Optimal In-Place Algorithms for 3-d Convex Hulls & 2-d Segment Intersection

Timothy Chan Eric Chen
School of CS
U of Waterloo

History Recap: 3-d Convex Hull Alg'ms

- $O(n \log n)$ time, $O(n)$ space

  [Preparata,Hong'77] by divide&conquer
  [Clarkson,Shor'88] by rand. incremental
  [Fortune'86] by sweep (for 2-d Voronoi diagrams)

History Recap: 2-d Line Segment Intersection Alg'ms

- $O((n + K) \log n)$ time, $O(n + K)$ (or $O(n)$) space
  [Bentley,Ottmann'79] (+ [Brown'81]) by sweep

- $O(n \log n + K)$ time, $O(n + K)$ space
  [Chazelle,Edelsbrunner'88] by a "complicated" sweep

- $O(n \log n + K)$ time, $O(n)$ space
  [Clarkson,Shor'88] by rand. incremental/sampling
  [Mulmuley'88] " " "
  [Balaban'95] by divide&conquer

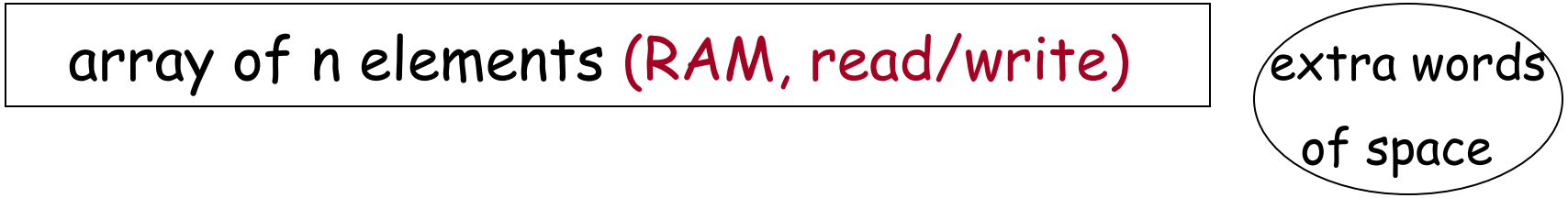
Sublinear space ??

CONSTANT space !!

In-Place Alg'ms

- Example: heapsort

- The model



(+ write-only output stream)

Known In-Place Results/Techniques

- In-place merging
  - $O(n)$ time, $O(1)$ space ['80s]
- In-place stable partitioning
  - $O(n)$ time, $O(1)$ space ['80s]
- In-place sorting with $O(n)$ moves
  - $O(n \log n)$ time, $O(1)$ space [Franceschini,Geffert'03]

Known In-Place Results/Techniques

- In-place/"implicit" data structures for searching
  - $O(\log^2 n)$ query/update time/space [Munro'84]
  - $O(\log^2 n / \log\log n)$ " " [Franceschini,Grossi,Munro,Pagli'02]
  - $O(\log n \log\log n)$ " " [Franceschini,Grossi'03]
  - $O(\log n)$ " ", $O(1)$ space [Franceschini,Grossi'03]

- Recent renaissance
  - In-place radix-sort ['07], in-place suffix sorting ['09], …

Known In-Place CG Alg'ms

- 2-d convex hull
  - $O(n \log h)$ time, $O(1)$ space
    [Brönnimann,Iacono,Katajainen,Morin,Morrison,Toussaint'02]
  - Simple polygonal chains: $O(n)$ time, $O(1)$ space
    [Brönnimann,Chan'04]
- 2-d maxima layers
  - $O(n \log n)$ time, $O(1)$ space [Blünck,Vahrenhold'06]
- 2-d red/blue closest pair
  - $O(n \log n)$ time, $O(1)$ space
    [Bose,Maheshwari,Morin,Morrison,Smid,Vahrenhold'04] by simple divide&conquer

...

