Convex Hull: Ordering the Points.

Elena Khramtcova

Algorithms and Data Structures.
Faculty of Informatics, USI
Spring 2013
Definition of convex hull (CH)

$P$ – set of $n$ points in the plane,
Definition of convex hull (CH)

\( P \) – set of \( n \) points in the plane,

**Problem(CH):** Compute \( \text{CH}(P) \).

\( \text{CH}(P) \): smallest convex set containing \( P \).
Definition of convex hull (CH)

$P$ – set of $n$ points in the plane,

Problem(CH): Compute $\text{CH}(P)$.

$\text{CH}(P)$: smallest convex set containing $P$. 

[Diagram of a convex hull enclosing a set of points]
Definition of convex hull (CH)

$P$ – set of $n$ points in the plane,

**Problem(CH):** Compute $\text{CH}(P)$.

$h$ – # of hull points
Sorting can be reduced to CH

$A$ – array of $n$ numbers,

**Problem**: Compute $B = \text{sorted } A$. 
Sorting can be reduced to CH

$A$ – array of $n$ numbers,

**Problem:** Compute $B = \text{sorted } A$. 
Sorting can be reduced to CH

\( A \) – array of \( n \) numbers,

**Problem:** Compute \( B = \text{sorted } A \).
Sorting can be reduced to CH

$A$ – array of $n$ numbers,

**Problem:** Compute $B = \text{sorted } A$. 

---

$p_1$ $p_2$ $p_3$ $p_4$ $p_5$ $p_6$ $p_7$ $p_8$ $p_9$
Sorting can be reduced to CH

$A$ – array of $n$ numbers,

**Problem:** Compute $B = \text{sorted } A$. 
CH algorithm 1: Jarvis’s march/Jarvis’s wrap

1. Find the lowest point $p_1$
2. Do
   Find $p_{next}$: min. angle with supporting line
   While $p_{next} \neq p_1$
Time complexity: $O(nh)$
CH algorithm 1: Jarvis’s march/Jarvis’s wrap

1. Find the lowest point $p_1$
CH algorithm 1: Jarvis’s march/Jarvis’s wrap

1. Find the lowest point \( p_1 \)
2. Do
   
   Find \( p_{next} \): min. angle with supporting line

   While \( p_{next} \neq p_1 \)

\( p_1 \)
CH algorithm 1: Jarvis’s march/Jarvis’s wrap

1. Find the lowest point $p_1$
2. Do
   Find $p_{\text{next}}$: min. angle with supporting line
   While $p_{\text{next}} \neq p_1$
CH algorithm 1: Jarvis’s march/Jarvis’s wrap

1. Find the lowest point $p_1$
2. Do
   Find $p_{next}$: min. angle with supporting line
   While $p_{next} \neq p_1$
CH algorithm 1: Jarvis’s march/Jarvis’s wrap

1. Find the lowest point $p_1$
2. Do
   
   Find $p_{next}$: min. angle with supporting line

While $p_{next} \neq p_1$
CH algorithm 1: Jarvis’s march/Jarvis’s wrap

1. Find the lowest point $p_1$
2. Do
   
   Find $p_{\text{next}}$: min. angle with supporting line

   While $p_{\text{next}} \neq p_1$
CH algorithm 1: Jarvis’s march/Jarvis’s wrap

1. Find the lowest point \( p_1 \)
2. Do
   Find \( p_{next} \): min. angle with supporting line
   While \( p_{next} \neq p_1 \)
CH algorithm 1: Jarvis’s march/Jarvis’s wrap

1. Find the lowest point $p_1$
2. Do
   Find $p_{next}$: min. angle with supporting line
   While $p_{next} \neq p_1$
CH algorithm 1: Jarvis’s march/Jarvis’s wrap

1. Find the lowest point \( p_1 \)
2. Do
   Find \( p_{next} \): min. angle with supporting line
   While \( p_{next} \neq p_1 \)
CH algorithm 1: Jarvis’s march/Jarvis’s wrap

1. Find the lowest point $p_1$
2. Do
   
   Find $p_{next}$: min. angle with supporting line
   
   While $p_{next} \neq p_1$
CH algorithm 1: Jarvis’s march/Jarvis’s wrap

1. Find the lowest point $p_1$

2. Do
   
   Find $p_{next}$: min.
   angle with
   supporting line
   
   While $p_{next} \neq p_1$

Time complexity: $O(nh)$
CH algorithm 2: ...

1. Divide points by a vertical line in two equal parts
2. Compute $CH(P_{left})$ and $CH(P_{right})$ recursively
3. Find two bridges
4. Delete all edges in-between the bridges

Time complexity: $O(n \log n)$
CH algorithm 2: ...

1. Divide points by a vertical line in two equal parts
CH algorithm 2: ...

1. Divide points by a vertical line in two equal parts
2. Compute $CH(P_{left})$ and $CH(P_{right})$ recursively

Time complexity: $O(n \log n)$
CH algorithm 2: ...

1. Divide points by a vertical line in two equal parts
2. Compute $CH(P_{\text{left}})$ and $CH(P_{\text{right}})$ recursively
3. Find two bridges

Time complexity: $O(n \log n)$
CH algorithm 2: ...

