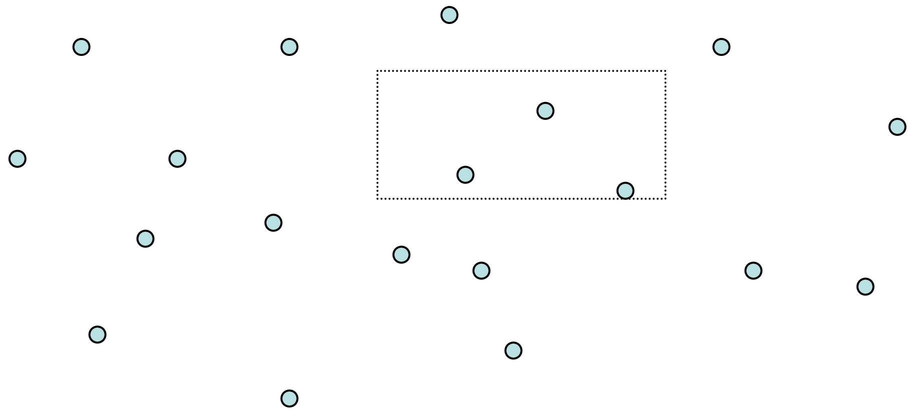
Orthogonal Range Queries: Basic Methods

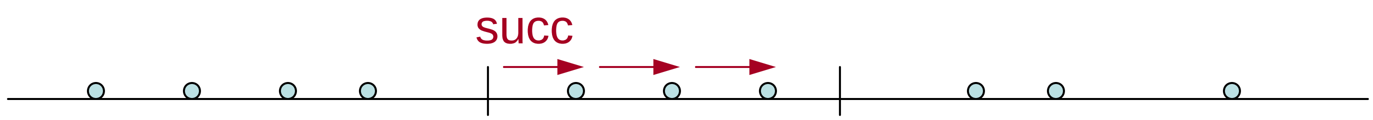
Timothy Chan

MADALGO Summer School'10
(Mon. Morning I)

Orthogonal Range Searching



- 1D:

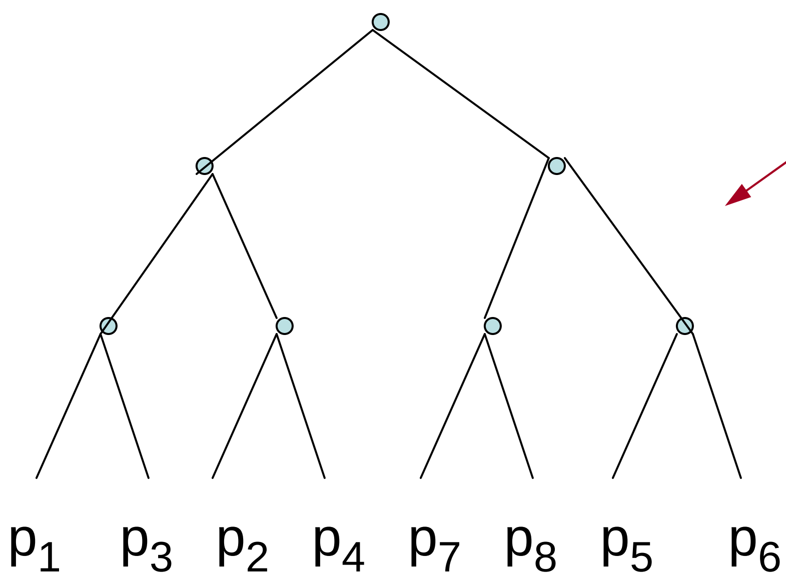


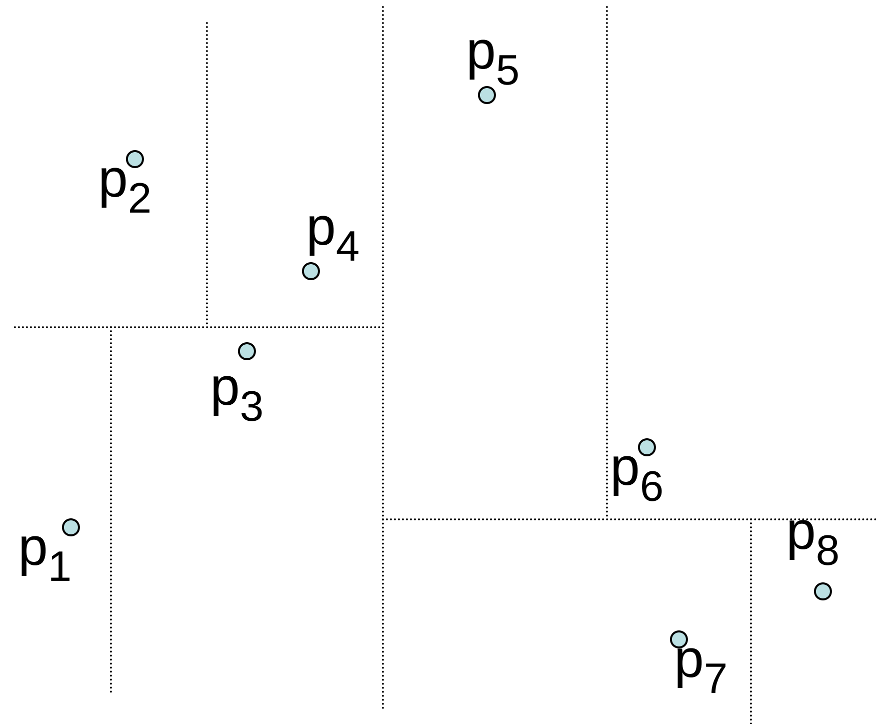
P(n) = O(n log n),   S(n) = O(n),   Q(n) = O(log n [+ k])

- 2D??

Method 0: k-d Tree

- Divide by median-x,
- Then by median-y,
- Then by median-x, Etc.





$$S(n) = \boxed{O(n)}$$

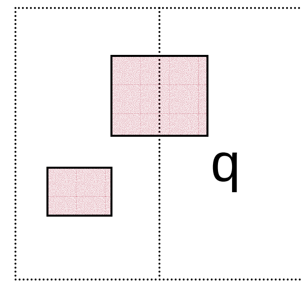
$$P(n) = 2\ P(n/2) + O(n)$$

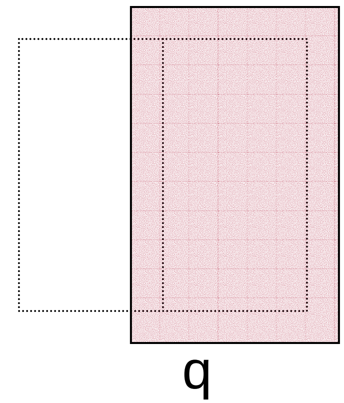
$$\Rightarrow \boxed{O(n \log n)}$$

- $Q(n) = O(\#\text{ cells crossing } \partial q)$

  $= O(4 \cdot \#\text{ cells crossing a line})$

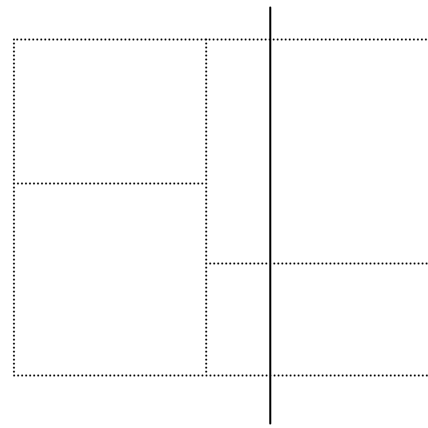
  Z(n)

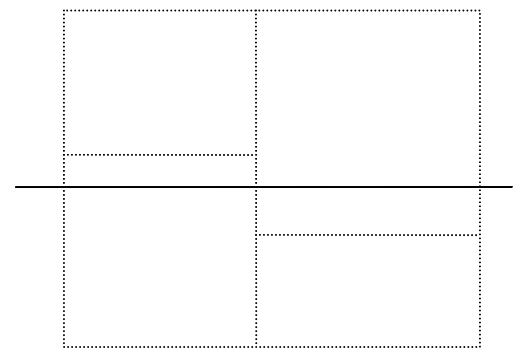




- $Z(n) = 2\, Z(n/4) + O(1)$

  $\Rightarrow \boxed{O(n^{1/2})}$





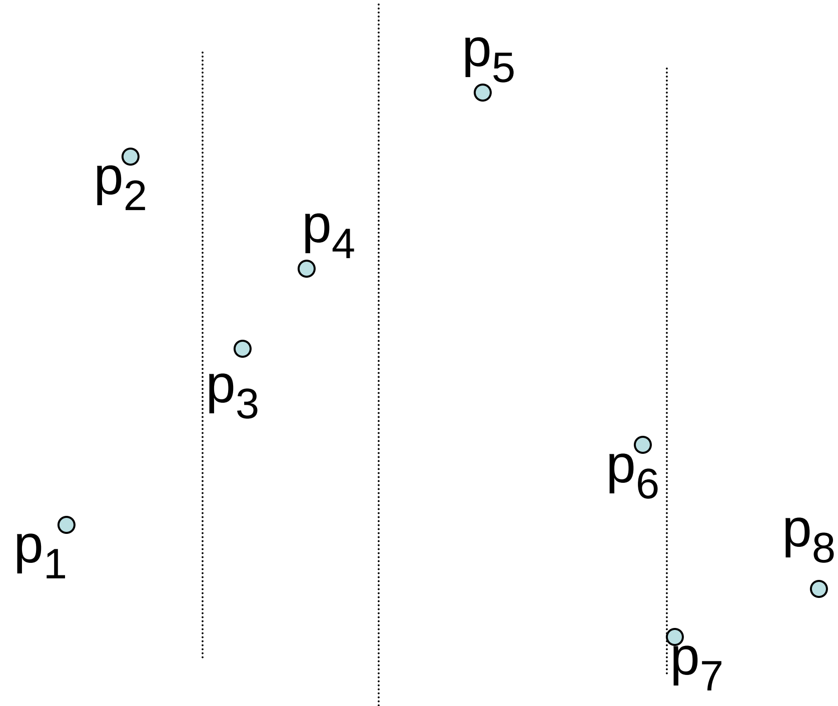
- Rmk: not good!

