"ULTIMATE PLANAR C.H. ALGORITHM?"

KIRKPATRICK-SEIDEL

It's a divide & conquer algorithm

Upper hull only
"ULTIMATE PLANAR C.H. ALGORITHM?"

KIRKPATRICK-SEIDEL

It's a divide & conquer algorithm

divide-conquer-merge

Upper hull only
"ULTIMATE PLANAR C.H. ALGORITHM?"
KIRKPATRICK-SEIDEL

It's a divide & conquer algorithm

divide-conquer-merge
divide-merge-conquer!

Upper hull only
"ULTIMATE PLANAR C.H. ALGORITHM?"

KIRKPATRICK-SEIDEL

It's a divide & conquer algorithm

divide-merge-conquer

Upper hull only

---

Diagram showing points and lines partitioning the space.
Goal: Get $O(n \log h)$

- We can’t sort anything.
- How do we split? Median $O(n)$
- We can also afford $O(n)$ to merge, i.e., to find a bridge.
Finding a Bridge in Linear Time

Let the bridge have slope $k^*$.

Suppose we guess slope $k$.

Sweep $k$.

Guess $k < k^*$

$\rightarrow$ sweep stops on blue

Guess $k > k^*$

$\rightarrow$ sweep stops on red

Guess $k = k^*$

$\rightarrow$ confirm bridge

$O(n)$ time to guess & verify
- Arbitrarily pair up points
- Arbitrarily pair up points
- Find median slope
- Guess $K = \text{median}$
Case 1: \( k > k^* \)

Half of the pairs have slope \( k' > k \), so \( k' > k^* \).
Case 1: $k > k^*$

Half of the pairs have slope $k' > k$, so $k' > k^*$.

$k^*$ can't sweep below $b$.

$\alpha$ can't be on bridge (it could be on C.H.)
Case 2

$K < K^*$

Half of the pairs have slope $K' \leq K$, so $K' < K^*$

$K^*$ can't sweep below $a$, and $b$ can't be on bridge (it could be on C.H.)

$a_x < b_x$
Case 1: $K > K^*$

Half of the pairs have slope $K' > K$, so $K' > K^*$

$\square$ can't be on bridge (it could be on C.H.)

Case 2: $K < K^*$

Half of the pairs have slope $K' < K$, so $K' < K^*$

$\bigcirc$ can't be on bridge (it could be on C.H.)

THROW AWAY ONE POINT (a or b) FROM HALF THE PAIRS

$\frac{1}{4}$ points
If we guess wrong: THROW AWAY ONE POINT (a or b) FROM HALF THE PAIRS

Then arbitrarily pair remaining points & "guess" again

Time: $c \cdot n$ for first wrong guess
$c \cdot \frac{3n}{4}$ for second " "
$c \cdot \frac{3}{4} \cdot \frac{3n}{4}$ for third.

etc

total: $O(n)$
"Prune & Search"

If you can throw out a constant fraction of your input whenever you fail, then you will still have a good algorithm.

\[ T(n) = F(n) + T\left(\frac{n}{c}\right) \quad [c > 1] \]

- \( O(\log n) \) \( O(1) \) : Binary search \([c=2]\)
- \( O(n) \) \( O(n) \) : Finding a bridge \([c=\frac{4}{3}]\)
- \( O(n^k) \) \( O(n^k) \) \( n^k + \frac{n^k}{2^k} + \frac{n^k}{4^k} + \ldots + \frac{n^k}{2^{i_k}} \) \([c=2]\)
- \( O(2^n) \) \( O(2^n) \) \( 2^n + 2^{n-1} + 2^{n-2} + \ldots + 2 \) \([c=2]\)

search leaves, if "fail" then search parents etc
Example of linear-time bridge finding

- Unknown bridge
- Upper hull only
- Median separator
- $X_{	ext{min}}$
- $X_{	ext{max}}$
Randomly pair points
Find median slope
Test slope:
- too steep
- only left side is extremal
Because slope is too steep:
Discard left endpoints of steeper pairs
Subset: \[ \leq \frac{3}{4} \text{ original} \]
Random pairs
Discard left endpoints of steeper pairs.
New random pairs and median slope
Too steep yet again
Discard...
Fourth attempt on $\leq \frac{3}{4} \cdot \frac{3}{4} \cdot \frac{3}{4}$ original
Discard right endpoints of shallower pairs
Find median slope.
Extreme finds one point on each side.

DONE
We know how to find a bridge in linear time.

Might as well throw out potential non-C.H. pts inside... it's "free"
We know how to find a bridge in linear time.

Might as well throw out potential non-C.H. pts inside ... it's "free"

Of course we might not throw anything out.
We know how to find a bridge in linear time.

Solve 2 smaller problems with \( n/2 \) half points each.

That still only gives us \( O(n \log n) \).

Do we have to find a bridge that “splits” the hull evenly?

If we at least find one new bridge on both sides then we get \( O(\log h) \) depth.

If we don’t find a bridge on one side, we must have thrown out \( n/2 \) pts.
Cost tree

Example

\[ \frac{c \cdot n}{2} \rightarrow \text{first bridge} \]

\[ \frac{c \cdot n}{4} \]

\[ \times \]

\[ \frac{c \cdot n}{4} \rightarrow \text{"only" 3 bridges} \]

Tree must have exactly \( h \) nodes

Actually a good thing
Cost tree:

- First bridge:
  - $c \cdot n$

- 2 more bridges:
  - $c \cdot n^\frac{1}{2}$
  - $c \cdot n^\frac{1}{2}$
  - $c \cdot n^\frac{1}{4}$

- "Only" 3 bridges:
  - $c \cdot n^\frac{1}{4}$

Balanced case:

- Depth $O(\log h)$
- Work $O(n \log h)$

$h$ can be $O(n)$
Cost tree

- First bridge
- 2 more bridges
- "Only" 3 bridges
- Exactly \( h \) nodes

Unbalanced case

- \( O(\log n) \) depth
- \( O(n) \) work

\[ c \cdot n \]

\[ c \cdot n^{\frac{1}{2}} \]

\[ c \cdot n^{\frac{1}{4}} \]

\[ c \cdot n^{\frac{1}{8}} \]

If you keep getting "unbalanced" hull edges, you will run out of points quickly, i.e., it can't keep happening! In this case, \( h \) cannot be \( O(n) \)!

(Unlike, say, quick-hull)
Cost tree

- $c \cdot n$
  - $c \cdot \frac{n}{2}$
    - $c \cdot \frac{n}{4}$
      - $c \cdot \frac{n}{4}$
        - $c \cdot \frac{n}{4}$
        - $c \cdot \frac{n}{4}$
  - $c \cdot \frac{n}{2}$
    - $c \cdot \frac{n}{4}$
      - $c \cdot \frac{n}{4}$

→ first bridge

→ 2 more bridges

→ "only" 3 bridges

exactly $h$ nodes

\[ \text{Swap nodes: } A < c \cdot \frac{n}{4} \]

(\& recursively the same)

~ANALYSIS?

for any tree
Every node ascends only. Weight per level: less than full tree case.
We get a full tree: depth $\log h$ \[O(n \cdot \log h)\]

See web notes for analysis (which isn't hard)