Cache-oblivious linked list

6.897
L15.3
Apr. 16, 2003

- using ordered-file maintenance:
  - Traversal(K) — $O(\lceil K/B \rceil)$ memory transfers
    - ↳ Note: all gaps of $O(1)$ size thanks to lower density thresh.
  - Insert/Delete — $O\left(\frac{\lg^2 N}{B}\right)$ memory transfers (if $M \geq 2B$)
    - ↳ interval grown left & right, so only 2 "contexts"
    - ↳ rebalancing also done by 2 parallel scans
- this is the best cache-oblivious update bound we know if we want $O(\lceil K/B \rceil)$ worst-case traversal bound.
- but don't need to keep elements in order —
- if allowed $O(\lceil K/B \rceil + [B^\varepsilon \text{ if } K \geq B^{1-\varepsilon}])$ for any $\varepsilon > 0$, then $O\left(\frac{(\lg \lg N)^{2+\varepsilon}}{B}\right)$ amortized memory transfers per update is best known [Bender, Cole, Demaine, Farach-Colton – ESA 2002]
- amortized/self-adjusting solution: (roughly in ↗)
  - insert: add element to the end, link it in — $O(1)$
  - delete: unlink, erase element (leaving hole) — $O(1)$
  - traverse: follow pointers
    - ↳ suppose there are r runs of adjacent elements
    - ↳ then cost is $\leq r + O(\lceil K/B \rceil)$.
    - ↳ charge $r-2$ of that to the insertions/deletions that created the runs
    - ↳ merge these $r-2$ runs by copying them in the right order to end of the array
      cost: $\leq r-2 + O(\lceil K/B \rceil)$ again
    - ↳ erase old runs & relink
    - ↳ if data structure size grows by const. factor, recompactify by shifting elements left to fill the holes. $O(N)$ memory transfers, but amortizes to $O(1)$.

TOPS FORM 33041
LITHO IN U.S.A.