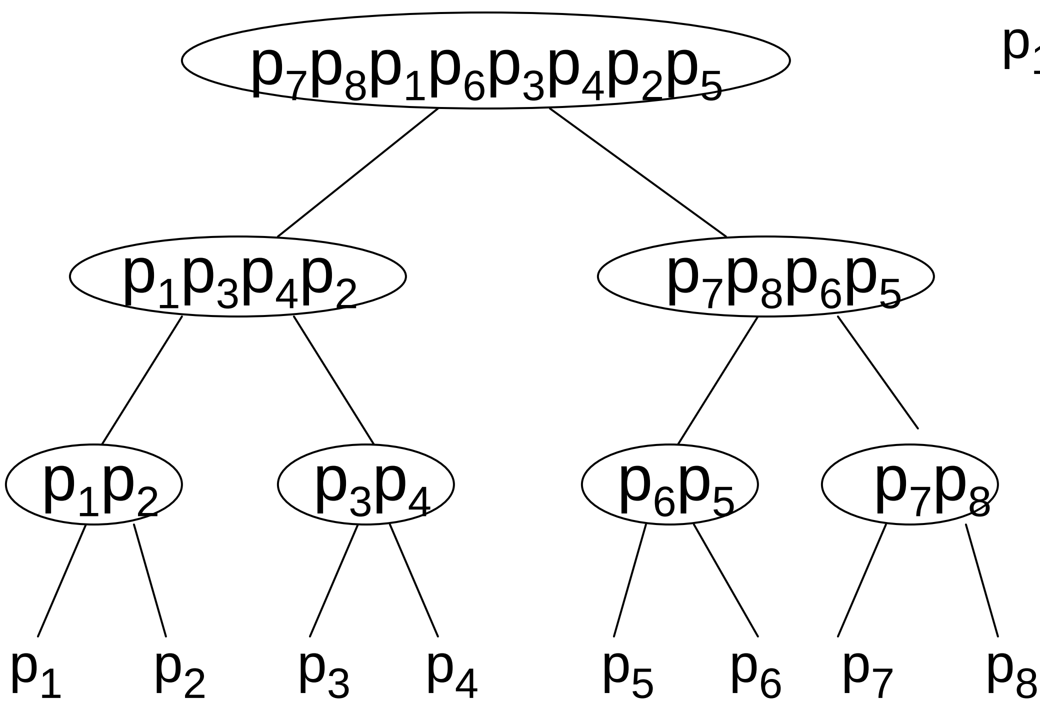
  & worse in higher-D:

  $Z(n) = 2^{d-1}\, Z(n/2^d) + O(1) \;\; \Rightarrow \;\; O(n^{1-1/d})$

Method 1: Range Tree

- Store points in sorted y-order
- Divide by median-x only
- Recurse on left & right





$$S(n) = 2S(n/2) + O(n)$$

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

by pre-sorting

$$P(n) = 2P(n/2) + O(n \cancel{\log} n)$$

$$\Rightarrow \boxed{O(n \log^{\cancel{2}} n)}$$

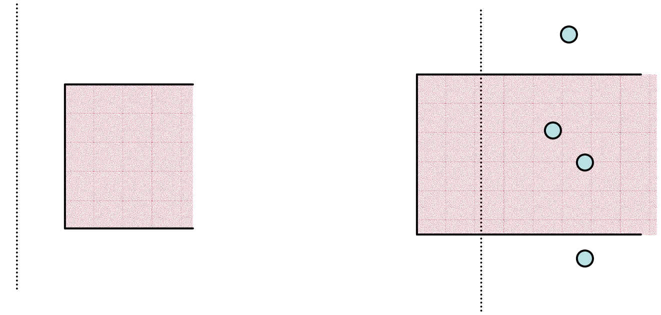
- 3-sided query:

$Q_{3side}(n) = Q_{3side}(n/2) + O(1)$

or $Q_{3side}(n/2) + O(\log n)$

search in y-sorted list

$\Rightarrow Q_{3side}(n) = O(\log^2 n)$



- General query:

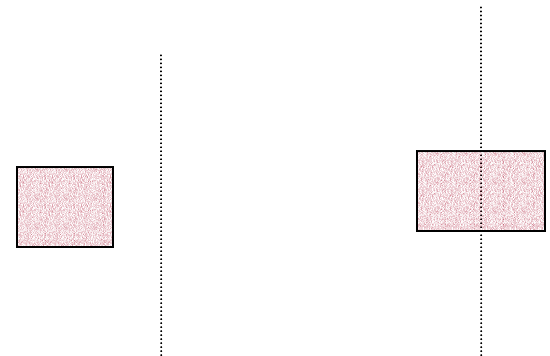
$Q(n) = Q(n/2) + O(1)$

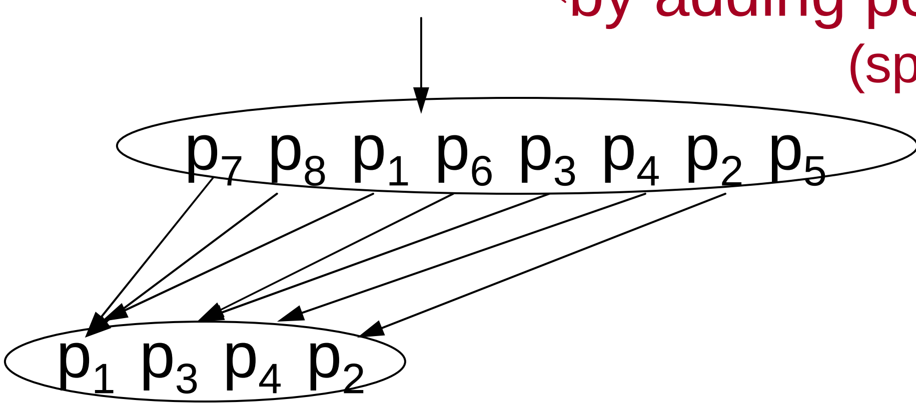
or $\cancel{2\, Q_{3side}(n/2) + O(1)}$ $O(\log^2 n)$

$\Rightarrow Q(n) = \boxed{O(\log^{\cancel{2}} n)}$

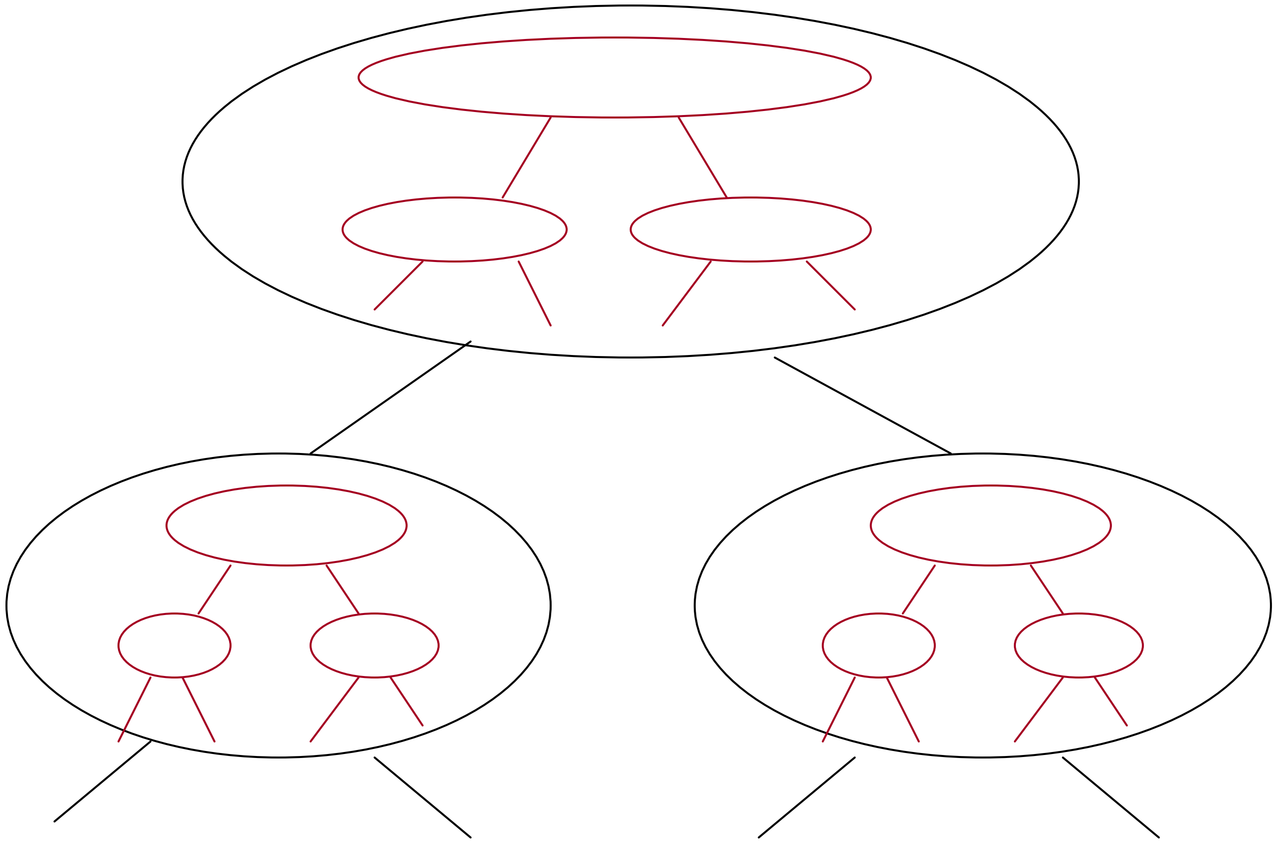
by adding pointers between layers

(special case of “fractional cascading”)





Higher-D



- $S_d(n) = 2S_d(n/2) + S_{d-1}(n)$

  $\Rightarrow$ $S_d(n) = O(S_{d-1}(n) \log n)$ $\Rightarrow$ $S_d(n) = \boxed{O(n \log^{d-1} n)}$
- $Q_d(n) = O(Q_{d-1}(n) \log n)$ $\Rightarrow$ $Q_d(n) = \boxed{O(\log^{d-1} n)}$

- Rmks:
  - Example of “multi-level” data structure
  - Dynamic: query/update time ~~$O(\log^d n)$~~ by standard balanced tree techniques $O(\log^{d-1} n \log\log n)$ by dynamic fractional cascading
  - (Mild) trade-off: by degree-b range tree

    $S(n) = O(n (\log_b n)^{d-1})$, $Q(n) = O((b \log_b n)^{d-1})$

    or $S(n) = O(n (b \log_b n)^{d-1})$, $Q(n) = O(\log n (\log_b n)^{d-2})$

Recap:

- Orthogonal range searching:
  $O(n \log^{d-1} n)$ space, $O(\log^{d-1} n)$ time

Next:

