Today: Memory Hierarchies II (of 3)
- ordered file maintenance (for B-tree in L7)
- list labeling (for persistence in L1)
- 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)
\[ \Rightarrow \text{scanning } K \text{ consecutive elts. costs } O\left(\frac{K}{B}\right) \text{ mem. trans.} \]
subject to elt. deletion & insertion between 2 elts. by re-arranging elts. in array interval of \( O\left(\log^2 N\right) \) amortized elts., via \( O(1) \) interleaved scans
\[ \Rightarrow \text{costs } O\left(\frac{\log^2 N}{B}\right) \text{ amortized memory transfers} \]

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 \( \Theta(\log n) \)-size chunks of array.
Update:
1. update leaf by rewriting $\Theta(\log n)$-size chunk
2. walk up tree until reach ancestor whose
   density(node) = \# elts. stored below node
   \# array slots in interval
   is within threshold at its depth $d$:
   \[-\text{density} \geq \frac{1}{2} - \frac{1}{4} \frac{d}{h} \in \left[\frac{1}{4}, \frac{1}{2}\right]\] (not too sparse)
   \[-\text{density} \leq \frac{3}{4} + \frac{1}{4} \frac{d}{h} \in \left[\frac{3}{4}, 1\right]\] (not too dense)
3. evenly rebalance elements below node

Analysis:
- thresholds get tighter as we go up
  \Rightarrow rebalancing node puts children far within threshold:
  \left|\text{density} - \text{threshold}\right| \approx \frac{1}{4} \frac{1}{h} = \Theta(\frac{1}{\log N})
- this node won't be rebalanced again until
  \geq 1 child out of threshold
  \Rightarrow \Omega(\frac{\text{capacity}}{\log N}) updates to charge to
  \Omega(1) because leaf = chunk has size $\Theta(\log N)$
  \Rightarrow $O(\log N)$ amortized rebuild cost
to update element below a node
  - each leaf is below $h = \Theta(\log N)$ ancestors
  \Rightarrow $O(\log^2 N)$ amortized cost per update


Conjecture: $\Omega(\log^2 N)$ 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>$(1+\varepsilon) n \cdots n \lg n$</td>
<td>$O(\lg^2 n)$ - ordered file maintenance</td>
</tr>
<tr>
<td>$n^{1+\varepsilon} \cdots n$</td>
<td>$O(\lg n)$ - $\Omega$ via modified threshold: density $\leq \frac{1}{\alpha^2}, 1 &lt; \alpha &lt; 2$</td>
</tr>
<tr>
<td>$2^n$</td>
<td>$\Theta(1)$ - trivial [Dietz, Seiferas, Zhang - SIDMA 2005]</td>
</tr>
</tbody>
</table>

List order maintenance: easier problem, from L1
maintain linked list subject to insert/delete node here
& order query: is node $x$ before node $y$?
- $O(1)$ solution via indirection: [Dietz & Sleator - STOC 1987; Bender, Cole, Demaine, Farach-Colton, Zito - ESA 2002]

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

- can compare two labels in $O(1)$ time
- top updates change many implicit labels at once
- bottom chunks slow top updates by $O(\lg n)$ factor
- $O(1)$ amortized cost
- worst-case bounds possible [same refs.]
Cache-oblivious priority queue: (as in Arge et al. 2007)
- \( \log \log n \) levels of size \( N, N^{3/4}, N^{4/9}, \ldots, c=O(1) \)
- level \( X^{3/2} \) has 1 up buffer of size \( X^{3/2} \)
  \& \( \leq X^{1/2} \) down buffers each of size \( \Theta(X) \)
where all but first is const. frac. full

Layout: store levels in order, small to large

Invariants:
- down buffers ordered in a level (but unsorted)
- down buffers \( @X^{3/2} < \) down buffers \( @X^{9/4} \)
- down buffers \( < \) up buffer in same level
Find-min: smallest element in smallest down buffer
Delete-min: delete from down buffer; if empty, pull

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 > down buffers at level $X$ & below
1. sort elements (see L9 for cache-obliv alg.)
2. distribute among down & up buffers:
   - scan elements, visiting down buts in order
   - when down buf. overflows, split in half & link
   - when #down buts overflows, move last to up buf.
   - when up buf. overflows, push it up to $X^{9/4}$

Pull X smallest elts. from level $X^{3/2}$ (& above)
1. sort first two down buts & extract leading elts.
2. if $<X$: pull $X^{3/2}$ smallest elts. from $X^{9/4}$ (& above)
   sort these elements & up buffer
   refill up buffer to previous size
   with largest elements
   extract needed smallest elts. till $X$ total
   split rest up into down buffers
Analysis: push/pull at level $X^{3/2}$ sans recursion costs $O\left(\frac{X}{B} \log mB \frac{X}{B}\right)$ memory transfers

- assume all levels of size $\leq M$ stay in cache
- tall cache assumption: $M \geq B^a$ (say)
- push at level $X^{3/2} \geq B^2 \Rightarrow X \geq B^{4/3} \Rightarrow \frac{X}{B} > 1$
- sort costs $O\left(\frac{X}{B} \log mB \frac{X}{B}\right)$ memory transfers
- distribute costs $O\left(X^{1/2} + \frac{X}{B}\right)$ mem. transf.

  startup per down buff.  \Rightarrow scan

- if $X \geq B^2$ then cost $= O\left(\frac{X}{B}\right)$
- else: only one such level: $B^{4/3} \leq X \leq B^2$
  can keep 1 block per down buff. in cache:

  $X \leq B^3 \Rightarrow X^{1/2} \leq B \leq \frac{M}{B}$ by tall cache
  so just pay $O\left(\frac{X}{B}\right)$ at this level too

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

Total: each element goes up & then down

(roughly—real proof harder)

& costs $O\left(\frac{1}{B} \log mB \frac{X}{B}\right)$ per push & pull @ $X$

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

exp. geometric $\Rightarrow$ geometric

$= O\left(\frac{1}{B} \log mB \frac{N}{B}\right)$. 