Convex Hulls

Carola Wenk
Convex Hull Problem

- Given a set of pins on a pinboard and a rubber band around them. How does the rubber band look when it snaps tight?
- The convex hull of a point set is one of the simplest shape approximations for a set of points.
Convexity

- A set $C \subseteq \mathbb{R}^2$ is **convex** if for all two points $p, q \in C$ the line segment $\overline{pq}$ is fully contained in $C$. 

convex | non-convex
Convex Hull

- The convex hull \( CH(P) \) of a point set \( P \subseteq \mathbb{R}^2 \) is the smallest convex set \( C \supseteq P \). In other words \( CH(P) = \bigcap_{C \supseteq P, \text{convex}} C \).
Convex Hull

• Observation: \( \text{CH}(P) \) is the unique convex polygon whose vertices are points of \( P \) and which contains all points of \( P \).

• We represent the convex hull as the sequence of points on the convex hull polygon (the boundary of the convex hull), in counter-clockwise order.
A First Try

**Algorithm** SLOW_CH($P$):

/* CH($P$) = Intersection of all half-planes that are defined by the directed line through ordered pairs of points in $P$ and that have all remaining points of $P$ on their left */

**Input:** Point set $P \subseteq \mathbb{R}^2$

**Output:** A list $L$ of vertices describing the CH($P$) in counter-clockwise order

$E := \emptyset$

for all $(p, q) \in P \times P$ with $p \neq q$ // ordered pair

valid := true

for all $r \in P$, $r \neq p$ and $r \neq q$

if $r$ lies to the right of directed line through $p$ and $q$ // takes constant time

valid := false

if valid then

$E := E \cup \overline{pq}$ // directed edge

Construct from $E$ sorted list $L$ of vertices of CH($P$) in counter-clockwise order

- Runtime: $O(n^3)$, where $n = |P|$  
- How to test that a point lies to the left?
Orientation Test / Halfplane Test

- **positive orientation** (counter-clockwise)
  - $r$ lies to the left of $pq$

- **negative orientation** (clockwise)
  - $r$ lies to the right of $pq$

- **zero orientation**
  - $r$ lies on the line $\overrightarrow{pq}$

- $\text{Orient}(p,q,r) = \text{sign det} \begin{bmatrix} 1 & p_x & p_y \\ 1 & q_x & q_y \\ 1 & r_x & r_y \end{bmatrix}$, where $p = (p_x, p_y)$

- Can be computed in constant time
Convex Hull: Divide & Conquer

- Preprocessing: sort the points by x-coordinate
- Divide the set of points into two sets $A$ and $B$:
  - $A$ contains the left $\lfloor n/2 \rfloor$ points,
  - $B$ contains the right $\lceil n/2 \rceil$ points
- Recursively compute the convex hull of $A$
- Recursively compute the convex hull of $B$
- Merge the two convex hulls
Merging

- Find upper and lower tangent
- With those tangents the convex hull of \( A \cup B \) can be computed from the convex hulls of \( A \) and the convex hull of \( B \) in \( O(n) \) linear time
Finding the lower tangent

\[ a = \text{rightmost point of A} \]
\[ b = \text{leftmost point of B} \]

while \( T=ab \) not lower tangent to both convex hulls of A and B do{

while T not lower tangent to convex hull of A do{

\[ a = a-1 \]

}\}

while T not lower tangent to convex hull of B do{

\[ b = b+1 \]

}\}

check with orientation test
Convex Hull: Runtime

• Preprocessing: sort the points by x-coordinate \( \mathcal{O}(n \log n) \) just once

• Divide the set of points into two sets \( A \) and \( B \):
  • \( A \) contains the left \( \lfloor n/2 \rfloor \) points,
  • \( B \) contains the right \( \lceil n/2 \rceil \) points

• Recursively compute the convex hull of \( A \) \( \mathcal{T}(n/2) \)

• Recursively compute the convex hull of \( B \) \( \mathcal{T}(n/2) \)

• Merge the two convex hulls \( \mathcal{O}(n) \)
Convex Hull: Runtime

- Runtime Recurrence:
  \[ T(n) = 2 \ T(n/2) + cn \]

- Solves to \( T(n) = \Theta(n \log n) \)
Recurrence
(Just like merge sort recurrence)

1. **Divide**: Divide set of points in half.

2. **Conquer**: Recursively compute convex hulls of 2 halves.

3. **Combine**: Linear-time merge.

\[
T(n) = 2T\left(\frac{n}{2}\right) + O(n)
\]
Recurrence (cont’d)

\[ T(n) = \begin{cases} 
\Theta(1) & \text{if } n = 1; \\
2T(n/2) + \Theta(n) & \text{if } n > 1. 
\end{cases} \]

- How do we solve \( T(n) \)? I.e., how do we find out if it is \( O(n) \) or \( O(n^2) \) or …?
Recursion tree

Solve $T(n) = 2T(n/2) + dn$, where $d > 0$ is constant.
Recursion tree

Solve $T(n) = 2T(n/2) + dn$, where $d > 0$ is constant.
Recursion tree

Solve $T(n) = 2T(n/2) + dn$, where $d > 0$ is constant.
Recursion tree

Solve $T(n) = 2T(n/2) + dn$, where $d > 0$ is constant.
**Recursion tree**

Solve $T(n) = 2T(n/2) + dn$, where $d > 0$ is constant.
Recursion tree

Solve $T(n) = 2T(n/2) + dn$, where $d > 0$ is constant.

$h = \log n$

$\Theta(1)$
Recursion tree

Solve $T(n) = 2T(n/2) + dn$, where $d > 0$ is constant.

$h = \log n$

$\Theta(1)$
Recursion tree

Solve $T(n) = 2T(n/2) + dn$, where $d > 0$ is constant.

$h = \log n$

$\Theta(1)$
Recursion tree

Solve $T(n) = 2T(n/2) + dn$, where $d > 0$ is constant.
Recursion tree

Solve $T(n) = 2T(n/2) + dn$, where $d > 0$ is constant.

$h = \log n$

$\Theta(1)$ -----> #leaves = $n$ -----> $\Theta(n)$
Recursion tree

Solve $T(n) = 2T(n/2) + dn$, where $d > 0$ is constant.

$h = \log n$

$\Theta(1)$

#leaves = $n$

Total $\Theta(n \log n)$
The divide-and-conquer design paradigm

1. **Divide** the problem (instance) into subproblems.
   - *a* subproblems, **each** of size $n/b$

2. **Conquer** the subproblems by solving them recursively.

3. **Combine** subproblem solutions.
   
   Runtime is $f(n)$
Master theorem

\[ T(n) = a T(n/b) + f(n) \]

where \( a \geq 1 \), \( b > 1 \), and \( f \) is asymptotically positive.

**Case 1:** \( f(n) = O(n^{\log_b a - \varepsilon}) \)

\[ \Rightarrow T(n) = \Theta(n^{\log_b a}) . \]

**Case 2:** \( f(n) = \Theta(n^{\log_b a \log^k n}) \)

\[ \Rightarrow T(n) = \Theta(n^{\log_b a \log^{k+1} n}) . \]

**Case 3:** \( f(n) = \Omega(n^{\log_b a + \varepsilon}) \) and \( af(n/b) \leq cf(n) \)

\[ \Rightarrow T(n) = \Theta(f(n)) . \]

**Convex hull:** \( a = 2, \ b = 2 \Rightarrow n^{\log_b a} = n \)

\[ \Rightarrow \text{CASE 2 } (k = 0) \Rightarrow T(n) = \Theta(n \log n) . \]