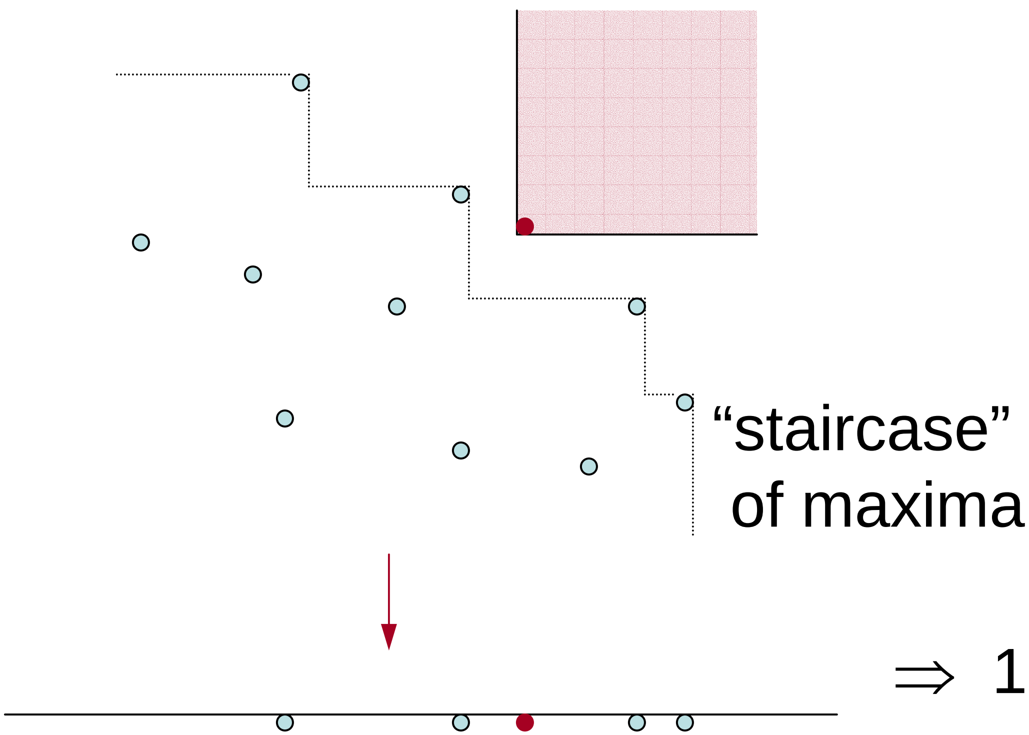
- Improvements, by starting with better base cases
  (1D, 2D, 3D)??

Orthogonal Range Queries:
3D Reporting

(Mon. Morning II)

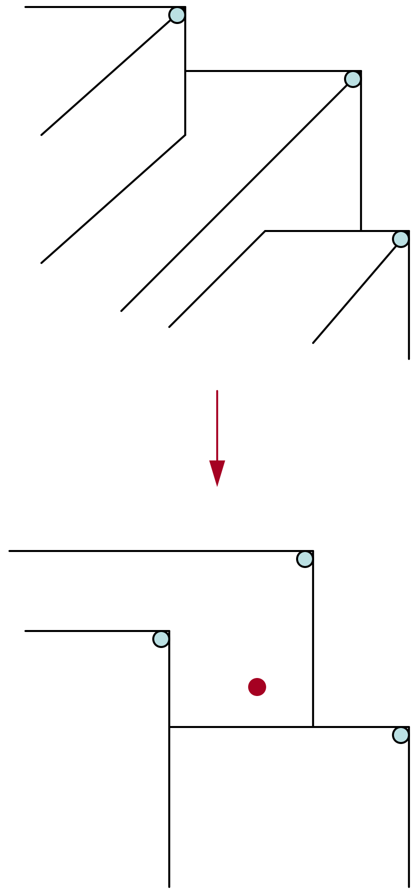
- Goal: 3D reporting
  O(n polylog n) space, O(loglog U + k) time?

- Warm-up: 2D dominance emptiness



$\Rightarrow$ 1D predecessor search

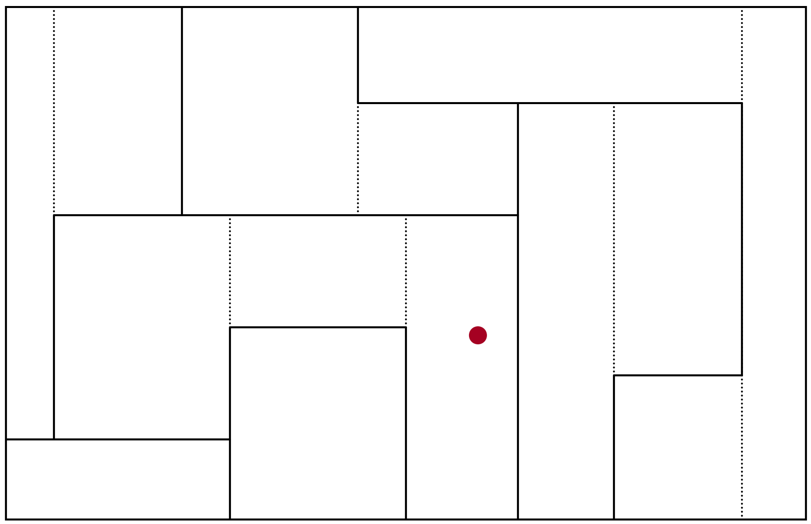
- 3D dominance emptiness



“staircase” of maxima

$\Rightarrow$ 2D orthogonal point location

Orthogonal 2D Point Location



Store n disjoint rectangles in 2D
s.t. can locate rectangle
containing query pt

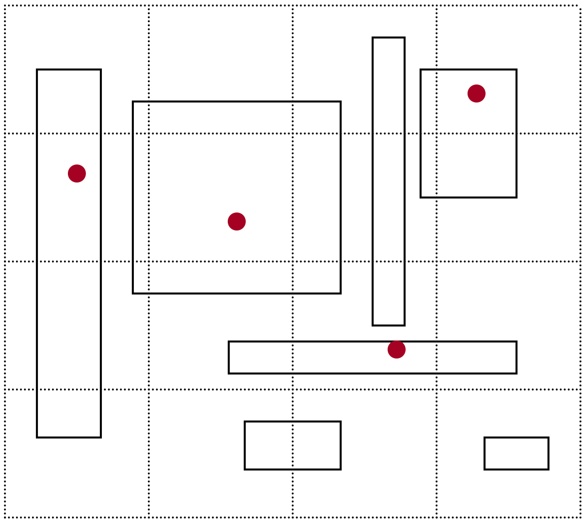
[non-orthogonal pt location:
wait till Wed. (John)…]

- Previous methods: (Dietz'89/de Berg-van Kreveld-Snoeyink'95)
  O(n) space, O((loglog U)^2) time

- Brand new method: (Chan'11…)
  O(loglog U) time!

Recursive Method: 1st Attempt

- Assume universe is W x H (initially W = H = U)
- Idea: use $n^{1/2}$ x $n^{1/2}$ grid, like Alstrup-Brodal-Rauhe
  (column width $W/n^{1/2}$, row height $H/n^{1/2}$)
- For each column/row,
  recurse on all rectangles that have
  a vertex inside the column/row
  (each rectangle stored ≤ 4 times)
- For each grid cell, remember in table
  if it is covered by a rectangle, or
  if a horizontal or vertical edge cuts thru it



- $Q(n,W,H) = O(1) + \max\{ Q(n_i, W/n^{1/2}, H), Q(n_j, W, H/n^{1/2}) \}$

locate grid cell

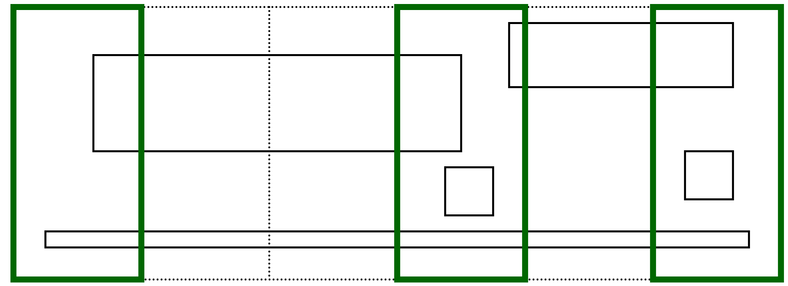
1 recursive call only!

- $Q(n,W,H) = O(1) + \max\{ Q(n_i, W/n^{1/2}, H), Q(n_j, W, H/n^{1/2}) \}$

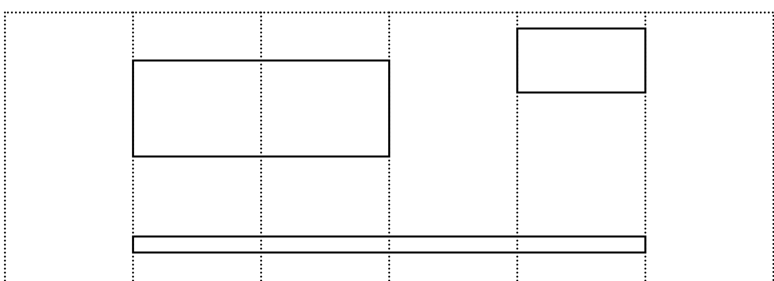
- Rmks:
  - as n gets smaller relative to W,H, recursion doesn't shrink W,H as much…
  - could apply rank space reduction to equalize n & W,H

    but cost extra loglog factor due to van Emde Boas!

Recursive Method: 2nd Attempt

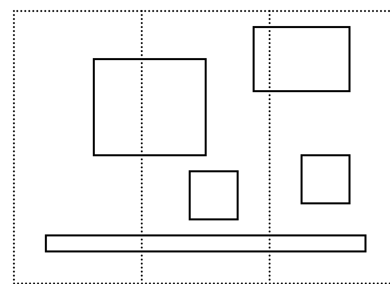


- Idea: imitate van Emde Boas
  divide into $W^{1/2}$ columns
  (of width $W^{1/2}$)
- Build hash table for D = all "nonempty" columns
- Recurse w. rounded input + Recurse on universe with empty columns removed



(universe $W^{1/2}$ x H)

(each rectangle stored ≤ 2 times)



(universe $nW^{1/2}$ x H)

- $Q(n,W,H) = O(1) + Q(n, nW^{1/2}, H)$

- $Q(n,W,H) = O(1) + Q(n, nW^{1/2}, H)$
- Or $Q(n,W,H) = O(1) + Q(n, n, nH^{1/2})$
- Rmk: but this recursion can't shrink W,H to const!

