COP 3502 - 3/16/22

1. Quick Sort
2. Comparison btw Quick / Merge
3. Avg Case Run Time Analysis QuickSort
4. Q3 Review

Quick Sort works based on the Partition

Partition

8, 3, 11, 6, 1, 9, 5, 4

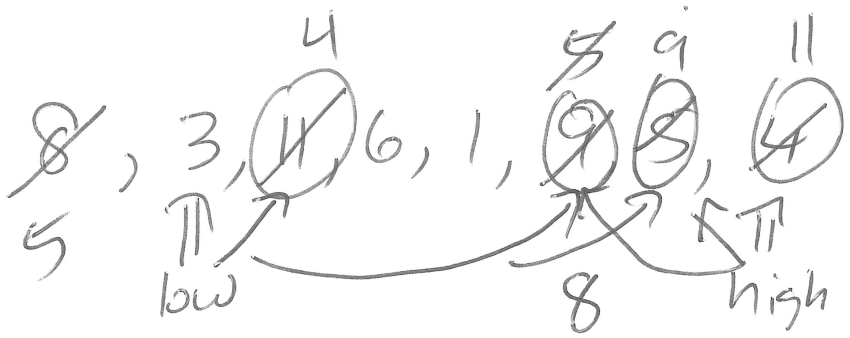
↑ Partition Element

Visually

3 6 1 5 4 [8] 9 11

This idea while easier to understand visually, uses a new array that is unnecessary

⇒ In-Place Partition



In loop
1. Move low to right until it finds an out of place item.
2. Move high to left until it finds an out of place item
3. As long as low < high, swap the values in those locations, else stop.

4. Swap index partition w/ index high.

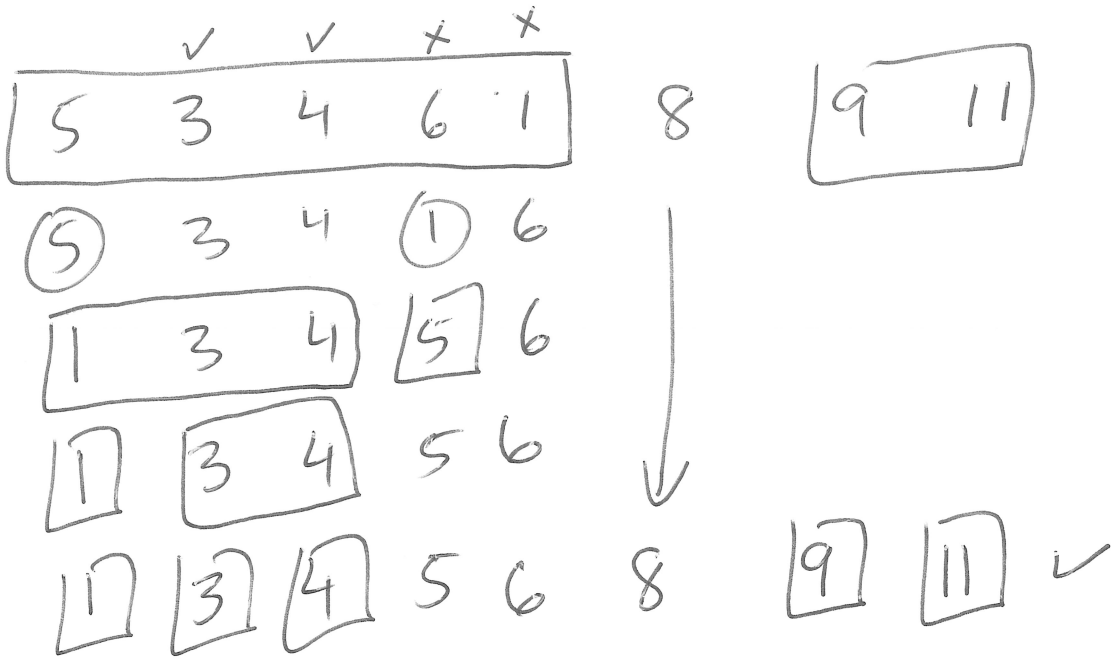
5 3 4 6 1 [8] 9 11

After one partition, partition element is in its correct sorted order.
Everything left ⇒ less than it
Everything right ⇒ ≥ to it.

→ SORT LEFT SIDE
→ SORT RIGHT SIDE

```
QuickSort (array, low, high) {
    if (high ≤ low) return;
    mid = partition (array, low, high)
    QuickSort (array, low, mid-1);
    QuickSort (array, mid+1, high);
}
```

Trace



Run-Time

Best Case array always splits half half

So recurrence relation $T(n) = T(\frac{n}{2}) + T(\frac{n}{2}) + O(n)$

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

$O(n \lg n)$

Worst Case

Always get 0 / n-1 split

$n + (n-1) + (n-2) + \ldots + 1 = \frac{n(n+1)}{2}$

$= O(n^2)$

(n: 1st part; (n-1): 2nd part; (n-2): 3rd part)

Questions

Why use Quick Sort ever?

How to reduce probability of worst case behavior of Quick Sort?

In practice, can we combine these ideas to create a sort that's even a bit faster in practice?

One difference: Quick Sort in place
Merge Sort is NOT.
↳ alloc of a new array for each merge, 2 copies

In practice, on average, Quick Sort is faster because it's done in place + you don't need new memory to copy stuff into and back.

Reduce Worst Case Behavior

1. Choose random elem partition
2. Choose 3 random elems + select their median to be the partition
3. Choose 5 random elems...

In practice to sort fast

1. Large array median 3 or 5 Quick Sort
2. Get down to about 40 elems ⟹ Insertion Sort

(40 varies)

Merge Sort is STABLE.
Quick Sort is NOT.

Stable Sort: if $a[i] == a[j]$ in the original array with $i < j$, then in the sorted array $a[i]$ will be stored in an index less than $a[j]$.

	Best	Avg	Worst	In Place	Stable
Merge	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$	NO	YES
Quick	$O(n \log n)$	$O(n \log n)$	$O(n^2)$	YES	NO

Skipping Avg Case Analysis of Quick Sort

QUIZ ITEMS

Big Oh Runtimes
Analyzing Code Segments
Predicting runtimes
Sums
* Recurrence Relations
Still know Master Thm.
Bubble Sort
Insertion Sort
Selection Sort
Merge Sort
Quick Sort