COP3502 10/20/23

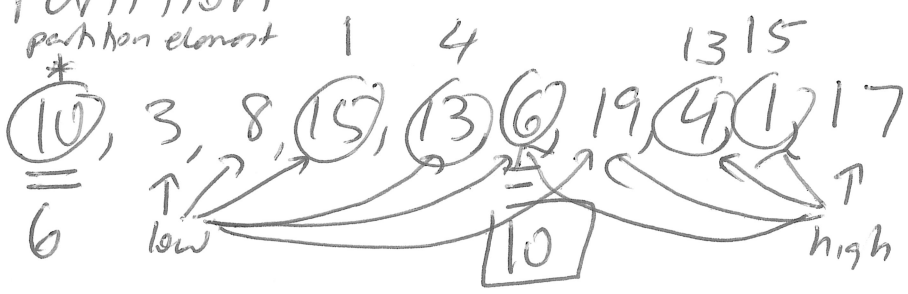
✓ (0) Candy

(1) Greatest Exam Solutions

(2) Quicksort

Partition

partition element



while (low <= high) {

(1) move low to right until it hits an item that's > a[part]

(2) move high to left until it hits an item <= a[part].

(3) if (low < high) swap a[low], a[high]

}

swap a[part], a[high]

6, 3, 8, 1, 4, [10], 19, 13, 15, 17

< 10 | sorted position | ≥ 10

Sort | Sort

```
void qsort(int * arr, int low, int high) {
    if (low >= high) return;
    int part = partition(arr, low, high);
    qsort(arr, low, part-1);
    qsort(arr, part+1, high);
}
```

~~Q(4,5)~~
~~Q(1,2)~~ ~~Q(6,8)~~
~~Q(1,5)~~ ~~Q(7,8)~~
~~Q(1,8)~~
~~Q(0,8)~~

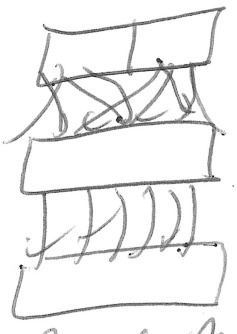
In practice, on average quick sort beats merge sort.

Seems strange because QS can split unevenly.

Worst Case $T(n) = T(n-1) + O(n) \Rightarrow O(n^2)$ (bad case part)

Best Case $T(n) = 2T(\frac{n}{2}) + O(n) \Rightarrow O(n \log n)$

Ans: Merge Sort requires extra mem + copy back step



Quick Sort is IN PLACE

Inefficiency of not splitting ideally isn't as "bad" + extra overhead of Merge.

Mitigate bad cases in a few ways:
(1) median of 3 or 5 partition.

AVG RUN TIME QSORT $O(n \log n)$