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 $O(k^3 B \log MB + k)$ mem. transf.

Recursive layout: each $\Delta$ stored consecutive
- fill buffer by merging 2 child buffers;
  if one empties, recursively fill it
- $N^{1/3}$-way mergesort with $N^{1/3}$-funnel merger
  sorts in $O(N \log MB / B)$ (as needed in L8 priority queue)
  assuming tall cache
Distribution sweeping: [Brodal & Fagerberg—ICALP 2003]
- 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 $O\left(\frac{N \log \frac{N}{b}}{b} + \frac{\text{output}}{b}\right)$
- 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

- maintain \( c_L = \#\text{active rectangles} \) stabbed by sweep line with left corners in \( L \) & spanning \( R \) (right corners are right of \( R \))
- symmetrically \( c_R = \#\text{active rectangles} \) with right corners in \( R \) & spanning \( L \)
- when encountering a point in \( L[R] \), add \( c_R [c_L] \) 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 \(O(\log B N + \frac{\text{out}}{B})\)

- query: \(O(\log B N + \frac{\text{out}}{B})\)
- space:
  - 2-sided: \(O(N)\)
  - 3-sided: \(O(N \frac{\log^2 N}{\log \log N})\)
  - 4-sided: \(O(N \frac{\log^2 N}{\log \log N})\)
  - static: \(\Omega(N(\log \log N))^2\)

Compare with space in \(O(\log N)\) RAM DS: \[L^3\]

- 2-sided: \(O(N)\)
- 3-sided: \(O(N)\)
- 4-sided: \(O(N \frac{\log N}{\log \log N})\) space
2-sided: \([A206]\)
- static search tree on points, keyed by \(y\)
- array of points, with duplication

Query: \((\leq x, \leq y)\)
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 \((\leq x, \leq y)\)

Claims:
- find all points in \((\leq x, \leq y)\)
- \# scanned points is \(O(\# \text{output points})\)
- array has size \(O(N)\)

Density:
- query \((\leq x, \leq y)\) dense in \(S\)
  if \(\# \text{points in } (\leq x, x) \leq \alpha \cdot \# \text{points in } (\leq x, \leq y)\)
  i.e. sorting \(S\) by \(x\) & scanning \((-\infty, x)\) visits \# points \(\leq \alpha \cdot \# \text{outputs points in } S\)
- else \((\leq x, \leq y)\) sparse in \(S\)

\(\alpha > 1\)
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 \( S_i = S_{i-1} \cap (\ast, \leq y_i) \) (sorted by \( x \))
- repeat until \( S_k \) of constant size
- array = \( S_0 \cdot S_1 \cdot S_2 \cdot \ldots \cdot S_k \)
- 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 \( P_{i-1} = S_{i-1} \cap (\leq x_i, \ast) \)
- let \( S_i = S_{i-1} \cap (\ast, \leq y_i) \cup (\ast, > x_i, \ast) \)
- array = \( P_0 \cdot P_1 \cdot P_2 \cdot \ldots \cdot P_{i-1} \cdot S_i \)
- \( O(1) \) 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: \( |P_{i-1} \cap S_i| < \frac{1}{\alpha} \cdot |P_{i-1}| \)
  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 \(O\left(\frac{N^2}{B} \log_B \frac{N}{B}\right)\) [Brodal]
3-sided: \[ \mathcal{A}_{206} \]

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 \(x\):

\[
\text{VEB}
\]

stores two 2-sided structures for \(\leq\) & \(\geq\) on points in the subtree

\[\Rightarrow O(N \lg N)\text{ space}\]

query \([x_1, x_2], y_2)\):
- find \(\text{lca}(l, r)\) (VEB analysis)
- query \(\geq x_1, y_2\) in left child
- query \(\leq x_2, y_2\) in right child

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: $[\text{ABFLO5}]$

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

- Static search tree on leaves = points, keyed by $y$
  - Conceptually contract $\frac{1}{3} \log \log n$-height subtrees into $\log n$-degree nodes:
    - $\Rightarrow$ height $= O(\frac{\log n}{\log \log n})$

- For each such node, store
  - Two 3-sided structures for on points in subtree
  - $\log n$ static search trees, keyed by $x$, on points in each interval of children

- Query $([x_1, x_2], [y_1, y_2])$:
  - Find $\text{LCA}(y_1, y_2)$ in tree
  - Query $([x_1, x_2], y_1)$ in (left) child $\exists y_1$
  - Query $([x_1, x_2], y_2)$ in (right) child $\exists y_2$
  - Query $([x_1, x_2], \ast)$ in children in between

- Space:
  - $O(N \log N \frac{\log N}{\log_B N})$
  - $3$-sided
  - # repetitions of element
  - Tree # trees