1/30/23 Sorting

regular #s have a natural total order

partial order - complex #s

Bogo sort random $O(n) = n!$

selection sort - look for smallest item over & over $n^2 = O(n)$

Insertion Sort - start w/ tiny list of length 1, continue checking array of items & insert them where needed $n^2 = O(n)$

Binary search - fastest
- $n\log(n)$

Bubble Sort - look @ elements & swap n^2
- items bubble up

(RAM)
Memory is 100 x slower than CPU (cache is fast)
Cache affinity

Merge Sort - 2 lists, take two elements, compare, into 1 list, cut in 1/2, do exact same thing, it creates "runs" of 4. $n\log n$

Shell sort $On^{\frac{5}{3}}$
- rly weird
- faster than quick sort when < [illegible]

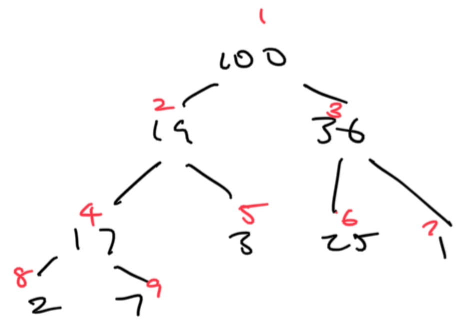
$n\log n$
merge sort
heap sort
quicksort (average case)
smallest constant

Quick Sort
- pick #, average of 3 items this is pivot
  - people pick different pivot algorithms
- 3 arrays of whats smaller, equal & bigger
- sort smaller & larger using the same algorithm of either

Heapsort n 3
- fastest

Heaps
- many kinds
- a binary tree
- 1 node = heap
- parent node is > than leaves
- can have 0, 1, or 2 child

n log n
- not recursive



left child is always $i*2$
right is $(i*2)+1$

Build heap n log n
- start @ last element

merge sort works for magnetic tape storage bc you can't jump back & forth, just need 3 tapes, assuming you don't have RAM
using comparisons, smallest is n log n

Radix sort (kinda n log n)
- look @ lowest digit
- doesn't use comparison