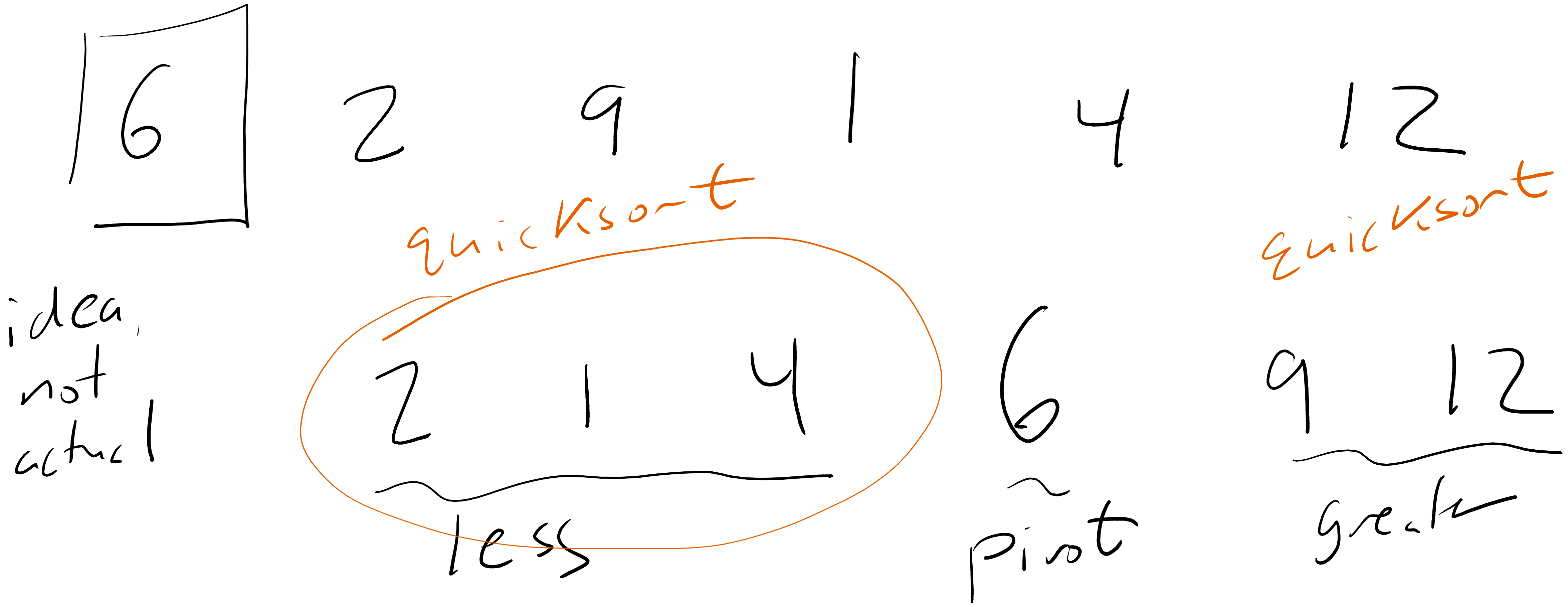
Quicksort

Idea (details vary a little)

- pick one item: "pivot"
  (common choice: first one)

"split" items so less than pivot are on one side, and greater than the pivot are on the other



* Example:

54 26 93 17 77 31 44 55 20

pivot left ~~up~~ up right down

as long as up < pivot, advance up
as long as down > pivot, slide down
→ swap items at up and down

swap

54 26 20 17 77 31 44 55 93

~~up~~ up down ~~down~~

repeat

54 26 20 17 44 31 77 55 93

up down

54 26 20 17 44 31 77 55 9

~~up~~ ~~down~~

repect

down up

crossed!

