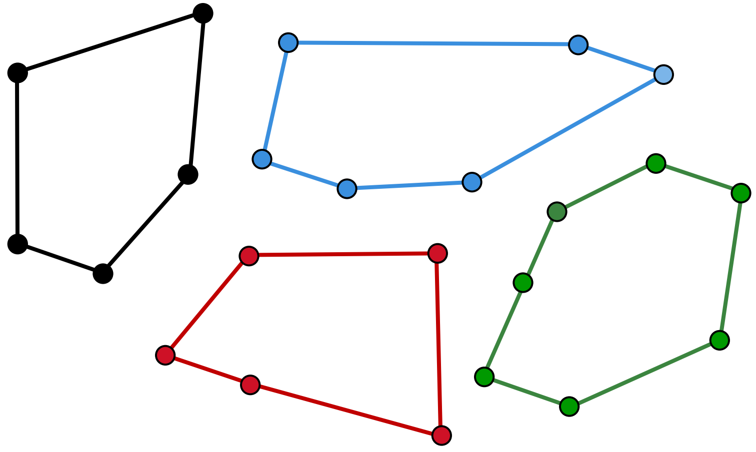
1. Partition the point set into n/h subsets of size h [shattering]
2. Compute the CH of each subset [n/h · O(h log h) = O(n log h)]



CH algorithm 7: Chen's shattering algorithm

Assume we know the value h [to be discussed later]

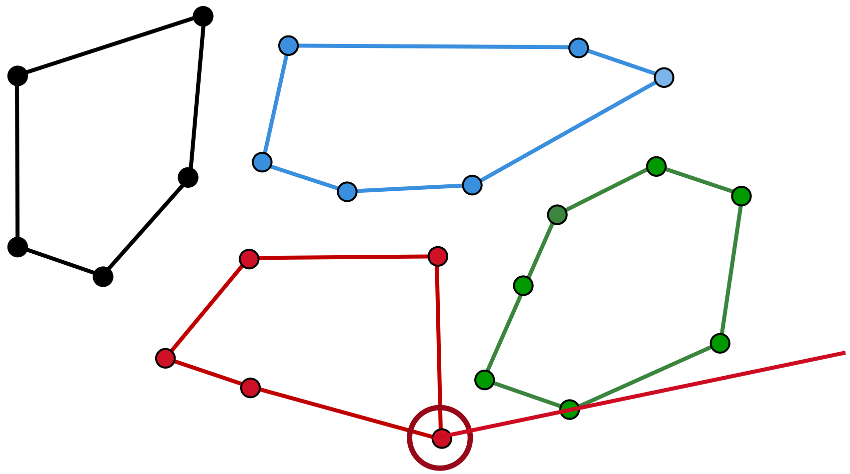
1. Partition the point set into n/h subsets of size h [shattering]
2. Compute the CH of each subset [$n/h \cdot O(h \log h) = O(n \log h)$]
3. Use algorithm 4 (with binary search for each component)



CH algorithm 7: Chen's shattering algorithm

Assume we know the value h [to be discussed later]

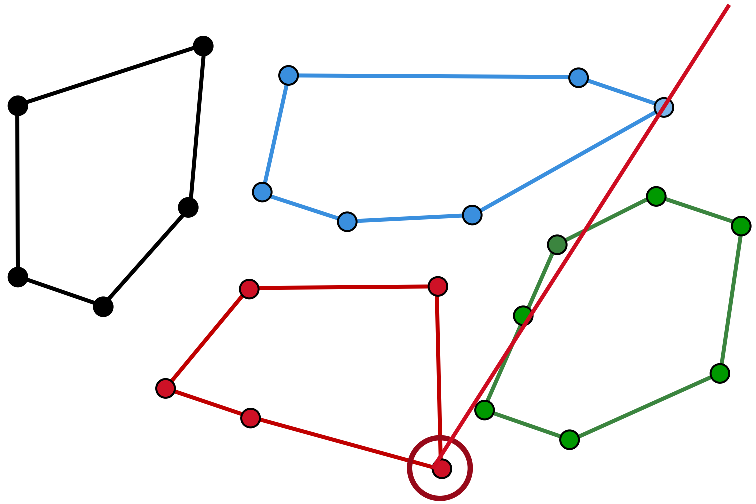
1. Partition the point set into n/h subsets of size h [shattering]
2. Compute the CH of each subset [n/h · O(h log h) = O(n log h)]
3. Use algorithm 4 (with binary search for each component)



