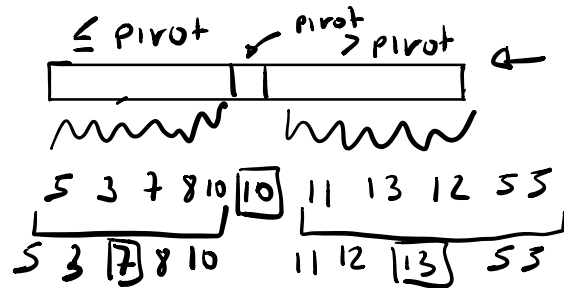


## Quick Sort Corman 7.1, 7.2, 7.3

Another sorting algorithm based on D&C approach

Merge Sort  $\begin{cases} \text{Divide} & \text{Constant time} \\ \text{Combine} & \text{Linear time} \end{cases}$

Quick Sort  $\begin{cases} \text{Divide} & \text{Linear time} \\ \text{Combine} & \text{Constant time} \end{cases}$



**Divide** Partition the array  $A[p \dots r]$  in two (possibly empty) subarrays

$A[p \dots q-1]$  and  $A[q+1, \dots r]$

such that

- each element in  $A[p \dots q-1]$

$\leq A[q]$

- each element in  $A[q+1, \dots, r]$

$> A[q]$

**Conquer** Sort  $A[p \dots q-1]$  and  $A[q+1 \dots r]$  recursively

**Combine** do nothing

How to choose the pivot?

it's the element in position  $A[p]$

$A_{\text{before}}$   
1 5 3 2 3 4  
p r

$A_{\text{after}}$   
1 2 3 5  
p q r

**QuickSort** ( $A, p, r$ )

if  $p < r$

$q = \text{Partition}(A, p, r)$

QuickSort ( $A, p, q-1$ )

QuickSort ( $A, q+1, r$ )

**Partitioning the array**

Partition ( $A, p, r$ )

$x = A[r]$

$i = p - 1$

for  $j = p$  to  $r - 1$

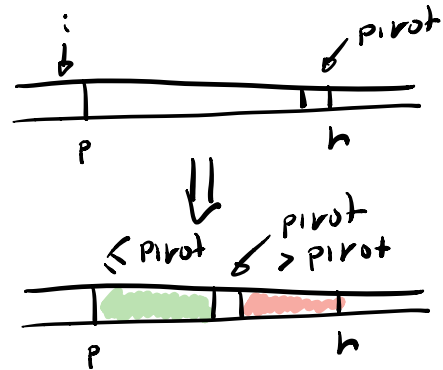
if  $A[j] \leq x$

$i = i + 1$

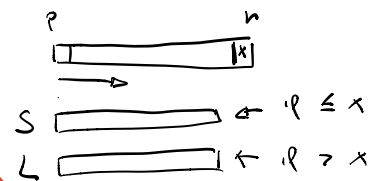
swap ( $A[i], A[j]$ )

swap ( $A[i + 1], A[r]$ )

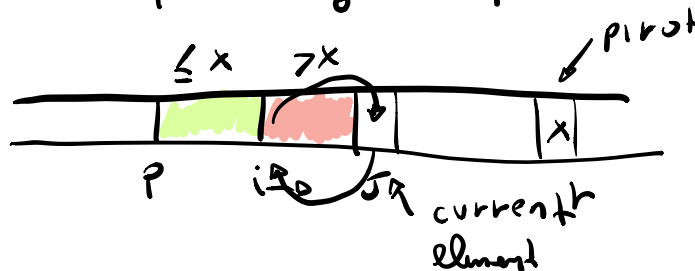
return  $i + 1$



if not inplace, easy



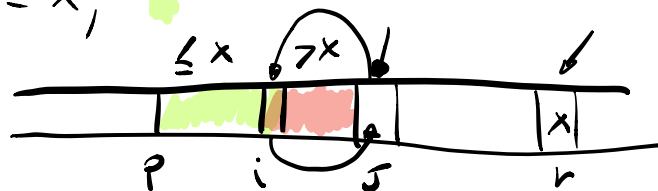
Partition keeps two indexes  $i$  and  $j$  and following loop invariant



