CSE 202 - Algorithms

Quicksort vs Heapsort:

the "inside" story

or

A Two-Level Model of Memory

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.

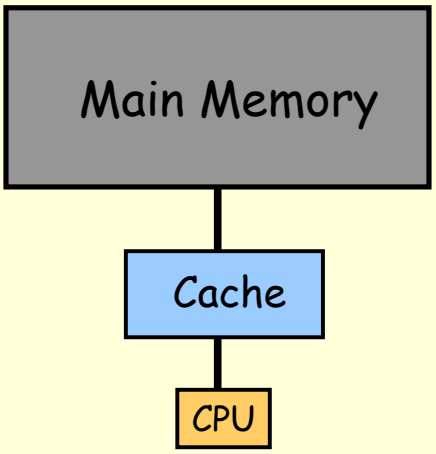
2-level memory hierarchy model (MH_2)

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.



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

2-level memory hierarchy model (MH_2)

For asymptotic analysis, we want b to grow with n

- b = 1/3 or 1/4 are plausible choices.

	"block" (Bytes)	"cache" (Bytes)	time (cycles)	"memory" (Bytes)
SRAM/DRAM	2^6 - 2^8	2^{13} - 2^{20}	2^5 -2^7	2^{27} - 2^{30}
$b = n^{1/4}$	2^7	2^{14}	2^7	2^{28}
DRAM/disk	2^{12} - 2^{13}	2^{26} - 2^{30}	2^{15} - 2^{20}	2^{32} - 2^{36}
$b=n^{1/3}$	2^{13}	2^{26}	2^{13}	2^{39}

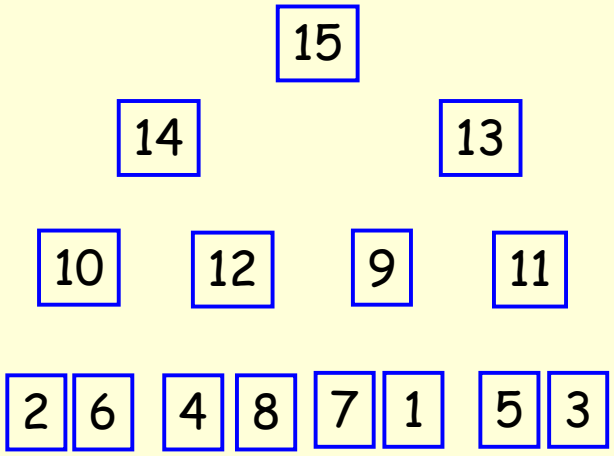
A worst-case Heapsort instance

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.

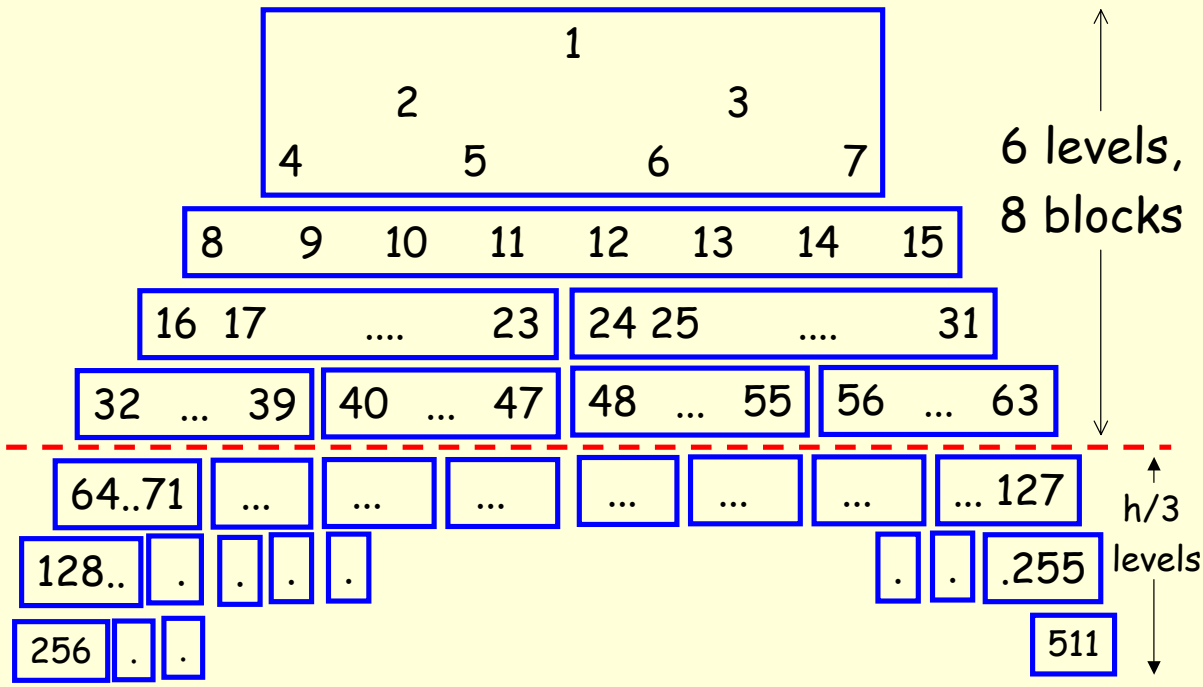
Construct starting with 1-node heap



MH_2 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 $h = \lfloor \lg n \rfloor$ 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.
    - (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 $\lceil (2/3)h \rceil$ 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 $h = \lfloor \lg n \rfloor$.
- Thus, $T(n)$ is $\Omega(n^{4/3} \lg n)$.
- 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.