Known In-Place CG Alg'ms (Cont'd)

- 2-d orthogonal segment intersection
  - $O(n \log n + K)$ time, $O(1)$ space
    [Bose,Maheshwari,Morin,Morrison,Smid,Vahrenhold'04] by simple divide&conquer

- 2-d segment intersection
  - $O((n + K) \log^2 n)$ time, $O(\log^2 n)$ space
    [Chen,Chan,CCCG'03] by modifying Bentley-Ottmann
  - $O(n \log^2 n + K)$ time, $O(1)$ space [Vahrenhold,WADS'05] by modifying Balaban

Known In-Place CG Alg'ms (Cont'd)

- 3-d convex hull
  - $O(n \log^3 n)$ time, $O(1)$ space
    [Brönnimann,Chan,Chen,SoCG'04] by clever divide&conquer

- 2-d nearest neighbor search
  - $O(\log^2 n)$ time, $O(1)$ space [Brönnimann,Chan,Chen,SoCG'04]
  - $O(\log^{1.7096} n)$ time, $O(1)$ space [Chan,Chen,SODA'08]

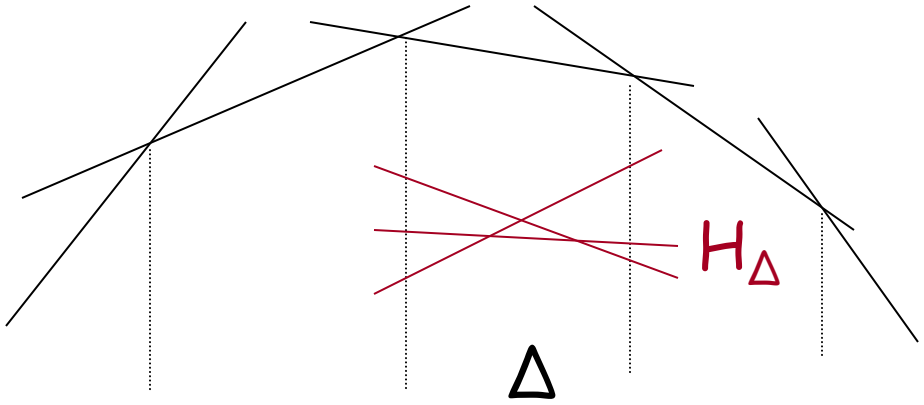
New Results

- 3-d convex hulls
  - $O(n \log n)$ time (rand.), $O(1)$ space
- 2-d segment intersection
  - $O(n \log n + K)$ time (rand.), $O(1)$ space

OPTIMAL !!

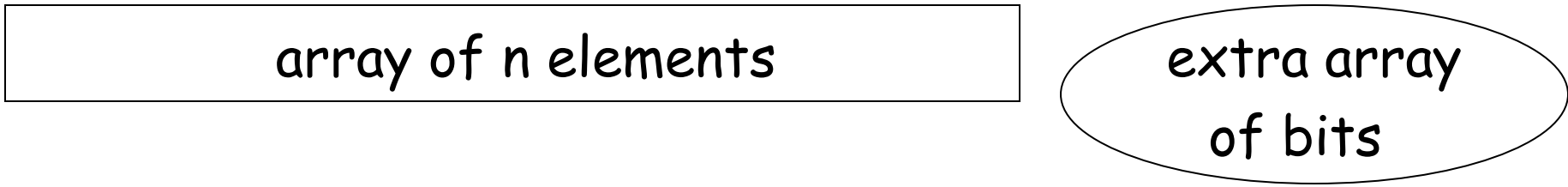
3-d Convex Hulls: Preliminaries

- Dual problem: Given n planes H in 3-d, output the vertices of the lower envelope (LE) of H
- Basic rand. divide&conquer approach:
  - Take sample R of size r
  - For each cell Δ of "canonical triangulation" of LE of R:
    - Compute "conflict list" H_Δ = all planes intersecting Δ
    - Recursively compute LE of H_Δ inside Δ



$\Rightarrow$ O(r) subproblems of size ~ O(n/r) (by Clarkson,Shor)

An Intermediate Model: "Permutation+Bits"



- Allow possibly large ($O(n \text{ polylog } n)$) # of extra bits
- But each bit access costs $O(1)$ time

- Note: pointers can be stored in the array of bits, but each pointer op would cost $O(\log n)$