stopping condition

swap pivot with the value at location down

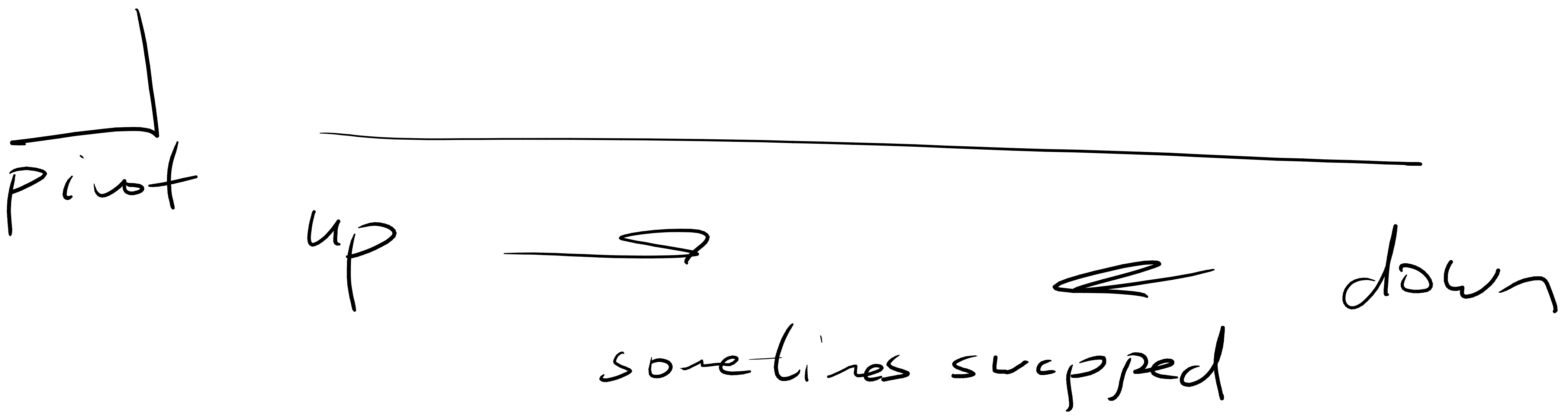
31 26 20 17 44 54 77 55 93

Quicksort

Quicksort

Performance:

To do what we just did: (one pass)
how much work?