CH algorithm 7: Chen's shattering algorithm

Assume we know the value h [to be discussed later]

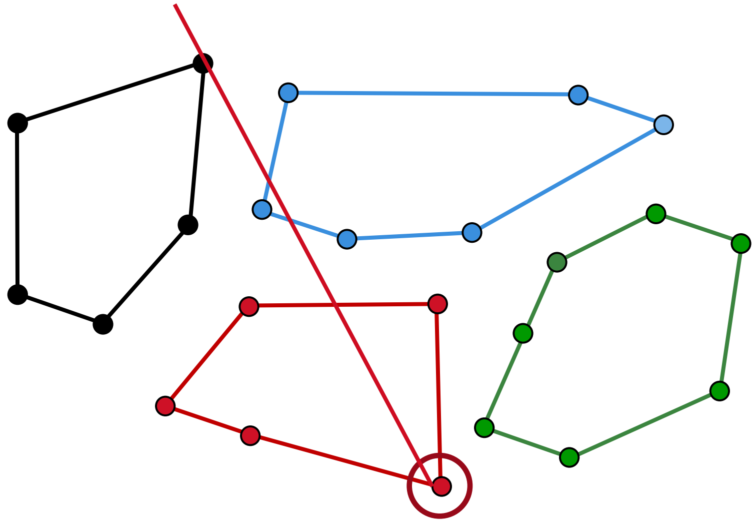
1. Partition the point set into n/h subsets of size h [shattering]
2. Compute the CH of each subset [$n/h \cdot O(h \log h) = O(n \log h)$]
3. Use algorithm 4 (with binary search for each component)



CH algorithm 7: Chen's shattering algorithm

Assume we know the value h [to be discussed later]

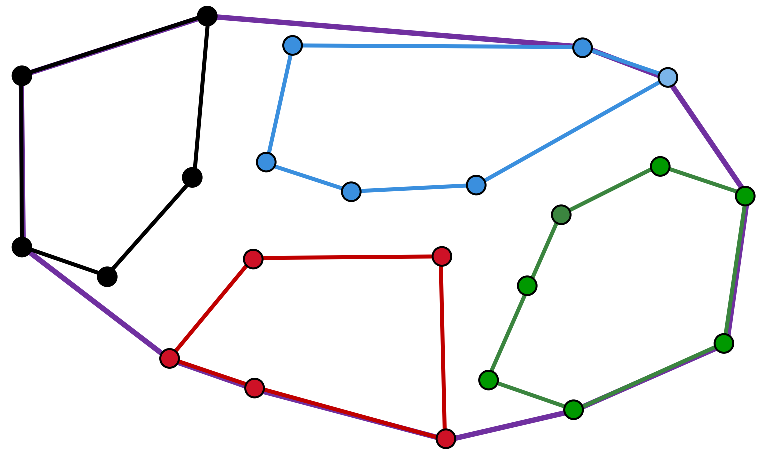
1. Partition the point set into n/h subsets of size h [shattering]
2. Compute the CH of each subset [n/h · O(h log h) = O(n log h)]
3. Use algorithm 4 (with binary search for each component)



CH algorithm 7: Chen's shattering algorithm

Assume we know the value h [to be discussed later]

1. Partition the point set into n/h subsets of size h [shattering]
2. Compute the CH of each subset [n/h · O(h log h) = O(n log h)]
3. Use algorithm 4 (with binary search for each component)
   [O(n/h · log h) to find next vertex of CH]
   $\Rightarrow$ [O(h · n/h · log h)=O(n log h)]



CH algorithm 7: Chen's shattering algorithm

What if we do not know h?

1. Set h=3
2. Run the algorithm (Algorithm 4) trying to build a CH of size at most h, after h iterations stop. [even if CH is not finished]
3. If CH complete
      return CH
   else
      set $h=h^2$ and goto step 2.

Running time?

CH algorithm 7: Chen's shattering algorithm

What if we do not know h?

1. Set h=3
2. Run the algorithm (Algorithm 4) trying to build a CH of size at most h, after h iterations stop. [even if CH is not finished]
3. If CH complete
   return CH
   else
   set $h=h^2$ and goto step 2.

Running time: $O(n \log 3) + O(n \log 3^2) + \ldots + O(n \log 3^{2^k})$ where $h \approx 3^{2^k}$

CH algorithm 7: Chen's shattering algorithm

What if we do not know h?