- Ex: binary search in an arbitrary list now costs $O(\log^2 n)$

Permutation+Bits Implies In-Place

- Reduction 1: S(n) bits of space $\Rightarrow$ ~ S(n/log n) bits

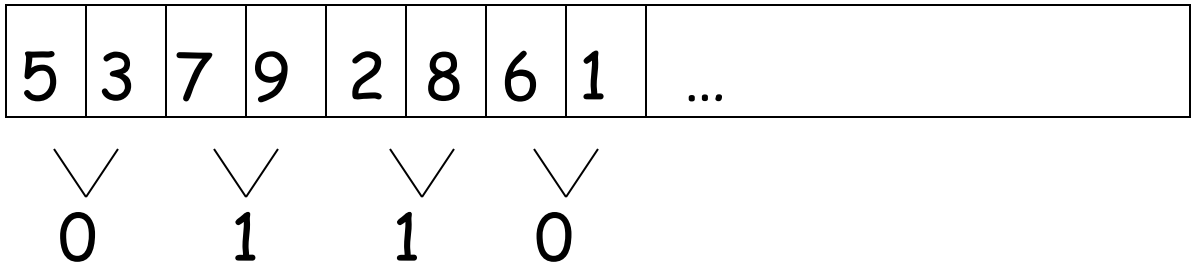
  Pf: Take sample of small size r ~ log n

  Solve each subproblem one by one

  Conflict lists computable in O(n log n) time

- Reduction 2: εn bits of space $\Rightarrow$ in-place

  Pf: By "bit-encoding trick" (permuting pairs)



CH Alg'ms in the Permutation+Bits Model

- Standard divide&conquer alg'm:

  $T(n) = 2T(n/2) + O(n)$ in standard model

  $T(n) = 2T(n/2) + O(n \log n)$ in permutation+bits model

  $\Rightarrow \quad T(n) = \boxed{O(n \log^2 n)}$

- New idea: can't reduce overhead $O(n \log n)$, but try to divide into larger # of subproblems in $O(n \log n)$ time…

Our CH Alg'm in Permutation+Bits Model

- Take sample of large size r
- For each plane h, can determine the conflict lists that h participates in, by point location

$\Rightarrow$ $T(n) = O(r)\, T(n/r) + O($time for n pt location queries in 2-d subdivision of size $O(r))$

- Standard point location methods:

  $O(\log r)$ query time in standard model

  $O(\log^2 r)$ query time in permutation+bits model

  TOO MUCH !!

Our Method for Point Location

- Modify a known pt location method?
  - Lipton,Tarjan's separator method
  - Kirkpatrick's hierarchical method
  - Preparata's trapezoid method
  - Edelsbrunner,Guibas,Stolfi's chain method
  - Sarnak,Tarjan's persistent search trees
  - Mulmuley's rand. incremental method
  - …