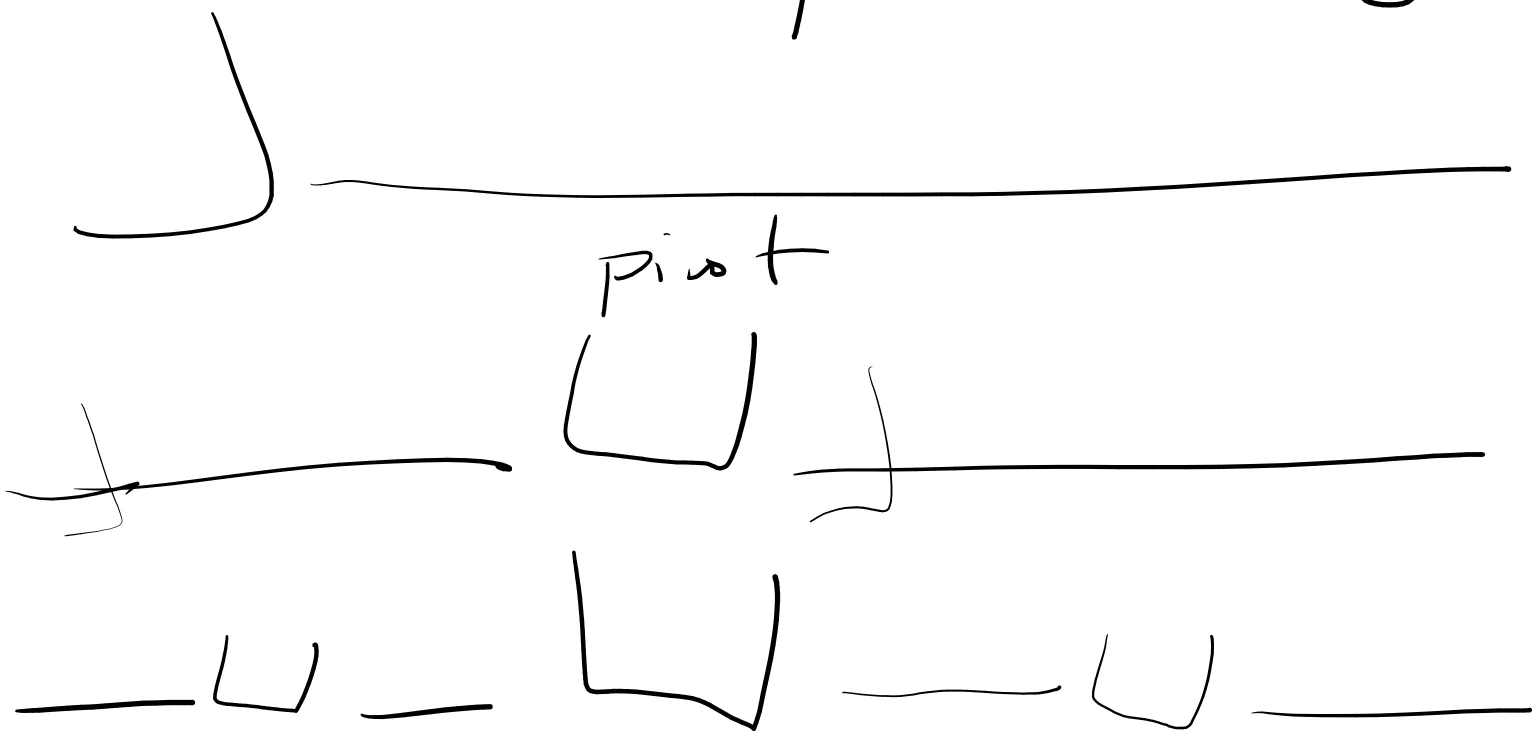


O(n) for one pass / one level

How many levels?

If things split nicely:

if pivot is approx in the middle after sweeping through



≈ log n levels
so whole alg O(n log n)

If things don't split nicely
(if piot is always less than all
(greater) else)

$$2 \;] \; \underbrace{3 \quad 8 \quad 9 \quad 15 \quad 18}_{\text{Quicksort}}$$

$$3 \;] \; \underbrace{8 \quad 9 \quad 15 \quad 18}_{\text{Quicksort}}$$

etc.

# of level is $\approx n$

total work is $O(n * n)$

$= O(n^2)$

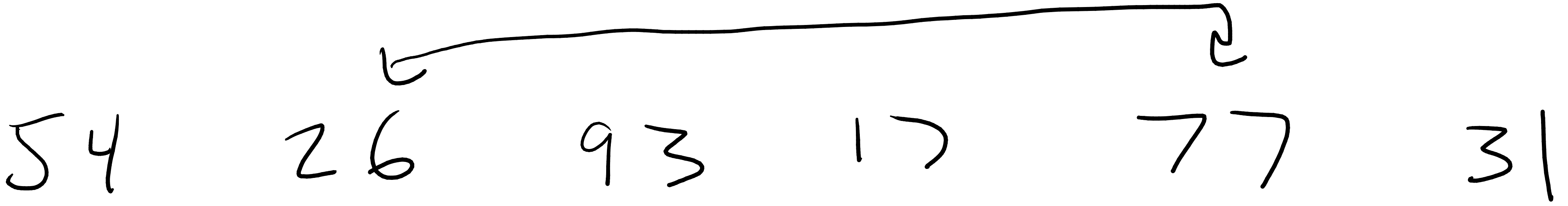
which is terrible.

Even though quicksort and mergesort are both $O(n \log n)$ [for typical quicksort], quicksort is faster in practice (smaller "c")