1. Set h=3
2. Run the algorithm (Algorithm 4) trying to build a CH of size at most h, after h iterations stop. [even if CH is not finished]
3. If CH complete
   return CH
   else
   set h=h^2 and goto step 2.

Running time: $O(n \log 3) + O(n \log 3^2) + \ldots + O(n \log 3^{2^k})$ where $h \approx 3^{2^k}$

$= O(n \log 3 + 2n \log 3 + \ldots + 2^k n \log 3)$

CH algorithm 7: Chen's shattering algorithm

What if we do not know h?

1. Set h=3
2. Run the algorithm (Algorithm 4) trying to build a CH of size at most h, after h iterations stop. [even if CH is not finished]
3. If CH complete
   return CH
   else
   set $h=h^2$ and goto step 2.

Running time: $O(n \log 3) + O(n \log 3^2) + \ldots + O(n \log 3^{2^k})$ where $h \approx 3^{2^k}$

$= O(n \log 3 + 2n \log 3 + \ldots + 2^k n \log 3)$
$= O(2^k n) = O(n \log h)$

What is known?

Preparata & Hong’77	O(n log n)
Kirkpatrick & Seidel’86	O(n log h)
Chen’93	O(n log h)
Dynamic convex hull	
Brodal & Jacob’02	O(log n) time/update

[This was an open problem since 1981]

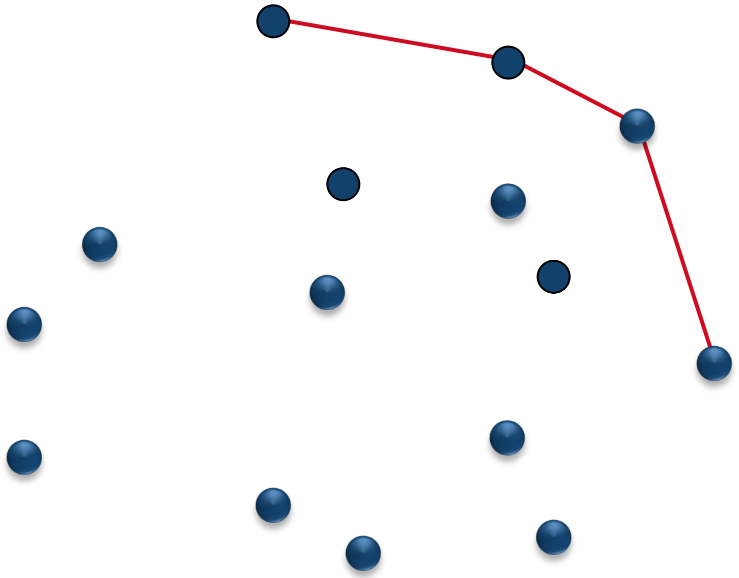
d dimensions	
Chazelle’93	$\Theta(n \log n + n^{\lfloor d/2 \rfloor})$

What's the expected size of CH?

Assumption: Points uniformly distributed in a unit square in 2D

Theorem: If n points are sampled from a uniform distribution in a unit square, then the expected number of points on the convex hull is O(log n).

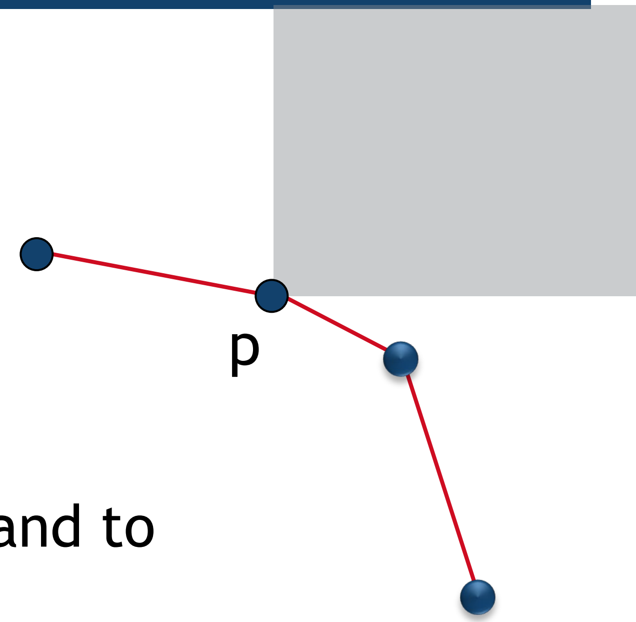
Proof: Consider the upper right (UR) part of the CH.



What’s the expected size of CH?

