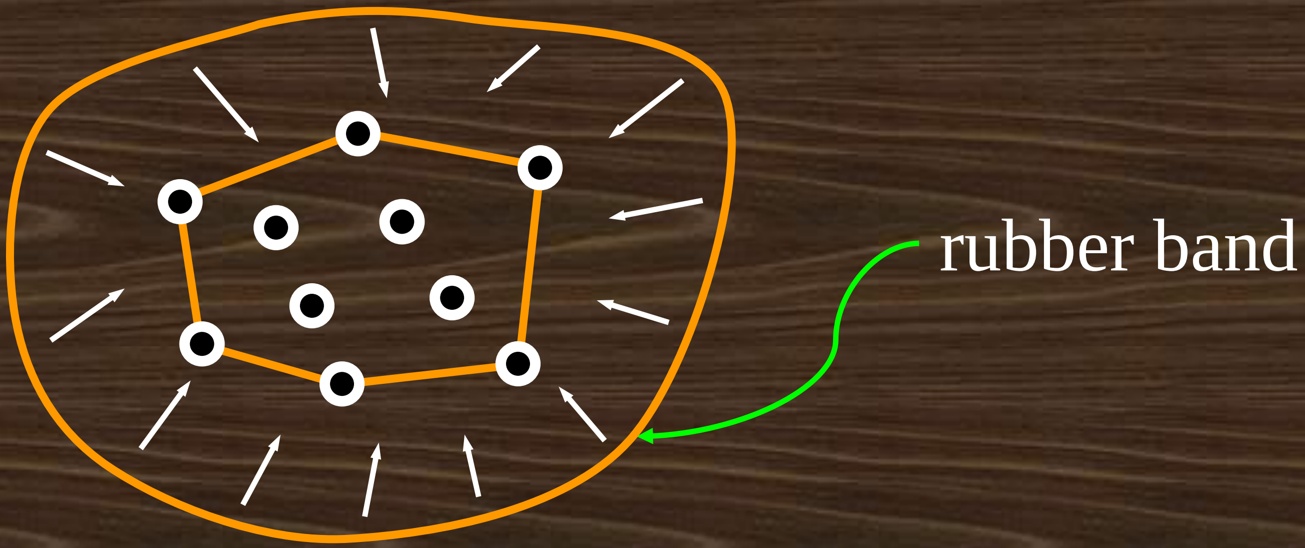
computing
CONVEX HULLS

Presentation Outline

- 2D Convex Hulls
  - Definitions and Properties
  - Approaches:
    - *Brute Force*
    - *Gift Wrapping*
    - *QuickHull*
    - *Graham Scan*
    - *Incremental*
    - *Divide and Conquer*
    - *By Delaunay Triangulation & Voronoi Diagram (later)*

Definitions and Properties

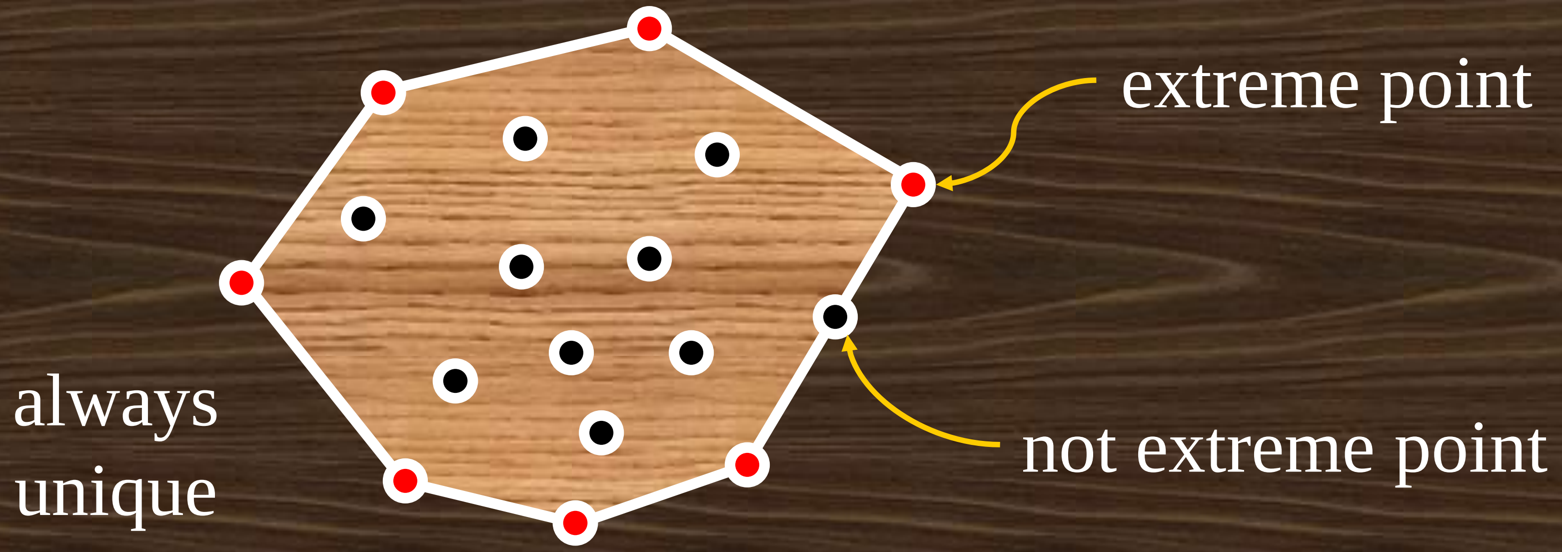


- Intersection of all convex sets containing *P*
- Smallest convex set containing *P*
- Intersection of all half-planes containing *P*
- Union of all triangles formed by points of *P*

Definitions and Properties

– Smallest *convex polygon* containing *P*

– *All vertices of hull are some points of P*



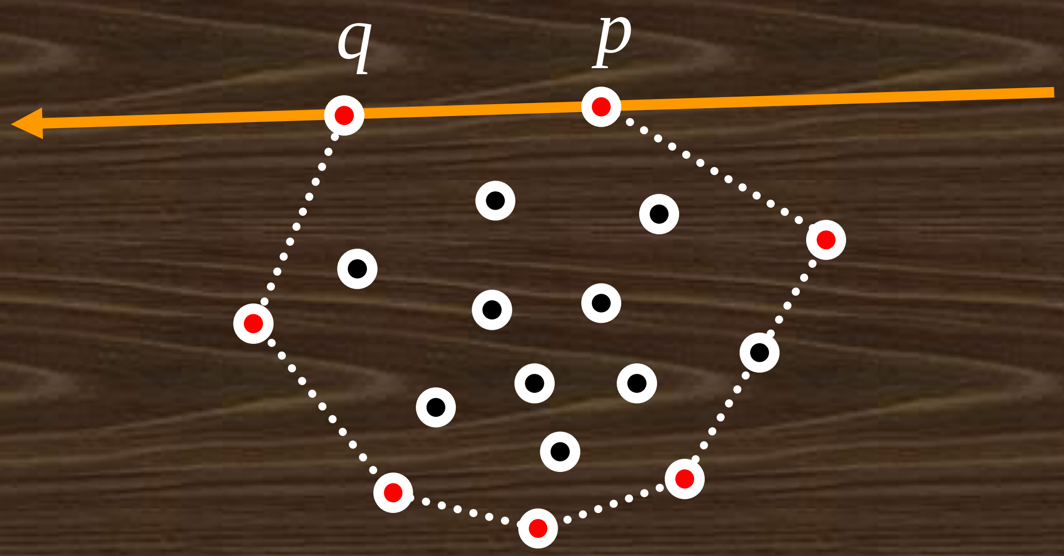
– **NOTE: convex hull is the closed solid region, not just the boundary**

