The Convex Hull Prob
(Sorting Prob of CG)

Def \( A \subseteq \mathbb{R}^d \) is convex if closed under convex combinations.

Def \( \text{Convex Closure} (A) = \text{CC}(A) = \text{smallest convex set} \supseteq A \)

2 Defs of Convex Hull

Def 1 \( \text{CH}(A) = \bigcap \text{CC}(A) \)

Def 2 \( \text{CH}(A) = \text{CC}(A) \)

We will use Def 1

A finite set

Thus in 2D \( \text{CH}(A) \) is a simple closed polygon.

(say CCW)
The Convex Hull Prob
(Sorting Prob of CG)

Def \( A \subset \mathbb{R}^d \) is convex if closed under Convex Combinations.

Def \( \text{ConvexClosure}(A) = \text{CC}(A) = \text{smallest convex set} \supseteq A \)

2 Defs of Convex Hull

Def 1 \( \text{CH}(A) = \bigcap \text{CC}(A) \)

Def 2 \( \text{CH}(A) = \text{CC}(A) \)

We will use Def 1

A finite set

Thus in 2D \( \text{CH}(A) \) is a simple closed polygon.

(say CCW)
We will use the following characterization:

Claim: \([a, b]\) is on \(CH(A)\) iff \(a \neq b\)

1) \(a, b \in A\)
2) \(\forall a' \in A\) unless \(a'\) left of \([a, b]\)
   or \(a' \in [a, b]\)

2D Convex Hull by divide-and-conquer

\(A = \{P_1, \ldots, P_n\}\)
\(P_i = (x_i, y_i)\)

Preprocess: sort \(A\) by \(x\)-coordinate

2D-CH(A)
if \(|A| = 1\) return \(P_1\)
else \(CH_L = 2D-CH(P_1, \ldots, P_{n-1})\)
    \(CH_R = 2D-CH(P_{n-1}, \ldots, P_n)\)
    STITCH \((CH_L, CH_R)\)
STITCH \((L, R)\)

1. \(L\) - lower bridge \((L, R)\)
   - \(a = \text{rightmost } L\)
   - \(b = \text{leftmost } LR\)

Repeat \(x)\) \(x\)

\(x)\) While \(a < \text{Right}(a, b)\) set \(a = a\)

\(xx)\) While \(b < \text{Right}(a, b)\) set \(b = b\)

Upper bridge \((L, R) = \) ?

---

Correctness

\(x)\) generates triangles \((a, b, a)\)

\(xx)\) " " \((a, b, b)\)

1) The \(\Delta\)'s are disjoint
   They are ordered by their intersection with vertical line \(L\).
2) They are in \(CC(A)\).

Thus termination:

At termination \(a, \bar{a}, b, \bar{b}\) are all lift of \((a, b)\)
Since \((a, \bar{a}), (b, \bar{b})\) and \((b, \bar{b}), (b, \bar{b})\) are on CH(L) & CH(R) respectively.

Done

Timing: Preprocess \(O(n \log n)\) to sort

STITCH in \(O(n)\)

\[ T(n) = 2T(n/2) + cn \]

\[ T(n) = O(n \log n) \]
Lower bounds

Sorting reducible to CH

Input: \( x_1, \ldots, x_n \)

\( \text{CH}\left( (x_1,x_1^2), \ldots, (x_n,x_n^2) \right) \)

The CH will be \( x_i \)'s in sorted order.

An important use for CH

\( \bar{p}_i \rightarrow \bar{p}_n \in \mathbb{R}^2 \)

\( \bar{p}_i = (p_x, p_y, p_x^2 + p_y^2) \)

\( \text{CH}(\bar{p}_1, \ldots, \bar{p}_n) = \text{Triangulated surface} \)

The Delaunay Triangulation
Quick Sort & Backwards Analysis

Consider
\( QS(M) \) (distinct keys)
1) pick random \( a \in M \)
2) split \( M \) : \( s < a < l \) \( (|M|-1) \) comparisons
3) return \( QS(s) \times a \times QS(l) \)

Goal: Expect \# comparisons

Consider dart game:
Init: empty board

While non-empty square
pick random empty sq

cost = \# empty sqs to left & right of dart.

Claim
Expect cost of dart game = Expect cost QS.
Backwards game:

Init: full board

While I dart remove random dart.
Cost: # empty Ds left & right.

Claim: \( \text{Expect cost} \ DG = \text{Expect cost} \ BW \ DG \)

Analysis backwards game
Assume i darts on board
\( T_i = \text{Expected cost to remove random dart.} \)

Total Cost = \( \sum \) cost of 1 dart

\[
T_i \leq \frac{2(n-i)}{i} \leq \frac{2n}{i}
\]

\[
E(DG) = \sum T_i \\
\leq \sum \frac{2n}{i} = 2n H_n \\
= O(n \log n)
\]
Random Incremental CH

Procedure Random Incremental CH \( (P) \)

1. Make \( \Delta = (P_i, P_j, P_k) \) pick \( C \) in interior \( \Delta \)
2. Construct ray from \( C \) to each \( P_i \)
3. Partition \( P_i \) by edge of \( \Delta \) they cross.
4. Randomly permute \( P_1, \ldots, P_n \).
   
   For \( i = 1 \) to \( n \)
   
   Let \( e \) be edge crossed by ray \( C \rightarrow P_i \)
   
   Build Tent \( (P, e) \)

Procedure Build Tent \( (P, e) \)

1. Find edges of CH "visible" to \( P \) by searching out from \( e \).
2. Replace visible edges with 2 new edges.
3. Assign rays to the new edges.
An Example

$n$-points on a circle

Worst case: Incremental order $p_1 \rightarrow p_n$

"Best" case $p_1, p_2, p_3, p_{1/2}, p_{3/4}, p_{3n/4} \ldots$
Correctness?

Timing

$O(n)$ work other than BuildTest.

Consider steps 1 & 2 in BuildTest

1) At most an edge generated over life of algo.
2) Charging rule for line-side tests
   a) Not visible tests: we charge $P_i$
       each visible test: we charge to the edge

   total $2n + 2n = 4n$ tests.

Consider step 3 in BuildTest.

Ray-costs

Backwards analysis

2-3 points to pick from say $P_i$

$\text{Cost}(P_i) = \begin{cases} 
0 & \text{if } P_i \text{ not on hull} \\
\# \text{ray crossing to left & right} & \text{otherwise}
\end{cases}$
$C_i = \text{cost}$

$E(C_i) \leq \frac{2(n-i)}{i-3}$

$C = \text{total cost}$

$E(C) = \sum_{i=4}^{n} E(C_i) \leq \sum_{i=4}^{n} \frac{2(n-i)}{i-3} \leq 2n \sum_{i=1}^{n} \frac{1}{i}$