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]
- $t$th 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 alg.) 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
  $\Rightarrow$ messing up advances $t$ & changes $M(t)$
- **speed augmentation**: algorithm gets to run \( c \) times faster than \( \text{OPT} \)
- usually same as saying algorithm uses \( c \) times more time than \( \text{OPT} \)
  - here makes a big difference
- page replacement results:
  - **offline**: still optimal to evict block to be used farthest in future
  - \( \text{LRU} \) is competitive with \( M(t) \)
    - \( 4\)-speed-augment. & \( 4\)-resource-augment.
- not all cache-oblivious alg. are good:
  - e.g. scan \( N^2 \) numbers \( \rightarrow O(N^2/B) \)
  - multiply \( N \times N \) matrices \( \rightarrow O(N^3/B^{1/2}M) \)
- \( M(t) \) could be big then small
  - so should have done opposite order
- get separation of \( \Omega(M/B) \)
  - \( M \) max or average
  - **OPEN**: bigger separation?
- many cache-oblivious algorithms are optimal with \( M(t) \) & \( O(1) \)-speed/resource-augm.
  - divide & conquer with \( f(N) \) recursive calls of \( g(N) \) size, & \( O(1) \) split/combine cost
  - e.g. matrix multiply/transpose, Gaussian elimination, all-pairs short. paths
- also lazy funnel sort
List labeling: \([\text{Bulanék, Koucký, Saks - STOC 2012}]\)

- **label space**
  - \(n\)
  - \(n + O(n^{1-\varepsilon})\)
  - \(n + f(n)\) \(\Theta(n^{1+\varepsilon})\) \(\Theta(n^{\frac{\log^2 n}{f(n)}})\)

- **label changes/update**
  - \(O(\log^3 n)\)
  - \(\Omega(\log^3 n)\)
  - \(\Omega(\log^2 n \cdot \log \frac{n}{f(n)})\)
  - \(\Omega(\log^2 n / \log f(n))\)
  - \(O(\log n / \log \log n)\)
  - \(\Omega(\log n / \log f(n))\)
  - \(O(1)\)

- **ref.**
  - [Zhang - PhD 1993]
  - [BKS12]
  - [BKS12]
  - [Zhang - PhD 1993]
  - [BKS12]
  - [Itai, Konheim, Roteh - ICALP 1981]
  - [BBCKS - manuscript 2012]
  - [IKR81]
  - [Dietz, Seiferas, Zhang - SODA 2004]
  - [Bulanék, Koucký, Saks - ICALP 2013]

Tiny gap:
  - \(\Theta(n^{\frac{\log^2 n}{f(n)}})\)
  - \(\Omega(\log n / \log f(n))\)

- **correction**
  - [Bulyka, Bulanék, Čunát, Koucký, Saks - ESA 2012]

Also randomized:
  - [Bulanék, Koucký, Saks - ICALP 2013]
Problem 1: Achieve $O(\log^3 n)$ amortized moves per insert into size-\(n\) ordered file (initially empty)

- hint: consider first \(\frac{n}{2}\) 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 $O(1 + K/\log B)$ mem. transfers