Brute-Force Approach

- **Determine extreme edges**

  for each pair of points *p*,*q*∈*P* do

  if all other points lie on one side of line passing thru *p* and *q* then keep edge (*p*, *q*)



Brute-Force Approach

- Next, sort edges in counterclockwise order
  - we want output in counterclockwise
- Running time: $O(n^3)$
  - bad but not the worst yet

Gift Wrapping

$p \leftarrow$ the lowest point p_0

repeat

  for each $q \in P$ and $q \neq p$ do

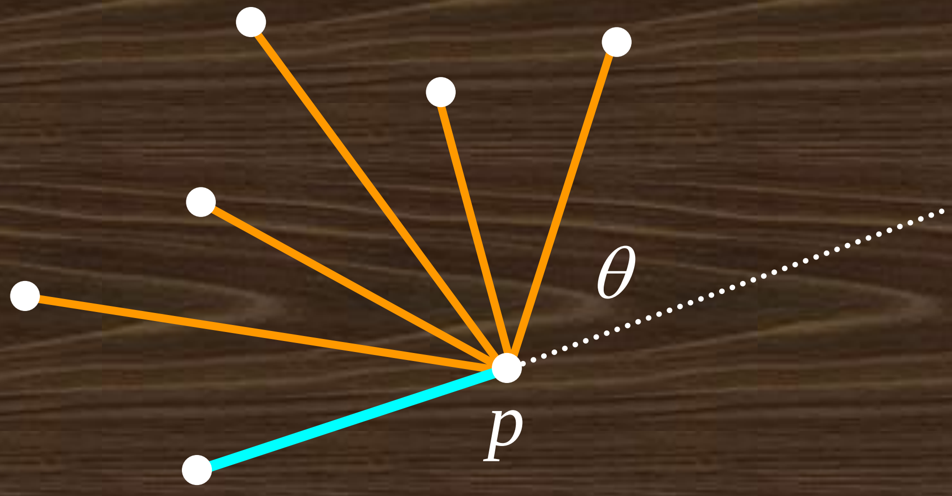
    compute counterclockwise angle θ from previous hull edge

  let r be the point with smallest θ

  output (p, r) as a hull edge

  $p \leftarrow r$

until $p = p_0$



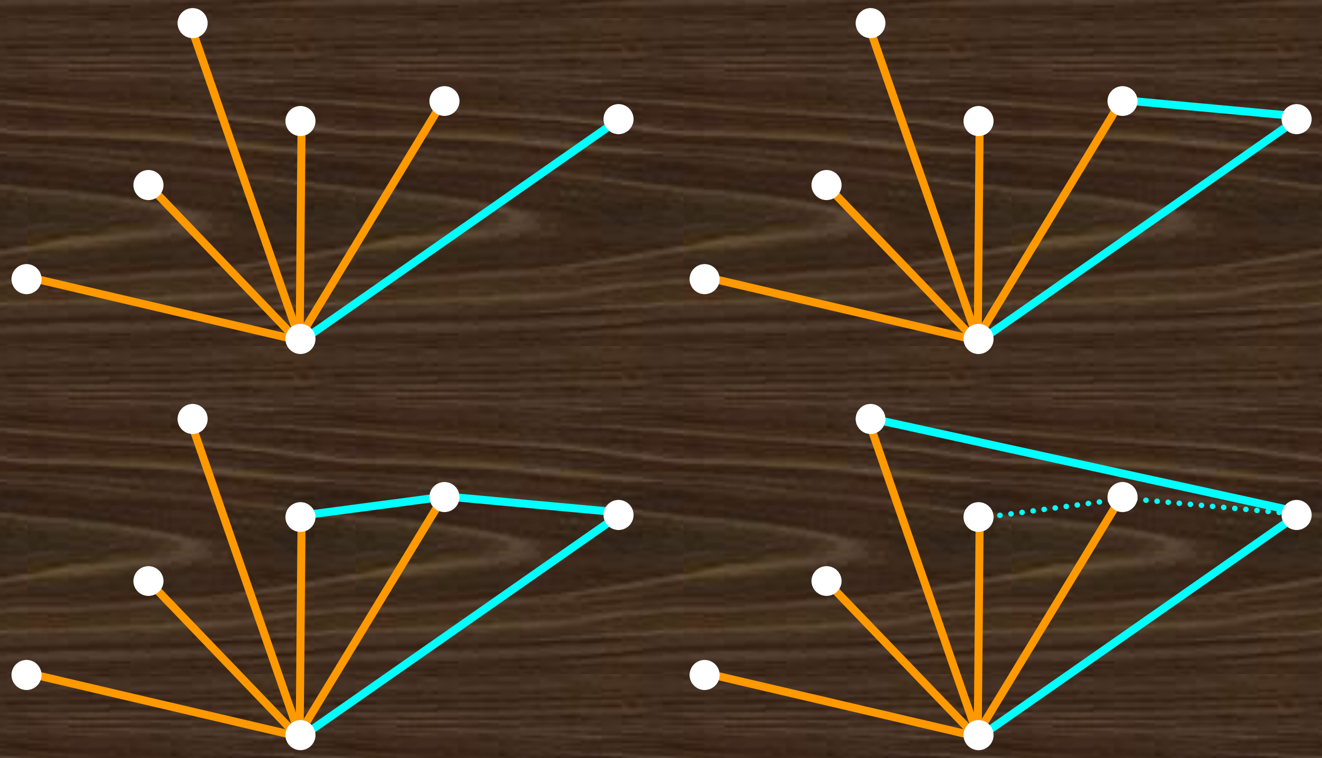
Gift Wrapping

- First suggested by Chand and Kapur (1970)
- Worst-case time: $O(n^2)$
- Output-sensitive time: $O(nk)$
  - where k is the # of vertices of hull
- Can be extended to higher dimension
  - was the primary algorithm for higher dimensions for quite some time