Summary So Far…

- Method 1:

  $Q(n,W,H) = O(1) + \max\{ Q(n_i, W/n^{1/2}, H), Q(n_j, W, H/n^{1/2}) \}$

- Method 2a:

  $Q(n,W,H) = O(1) + Q(n, nW^{1/2}, H)$

- Method 2b:

  $Q(n,W,H) = O(1) + Q(n, n, nH^{1/2})$

- Final Method: just combine!!
  - If $n \geq W^{1/3}$ & $n \geq H^{1/3}$ then Method 1
  - If $n < W^{1/3}$ then Method 2a; if $n < H^{1/3}$ then Method 2b

$Q(n,W,H) = O(1) + \max\{ Q(n_i, W^{5/6}, H), Q(n_j, W, H^{5/6}) \}$

$$Q(n,W,H) = O(1) + \max\{ Q(n_i, W^{5/6}, H), Q(n_j, W, H^{5/6}) \}$$

$$\Rightarrow O(\log\log W + \log\log H) = \boxed{O(\log\log U)}$$

- $S(n) = O(4^{O(\log\log U)}\, n) = \boxed{O(n \operatorname{polylog} U)}$

- Rmk: can reduce space to O(n) by more work…

In Conclusion…

- 3D dominance emptiness:

  O(n polylog n) space, O(loglog U) time

- 3D dominance reporting:

  similarly, w. additional ideas (Tues. afternoon),

  O(n polylog n) space, O(loglog U + k) time

$\Rightarrow$ 3D general reporting:

by adding sides w. 3 extra log factors in space,

$\boxed{\text{O(n polylog n)}}$ space, $\boxed{\text{O(loglog U + k)}}$ time

[can save space by Alstrup-Brodal-Rauhe idea (Karpinski-Nekrich’10)…]

- Higher-D reporting:

  by range trees w. d-3 extra log factors in space & time,
  O(n polylog n) space, $O(\log^{d-3} n \log\log n + k)$ time

  by degree-b range trees w. $b=\log^{\varepsilon} n$,
  $\boxed{O(n \text{ polylog } n)}$ space, $\boxed{O((\log n/\log\log n)^{d-3} \log\log n + k)}$ time

  current record query time!

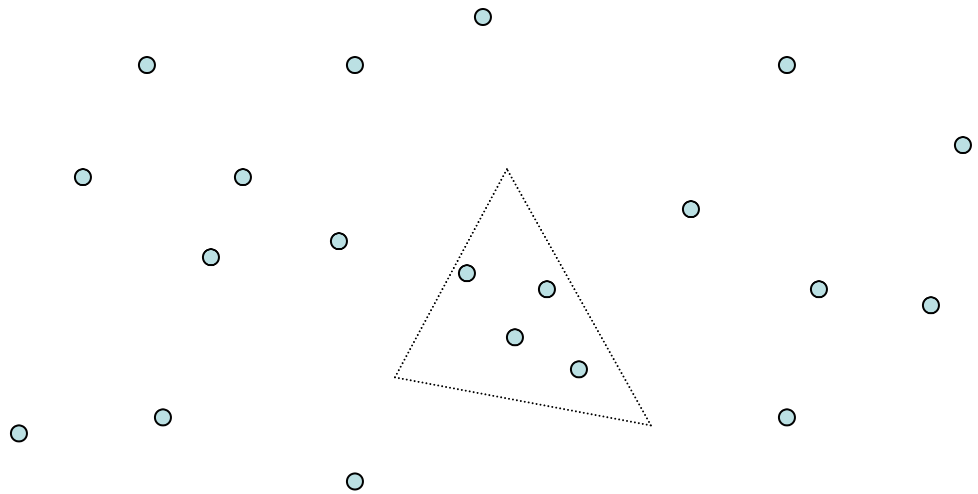
End of Orthogonal Range
Query Upper Bounds

[why not O(loglog U) in 4D?? wait for Tues. (Mihai)…]

Non-Orthogonal Range Queries

(Mon. Afternoon II +
Tues. Afternoon I + II)

Simplex Range Searching



- An illustrative case: 2D halfplane counting

History in 2D

	P(n)	S(n)	Q(n)
• Willard'82		n	$n^{0.793}$
		n	$n^{0.774}$
• Edelsbrunner-Welzl'86		n	$n^{0.695}$
• Haussler-Welzl'87		n	$n^{0.667}$ (rand.)
• Welzl'88		n	$n^{1/2} \log n$
• Chazelle-Sharir-Welzl'92	$n^{1+\varepsilon}$	$n^{1+\varepsilon}$	$n^{1/2+\varepsilon}$
• Matoušek'92	n log n	n	$n^{1/2}$ polylog n
• Matoušek'93	$n^{1+\varepsilon}$	n	$n^{1/2}$
• Chan'10	n log n	n	$n^{1/2}$ (rand.)

$n^{1-1/d}$

(near opt.)

	P(n)	S(n)	Q(n)
• Clarkson'87		$n^{2+\varepsilon}$	log n (rand.)
• Chazelle'93/Matoušek'93		n^2	$\log^3 n$

n^d $\log^{d+1} n$

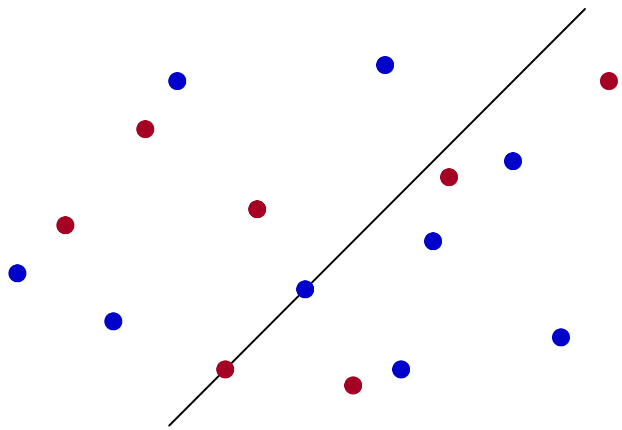
• Trade-off		m	$(n / m^{1/d}) \log^{d+1} n$
			(near opt.)

Method 0 (Willard'82)

"ham" "bread"

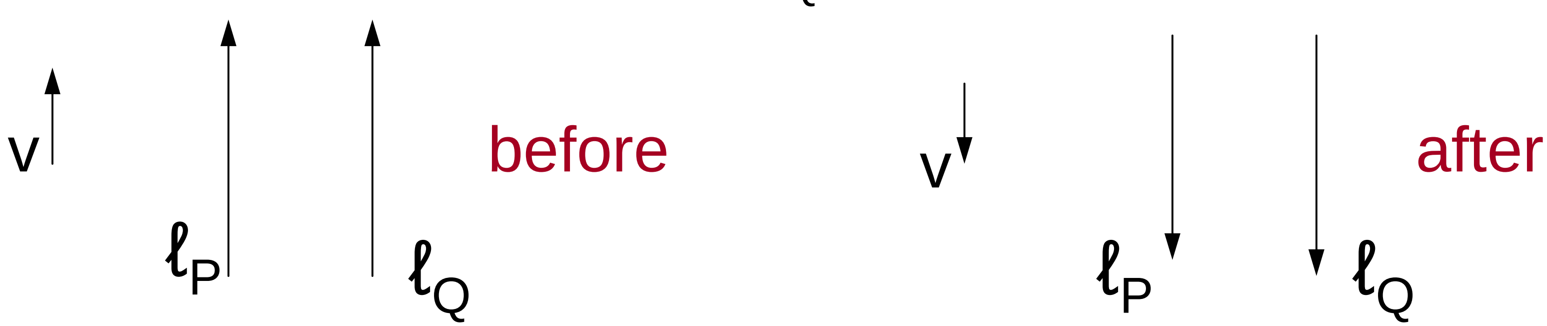
- "Ham-Sandwich Cut" Thm: Given 2 point sets P & Q in 2D,

  ∃ line that simultaneously bisects P & Q



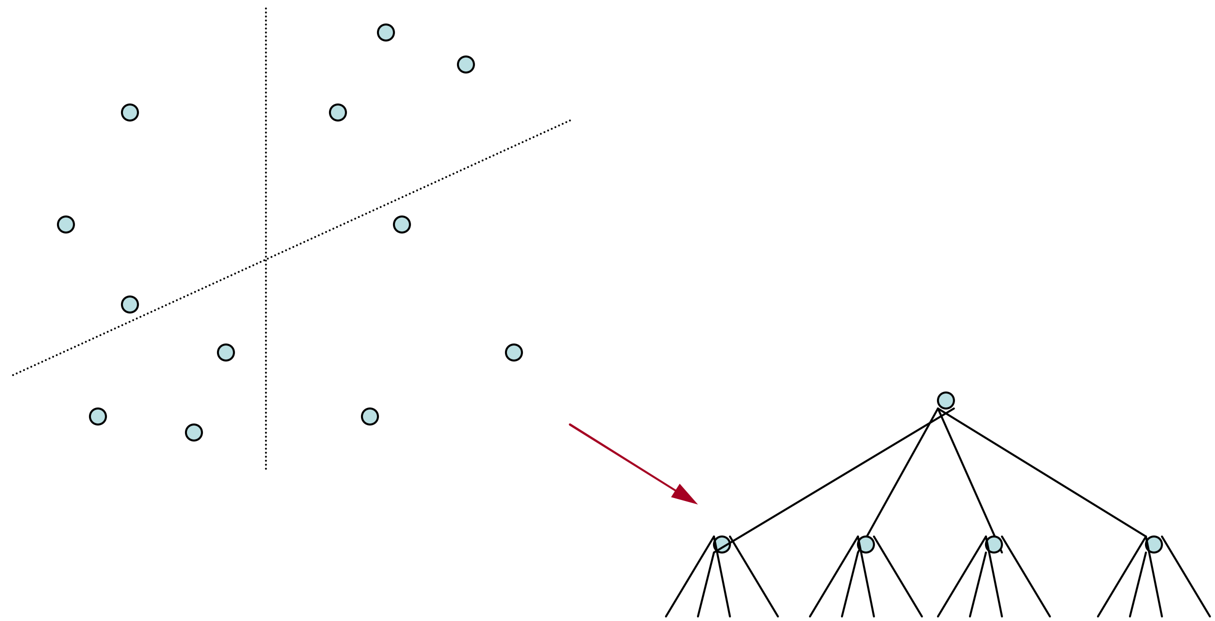
- Pf Sketch: Given dir. v, let ℓ_P = line bisecting P along dir v

