Eric Liu & Tom Morgan:
- sorting/priority queue equivalence
- Thorup transforms $O(nS(n))$ sorting alg.
  into $O(S(n))$ priority queue
  [Thorup 2007: Lecture 12]
- group list into blocks of size $\Theta(lg n)$
- exponentially space out $\Theta(lg n)$
  geometrically increasing buffers
- atomic heap over the buffers
- maintain these properties over updates
- can be lazier: flush buffers only when
  head shrinks too much
- working toward $O(1)$ amortized Decrease-Key
Jingjing Liu: LCAs in DAGs

- \( \text{LCA}(x, y) = \text{common ancestor} \ z \text{ of } x \text{ and } y \) 
  s.t. no descendant of \( z \) is common ancestor.
- no longer unique
- one example: common ancestor of max. depth
  length of longest path
- if just want representatives, can compute
  all-pairs LCAs in \( O(n^{2.688} \log n) \) time
  via approx. shortest paths + search,
  & in \( O(nm) \) time via "list merging", etc.
  \( \uparrow \) #edges
- fully dynamic: update = change edges around \( v \)
- Eckhardt et al. 2007: \( O(n^{2.5}) \) updates, \( O(1) \) query
- Kowaluk et al. 2008: path cover method
- as hard as dynamic transitive closure

**OPEN**: bounded in/out degree, approximation algorithms?
shortest-distance LCA?
Kevin Kelley & TB Schardl: Models of GPU

- **kernel** = “inside of loop” written in extension of subset of C
- TPUs, SMs, SPs, half-warps, ...
- 6 types of memories, mostly explicitly managed
- coalescing of structured memory reads
  ⇒ many reads for price of one

- model = fork-join parallelism + external mem.
  + no branching
- analyzed matrix transpose, matrix multiply
- some crazy behavior in experiments depending on divisibility of problem size
- still work to do
Nicholas Zehender:

faster functional tries

- $O(\min \{d, \lg D, \lg n\})$
- cf. $O(d \lg A)$ & $O(\lg n)$ previous bounds

[Demaine, Langerman, Price: Lecture 19]

- key idea: modify weights $w \rightarrow w'$ in globally biased trees

so $w'/w = O(W/w) \Rightarrow$ still $O(\lg n)$

& $w'/w = O(d)$
Mark Chen & Haitao Mao:

- Performance of integer DSs
  - Van Emde Boas: lots of space overhead, even if you store only nonempty parts.
    (But without hash table optimization)
  - x-fast tree: build time \( O(nw) \) was slow
  - y-fast tree: much better
  - Fusion trees: not useful on 32-bit words (degenerates to slow BST)
    - Estimate \( w \approx 1000 \) to be good?
  - Exponential trees: simple
  - Signature sort: still in progress
    - Again really needs large words \( w = \Omega(\log^{1+\varepsilon} w) \)
  - Timing data has lots of spikes w.r.t. \( n \)
  - Many DSs run out of memory (on 1GB)