Graham Scan

- By Graham (1972)
- First algorithm to achieve optimal running time
- Uses angular sweep

Graham Scan



- Animated demo
  - http://www.gris.uni-tuebingen.de/gris/grdev/java/algo/solutions/lesson12/ConvexHull.html

Graham Scan

Find rightmost lowest point p_0

Sort all other points angularly about p_0,

break ties in favor of closeness to p_0;

label them $p_1, p_2, \ldots, p_{n-1}$

Stack $S = (p_{n-1}, p_0) = (p_{t-1}, p_t)$; t indexes top

$i \leftarrow 1$

while $i < n$ do

if p_i is strictly left of (p_{t-1}, p_t) then

Push(S, i); i++

else Pop(S)

Graham Scan

- Running time: O(*n* lg *n*)
  - the whole sweep takes O(*n*) only because each point can be pushed or popped at most once
  - O(*n* lg *n*) due to sorting of angles
- No obvious extension to higher dimensions

Lower Bound

- Is $\Theta(n \lg n)$ the lower bound running time of any 2D convex hull algorithm?
- Yes
  - Proven by Yao (1981) using decision tree
- Can be proven by reducing sorting to finding convex hull
  - Sorting has lower bound $\Theta(n \lg n)$

Lower Bound

- We can use a convex hull algorithm to sort (Shamos, 1978)

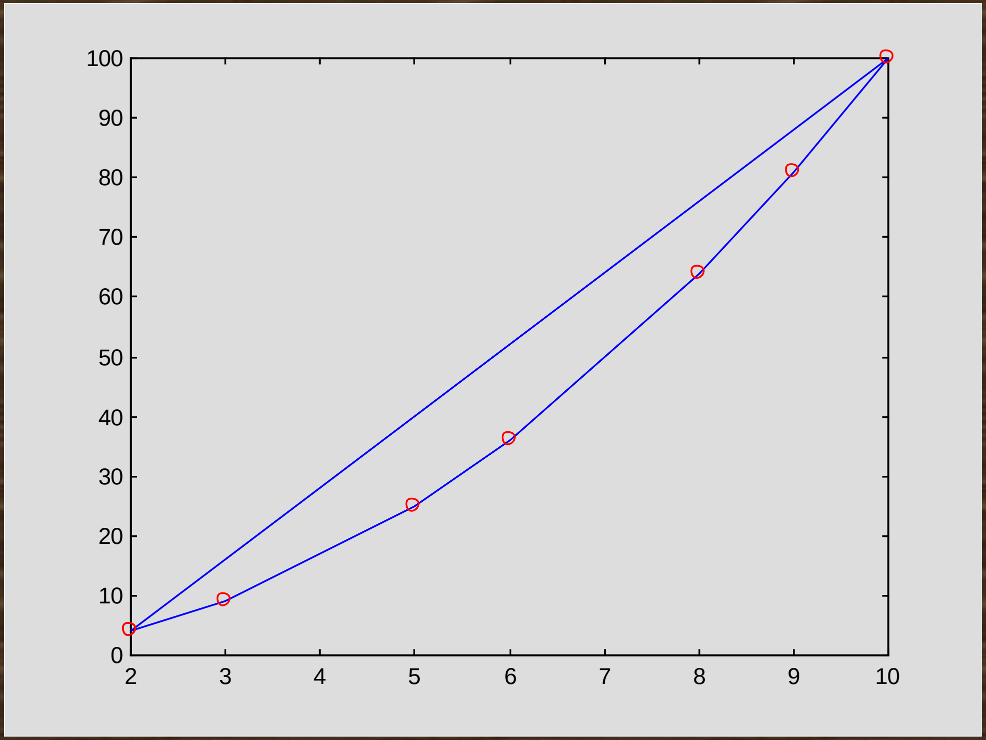
for each input number x do

create a 2D point (x, x^2)

Construct a hull for these points

Find the lowest point on the hull and follow the vertices of the hull

Lower Bound

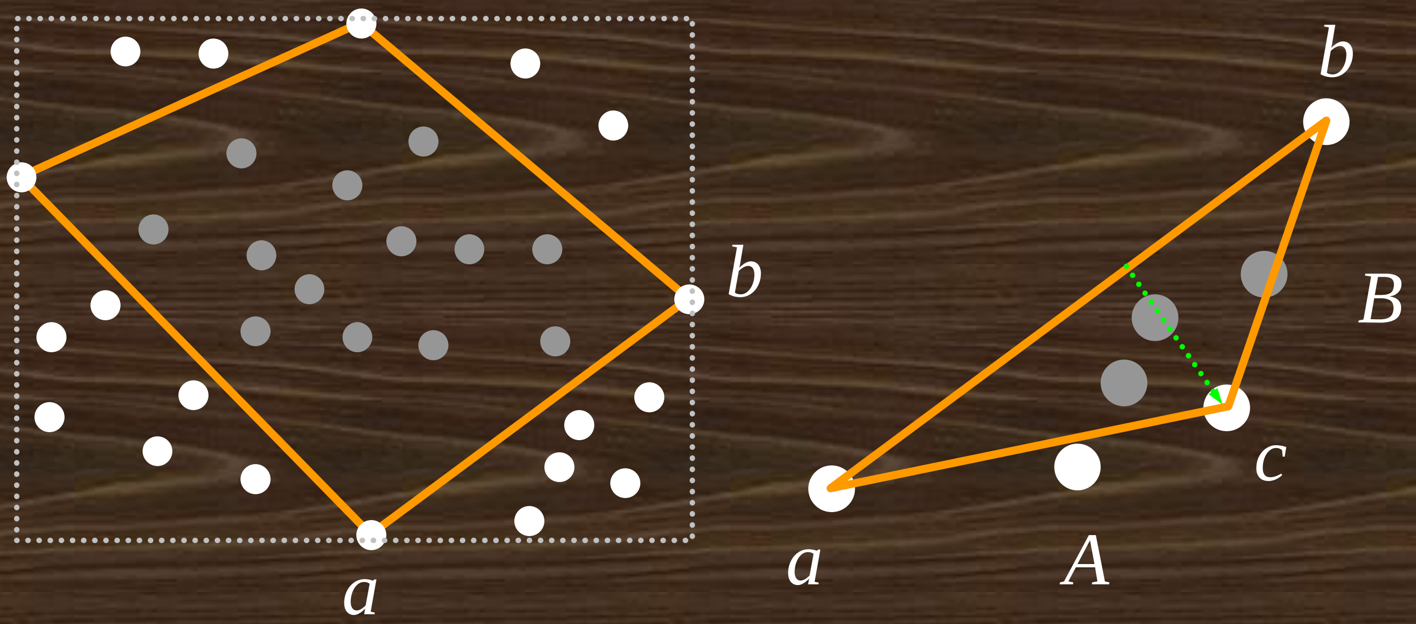


- If we can compute hull faster than $\Theta(n \lg n)$ then we can sort faster than $\Theta(n \lg n)$. Impossible!