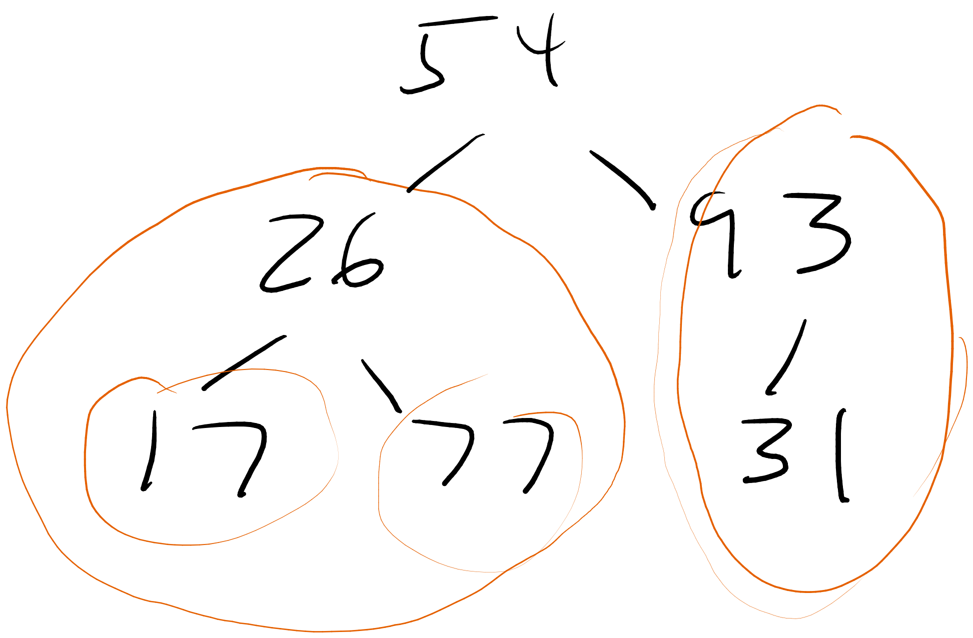
Reason: Quicksort doesn't have to copy all data like mergesort does (no temp list!)

Heapsort

- start with a list
- turn it into a heap ^{max}
- remove one-by-one, putting at end of list

