Where are we

• Traditional (RAM-model) analysis: Heapsort is better
  - Heapsort worst-case complexity is $\Theta(n \log n)$
  - Quicksort worst-case complexity is $\Theta(n^2)$.
    • average-case complexity should be ignored.
    • probabilistic analysis of randomized version is $\Theta(n \log n)$

• Yet Quicksort is popular.

• Goal: a better model of computation.
  - It should reflect the real-world costs better.
  - Yet should be simple enough to perform asymptotic analysis.
2-level memory hierarchy model (MH₂)

Data moves in “blocks” from Main Memory to cache.
   - A block is b contiguous items.
   - It takes time b to move a block into cache.
   - Cache can hold only b blocks.
     Least recently used block is evicted.

Individual items are moved from Cache to CPU.
   - Takes 1 unit of time.

For asymptotic analysis, we want b to grow with n
   - b = n^{1/3} or n^{1/4} are plausible choices

<table>
<thead>
<tr>
<th></th>
<th>block size = b (Bytes)</th>
<th>cache size = b² (Bytes)</th>
<th>transfer (cycles)</th>
<th>memory = n (Bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory = DRAM</td>
<td>2^6 - 2^8</td>
<td>2^{13} - 2^{20}</td>
<td>2^5 - 2^7</td>
<td>2^{26} - 2^{30}</td>
</tr>
<tr>
<td>Cache = SRAM</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Memory = disk</td>
<td>2^7</td>
<td>2^{14}</td>
<td>2^7</td>
<td>2^{28}</td>
</tr>
<tr>
<td>Cache = Dram</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b = n^{1/4}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b=n^{1/3}</td>
<td>2^{13}</td>
<td>2^{26}</td>
<td>2^{13}</td>
<td>2^{39}</td>
</tr>
</tbody>
</table>
**Cache lines of heap (b=8, n=511, h=9)**

- 6 levels, 8 blocks

**A worst-case Heapsort instance**

Each Extract-Max goes all the way to a leaf.

Visits to each node alternate between left and right child.

Actually, for any sequence of paths from root to leaves, one can create example.

Construct starting with 1-node heap
MH₂ analysis of Heapsort

• Assume \( b = n^{1/3} \).
  - Similar analysis works for \( b = n^a, 0 < a < \frac{1}{2} \).

• Effect of LRU replacement:
  - First \( n^{2/3} \) heap elements will "usually" be in cache.
    • Let \( h = \lfloor \log n \rfloor \) be height of the tree.
    • These elements are all in top \( \lceil (2/3)h \rceil \) of tree.
  - Remaining elements won’t usually be in cache.
    • In worst case example, they will never be in cache when you need them.
    • Intuition: Earlier blocks of heap are more likely to be references than a later one. When we kick out an early block to bring in a later one, we increase misses later.

MH₂ analysis of Heapsort (worst-case)

• Every access below level \( \lceil (2/3)h \rceil \) is a miss.

• Each of the first \( n/2 \) Extract-max’s “bubbles down” to the leaves.
  - So each has at least \( (h/3) - 1 \) misses.
  - Each miss takes time \( b \).

• Thus, \( T(n) > (n/2)((h/3) - 1) b \).
  - Recall: \( b = n^{1/3} \) and \( h = \lfloor \log n \rfloor \).

• Thus, \( T(n) \) is \( \Omega(n^{4/3} \log n) \).

• And obviously, \( T(n) \) is \( O(n^{4/3} \log n) \).
  - Each of \( cn \log n \) accesses takes time at most \( b = n^{1/3} \).
    (where \( c \) is constant from RAM analysis of Heapsort).
Quicksort $\mathcal{MH}_2$ complexity

- Accesses in Quicksort are sequential
  - Sometimes increasing, sometimes decreasing
- When you bring in a block of $b$ elements, you access every element.
  - Not 100% true, but I'll wave my hands
- We take $b$ time to get block for $b$ accesses
- Thus, time in $\mathcal{MH}_2$ model is same as RAM.
  - $\Theta(n \lg n)$

Bottom Line: MH2 analysis shows Quicksort has lower complexity than Heapsort!