QuickHull

- Suggested by researchers in late 1970s
- Dubbed the "QuickHull" algorithm by Preparata and Shamos (1985) because of similarity to QuickSort
- Works by recursively discarding points
- Very commonly implemented, just like QuickSort
  - Qhull
    - http://www.geom.umn.edu/software/qhull

QuickHull



- **Animated demo**
  - **http://www.piler.com/convexhull/**

QuickHull

```
function QuickHull(a, b, S)
   if S = {a, b} then return {a, b}
   else
      c ← point furthest from edge (a, b)
      A ← points right of (a, c)
      B ← points right of (c, b)
      return QuickHull(a, c, A) concatenate with
             QuickHull(c, b, B)
```

- Worst-case time: $O(n^2)$
  - when "partitions" are very unbalanced
- Best- and average-case time: $O(n \lg n)$

Divide & Conquer

- First applied to convex hull problem by Preparata and Hong (1977)

- The only technique known to extend to 3D and still achieve O(*n* lg *n*) time complexity

Divide & Conquer

Sort the points from left to right

Let A be the leftmost $\lceil n/2 \rceil$ points

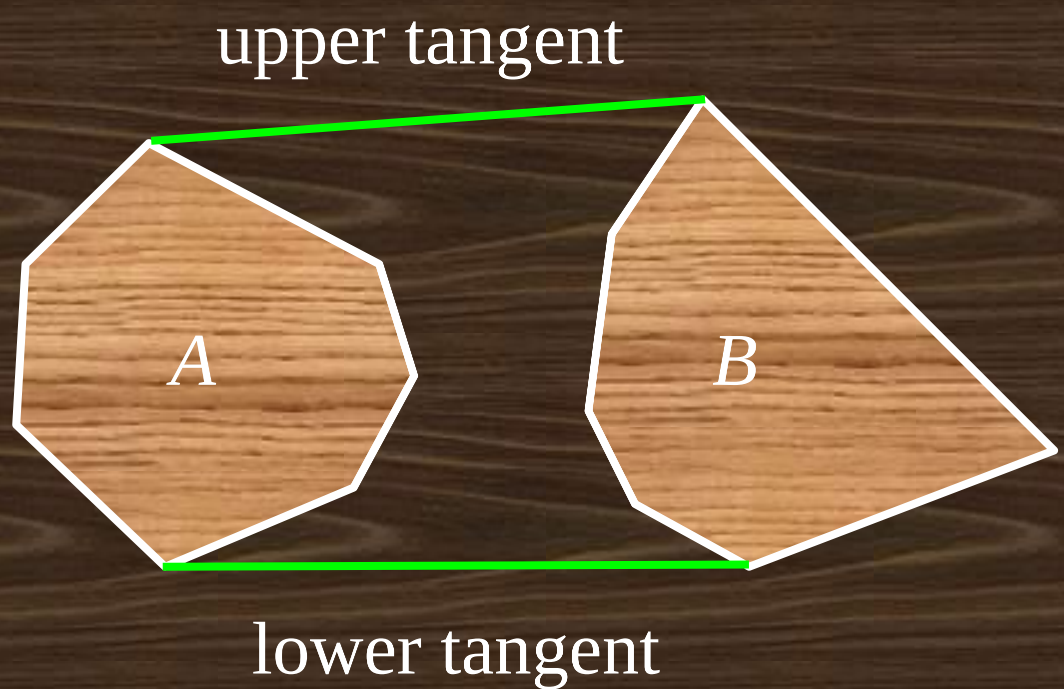
Let B be the rightmost $\lfloor n/2 \rfloor$ points

