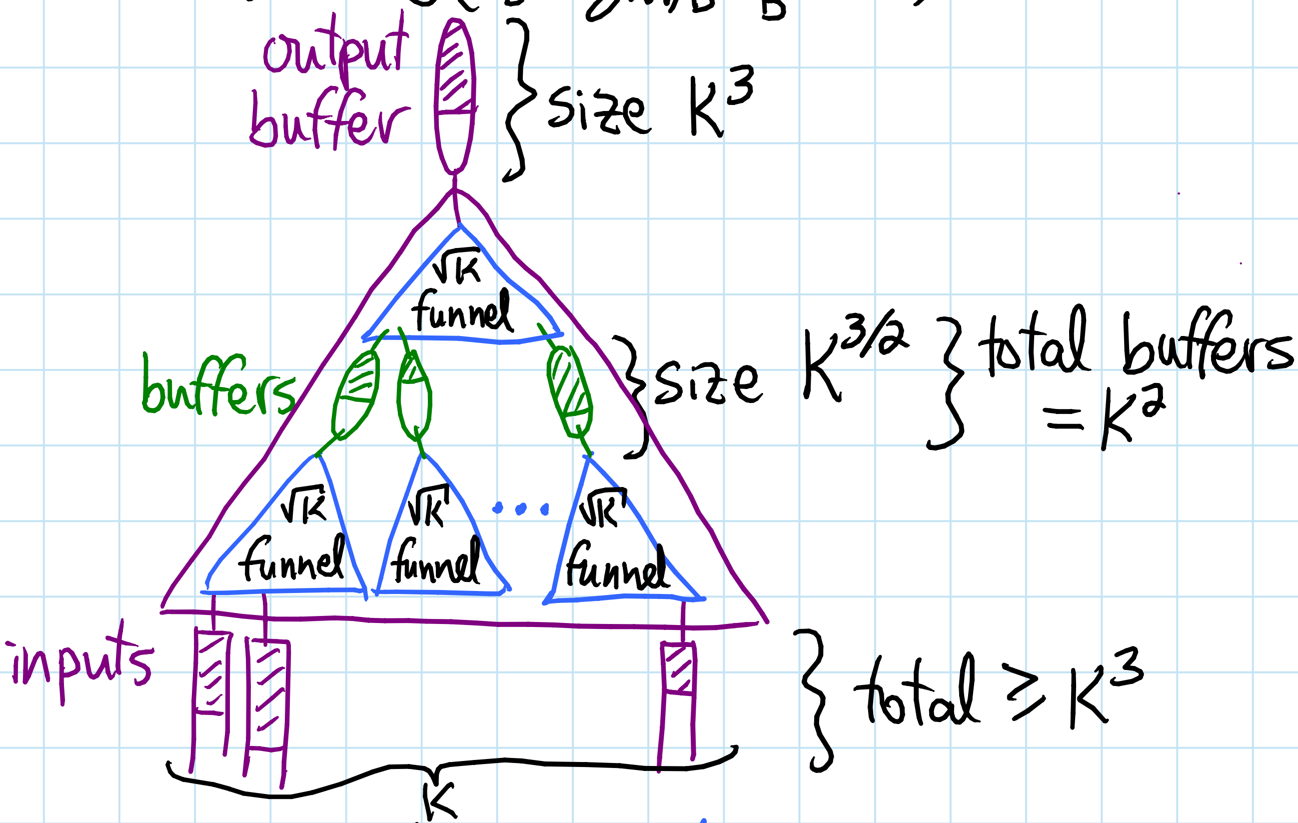
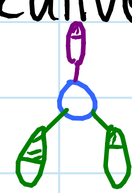
6.851	Lecture 9	Mar. 9. 2012

TODAY: Memory Hierarchies meet Geometry
- distribution sweeping via Lazy Funnelsort
- orthogonal 2D range searching:
  - batched
  - online

Lazy Funnelsort: [Brodal & Fagerberg - ICALP 2002]
- K-funnel: merges K sorted lists of total size $\Theta(K^3)$ in $O(\frac{K^3}{B} \log_{M/B} \frac{K}{B} + K)$ mem. transf.