  ℓ_Q = line bisecting Q along dir v



- Corollary: Given n points P in 2D,

  $\exists$ 2 lines which partition P into 4 subsets of n/4 points



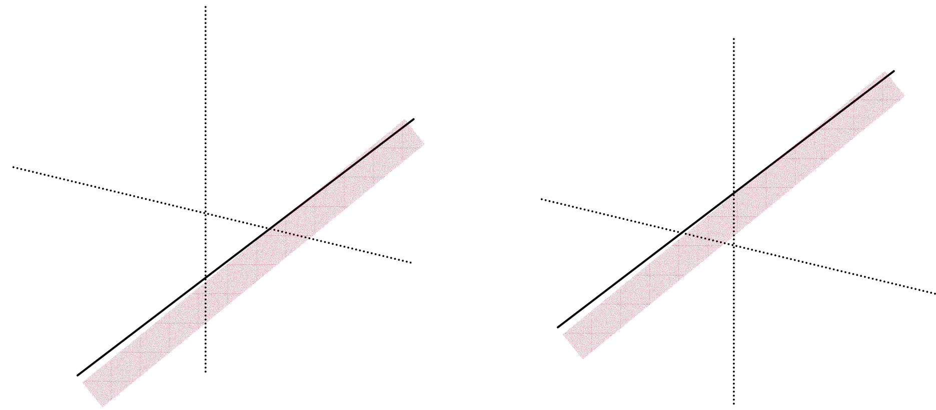
- Recurse $\Rightarrow$ "partition tree"
- S(n) = $\boxed{O(n)}$
- P(n) = 4 P(n/4) + O(n) $\Rightarrow$ $\boxed{O(n \log n)}$

Megiddo'85

- Halfplane query:

  $Q(n) = 3\,Q(n/4) + O(1)$

  $\Rightarrow O(n^{\log_4 3}) \approx O(n^{0.793})$



- Triangle query:

  $Q(n) = O(\#\text{ cells crossing } \partial q)$

  $= O(3 \cdot \#\text{ cells crossing a line})$

  $= \boxed{O(n^{0.793})}$

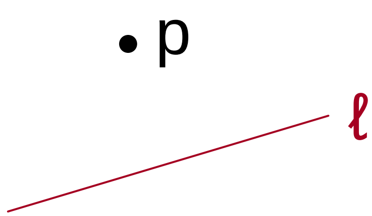
- Rmks: work also in 3D (8-partitioning), but not in 5D, …

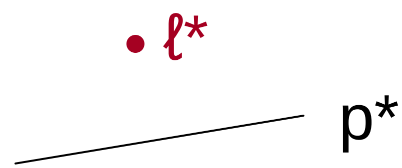
  in 2D, improve by partitioning into > 4 cells??

Method 1 (Dual)

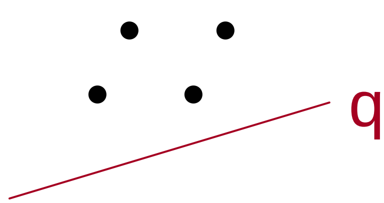
- Def: Given point $p = (a,b)$, define its dual line p^*: $y = ax - b$

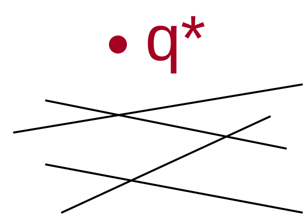
  Given line ℓ: $y = \alpha x - \beta$, define its dual point $\ell^* = (\alpha, \beta)$





$b > \alpha a - \beta \quad \Leftrightarrow \quad \beta > a\alpha - b$

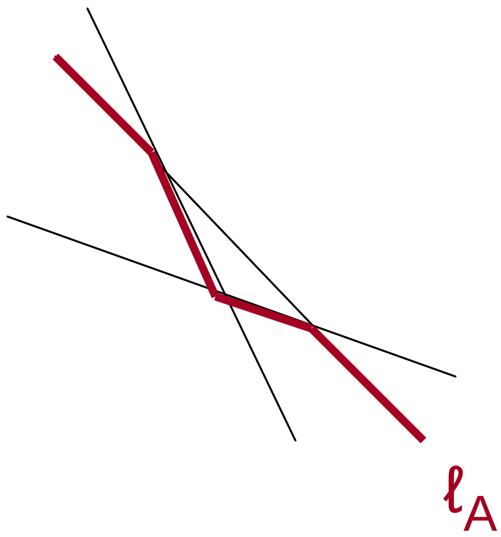


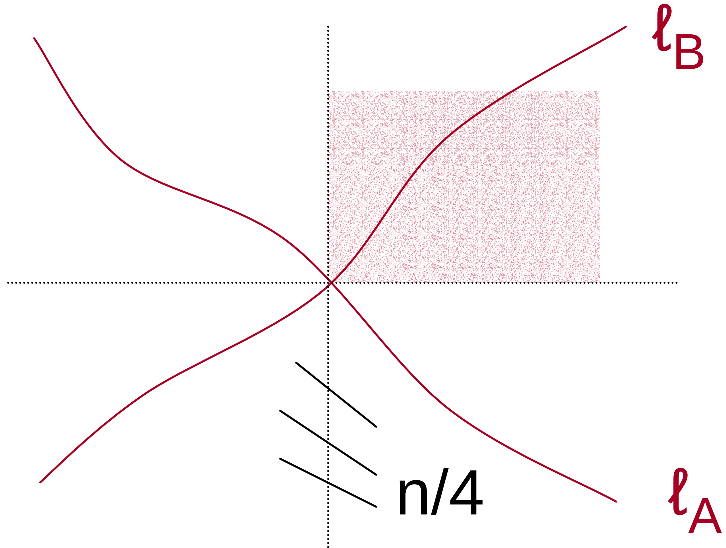


given n pts, count pts above query line $\Leftrightarrow$ given n lines, count lines below query pt

- Lemma: Given n lines in 2D, can cut the plane into 4 cells s.t. each cell intersects ≤ 3n/4 lines

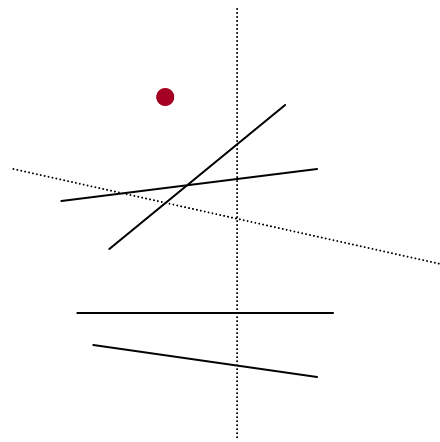
- Pf Sketch: Say median slope = 0
  Let A = lines w. slope < 0
  B = lines w. slope > 0
  Let ℓ_A = median level of A
  ℓ_B = median level of B





- Recurse $\Rightarrow$ "cutting tree"
- $S(n) = 4\ S(3n/4) + O(1)$

  $\Rightarrow\ O(n^{\log_{4/3} 4}) \approx \boxed{O(n^{4.82})}$



- Count # lines below query pt:

  $Q(n) = Q(3n/4) + O(1)$

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

- Rmks: more complicated in higher-D… (Megiddo'84/Dyer'86)

  in 2D, improve by cutting into > 4 cells??

Method 2 (Clarkson'87)

by Chazelle-Friedman'90
(bound is tight)

- Cutting Lemma: Given n lines in 2D, can cut into $O(r^2 \cancel{\log^2} r)$ disjoint cells s.t. each cell intersects $O(n/r)$ lines

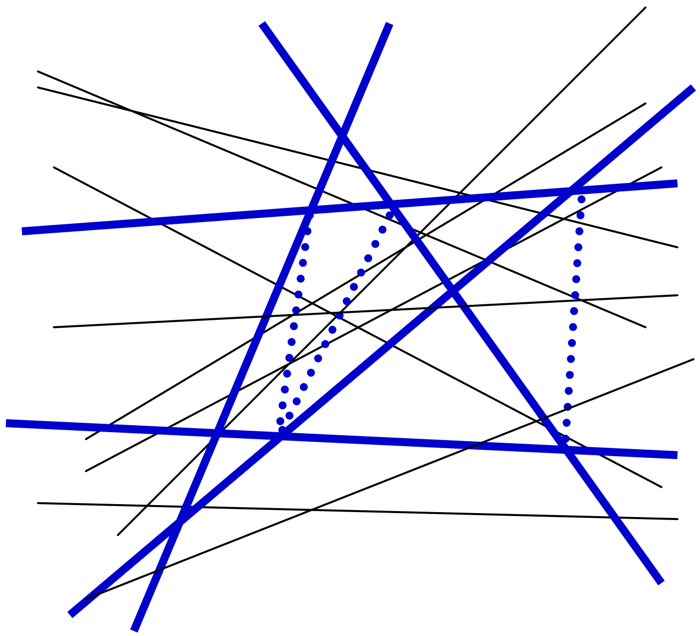
"(1/r)-cutting"

- Pf: Idea: "probabilistic method" ("ε-net"-type argument)

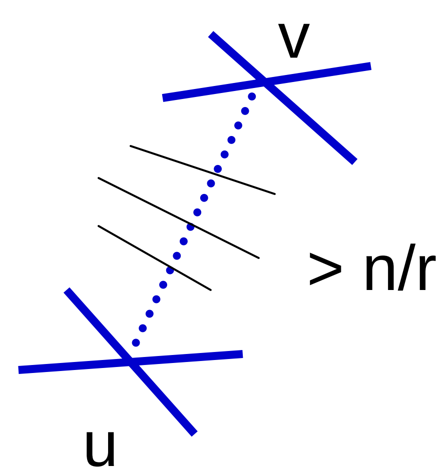
  Take random sample R of size cr

  Return a triangulation T(R) of the arrangement of R

  Success if every edge of T(R) intersects ≤ n/r lines



- Fix a line segment uv that intersects > n/r lines



- Pr{uv appears in T(R)}

  $<\approx \quad (cr/n)^4 (1 - cr/n)^{n/r}$

- Pr{failure} $<\approx \quad n^4 \cdot (cr/n)^4 (1 - cr/n)^{n/r}$

  $<\approx \quad (cr)^4 / e^c \quad << 1$

  by setting $c \approx 100 \log r$ Q.E.D.

