6s: QuickSort (divide and conquer/randomized)

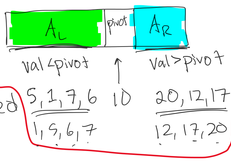
Friday, March 19, 2021
2:47 PM

QuickSort

Input: Array A of unique integers
Output: sorted A.

- If $|A| = 1$: Return A } Base case
- pivot = Randomly chosen elt } Preprocessing
- Partition (A, pivot)
- QuickSort (A_L)
- QuickSort (A_R) } Recursive calls on ~ equal sized sets

Effect of Partition:



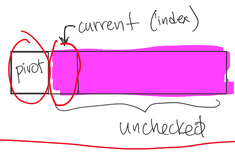
val < pivot | val > pivot

5,1,7,6 10 20,12,17
1,5,6,7 12,17,20

Partition (A, pivot)

← value of pivot

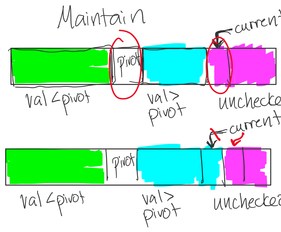
- Move pivot to start (swap first element with pivot)
- While current ≤ |A|



?

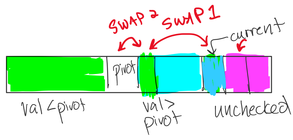
comparison → If A[current] > pivot

Maintain



O(1)

→ Else (A[current] < pivot)



Key Pts

- Partition is doing most of the work in QuickSort
- Runtime of partion scales like the # of comparisons

Idea: To determine runtime of Quicksort, count comparisons over the whole alg.

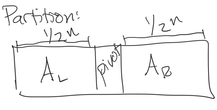
Q How many comparisons are done by partition on an array of size n?

A) $O(\sqrt{n})$ B) $O(n)$ C) $O(n \log n)$ D) $O(n^2)$

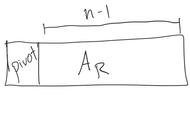
exactly n-1 comparisons

Effect of Partition:

Lucky:



Unlucky:



1. Suppose you got very lucky and pivot is always chosen to be median of A, every time partition is called.
   - Create recurrence relation for runtime of Quicksort
   - Solve

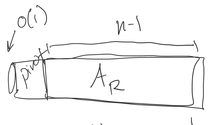
2. Suppose you got very unlucky and pivot is always chosen to be minimum of A, every time partition is called.
   - Create recurrence relation for runtime of Quicksort
   - Solve

1.

$$T(n) = \begin{cases} O(1), & n = 1 \\ 2T(\frac{n}{2}) + O(n) \end{cases}$$

$O(n \log n)$

2.



$$T_n = \begin{cases} O(1), & n = 1 \\ T(n-1) + O(n) \end{cases}$$

$$T(n) = T(n-1) + O(n)$$
$$= T(n-2) + O(n-1) + O(n)$$
$$= T(n-3) + O(n-2) + O(n-1) + O(n)$$
$$\vdots$$
$$= O(1) + O(2) + O(3) \cdots O(n)$$
$$= O\left(\sum_{i=1}^{n} i\right) = O(n^2)$$

Lucky	Unlucky
$O(n \log n)$	$O(n^2)$

Which is likely?
Which is average?