TODAY: memory hierarchies II
- ordered file maintenance (for B-tree in L20)
- list labeling (for persistence in L19)
- cache-oblivious priority queue

Ordered file maintenance: [Itai, Konheim, Roteh—ICALP 1981; Bender, Demaine, Farach-Colton—FOCS 2000]

Goal: store \( N \) elements in specified order in an array of size \( O(N) \) with gaps of size \( O(1) \)
\( \Rightarrow \) scanning \( K \) consecutive elts. costs \( O(\frac{K^2}{B}) \) mem.trans
subject to elt. deletion & insertion between 2 elts.
by re-arranging elts. in array interval of \( O(lg^2 N) \) amortized elts., via \( O(1) \) interleaved scans
\( \Rightarrow \) costs \( O(\frac{log^2 N}{B}) \) amortized memory transfers

Idea: upon updating element, ensure locally not too dense/sparse by redistributing elements in surrounding interval
- intervals defined by nodes in complete binary tree on \( \Theta(lg n) \)-size chunks of array:

\[ h = lg n - O(lg lg n) \]
Update:
- update leaf node (Θ(\log n) chunk) containing elt.
- walk up tree until reach node within threshold
- \text{density}(\text{node}) = \frac{\# \text{elts. in interval below node}}{\# \text{array slots in that interval}}
- density thresholds depend on depth d of node:
  - density ≥ \frac{1}{2} - \frac{1}{4^d} \in \left[\frac{1}{4}, \frac{1}{2}\right] \text{ (not too sparse)}
  - density ≤ \frac{3}{4} + \frac{1}{4^d} \in \left[\frac{3}{4}, 1\right] \text{ (not too dense)}
  - stricter at top of the tree (larger interval)
- evenly rebalance descendant elts. in node's interval

Analysis:
- thresholds get tighter as we go up
  ⇒ rebalancing a node puts children far within threshold: 
    \text{density - threshold} = \frac{1}{4^h} = \Theta\left(\frac{1}{\log n}\right)
  ⇒ before this node is rebalanced again, must have \Ω\left(\frac{\text{capacity}}{\log n}\right) updates to bring child out of threshold
  \Ω(1) because leaves have size \Θ(\log n)
  ⇒ amortized rebuilding caused by update below a node = \Θ(\log n)
- each leaf is below \( h = \Θ(\log n) \) ancestors
  ⇒ amortized rebuilding/update = \Θ(\log^2 n)


Conjecture: \Ω(\log^2 n) necessary
List labeling: closely related problem
maintain explicit integer label in each node in a
linked list, subject to insert/delete node here,
such that labels are monotone at all times
(label = index in array)

<table>
<thead>
<tr>
<th>label space</th>
<th>best known time/update</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1+ε)n .. n lg n</td>
<td>O(\lg^2 n) - ordered file maintenance</td>
</tr>
<tr>
<td>n^{1+ε} .. n^ε</td>
<td>\Theta(\lg n) \leq \Omega \ [\text{Dietz, Seiferes, Zhang - SIDMA 2005}]</td>
</tr>
<tr>
<td>2^n</td>
<td>\Theta(1) - trivial</td>
</tr>
</tbody>
</table>

List order maintenance: easier problem, from L19
maintain linked list subject to insert/delete node here
& order query: is node x before node y?
- \Theta(1) solution via indirection: [Dietz, Sleator - STOC 1987; Bender, Cole, Demaine, Farach-Colton, Zito - ESA 2002]

\[ \Theta\left(\frac{n}{\lg n}\right) \] \{ \Theta(\lg n) solution using label space \ n \Theta(1) \}
\[ \Theta(\lg n) \Theta(\lg n) \cdots \Theta(\lg n) \] \{ trivial \(O(1)\) solution using exponential label space \}

- implicit node label = (top label, bottom label), \(0(\lg n)\) bits

\(\Rightarrow\) can compare two labels in \(O(1)\) time
- top updates change many implicit labels at once
- bottom chunks slow top updates by \(\Theta(\lg n)\) factor
- \(O(1)\) amortized cost
- worst-case bounds possible [same refs.]
Cache-oblivious priority queue: [Arge, Bender, Demaine, Holland-Minkley, Munro- SODA 2003; Stoc 2003; Brodal & Fagerberg- ICALP 2002]

- \( \lg \lg n \) levels of size \( N, N^{2/3}, N^{4/9}, \ldots, N^{c} \)
- Level \( X^{3/2} \) has 1 up buffer of size \( X^{3/2} \)
- \( \leq X^{1/2} \) down buffers each of size \( \Theta(X) \), all except first with \( \Theta(X) \) elements

- Levels stored consecutively, say smallest to largest

Invariants: keys in down buffers < keys in up buffer at same level
down buffers ordered in a level
down buffers ordered globally
Insert:
1. append to bottom up buffer
2. swap into bottom down buffers if necessary
3. if up buffer overflows: push

Push X elements into level $X^{3/2}$
- all > all els. in down buffers at level $X$ (& below)
1. sort elements
2. distribute among down buffers (& possibly up buffer):
   - scan els., visiting down buffers in order
   - when down buffer overflows: split in half, link list
   - when #down buffers overflows: move last to up buffer
   - when up buffer overflows: push it up to $X^{9/4}$

Delete-min:
1. if bottom down buffers underflow: pull
2. extract smallest elt. in bottom-left down buffer

Pull X smallest els. from level $X^{3/2}$ (& above)
1. sort first two down buffers & extract leading els.
2. if $<X$: pull $X^{3/2}$ smallest els. from level $X^{9/4}$ above
   - sort these els. + up buffer
   - put larger els. in up buffer (same # as before)
   - extract smallest els. to get X total smallests
   - split rest into down buffers
Analysis: push/pull at level $X^{3/2}$, sans recursion, costs $O\left(\frac{X}{B} \log M/B \frac{X}{B}\right)$ memory transfers

- Assume all levels of size $\leq M$ stay in cache $\Rightarrow$ free
- Tall-cache assumption: $M \geq B^2$ (else change $3/2$)
- Push at level $X^{3/2} > M \geq B^2 \Rightarrow X > B^{4/3} \Rightarrow \frac{X}{B} > 1$
- Sort costs $O\left(\frac{X}{B} \log M/B \frac{X}{B}\right)$ memory transfers
- Distribute costs $O\left(\frac{X}{B} + X^{1/2}\right)$ memory transfers
- If $X > B^2$ then cost $= O\left(\frac{X}{B}\right)$
- Else: only one such level with $B^{4/3} \leq X \leq B^2$
  
  can keep 1 block per down buffer in cache:
  $X \leq B^2 \Rightarrow X^{1/2} \leq B \leq \frac{M}{B}$ by tall cache
  
  So just pay $O\left(\frac{X}{B}\right)$ at this level too

- Pull at level $X^{3/2} > M \geq B^2$:
  - Sort costs $O\left(\frac{X}{B} \log M/B \frac{X}{B}\right)$ memory transfers
  - Another sort of $X^{3/2}$ elts. only when recursing $\Rightarrow$ charge to recursive pull

Totaling:
- $X$ elts. involved in push/pull costing $O\left(\frac{X}{B} \log M/B \frac{X}{B}\right)$
- Each elt. goes up & then down (more or less)
  
  $\lesssim$ real proof messier

$\Rightarrow O\left(\frac{1}{B} X \log M/B \frac{X}{B}\right)$ amortized cost per element

exp. geometric

$= O\left(\frac{1}{B} \log M/B \frac{X}{B}\right)$

$\square$