- Recurse $\Rightarrow$ cutting tree
- $S(n) = Cr^2\ S(n/r) + O(r^2)$

  $\Rightarrow\ O(n^{\log(Cr^2)/\log r}) = O(n^{2 + \log C/\log r})\ \Rightarrow\ \boxed{O(n^{2+\varepsilon})}$

  by setting r = suff large const

- Halfplane query:

  $Q(n) = Q(n/r) + O(r^2)\ \Rightarrow\ \boxed{O(\log n)}$

- Rmks: extends to triangle query by multi-level…

  $Q(n) = O(\text{polylog } n)$

  in higher-D: $S(n) = Cr^d\ S(n/r) + O(r^d)\ \Rightarrow\ O(n^{d+\varepsilon})$

Recap:

- Simplex range searching (2D):
  - Method 0 (Willard's partition tree):
    O(n) space, $O(n^{0.793})$ time
  - Method 1 (cutting tree):
    $O(n^{4.82})$ space, O(polylog n) query time
  - Method 2 (Clarkson):
    improve space of Method 1 to near $O(n^2)$

Next:

- improve time of Method 0 to $O(n^{1/2})$??

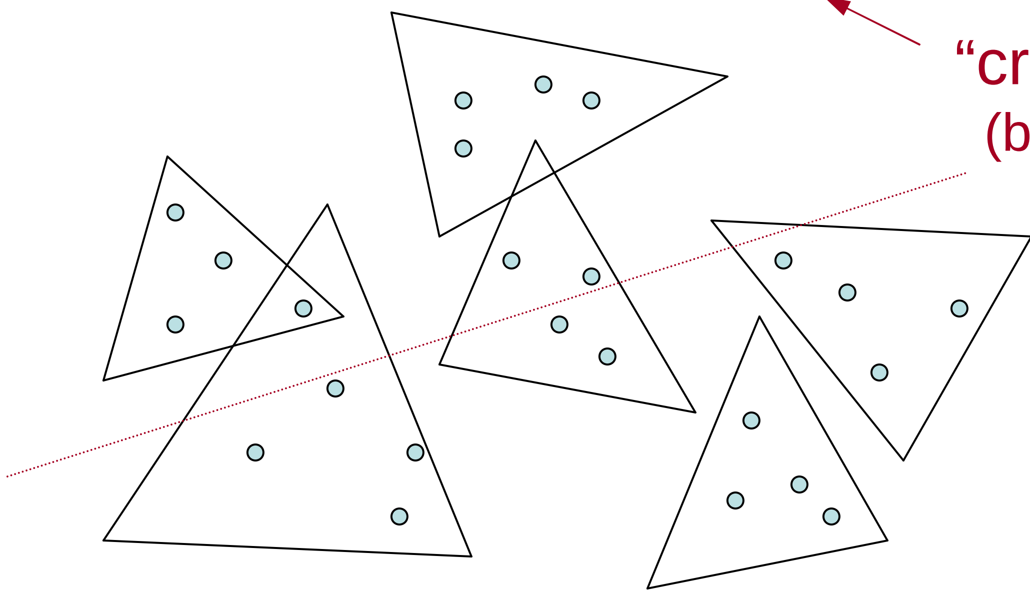
Method 3 (Matoušek'92)

- Back in primal…

- Partition Thm: Given n pts in 2D, can partition into t subsets of $\Theta(n/t)$ pts & enclose each subset P_i in a cell Δ_i

  s.t. any line crosses $O(t^{1/2})$ cells

"crossing #"
(bound is tight)



[Corollary: matching & spanning tree w. crossing # $O(n^{1/2})$]

- Recurse $\Rightarrow$ partition tree
- $S(n) = \boxed{O(n)}$
- Halfplane/triangle query:

  $Q(n) = Ct^{1/2}\, Q(n/t) + O(t)$

  $\Rightarrow O(n^{\log(Ct^{1/2})/\log t}) = O(n^{1/2 + \log C/\log t}) \Rightarrow O(n^{1/2+\varepsilon})$

  by setting t = suff large const

- Or set $t = n^{\varepsilon}$

  $\Rightarrow O(C^{O(\log\log n)}\, n^{1/2}) = \boxed{O(n^{1/2} \text{ polylog } n)}$

- Rmk: in higher-D:

  $Q(n) = Ct^{1-1/d}\, Q(n/t) + O(t) \Rightarrow O(n^{1-1/d} \text{ polylog } n)$

Pf of Matoušek's Partition Thm

- Suffice to prove crossing # for a finite set L of m "test lines" (m = $\cancel{O(n^2)}$) O(t)
- Intuition:
  1. Apply cutting lemma to L with $r = t^{1/2}$

     $\Rightarrow$ # cells $O(r^2) = O(t)$
  2. Subdivide cells to ensure each has O(n/t) pts

     $\Rightarrow$ O(t) extra cuts
  3. Total crossings between lines & cells

     $= O(t \cdot m/r) = O(m\, t^{1/2})$

     $\Rightarrow$ average # crossings per line $= O(t^{1/2})$
- Challenge: turn average to max??

- Idea: “iterative reweighting” (Welzl’88)

“weight”

- Maintain a multiset $L^{\#}$ initially containing L (multiplicity 1)
- For i = t, …, 1 do: // assume i(n/t) pts remain

1. Apply cutting lemma to $L^{\#}$ with $r = ci^{1/2}$

   $\Rightarrow$ # cells $O(r^2) \leq i$
2. Pick cell Δ_i containing ≥ n/t pts
3. Shrink Δ_i s.t. it contains exactly n/t pts P_i & remove P_i
4. For each ℓ in L crossing Δ_i

   double multiplicity of ℓ in $L^{\#}$

- Analysis:

$|\{\ell \text{ in } L^{\#} : \ \ell \text{ crosses } \Delta_i\}| \ \leq \ |L^{\#}| / r \ = \ O(|L^{\#}| / i^{1/2})$

$\Rightarrow$ $|L^{\#}|$ increases by a factor of $1 + O(1/ i^{1/2})$

$\Rightarrow$ $|L^{\#}|$ increases by a factor of $1 + O(1/i^{1/2})$

- Final value of $|L^{\#}| \leq m \prod_{i=t,\dots,1} [1 + O(1/i^{1/2})]$

$$\leq m \exp\Big(O\Big(\Sigma_{i=t,\dots,1} 1/i^{1/2}\Big)\Big)$$

$$= m \exp(O(t^{1/2}))$$

- Final multiplicity of ℓ = $2^{\text{crossing \# of } \ell}$

$\Rightarrow$ max crossing # $\leq$ log (final value of $|L^{\#}|$)

$$\leq O(\cancel{\log} m + t^{1/2})$$

$$\leq O(t^{1/2}) \qquad \text{Q.E.D.}$$

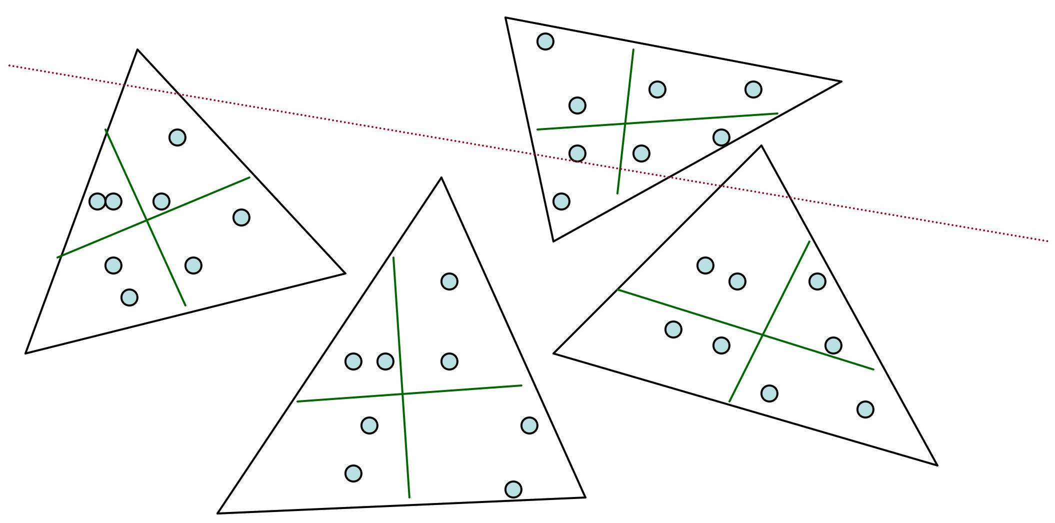
New Method (C'10)

- Idea: instead of recursion, apply iterative reweighting to an entire level of the partition tree

- Partition Refinement Thm: Given a partition with t disjoint cells each with O(n/t) pts s.t. crossing # is Z,

  can subdivide each cell into O(b) disjoint subcells each with O(n/bt) pts s.t. overall crossing # is

$$O((bt)^{1/2} + Z + b \text{ polylog } n)$$



- Repeat level by level $\Rightarrow$ partition tree
- $Z(bt) = O(Z(t) + (bt)^{1/2} + \cancel{b \text{ polylog } n})$ ignore

  i.e., $Z(u) = C\, Z(u/b) + O(u^{1/2})$

  $\Rightarrow Z(u) = O(u^{1/2})$

  by setting b = suff large const

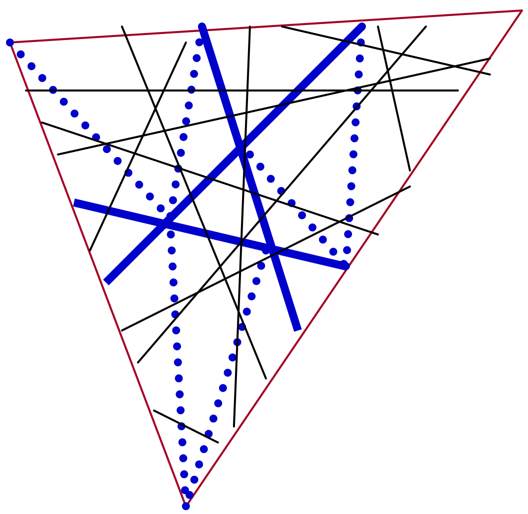
- $Q(n) = O\left(\Sigma_{t=1,b,b^2,\ldots} Z(t)\right) = \boxed{O(n^{1/2})}$ (no extra logs!)