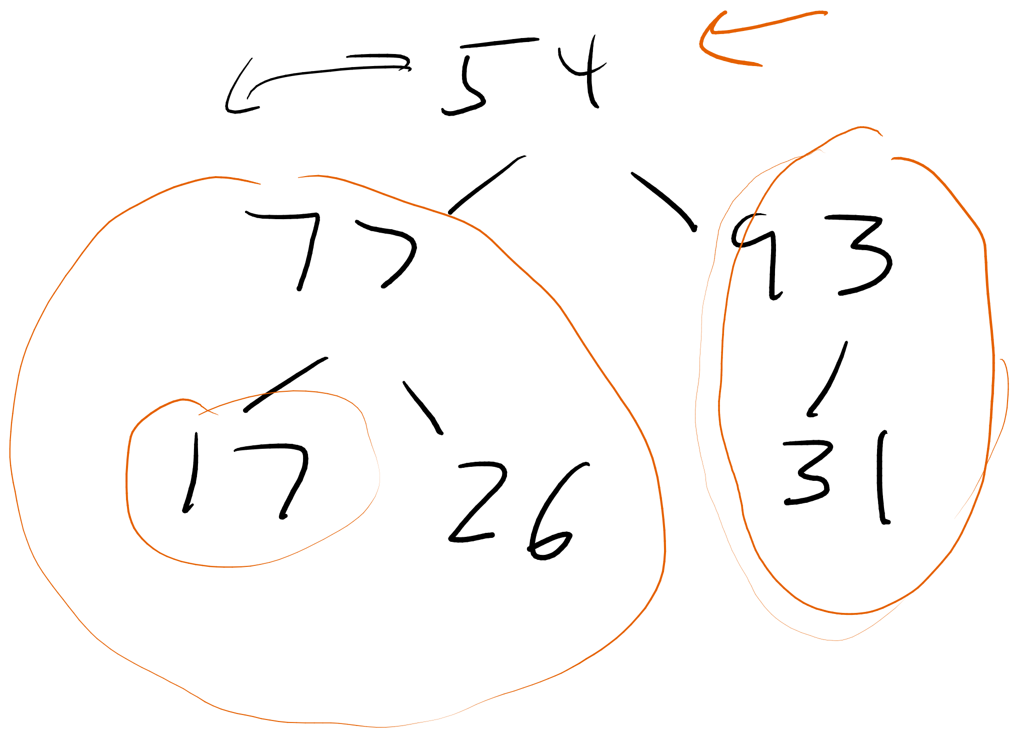


Think of as a screwed up heap

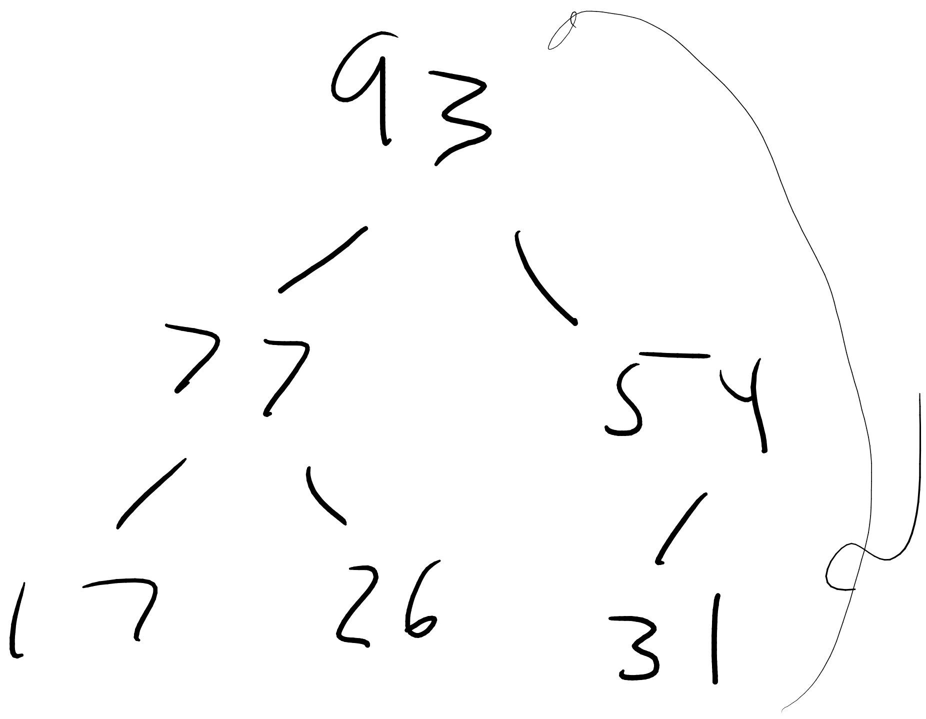


compare. ok? ✓

To fix it, we start with rightmost leaf, then go left backwards through the heap, fixing as we go. "heapify"