- recursive layout: each △ stored consecutive
- fill buffer by binary merging 2 child buffers; if one empties, recursively fill it
- $N^{1/3}$-way mergesort with $N^{1/3}$-funnel merger sorts in $O(\frac{N}{B} \log_{M/B} \frac{N}{B})$ (as needed in L8 prio. queue) assuming tall cache



Distribution sweeping: [Brodal & Fagerberg – ICALP 2002]

- use lazy funnelsort to drive divide & conquer
- replace binary merger by thinking about streams of inputs & output, adding extra data along the way

Problems: all solved in $O(\frac{N}{B} \log_{M/B} \frac{N}{B} + \frac{\text{output}}{B})$

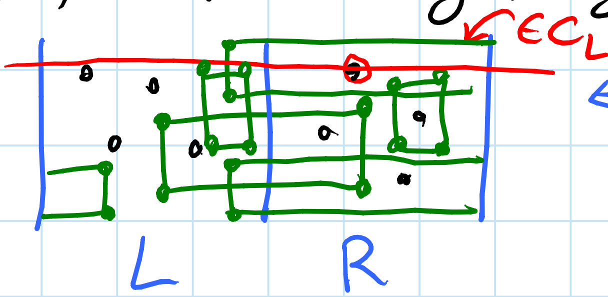
- measure of 2D rectangles
- batch orthogonal range queries
- orthogonal line segment intersection
- pairwise rectangle intersection
- line segment visibility from a point
- all Euclidean 2D nearest neighbors
- all maximal points in 3D

Batch orthogonal range searching: given N points & N rectangles, report which points are in which rectangles

- first count # rectangles containing each pt.:

① sort points & corners by x coordinate

② divide & conquer in x via lazy funnelsort in y (!) where binary merger = upward sweep

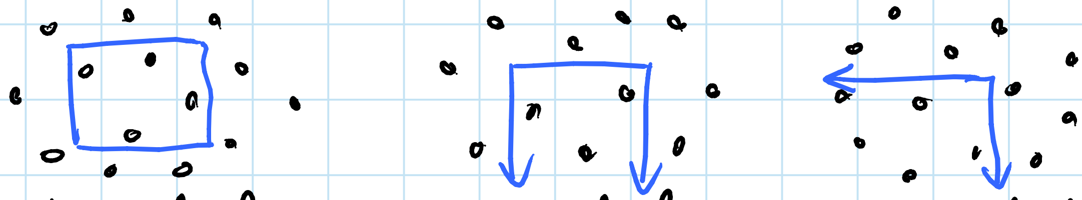


← split by x, sorted by y, merging y orders = sweep

- maintain c_L = # active rectangles ↳ stabbed by sweep line with left corners in L & spanning R (right corners are right of R)
- symetrically c_R = # active rectangles with right corners in R & spanning L
- when encountering a point in L, add c_R to its counter

- similarly compute # outputs from each merge
- allocate that much space for reporting pass
- split up recursion into $O(N)$-space parts (necessary for analysis to work out — see Brodal & Fagerberg)

Orthogonal 2D range searching: preprocess set of points to support reporting queries in $O(\log_B N + \frac{\text{output}}{B})$



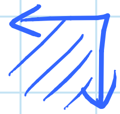
4-sided 3-sided 2-sided

- query: $O(\log_B N + \frac{\text{out}}{B})$
- space:
  - 2-sided: $O(N)$ } [Arge & Zeh - SoCG 2006]
  - 3-sided: $O(N \lg N)$ }
  - 4-sided: $O\left(N \frac{\lg^2 N}{\lg \lg N}\right)$ } [Arge, Brodal, Fagerberg, Laustsen - SoCG 2005]