- Rmk: in higher-D,

  $Z(u) = Cb^{1-1/(d-1)} Z(u/b) + O(u^{1-1/d}) \Rightarrow O(n^{1-1/d})$

Pf of Partition Refinement Thm

- Refined Cutting Lemma: Given n lines in 2D & triangle Δ with X intersections inside, can cut Δ into $O(r + X (r/n)^2)$ disjoint cells s.t. each cell intersects ≤ n/r lines



complexity of arrangement of sample R of size r

- Maintain multiset $L^{\#}$
- For i = t,…,1 do: // assume i cells remain
  1. Pick a remaining cell Δ_i with $X_i \leq O(|L^{\#}|^2 / i)$ intersections inside & with $m_i \leq O(|L^{\#}|$?Z$/ i)$ lines crossing it
  2. Apply cutting lemma to $L^{\#}$, Δ_i with $r = \min\{ m_i(b/X_i)^{1/2}, b \}$

     $\Rightarrow$ # subcells $O(r + X_i (r/m_i)^2) = O(b)$
  3. Further subdivide s.t. each subcell of Δ_i has O(n/bt) pts

     $\Rightarrow$ O(b) extra cuts
  4. For each ℓ in L

     multiply multiplicity of ℓ in $L^{\#}$ by $(1+1/b)^{z_i(\ell)}$

     where $z_i(\ell)$ = # new subcells of Δ_i crossed by ℓ

- Analysis:

$$\sum_{\ell \text{ in } L^{\#}} z_i(\ell) \leq O(b \cdot m_i / r) \leq O(b \cdot [(X_i / b)^{1/2} + m_i / b])$$

$$\Sigma_{\ell \text{ in } L^\#} z_i(\ell) \le O(b \cdot m_i / r) \le O(b \cdot [(X_i / b)^{1/2} + m_i / b])$$

$$\Rightarrow \text{ increase in } |L^\#| = \Sigma_{\ell \text{ in } L^\#} [(1+1/b)^{z_i(\ell)} - 1]$$

$$\le O\left(\Sigma_{\ell \text{ in } L^\#} z_i(\ell) / b\right)$$

$$\le O((X_i / b)^{1/2} + m_i / b)$$

$[\text{recall } X_i \le O(|L^\#|^2 / i), \quad m_i \le O(|L^\#| Z / i)]$

$$\le O(|L^\#| / (bi)^{1/2} + |L^\#| Z / (bi))$$

- Final value of $|L^\#| \le m \prod_{i=t,\ldots,1} [1 + O(1/ (bi)^{1/2} + Z / (bi))]$

$$\le m \exp\left(O\left(\Sigma_{i=t,\ldots,1} [1/ (bi)^{1/2} + Z / (bi)]\right)\right)$$

$$= m \exp(O((t / b)^{1/2} + (Z \ln t) / b))$$

$$= \quad m \;\; \exp(O((t / b)^{1/2} + (Z \ln t) / b))$$

- Final multiplicity of ℓ $= (1+1/b)^{\text{crossing \# of } \ell}$

$\Rightarrow$ max crossing # $\leq O(b \; \log(\text{final value of } |L^{\#}|))$

$$\leq \quad O(b \log m + (bt)^{1/2} + Z \cancel{\ln} \, t) \qquad \text{Q.E.D.}$$

by more work

- Rmk: $P(n) = O(n \log n)$ requires yet more work…

Recap:

- Simplex range searching:

  $O^*(n^d)$ space, $O(\text{polylog } n)$ query time

  or $O(n)$ space, $O(n^{1-1/d})$ time

Next:

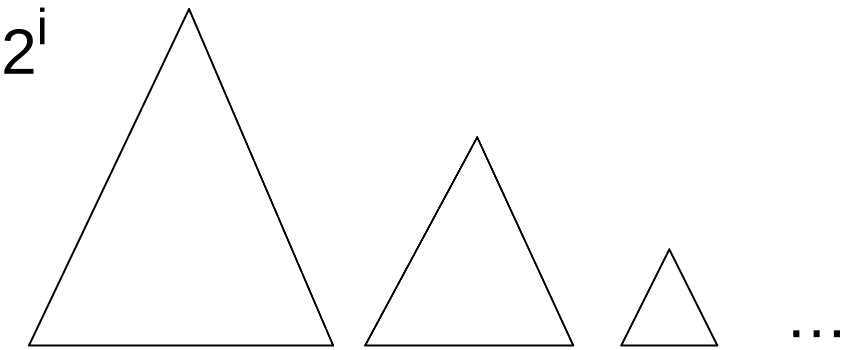
- Extensions & applications…

Extension 0: Dynamic

- Insertion: the “logarithmic method” (Bentley,Saxe’80)
  - Insert by building new subset of size 1
  - While $\exists$ 2 subsets of size 2^i

    merge by building

    new subset of size 2^{i+1}

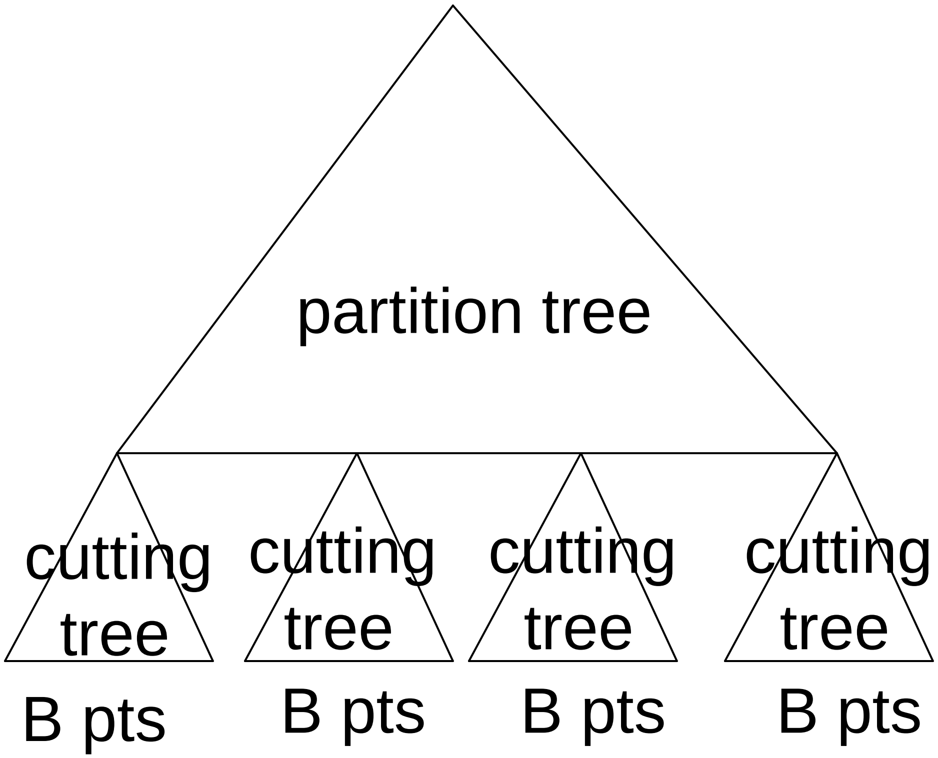


  - Total insert time = $O\left(\sum_i (n/2^i)\, 2^i \log 2^i\right) = O(n \log^2 n)$

    $\Rightarrow$ amort. time $\boxed{O(\log^2 n)}$
  - Query is “decomposable”: $Q(n) = O\left(\sum_i (2^i)^{1-1/d}\right) = \boxed{O(n^{1-1/d})}$
- Deletion: be lazy…

Extension 1: Trade-Offs

- Combine...



$$S(n) = O^*((n/B) \cdot B^d) = O^*(nB^{d-1}) = \boxed{O^*(m)}$$

by setting $B = (m/n)^{1/(d-1)}$

$$Q(n) = O((n/B)^{1-1/d} \cdot \text{polylog } B) = \boxed{O^*(n/m^{1/d})}$$

Extension/Appl'n 2: Off-Line Problems

- Ex: Given n lines & n pts in 2D, count # of pairs (p, ℓ) s.t. point p is above line ℓ (“Hopcroft's problem”)

  $O^*(m + n \cdot n/m^{1/2}) = \boxed{O^*(n^{4/3})}$

  by setting $m = n^{4/3}$

- Rmks: alg'mic pfs of combinatorial geometry problems

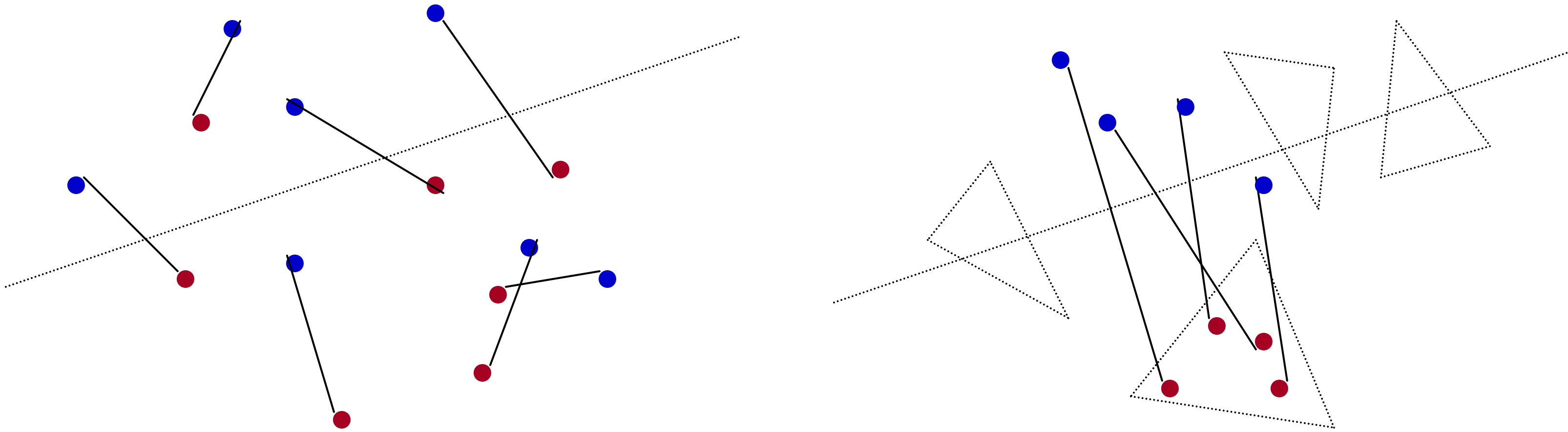
  lots & lots of other appl'ns…

  (sometimes cutting lemma suffices)