Compute convex hulls $H(A)$ and $H(B)$

Compute $H(A \cup B)$ by merging $H(A)$ and $H(B)$

- Merging is tricky, but can be done in linear time

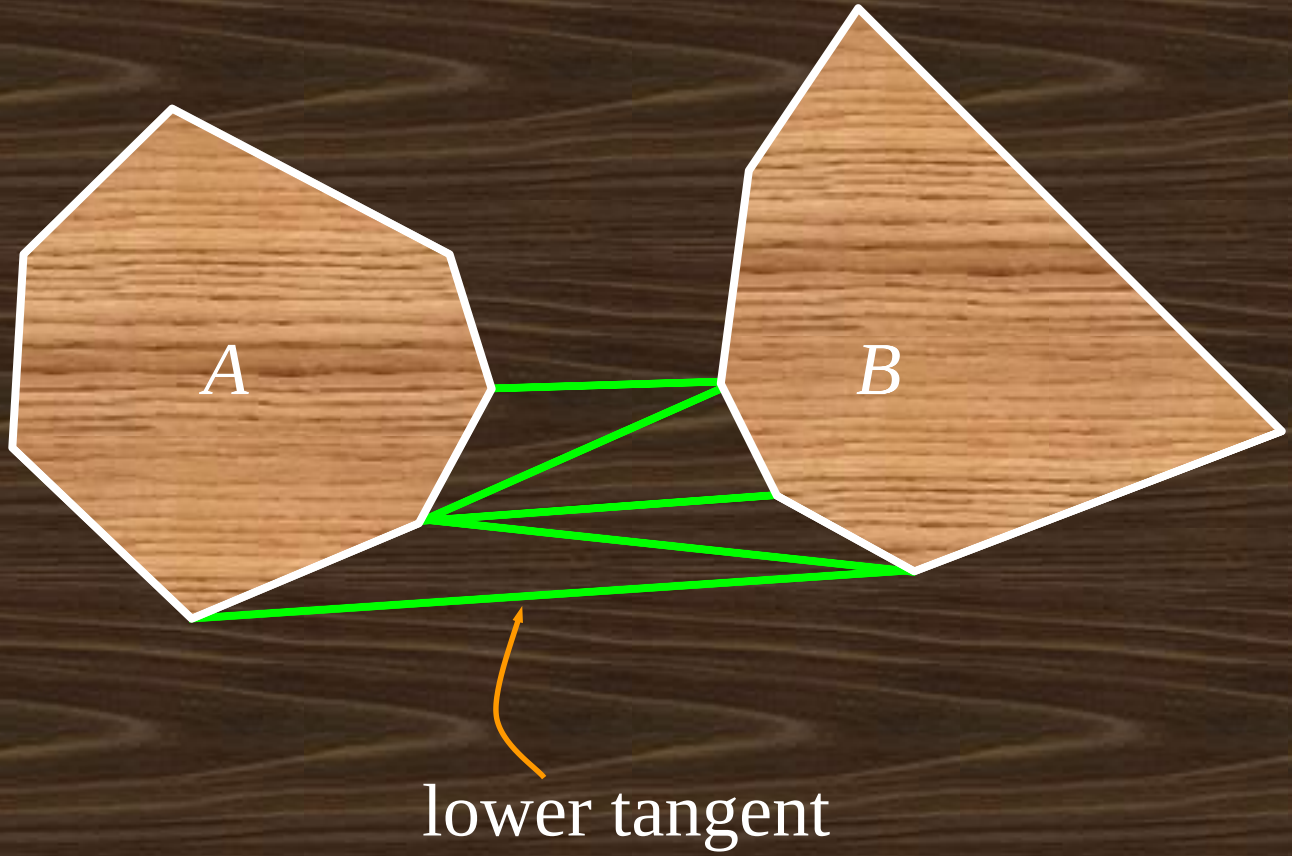
Divide & Conquer



- Need to find the upper and lower tangents
- They can be found in linear time

Divide & Conquer

- Find lower tangent



Divide & Conquer

- Find lower tangent

  Let a be the rightmost point of A

  Let b be the leftmost point of B

  while ab not lower tangent of both A and B do

      while ab not lower tangent to A do

          $a \leftarrow$ next clockwise point on $H(A)$

      while ab not lower tangent to B do

          $b \leftarrow$ next counterclockwise point on $H(B)$

Incremental Algorithm

- Basic idea

  $H_2 \leftarrow \text{conv}\{ p_0, p_1, p_2 \}$

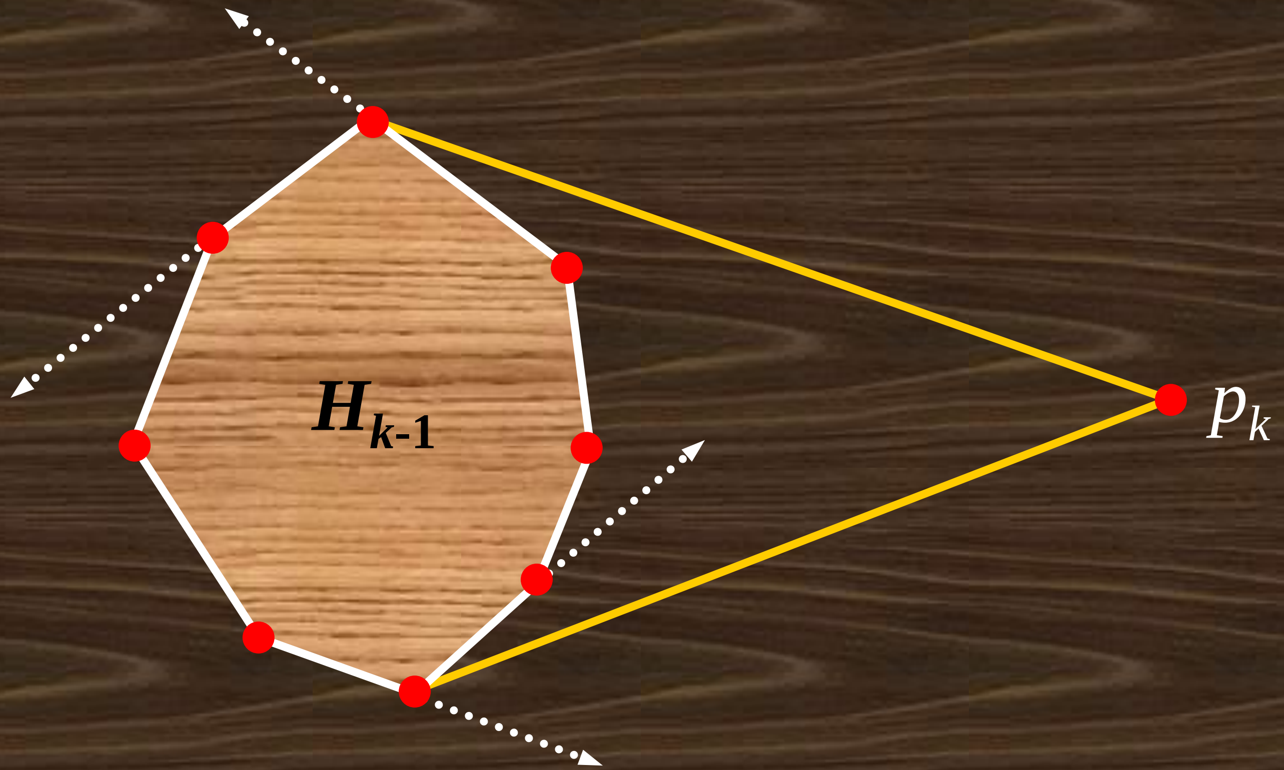
  for $k \leftarrow 3$ to n - 1 do

  $H_k \leftarrow \text{conv}\{ H_{k-1} \cup p_k \}$

- Two cases to consider
  - case 1: $p_k \in H_{k-1}$
    - *need not update hull*
  - case 2: $p_k \notin H_{k-1}$
    - *need to update hull*

Incremental Algorithm

- **When $p_k \notin H_{k-1}$**



Incremental Algorithm

- Requires $O(n^2)$ time but points can be dynamically added to P
- Can be done in randomized expected time $O(n \lg n)$
- Can be improved to worst-case $O(n \lg n)$ (Edelsbrunner, 1987)