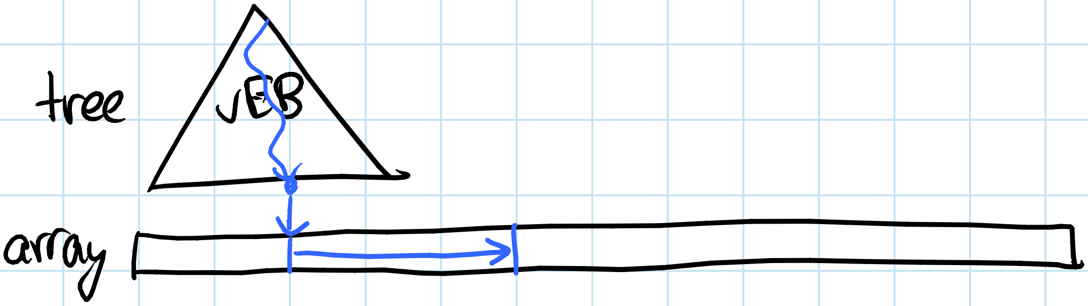
(static)

- compare with RAM: $O\left(N \frac{\lg N}{\lg \lg N}\right)$ space [L3]

2-sided: [AZ06]

 $(\le x, \le y)$

- static search tree on points, keyed by y
- array of points, with duplication



Query: $(\le x, \le y)$

1. binary search for y in tree
2. follow pointer into array
3. scan array to the right
   until reach a point whose x coord > query x
   - output unique points in $(\le x, \le y)$ ↖ filter

Claims:
- find all points in $(\le x, \le y)$
- # scanned points is O(# output points)
- array has size $O(N)$

$\alpha > 1$

Density:

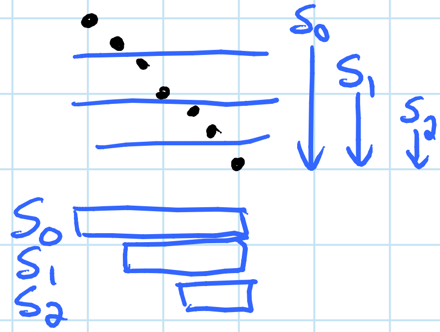
NEXT TIME: USE $\alpha = 2$

- query $(\le x, \le y)$ <u>dense</u> in S
  if # points in $(\le x, *) \le \alpha \cdot$ # points in $(\le x, \le y)$
  i.e. sorting S by x & scanning $(-\infty, x)$
  visits # points $\le \alpha \cdot$ # outputs points in S
- else $(\le x, \le y)$ <u>sparse</u> in S

First try:

- let S_0 = all points (sorted by x)
- observation: $(\leq x, \leq y)$ is surely dense in S_0 for y large e.g. $y \geq$ max y coord.
- let y_i = largest y where some query $(\leq x, \leq y_i)$ is sparse in S_{i-1}
- let $S_i = S_{i-1} \cap (*, \leq y_1)$ (sorted by x)
- repeat until S_k of constant size
- array = $S_0, S_1, S_2, \ldots, S_k$

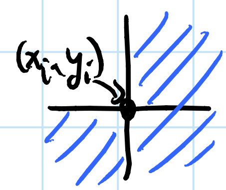
→ correct & fast queries but quadratic space:



Correct attempt: maximize common suffix

- define y_i (but not S_i) as before
- let x_i = max. where $(\leq x_i, \leq y_i)$ is sparse for S_{i-1}
- let $P_{i-1} = S_{i-1} \cap (\leq x_i, *)$
- let $S_i = S_{i-1} \cap ((*, \leq y_i) \cup (> x_i, *))$
- array = $P_0, P_1, P_2, \ldots, P_{i-1}, S_i$

↑ $O(1)$ size



Proof of claims:

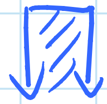
- correctness: the repeated elements always have x coord. < last seen point. in any query
  - can avoid duplicates by focusing on monotone sequence of x coords.

