Where are we?

- Traditional (RAM-model) analysis: Heapsort is better
  - Heapsort worst-case complexity is $\Theta(n \lg n)$
  - Quicksort worst-case complexity is $\Theta(n^2)$.
    - average-case complexity should be ignored.
    - probabilistic analysis of randomized version is $\Theta(n \lg n)$
- 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 (MH$_2$)

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.

![Diagram of memory hierarchy]

Note - "$b$" affects:
1. block size
2. cache capacity ($b^2$)
3. transfer time

For asymptotic analysis, we want $b$ to grow with $n$

- $b = \frac{1}{3}$ or $\frac{1}{4}$ are plausible choices.

<table>
<thead>
<tr>
<th></th>
<th>“block” (Bytes)</th>
<th>“cache” (Bytes)</th>
<th>time (cycles)</th>
<th>“memory” (Bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRAM/DRAM</td>
<td>$2^6 - 2^8$</td>
<td>$2^{13} - 2^{20}$</td>
<td>$2^{5} - 2^{7}$</td>
<td>$2^{27} - 2^{30}$</td>
</tr>
<tr>
<td>$b = n^{1/4}$</td>
<td>$2^7$</td>
<td>$2^{14}$</td>
<td>$2^{7}$</td>
<td>$2^{28}$</td>
</tr>
<tr>
<td>DRAM/disk</td>
<td>$2^{12} - 2^{13}$</td>
<td>$2^{26} - 2^{30}$</td>
<td>$2^{15} - 2^{20}$</td>
<td>$2^{32} - 2^{36}$</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>
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 $b = n^a$, $0 < a < \frac{1}{3}$.

• Effect of LRU replacement:
  - First $n^{2/3}$ heap elements will “usually” be in cache.
    • Let $h = \lceil \log n \rceil$ 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.
    • (Caution: hand waving) In general, an earlier block of array is more likely to be accessed than a later one. When we kick out an early block to bring in a later one, we increase misses later.
Cache lines of heap (b=8, n=511, h=9)

MH$_2$ analysis of Heapsort (worst-case)

- Every access below level ⌈(2/3)h⌉ is a miss.
- Each of the first n/2 Extract-max's "bubbles down" to the leaves.
  - So it 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 = ⌊lg n⌋.
- Thus, T(n) is Ω(n$^{4/3}$ lg n).
- And obviously, T(n) is O(n$^{4/3}$ lg n).
  - Each of c n lg n accesses takes time at most b = n$^{1/3}$.
    (where c is constant from RAM analysis of Heapsort).
Quicksort 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%, but I'll wave my hands

- We take $b$ time getting block for $b$ accesses

Thus, time in MH$_2$ model is same as RAM.