Consider a point p on the UR hull.

What’s known about p?

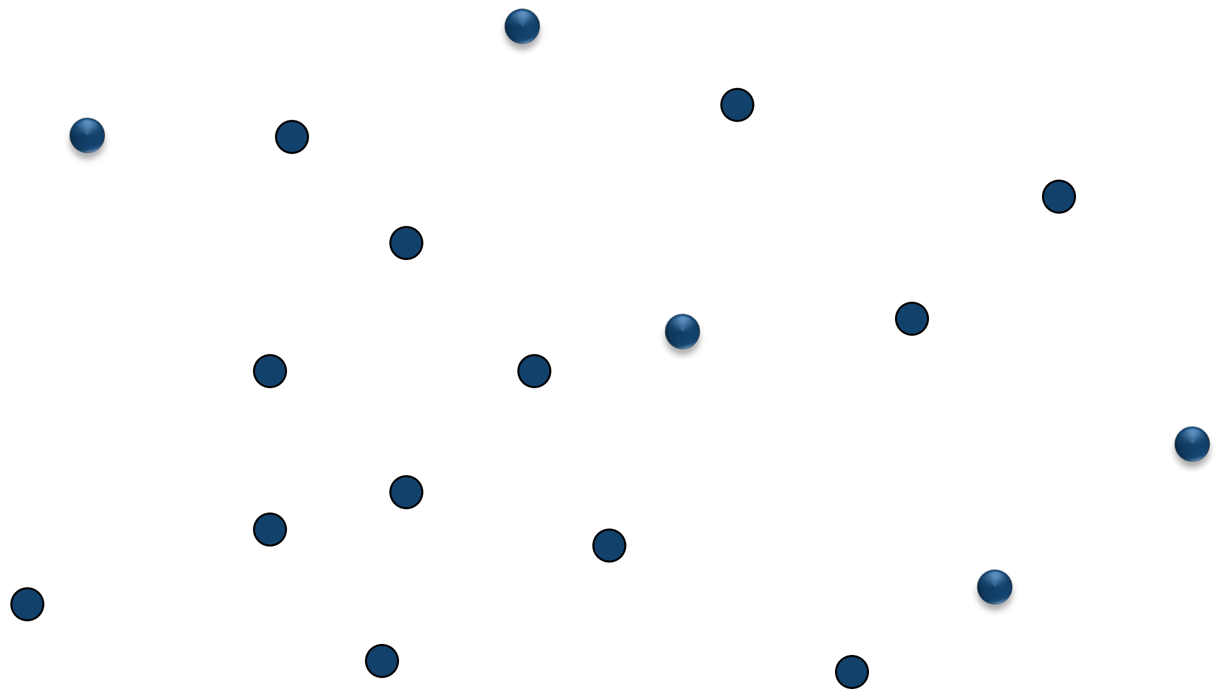


Observation: No other point can lie above and to the right of p.
[Not dominated by any other point]

Question: How many points are not dominated by others?

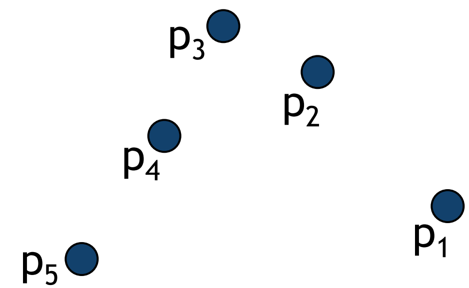
What’s the expected size of CH?

Question: How many points are no dominated by others?



What's the expected size of CH?

Question: How many points are not dominated by others?



Equivalent to:

$p_{i.y}$ is greater than $p_{1.y}$, $p_{2.y}$, … , $p_{(i-1),y}$

12, 45, 65, 23, 5, 87, 34, 14, 76, 34, 9, 81, 36

What is the probability that $p_{i.y}$ is greater than $p_{1.y}$, $p_{2.y}$, … , $p_{(i-1).y}$?
- Independent from a uniform distribution

Probability is 1/i
(each point is equally likely to be the maximum)

What's the expected size of CH?

Let us sum up the probability for all the points.

$$\sum_{i=1}^{n} 1/i = 1+1/2+1/3+ \ldots +1/n < \log n + 1$$

Answer: For points uniformly distributed in a unit square the expected number of points on the CH is $O(\log n)$.

Disk: For points uniformly distributed in a disk the expected number of points on the CH is $O(n^{1/3})$. [Raynaud'70]