Sort points from left to right

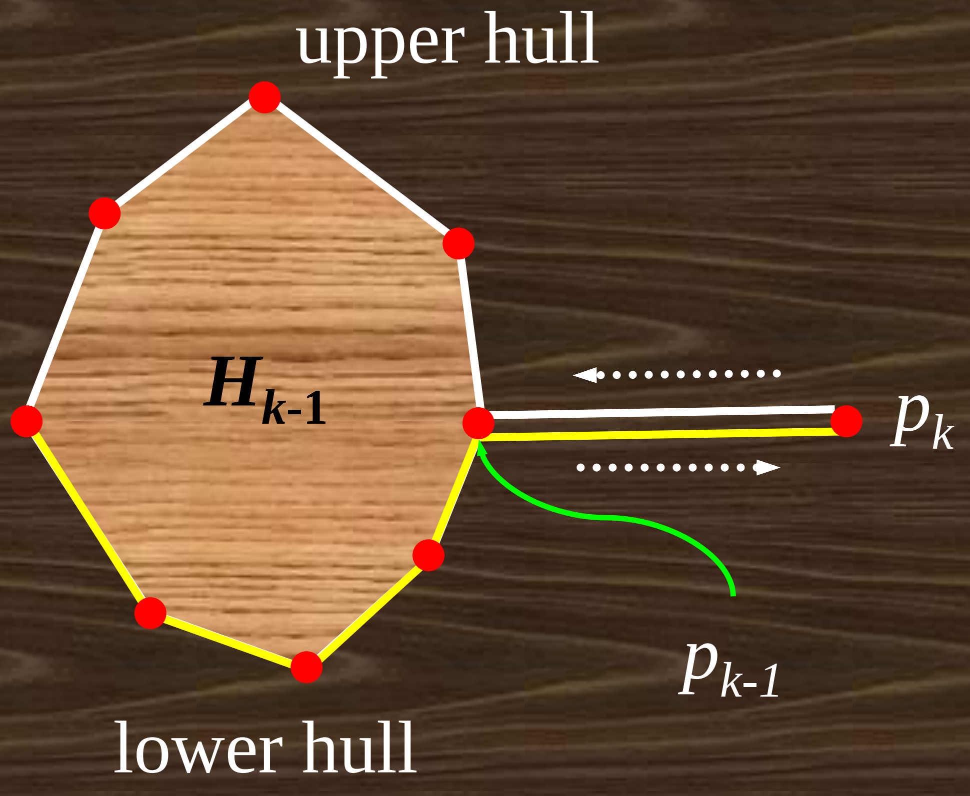
$H_2 \leftarrow$ conv2{ p_0, p_1, p_2 }

for $k \leftarrow 3$ to $n - 1$ do

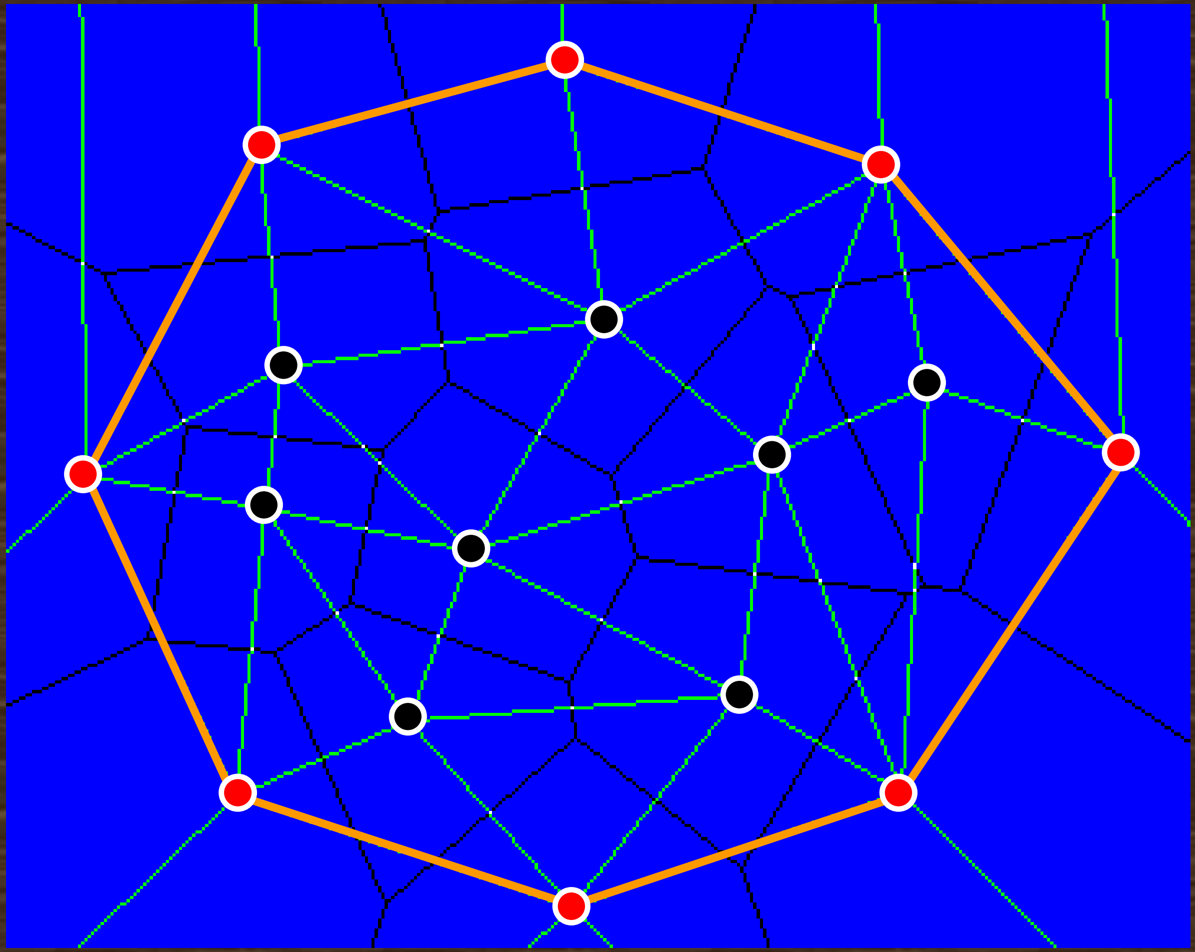
$\quad H_k \leftarrow$ conv2{ $H_{k-1} \cup p_k$ }

Incremental Algorithm

- Always $p_k \notin H_{k-1}$



By Delaunay Triangulation & Voronoi Diagram



By Delaunay Triangulation

Compute Delaunay triangulation of P

$p \leftarrow$ the rightmost lowest point p_0 in P

repeat

    for each point adjacent to p do

        compute counterclockwise angle θ from previous hull edge

    let q be the point with smallest θ

    output (p, q) as a hull edge

    $p \leftarrow q$

until $p = p_0$

By Delaunay Triangulation

- Delaunay triangulation can be computed in O(*n* lg *n*) time
- The rest takes O(*n*) time
- Therefore, total time is O(*n* lg *n*)

- Can use Voronoi diagram similarly since it is the dual of Delaunay triangulation

3D CONVEX HULLS

3D Convex Hulls

- 3D convex hull is the smallest convex polyhedron or 3-polytope enclosing P

- Complexity of 3D convex hull
  - Euler's formula: $V - E + F = 2$
  - F and E are $O(n)$ where $n = V$