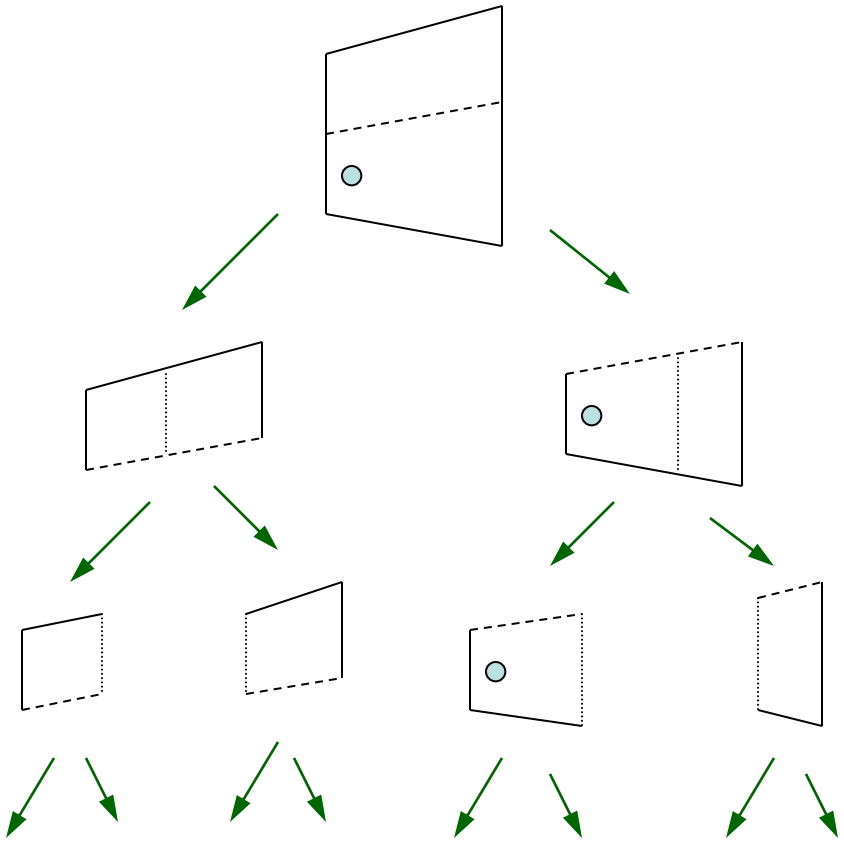
Our Method for Point Location

- Modify a known pt location method?

  - Lipton,Tarjan's separator method
  - Kirkpatrick's hierarchical method
  - Preparata's trapezoid method ⇐ BINGO !!
  - Edelsbrunner,Guibas,Stolfi's chain method
  - Sarnak,Tarjan's persistent search trees
  - Mulmuley's rand. incremental method
  - …

Our Method for Point Location

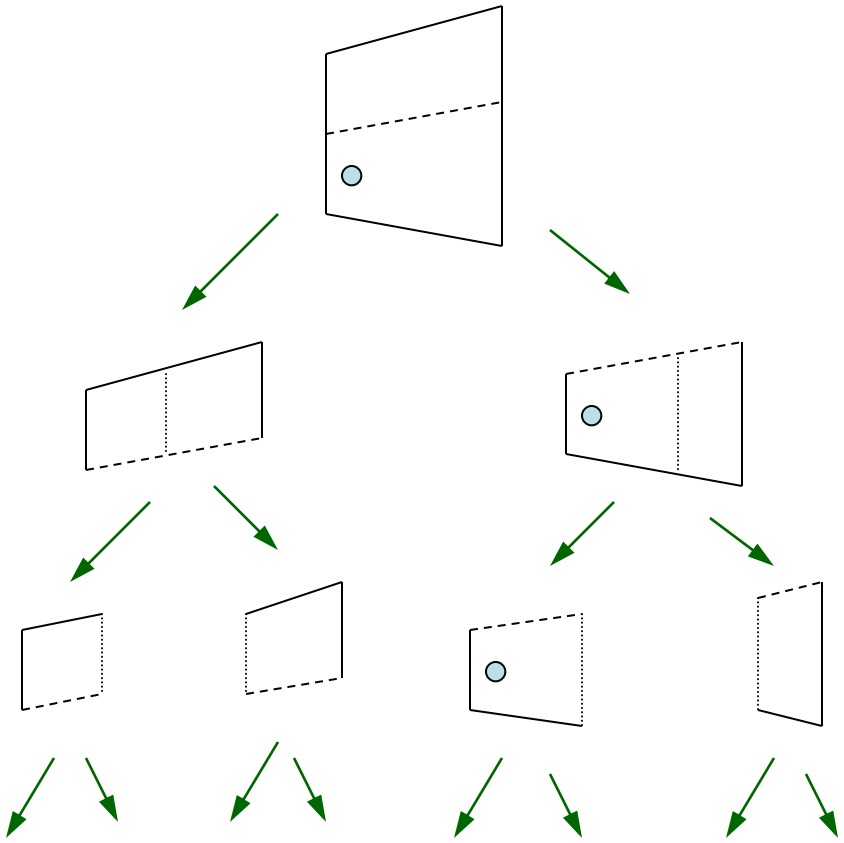
- Modify Preparata's trapezoid method:



