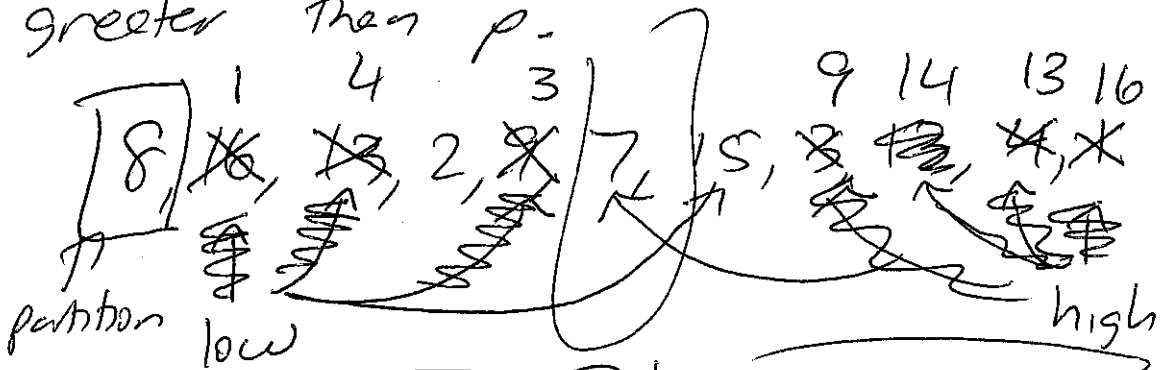
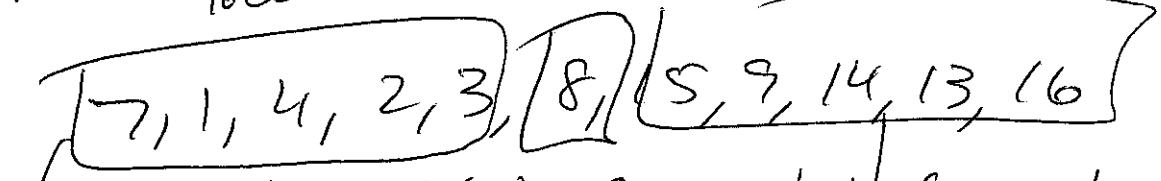
QUICK SORT + QUICK SELECT

Partition

Given a set of #s, split them into 2 groups values less than some value p, and values greater than p.





Run-time O(n) for a list of n items.

recursively quick sort

recursively quick sort

```
quicksort(int *array, int low, int high) {
    if (low < high) {
        int partIndex = partition(array, low, high);
        quicksort(array, low, partIndex-1);
        quicksort(array, partIndex+1, high);
    }
}
```

$T(n) = cn^2$

$T(40{,}000) = c\,40{,}000^2 = 13\,\text{sec}$

$c = \frac{13\,\text{sec}}{40{,}000^2}$

$T(4\times10^6)$

$= c \cdot (4\times10^6)^2$

$= \frac{13 \cdot 4^2 \times 10^{12}}{4^2 \times 10^8}\ 10^4 = 13\times10^4\,\text{sec}$

$\frac{130000}{60} = \text{min}$

2166 min

$6\overline{)13000}$

10

40

$\frac{2166}{60} = 36\,\text{hrs}$

---

(1) Run-time analysis
- Best
- Worst
- Approx of Avg

2 (3) Median of 3 or 5 idea

(3) In practice

Towers
$T(n) = 2T(n-1) + 1$

Best Case

let $T(n)$ = QS run time in best case

$$T(n) = O(n) + T\left(\frac{n}{2}\right) + T\left(\frac{n}{2}\right)$$

$T(n) = T\left(\frac{n}{2}\right) + 1$ ← bin search

- $O(n)$ ↑ Partition
- $T\left(\frac{n}{2}\right)$ ↑ left side qs on perfect split
- $T\left(\frac{n}{2}\right)$ ↑ right side qs on perfect split

$$T(n) = 2T\left(\frac{n}{2}\right) + O(n) \rightarrow O(n \lg n)$$

Merge
$T(n) = 2T\left(\frac{n}{2}\right) + O(n)$
↑ MS Left ↑ Merge
MS Right runtime

In the best case in practice, quick sort BEATS merge sort because it doesn't have to copy stuff back and forth!

Worst Case

Partition
0 elems [P] n-1 elem

$$T(n) = O(n) + T(0) + T(n-1)$$

- $O(n)$ ↑ Partition
- $T(0)$ ↑ empty side
- $T(n-1)$ ↑ big side

$$T(n) = T(n-1) + O(n)$$
$$= T(n-2) + (n-1) + n$$
$$= T(n-3) + (n-2) + (n-1) + n$$
$$= \sum_{i=1}^{n} i = \frac{n(n+1)}{2} = O(n^2)$$

Avg Case Intuition

$$T(n) = O(n) + T\left(\frac{n}{4}\right) + T\left(\frac{3n}{4}\right)$$

Part — shorter — larger

avg split

$\frac{n}{4}$ Left — $\frac{3n}{4}$ Right

$$T(n) \leq 2T\left(\frac{3n}{4}\right) + O(n)$$

$$T(n) = 2T\left(\frac{3n}{4}\right) + n$$

$$= 2\left[2T\left(\frac{9n}{16}\right) + \frac{3n}{4}\right] + n \qquad \frac{5}{2} + \frac{9}{4}$$

$$= 4T\left(\frac{9n}{16}\right) + n\left(1 + \frac{3}{2}\right)$$

$$= 4\left(2T\left(\frac{27n}{64}\right) + \frac{9n}{16}\right) + n\frac{5}{2}$$

$$= 8T\left(\frac{27n}{64}\right) + \frac{19}{4}n$$

2^k ↑ — $\left(\frac{3}{4}\right)^k n$ ↓ — harder but it's n or $n \lg n$

$$= O(n \lg n)$$

Even in the avg case, quick sort beats merge sort

How to sort quickly (Comparison using QS)

(1) When array size is large, use the median of 3 or 5 to pick the partition element.

(2) Insertion Sort, Selection works better for arrays ≤ 40 elements

```
if (low < high)
    :
    :
```

```
if (high - low < 40)
    insertionSort(array, low, high);
else {
    :
    :
}
```