3D Gift Wrapping

- Basic idea

Let partial hull $H \leftarrow$ a triangle on the hull

for each edge e on the boundary of H do

let $F \in H$ be the face bounded by e

let π be the plane containing F

"bent" π over e toward the other points until the first point p is encountered

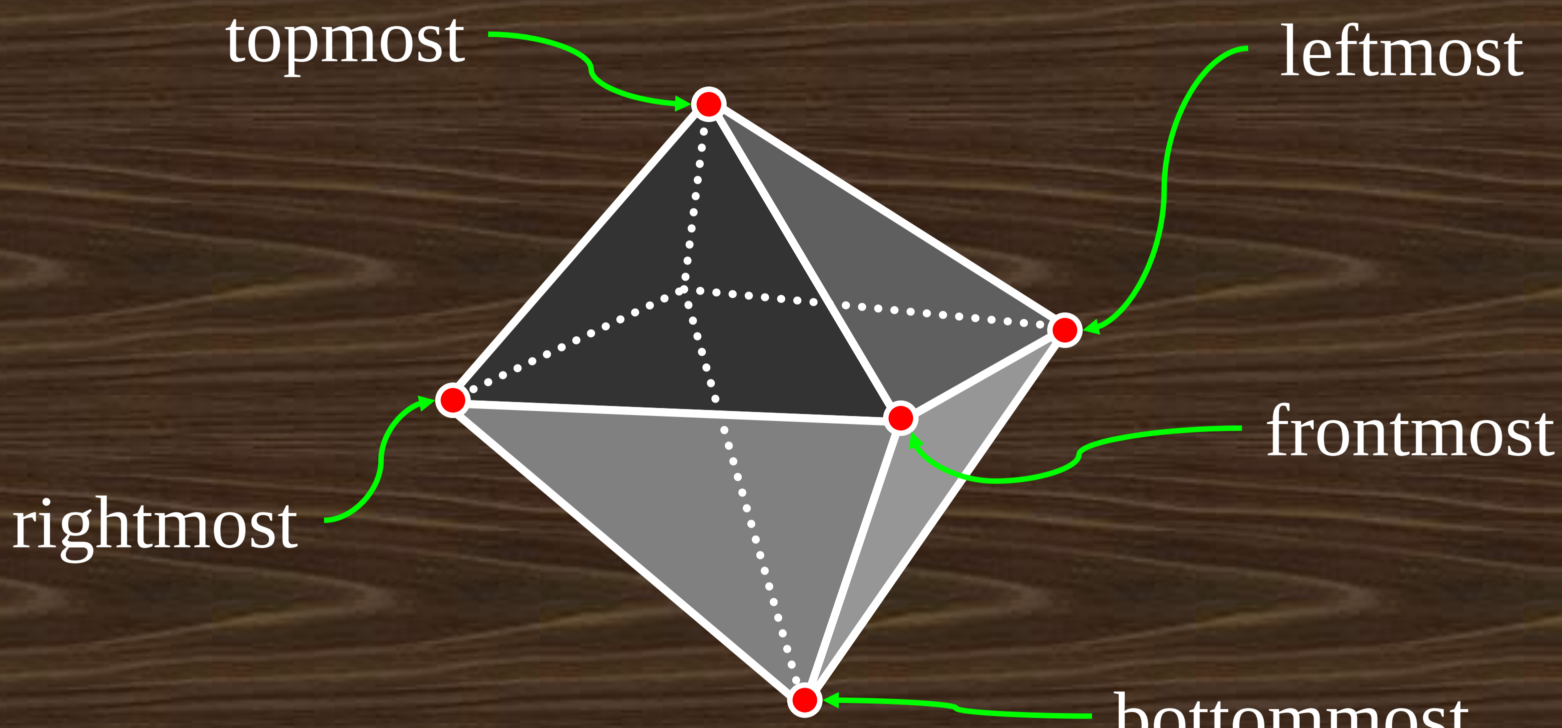
$H \leftarrow H \cup$ {triangle formed by e and p}

3D Gift Wrapping

- Worst-case time complexity: $O(n^2)$
- Output-sensitive time: $O(nF)$
  - where F is the number of faces on the hull

3D QuickHull

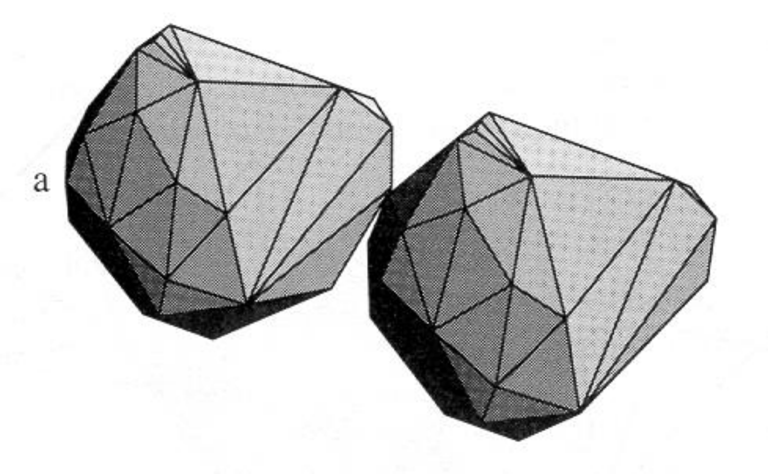
- Similar to the 2D algorithm, but begins with a 8-face polytope



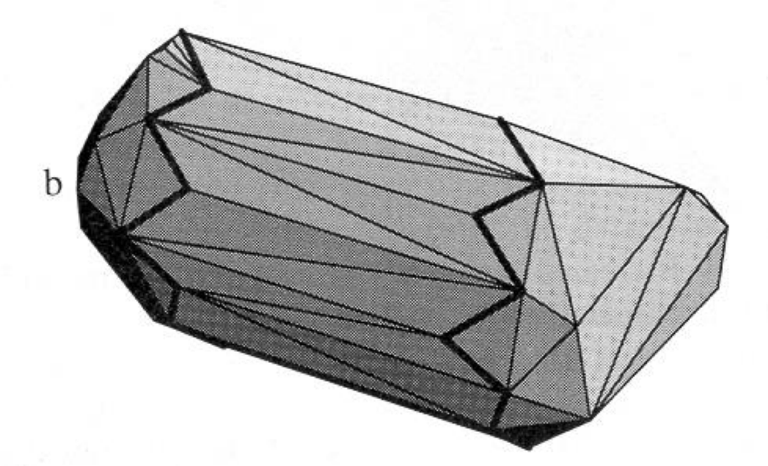
3D QuickHull

- **Worst-case time: O(n^2)**
  - **when "partitions" are very unbalanced**
- **Best- and average-case time: O($n \lg n$)**