  in higher-D, $O^*(m + n \cdot n/m^{1/d}) \Rightarrow O^*(n^{2 - 2/(d+1)})$

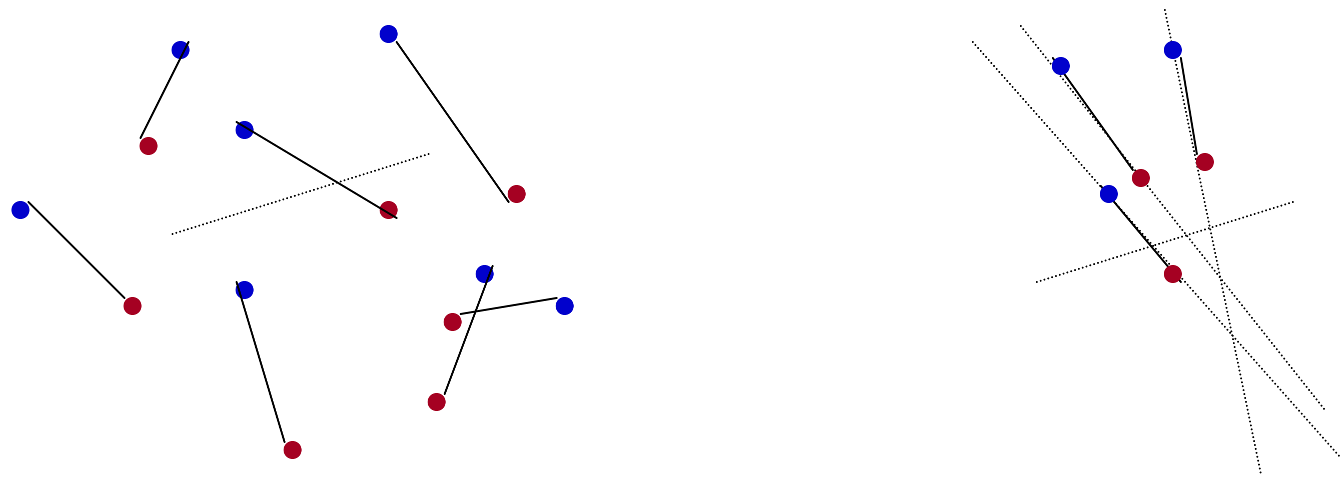
Extension/Appl'n 3: Multi-Level Data Structures

- Ex A: Count all line segments intersecting query line



- Build partition tree for red pts, where each node stores a partition tree for a subset of blue pts ("canonical subset")
- $S(n) = t\ S(n/t) + O(t \cdot n/t) \Rightarrow \boxed{O(n \log n)}$
- $Q(n) = Ct^{1/2}\ Q(n/t) + O(t \cdot (n/t)^{1/2}) \Rightarrow \boxed{O^*(n^{1/2})}$
  by setting t = suff large const

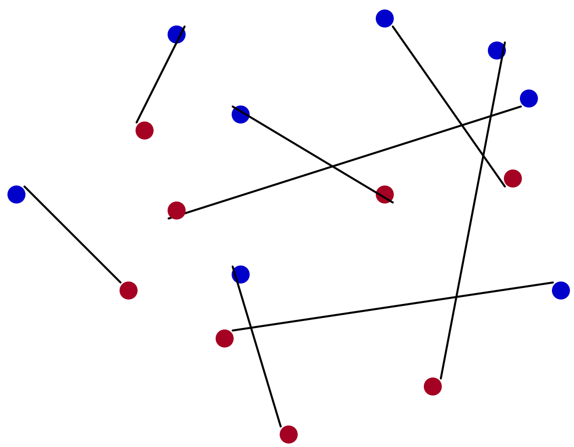
- Ex B: Count all line segments intersecting query line segment



- Build partition tree for dual of input lines, where each node stores data structure from Ex 1 for a canonical subset
- $S(n) = t\,S(n/t) + O(t \cdot (n/t) \log (n/t)) \;\Rightarrow\; \boxed{O(n \log^2 n)}$
- $Q(n) = Ct^{1/2}\,Q(n/t) + O^*(t \cdot (n/t)^{1/2}) \;\Rightarrow\; \boxed{O^*(n^{1/2})}$

  by setting t = suff large const

- Ex C: (Off-line problem) Given n line segments in 2D, count total # intersections



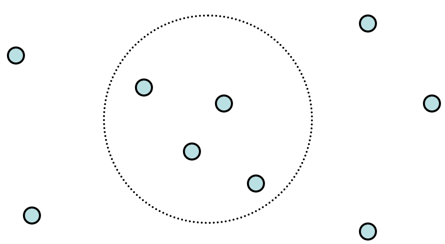
$\Rightarrow$ $\boxed{O^*(n^{4/3})}$ time alg'm

[Open: $O(n^{4/3})$ without extra factors?]

Extension 4: Non-Linear Ranges

1. By change of variables (“linearization”)

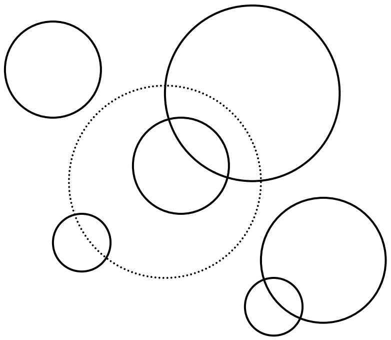
Ex A: 2D disk counting



$$
\begin{array}{lccccc}
 & (x - q_x)^2 + (y - q_y)^2 \le q_r^2 & & & & \\
\Leftrightarrow & \boxed{x^2+y^2} & - 2q_x x - 2q_y y + & \boxed{q_x^2+q_y^2-q_r^2} & \le 0 \\
 & \| & & \| & \\
\Leftrightarrow & z & - 2q_x x - 2q_y y + & q_z & \le 0
\end{array}
$$

$\Rightarrow$ 3D halfspace counting: $S(n) = \boxed{O(n)}$, $Q(n) = \boxed{O^*(n^{2/3})}$

Ex B: 2D disk intersection counting



$$(x - q_x)^2 + (y - q_y)^2 \le (r + q_r)^2$$

$$\Leftrightarrow \underbrace{\boxed{x^2+y^2-r^2}}_{z} - 2q_x x - 2q_y y - 2q_r r + \underbrace{\boxed{q_x^2+q_y^2-q_r^2}}_{q_z} \le 0$$

$$\Leftrightarrow z - 2q_x x - 2q_y y - 2q_r r + q_z \le 0$$

$\Rightarrow$ 4D halfspace counting: $S(n) = \boxed{O(n)}$, $Q(n) = \boxed{O^*(n^{3/4})}$

$n^{2/3}$

2. By directly extending partition thm, using combinatorial analysis of arrangements of surfaces (Agarwal-Matoušek'94)

$S(n) = O(n)$, $Q(n) = O^*(n^{1-1/b})$

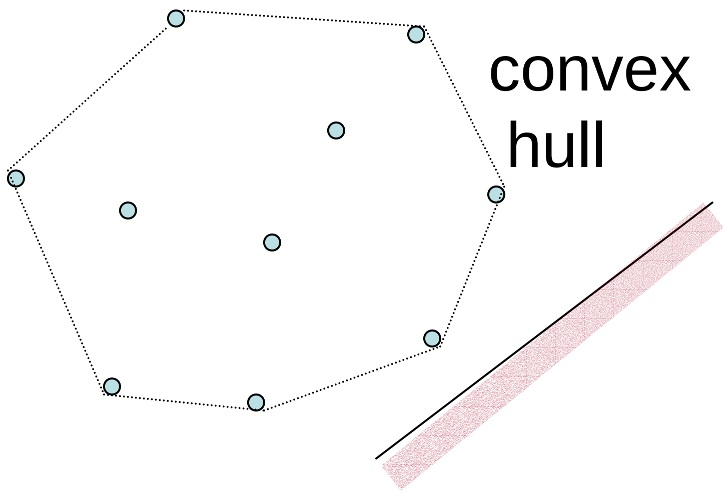
where $b = d$ if $d \le 4$; $b = \min\{ 2d - 4, \lfloor (d + \ell)/2 \rfloor \}$ else

\# vars

\# vars after linearization

Extension 5: Halfspace Reporting

- Warm-up: 2D halfspace emptiness



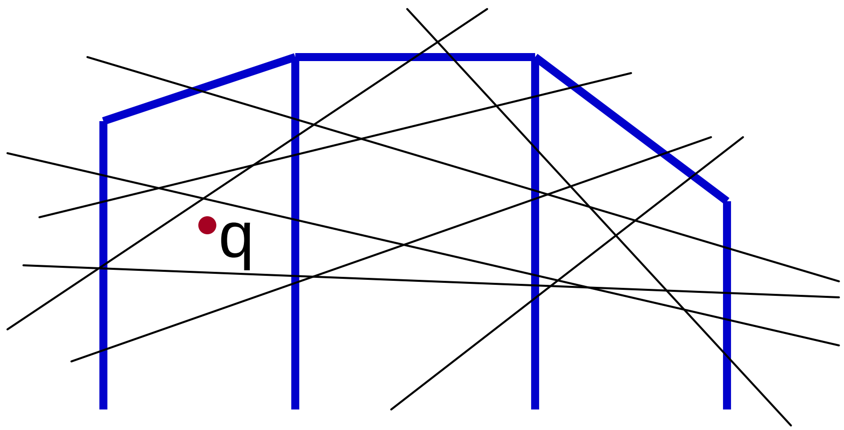
$S(n) = O(n)$

$Q(n) = O(\log n)$

- 3D halfspace emptiness
  – same (reduces to 2D point location in dual)

3D Halfspace Reporting (C'00)

- Shallow Cutting Lemma: (Matoušek'92) Given n planes in 3D, can cover all "(n/r)-shallow" pts with O(r) disjoint vertical cells s.t. each cell intersects O(n/r) planes



For r = 1, 2, 4, … do:

apply cutting lemma for r

store list L_Δ of planes

intersecting each cell Δ

$$S(n) = O\left(\sum_r r \cdot n/r\right) = \boxed{O(n \log n)}$$

- Query: take r ≈ n/k

$Q(n) = O(\log n + n/r) = \boxed{O(\log n + k)}$

point location to find Δ (John)

linear search in L_Δ

- Rmks: same approach works for 3D dominance reporting…
  can reduce space to O(n loglog n) by C'00/Ramos'99
  & to O(n) by Afshani-C'09

- Higher-D halfspace reporting:
  - By shallow versions of cutting lemma & partition thm
  - Matoušek'92 & Ramos'99/Afshani-C'09:
    $S(n) = O(n)$, $Q(n) = O(n^{1-1/\lfloor d/2 \rfloor}$ polylog $n + k)$ for any d
  - C'10:
    $S(n) = O(n)$, $Q(n) = O(n^{1-1/\lfloor d/2 \rfloor} + k$ polylog $n)$ for even d
  - Open: $O(n^{1-1/\lfloor d/2 \rfloor} + k)$? Same for odd d? Lower bds?
  - Lots of appl'ns: ray shooting, LP queries, exact nearest neighbor search, convex hull alg'ms, …

- An Open Problem: off-line halfspace counting for pts in convex position in d-D?

The End