$j$ -th iteration

if  $A[j] > x$ , ■ do nothing

if  $A[j] \leq x$ , ■



i	j
2	8

i	j
2	8

i	j
2	8

i	j
2	8

i	j
2	8

i	j
2	8

i	j
2	8

i	j
2	8

i	j
2	8

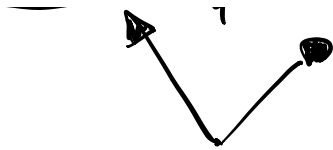
i	j
2	8

i	j
2	8

Partition (A, p, r)

- 1  $x = A[r]$
- 2  $i = p - 1$
- 3 for  $j = p$  to  $r - 1$
- 4 if  $A[j] \leq x$
- 5  $i = i + 1$
- 6 swap  $(A[i], A[j])$
- 7 swap  $(A[i + 1], A[r])$
- 8 return  $i + 1$

Stable? NO!!



they are in their  
correct relative order

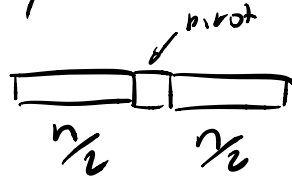
Time complexity of Partition  $\Theta(n)$

- Stable! No

- Inplace! Yes

## Analysis of Quick Sort

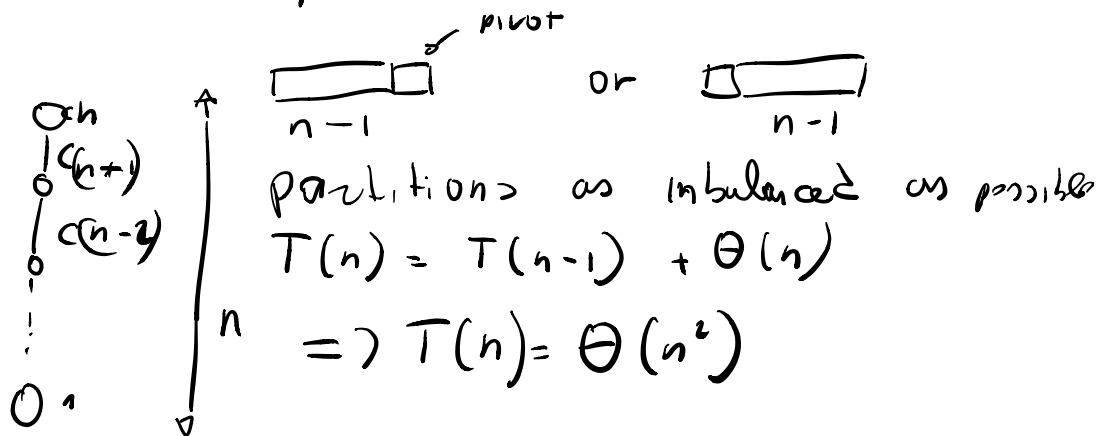
Best Case: Every time Pivot is such that



Partitions are balanced

$$T(n) = 2T\left(\frac{n}{2}\right) + \theta(n) \quad (\text{as MS})$$
$$= \theta(n \log n)$$

Worst Case: Every time pivot is such that



Balanced (or almost balanced) subproblem  
Good

unbalanced BAD

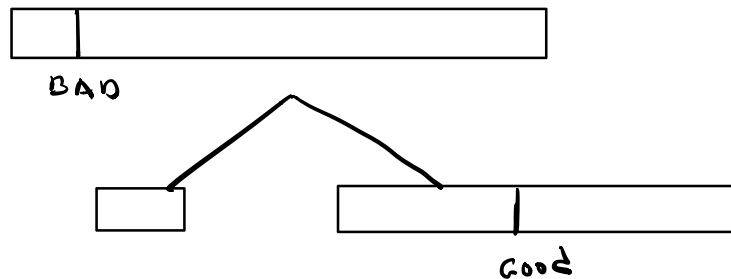
## Average Case analysis (just intuition)

Choose pivot uniformly at random



$$P(\text{GOOD}) = P(\text{BAD}) = \frac{1}{2}$$

Benefit of a good pivot  $\gg$   
penalty of a bad pivot



by selecting a bad pivot  
we just waste a  
recursive call

QS expected running time is  $O(n \log n)$