- space: $|P_{i-1} \cap S_i| \leq \frac{1}{\alpha} \cdot |P_{i-1}|$
  because (x_i, y_i) is sparse in S_{i-1}
  $\Rightarrow$ charge storing P_{i-1} to $P_{i-1} \setminus S_i$
  $\Rightarrow$ each point charged only once.
  factor $\frac{1}{1-\frac{1}{\alpha}} = \frac{\alpha}{\alpha-1}$
  $\Rightarrow \leq \frac{\alpha}{\alpha-1} \cdot N$ space

- query time: repetition is geometric series
  $\Rightarrow$ lose only $O(1) \times$

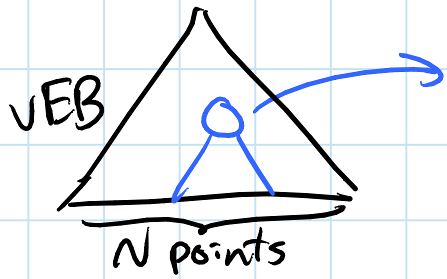
- can be computed in $O(\frac{N}{B} \log_{M/B} \frac{N}{B})$ [Brodal]

3-sided: [AZ06]



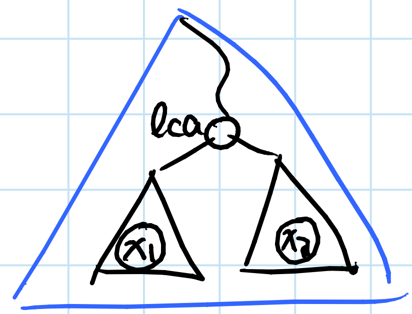
$O(\log_B N + \frac{\text{output}}{B})$ query: $O(N \lg N)$ space

- just like structure ③ in L4:
- static search tree where leaves = points, keyed by x:



stores two 2-sided structures for & on points in the subtree

$\Rightarrow O(N \lg N)$ space



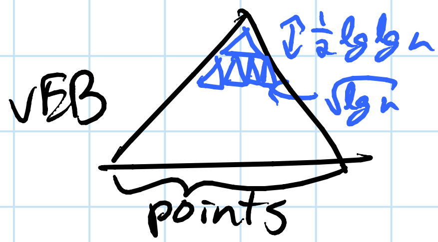
query $([x_1, x_2], \le y_2)$:
- find lca(l, r) (vEB analysis)
- query $(\ge x_1, \le y_2)$ in left child
- query $(\le x_2, \le y_2)$ in right child

$\boxed{\text{OPEN}}$: 3-sided range queries
$O(\log_B N + \frac{\text{output}}{B})$ query
$O(N)$ space
i.e. match persistent B-tree of external memory

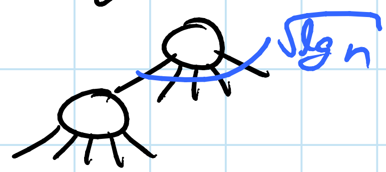
4-sided: [ABFL05]

$O(\log_B N + \frac{\text{output}}{B})$ query; $O(N \frac{\lg^2 N}{\lg \lg N})$ space

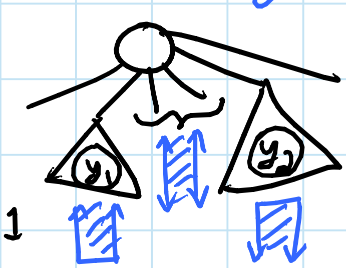
- static search tree on leaves = points, keyed by y



- conceptually contract $\frac{1}{2} \lg \lg n$-height subtrees into $\sqrt{\lg n}$-degree nodes:
  - $\Rightarrow$ height $= O(\frac{\lg n}{\lg \lg n})$



- for each such node, store
  - two 3-sided structures for on points in subtree } $O(K \lg K)$ space
  - $\lg n$ static search trees, keyed by x, on points in each interval of children } $O(K)$ spc $\cdot \lg K$
- query($[x_1, x_2], [y_1, y_2]$):
  - find lca(y_1, y_2) in tree
  - query($[x_1, x_2], \geq y_1$) in (left) child $\ni y_1$
  - query($[x_1, x_2], \leq y_2$) in (right) child $\ni y_2$
  - query($[x_1, x_2], *$) in children in between



- space:

$$O(\underbrace{\underbrace{N}_{\text{tree}} \underbrace{\lg N}_{\#\text{trees}}}_{3\text{-sided}} \underbrace{\frac{\lg N}{\lg \lg N}}_{\#\text{repetitions of element}})$$