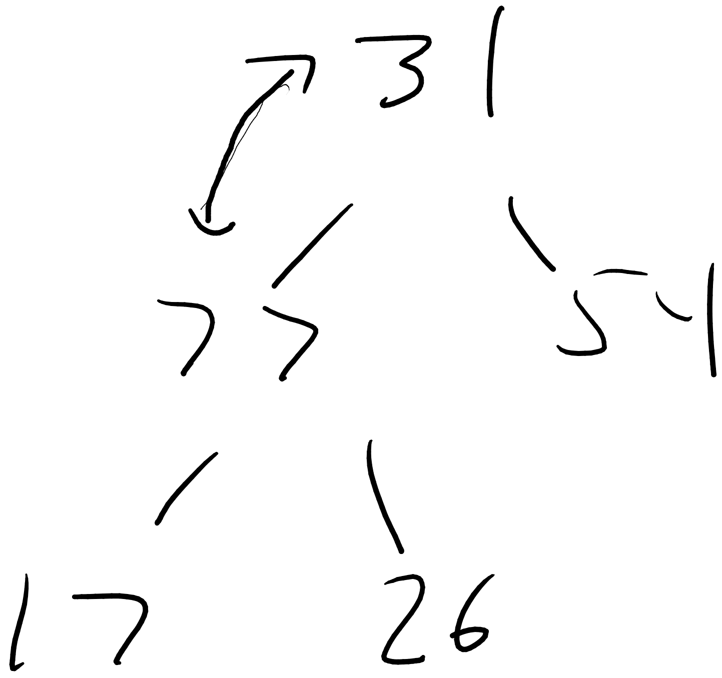
54 77 93 17 26 31



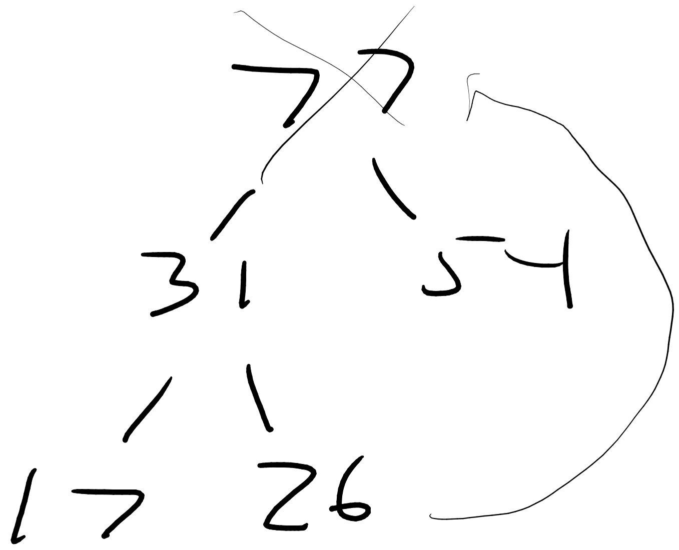
93 54 77 17 26 31

keep going if necessary

Remove one-by-one from heap (take out 93)



31 77 54 17 26 93

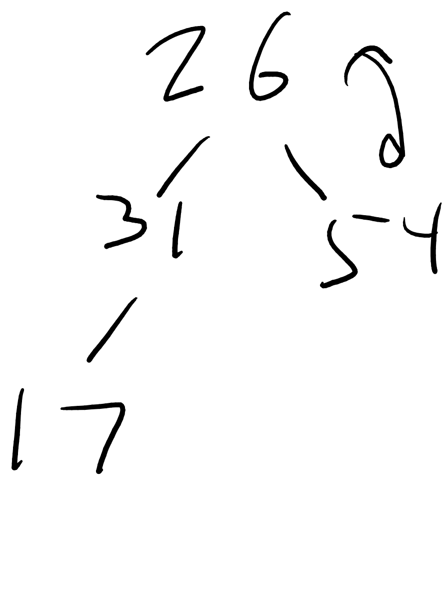


77 31 54 17 26 93

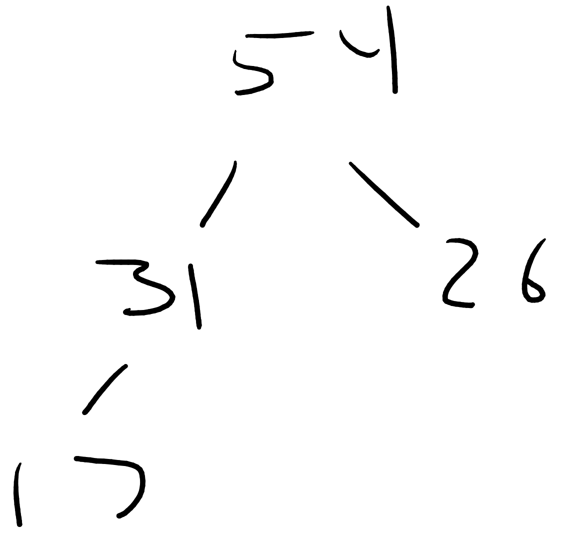
remaining heap

sorted!

Do it again Remove 77



26 31 54 17 77 93



$$\underbrace{54 \quad 31 \quad 26 \quad 17}_{\text{heap}} \qquad \underbrace{77 \quad 93}_{\text{done}}$$

To fix heap takes log n steps

We fix the heap n times

So ... O(n log n)

Why/what compared w/ mergesort/quicksort

Heapsort is O(n log n) worst case, just like mergesort. (No $O(n^2)$ like quicksort)

Doesn't require a second temp list like mergesort does (saves memory over mergesort).

But in practice, mergesort is a little faster than heapsort.

```
fun <T: Comparable<T>>
_quicksort(list: MutableList<T>,
left: Int, right: Int) {
    if (left < right) {
        val pivot = list[left]
        var up = left+1
        var down = right

        while (up < down) {
            while (up < right &&
_____________________________________ ) {
                up++
            }

            while (down > left &&
____________________________________) {
                down--
            }

            if (up < down) {
                swap(list,up,down)
            }
        }

        // Move pivot to the middle.
down is now at the rightmost spot
less
        // than or equal to pivot.
```

Fact: no sorting alg based on swapping things can ever be better than $O(n \log n)$ in the worst case!

Take CS 202.

```
        // than or equal to pivot.
        swap(list,left,down)

        _quicksort(list,
____________________,
________________________)

        _quicksort(list,
____________________,
________________________)
    }
}
```