6.851	Class 5	Mar. 5, 2014

TODAY: Memory hierarchy
- NEW: cache oblivious vs. changing M
- NEW: tight bounds on ordered files & list labeling

Changing M: (mentioned as OPEN in L7)
(originally considered by Demaine, Lincoln, Lynch in 6.851 Spring 2012 final project)

Instruction-aligned model: [Peserico - arXiv 2013 1304.6007]
- tth instruction has $M(t)$ cache
- page replacement results:
  - offline: still optimal to evict block to be used farthest in future
  - LRU & other "dynamically conservative" algorithms (but not all conservative algs.) are $O(1)$-competitive with $O(1)$ factor larger $M(t)$ ← "resource augmentation"

Time-aligned model: [Bender, Ebrahimi, Fineman, Ghasemiesfeh, Johnson, McCauley - SODA 2014]
- time = # cache misses
- at time t, have $M(t)$ cache

⇒ messing up advances t & changes $M(t)$

- <u>speed augmentation</u>: algorithm gets to run c times faster than OPT
- usually same as saying algorithm uses c times more time than OPT
  ~ here makes a big difference
- page replacement results:
  - <u>offline</u>: still optimal to evict block to be used farthest in future
  - <u>LRU</u> is competitive with 4-speed-augment. & 4-resource-augment. → $M(t)$
- not all cache-oblivious algs. are good:
  e.g. scan N_2 numbers → $O(N_2/B)$
  multiply $N_1 \times N_1$ matrices → $O(N_1^3/B\sqrt{M})$
  - $M(t)$ could be big then small so should have done opposite order
  - get separation of $\Omega(\sqrt{M}/B)$ ↳ max or average

  <u>OPEN</u>: bigger separation?

- many cache-oblivious algorithms <u>are</u> optimal with $M(t)$ & $O(1)$-speed/resource-augm.
  - <u>divide & conquer</u> with $f(N)$ recursive calls of $g(N)$ size, & <u>$O(1)$ split/combine cost</u>
  - e.g. matrix multiply/transpose, Gaussian elimination, all-pairs short. paths
  - also lazy funnel sort

List labeling: [Bulánek, Koucký, Saks – STOC 2012]

label space	label changes/update	ref.
n	$O(\lg^3 n)$	[Zhang – PhD 1993]
$n + O(n^{1-\varepsilon})$	$\Omega(\lg^3 n)$	[BKS12]
* $n + \underbrace{f(n)}_{o(n)}$	$\Omega(\lg^2 n \cdot \lg \frac{n}{f(n)})$	[BKS12]
STARTING EMPTY!	$O(\lg^2 n \cdot \lg \frac{n}{f(n)})$	[Zhang – PhD 1993]
$(1+\varepsilon)n$	$\Omega(\lg^2 n)$	[BKS12]
ordered file	$O(\lg^2 n)$	[Itai, Konheim, Rodeh – ICALP 1981]
* $n \cdot \underbrace{f(n)}_{o(n)}$	$\Omega(\lg^2 n / \lg f(n))$	[BBCKS – manuscript 2012]
	$O(\lg^2 n / \lg f(n))$	[IKR81]
$\Theta(n^{1+\varepsilon})$	$\Omega(\lg n)$	[Dietz, Seiferas, Zhang – SIDMA 2004]
	CORRECTED by [Babka, Bulánek, Čunát, Koucký, Saks – ESA 2012]	
	also randomized [Bulánek, Koucký, Saks – ICALP 2013]	
tiny gap — $\Theta(n^{\lg^2 n})$	$O(\lg n / \lg\lg n)$	[BKS12]
* $n^{\underbrace{f(n)}_{\Omega(\lg^2 n)}}$	$\Omega(\lg n / \lg f(n))$	[BBCKS12]
	$O(\lg n / \lg f(n))$	[BKS12]
2^{n^ε}	$O(1)$	[BKS12]

Problem 1: Achieve $O(\lg^3 n)$ amortized moves per insert into size-n ordered file (initially empty)

— hint: consider first $n/2$ insertions

Problem 2a: Cache-oblivious search tree supporting search in $O(\log_{B+1} N)$ & insert/delete-here in $O(1)$ amortized

Problem 2b: bulk insert/delete of K items in $O(1 + K/\lg B)$ mem. transfers