3D Divide & Conquer



before



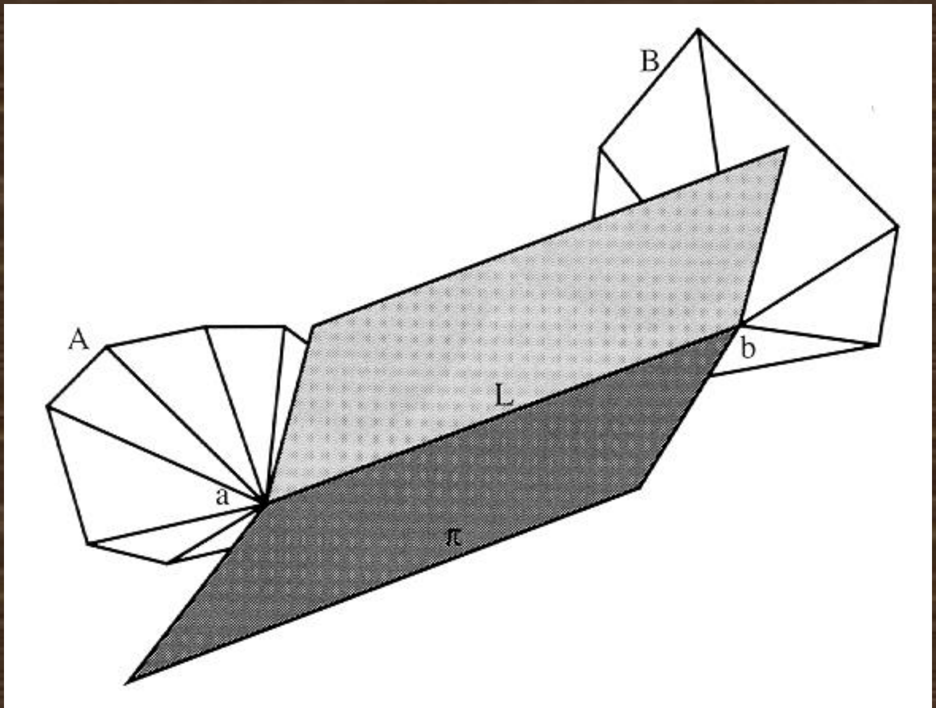
after

3D Divide & Conquer

- Same paradigm as 2D algorithm
  - Sort the points from left to right
  - Let A be the leftmost $\lceil n/2 \rceil$ points
  - Let B be the rightmost $\lfloor n/2 \rfloor$ points
  - Compute convex hulls $H(A)$ and $H(B)$
  - Compute $H(A \cup B)$ by merging $H(A)$ and $H(B)$
- Need to merge in $O(n)$ time in order to achieve overall $O(n \lg n)$ time

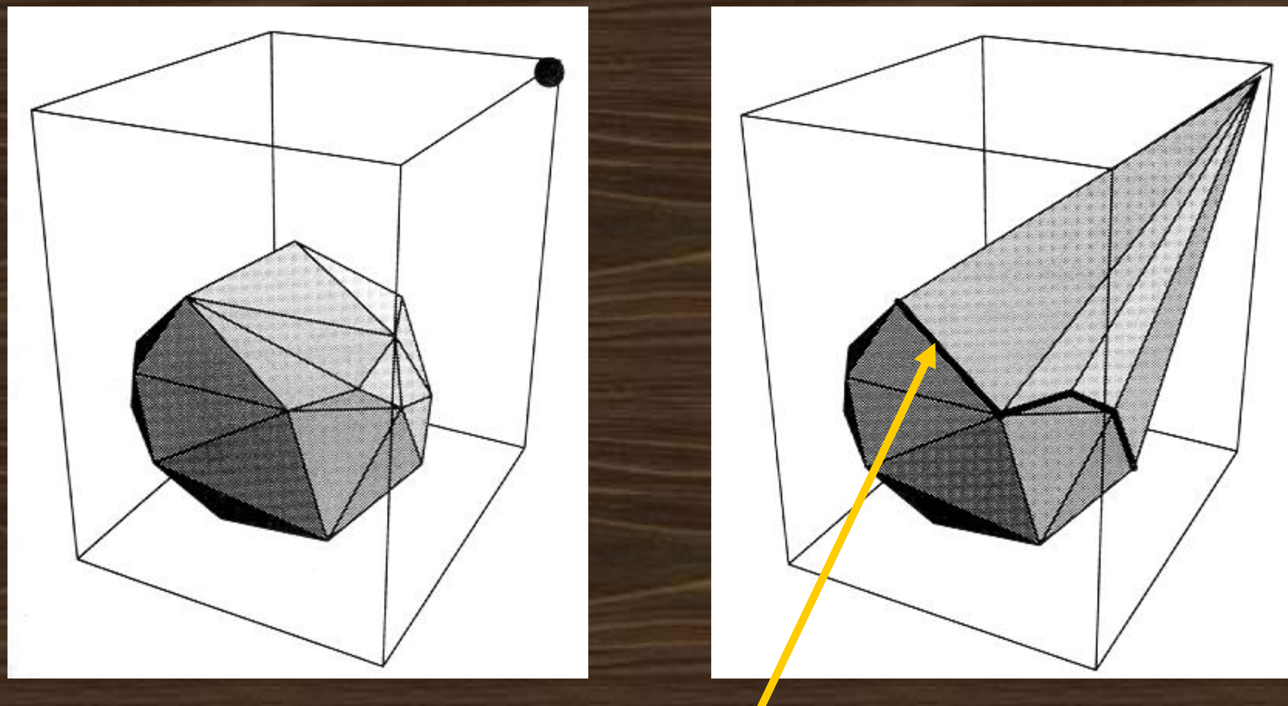
3D Divide & Conquer

- **Merging (basic idea)**
  - **find faces joining *H*(*A*) and *H*(*B*)**
  - **discard hidden faces**



find joining faces

3D Incremental Algorithm



horizon

- **Animated demo**
  - **http://imagery.mit.edu/imagery3/6.838/S98/students/bglazer/javaHull/hull.html**

3D Incremental Algorithm

Initialize H_4 to tetrahedron (p_0, p_1, p_2, p_3)

for $i \leftarrow 4$ to n-1 do

  for each face f of H_{i-1} do

    compute volume of tetrahedron formed by f and p_i

    mark f visible iff volume < 0

  if no faces are visible then

    discard p_i (it is inside H_{i-1})

  else

    for each horizon edge e of H_{i-1} do

      construct cone face determined by e and p

    for each visible face f do delete f

    update H_i

3D Incremental Algorithm

- Requires $O(n^2)$ time
- Can be improved to randomized expected time $O(n \lg n)$

Convex Hulls in Higher Dimensions

- Unfortunately, convex hull in d dimensions can have $\Omega(n^{\lfloor d/2 \rfloor})$ facets (proved by Klee, 1980)
  - therefore, 4D hull can have quadratic size
- No $O(n \lg n)$ algorithm possible for $d > 3$
- These approaches can extend to d > 3
  - gift wrapping
  - QuickHull
  - divide & conquer
  - incremental

References

- Main reference
  - "Computational Geometry in C" by Joseph O'Rourke, 1994
- Other references
  - our textbook "Computational Geometry - An Introduction" by Preparata and Shamos, 1985
  - "Computational Geometry: Algorithms and Applications" by M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, 1997.
  - "Introduction to Algorithms" by Cormen, Leiserson and Rivest, 1990