1. Divide points by a vertical line in two equal parts
2. Compute $CH(P_{left})$ and $CH(P_{right})$ recursively
3. Find two bridges

Time complexity: $O(n \log n)$
CH algorithm 2: ...

1. Divide points by a vertical line in two equal parts
2. Compute $\text{CH}(P_{left})$ and $\text{CH}(P_{right})$ recursively
3. Find two bridges
4. Delete all edges in-between the bridges

Time complexity: $O(n \log n)$
CH algorithm 2: Divide and Conquer

1. Divide points by a vertical line in two equal parts
2. Compute \( CH(P_{left}) \) and \( CH(P_{right}) \) recursively
3. Find two bridges
4. Delete all edges in-between the bridges

Time complexity: \( O(n \log n) \)
CH algorithm 3: ...

1. Find points $r$ and $l$
2. Divide the set by $rl$ into $A$ and $B$
3. Return $HULL(A, l, r) \cup HULL(B, r, l)$

If $A = lr$, return $(l, r)$ else

1. Find $z \in A$: farthest from $lr$
2. $R$ – points to the right of $lz$
3. $L$ – points to the left of $zr$
4. Return $\{HULL(L, l, z) \cup \{z\} \cup HULL(R, z, r)\}$

Time complexity: $O(n^2)$ w.c., $O(n \log n)$ avg.
1. Find points $r$ and $l$

2. Divide the set by $rl$ into $A$ and $B$

3. Return $HULL(A, l, r) \cup HULL(B, r, l)$

1. Find $z \in A$: farthest from $lr$

2. $R$ – points to the right of $lz$

3. $L$ – points to the left of $zr$

4. Return $\{HULL(L, l, z) \cup \{z\} \cup HULL(R, z, r)\}$

Time complexity: $O(n^2)$ w.c., $O(n \log n)$ avg.
CH algorithm 3: ...

1. Find points $r$ and $l$
2. Divide the set by $rl$ into $A$ and $B$
3. Return $\text{HULL}(A, l, r) \cup \text{HULL}(B, r, l)$

Time complexity: $O(n^2)$ w.c., $O(n \log n)$ avg.
CH algorithm 3: ...

1. Find points $r$ and $l$
2. Divide the set by $rl$ into $A$ and $B$
3. Return $\text{HULL}(A, l, r) \cup \text{HULL}(B, r, l)$

$\text{HULL}(A, l, r)$
- if $A = lr$, return $(l, r)$
- else
  1. Find $z \in A$: farthest from $lr$
  2. $R$ – points to the right of $lz$
  3. $L$ – points to the left of $zr$
  4. Return $\{\text{HULL}(L, l, z) \cup \{z\} \cup \text{HULL}(R, z, r)\}$
CH algorithm 3: ...

1. Find points $r$ and $l$
2. Divide the set by $rl$ into $A$ and $B$
3. Return $\text{HULL}(A, l, r) \cup \text{HULL}(B, r, l)$

$\text{HULL}(A, l, r)$
if $A = lr$, return $(l, r)$ else

1. Find $z \in A$: farthest from $lr$
2. $R$ – points to the right of $lz$
3. $L$ – points to the left of $zr$
4. Return
   $\{\text{HULL}(L, l, z) \cup \{z\} \cup \text{HULL}(R, z, r)\}$
CH algorithm 3: ...

1. Find points $r$ and $l$
2. Divide the set by $rl$ into $A$ and $B$
3. Return $HULL(A, l, r) \cup HULL(B, r, l)$

$HULL(A, l, r)$
if $A = lr$, return $(l, r)$ else
1. Find $z \in A$: farthest from $lr$
2. R – points to the right of $lz$
3. L – points to the left of $zr$
4. Return
   \[
   \{HULL(L, l, z) \cup \{z\} \cup HULL(R, z, r)\}
   \]
CH algorithm 3: Quickhull

1. Find points \( r \) and \( l \)
2. Divide the set by \( rl \) into \( A \) and \( B \)
3. Return \( \text{HULL}(A, l, r) \cup \text{HULL}(B, r, l) \)

\[
\text{HULL}(A, l, r) \\
\text{if } A = lr, \text{ return } (l, r) \text{ else}
\]

1. Find \( z \in A \): farthest from \( lr \)
2. \( R \) – points to the right of \( lz \)
3. \( L \) – points to the left of \( zr \)
4. Return \[
\{ \text{HULL}(L, l, z) \cup \{z\} \cup \text{HULL}(R, z, r) \}
\]

Time complexity: \( O(n^2) \) w.c., \( O(n \log n) \) avg.
CH algorithm 4: Heaphull. It does exist!

Uses a kinetic heap w.r.t. a certain “up” direction
CH algorithm 4: Heaphull. It does exist!

Uses a kinetic heap w.r.t. a certain “up” direction

Time complexity: $O(n \log^2 n)$
Not covered here

- Graham scan
- Chan’s algorithm
- Randomized incremental construction
- 3- and d-dimension