Ordered file maintenance [Itai, Konheim, Rodeh-ICALP 1981; Bender, Demaine, Farach-Colton-FOCS 2002]

Goal: store $N$ elements in specified order in array of size $O(N)$ with gaps of size $O(1)$

$\Rightarrow$ scanning $K$ consecutive elements costs $O(\frac{K^2}{N})$ mt.

subject to element deletion/insertion between two given elements by re-arranging elements in array interval of $O(\lg^2 N)$ amortized

$\Rightarrow$ costs $O(\frac{\lg^2 N}{N})$ amortized memory transfers

Rough idea: upon updating element,

ensure locally not too dense/sparse

- grow an interval around the element until not "too" dense or sparse $\sim$ ratio $\frac{\text{elements}}{\text{space}}$

- evenly redistribute elements in that interval

In fact: intervals grow by walking up complete binary tree on $O(\lg n)$-size chunks of array:

![Diagram of complete binary tree with array chunks](conceptual complete binary tree)
Update:
- update the leaf node (Θ(\(\log n\)) chunk) containing elt.
- walk up tree until reach node within threshold
- density(node) = \# elements in interval below node / \# array slots in interval
- density thresholds depend on depth d of node:
  - density \(\geq \frac{1}{2} - \frac{1}{4} \frac{d}{h} \in [\frac{1}{4}, \frac{1}{2}]\) \(\sim\) not too sparse
  - density \(\leq \frac{3}{4} + \frac{1}{4} \frac{d}{h} \in [\frac{3}{4}, 1]\) \(\sim\) not too dense
- evenly rebalance descendant elements in node's interval

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

List labeling: related problem
maintain explicit tag for each element in a linked list
such that tags are monotone through list
subject to linked-list updates: insert/delete here

Best results: tag space
time/update

\[(1+\varepsilon)n \cdots n \lg n\]
\[n^{1+\varepsilon} \cdots n \Theta(1)\]
\[2^n \]

\[O(\lg^2 n)\]
\[O(\lg n)\]
\[\Theta(1)\]

= ordered file maintenance
\[\Rightarrow \Theta \text{ via modified thresholds}\]
\[\Omega \text{ [Dietz, Sleator, Zhang - SODA 2005]}\]

Order queries in list: easier, problem from L7 (full persistence)
maintain linked list subject to insert/delete here
& 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 tag of elt. = (top tag, bottom tag) \sim O(\lg n) bits
\[\Rightarrow\text{can still compare two tags in } O(1) \text{ time}\]
- top updates change many implicit tags at once
- bottom chunks slow updates to top by \(O(\lg n)\) factor
\[\Rightarrow O(1) \text{ amortized cost}\]
- worst-case bounds also possible [same refs.]
Cache-oblivious priority queue [Arge, Bender, Demaine, Holland-Minkley, Munro - STOC2002/SICOMP; Brodal & Fagerberg - 2002]

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

- levels stored in order, say smallest to largest

**Invariants:**

- keys in down buffers < up buffer at same level
- keys in down buffers @ \( X^{3/2} \) < keys in down buf. @ \( X^{9/4} \)
- order between down buffers in a level: first < second < ...
Insert:
1. append to smallest up buffer
2. swap into smallest down buffers if necessary
3. if up buffer overflows: push

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

Delete-min:
1. if smallest down buffer underflows: pull
2. extract smallest elt. in smallest down buffer

Pull \( X \) smallest elts. from level \( X^{3/2} \) (and above)
1. sort first two down buffers & extract leading elts.
2. if \( <X \): pull \( X^{3/2} \) smallest elts. from level \( X^{9/4} \) above
   sort these elts. & up buffer
   put larger elts. in up buffer (same # as before)
   extract smallest elts. to get \( X \) total smallests
   distribute rest into down buffers
Analysis:

Claim: 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: $M \geq B^3$ (say)
- push at level $X^{3/2} \geq B^2 \Rightarrow X > B^{4/3} \Rightarrow \frac{X}{B} > 1$
  - 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
    scan startup each down buffer
- if $X \geq B^2$ 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 $X^{3/2} \geq B^2$
  - 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)
  - real proof messier
\[ \Rightarrow O\left(\frac{1}{B} \leq \frac{X}{X} \log m_B \frac{X}{B}\right) \text{ amortized cost per element} \]
  - exp. geometric
  - log geometric
  \[ = O\left(\frac{1}{B} \log m_B \frac{N}{B}\right) \]