tree of
height $O(\log r)$
size $O(r \log r)$

Our Method for OFFLINE Point Location

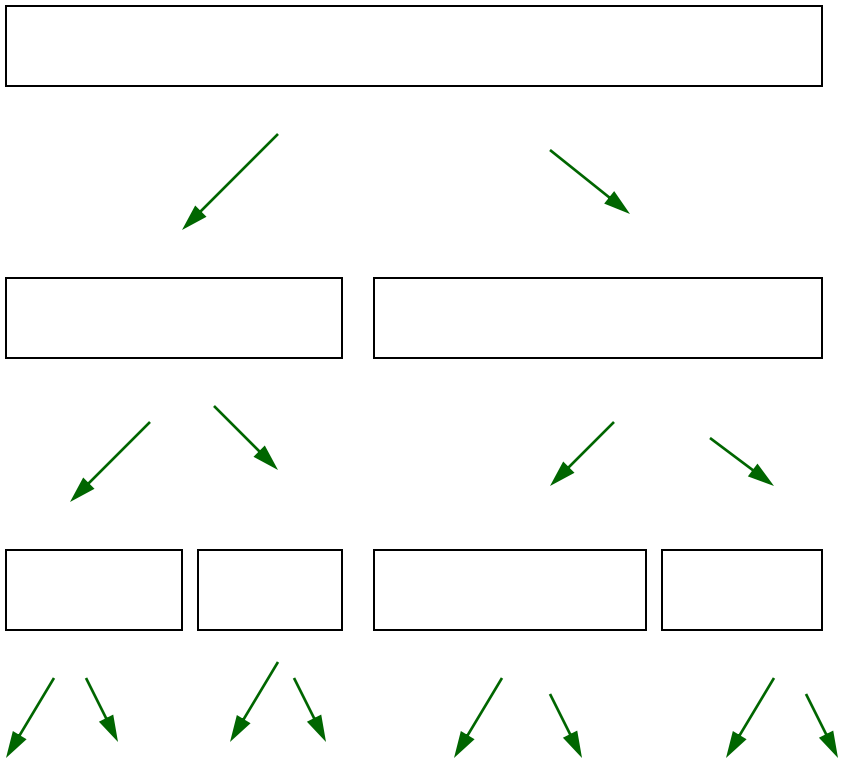
- Modify Preparata's trapezoid method:



tree of
height $O(\log r)$
size $O(r \log r)$

Our Method for OFFLINE Point Location

- Modify Preparata's trapezoid method:



cost of partitioning
= $O(n \log r)$

pointer ops = $O(r \log r)$

cost of pointer ops
= $O(r \log^2 r)$

(note: similarity to quicksort)

Final Analysis

$\Rightarrow$ $T(n) = O(r)\, T(n/r) + O(n \log r + r \log^2 r)$

- Choose $r = n/\log n$

$\Rightarrow$ $T(n) = O(n/\log n)\, T(\log n) + O(n \log n)$

$\Rightarrow$ $T(n) = \boxed{O(n \log n)}$

Remark:
A "New" CH Alg'm in Standard Model

- Take sample of size $r = n^{1-\varepsilon}$

$\Rightarrow$ $T(n) = cn^{1-\varepsilon}\, T(n^{\varepsilon}) + O(\text{time for n pt location queries in 2-d subdivision of size } n^{1-\varepsilon})$

$\Rightarrow$ $T(n) = cn^{1-\varepsilon}\, T(n^{\varepsilon}) + O(n \log n)$

$\Rightarrow$ $T(n) = \boxed{O(n \log n)}$ by choosing $\varepsilon < 1/c$

- Note: compare with other $O(n \log n)$ rand. 3-d CH alg'ms (e.g. [Clarkson,Shor'88], ... [Amenta,Choi,Rote,SoCG'03])

Remark:
A "New" CH Alg'm in Standard Model

- Take sample of size $r = n^{1-\varepsilon}$

$\Rightarrow$ $T(n) = cn^{1-\varepsilon}\, T(n^{\varepsilon}) + O($time for n pt location queries in 2-d subdivision of size $n^{1-\varepsilon})$

$\Rightarrow$ $T(n) = cn^{1-\varepsilon}\, T(n^{\varepsilon}) + O(n \log n)$

$\Rightarrow$ $T(n) = \boxed{O(n \log n)}$ by choosing $\varepsilon < 1/c$

- Note: leads to new optimal rand. cache-oblivious alg'ms for 3-d CH (simplifies [Kumar,Ramos'02]) & 2-d segment intersection (extends [Arge,Molhave,Zeh,ESA'08])

Some Open Problems

- Optimal deterministic in-place alg'ms for 3-d CH or 2-d segment intersection?
- In-place and cache-oblivious?
- In-place 2-d EMST?
- In-place nearest neighbor search in $O(\log n)$ time?
- General point location in $o(\log^2 n)$ time in permutation+bits model??