## QuickSort Algorithm

Wednesday, May 20, 2020 12:06 PM

rt(A, low, high) { ow < high){		
Pivot = partition(A,low,high)	//O(N)	
QuickSort(A,low,pivot-1)	//T(k)	
QuickSort(A, pivot+1, high)	//T(n-k)	

## Partition Method:

Alį	Algorithm		
PA	RTITION $(A, p, q) \triangleright A[p \dots q]$		
1	$x \leftarrow A[p] \qquad \rhd \text{ pivot} = A[p]$		
2	$i \leftarrow p$		
3	for $j \leftarrow p+1$ to $q$		
4	do if $A[j] \leq x$		
5	then $i \leftarrow i + 1$		
6	$exchange \ \mathcal{A}[i] \leftrightarrow \mathcal{A}[j]$		
7	exchange $A[p] \leftrightarrow A[i]$		
8	return i		



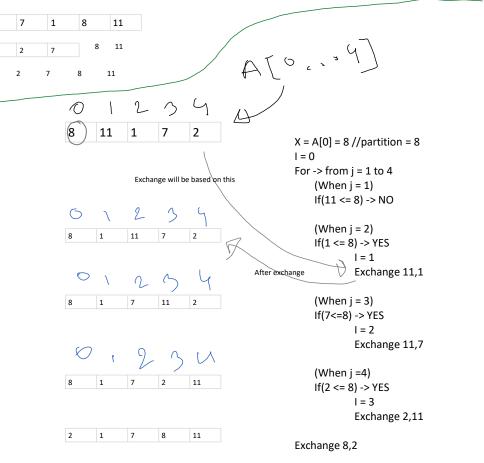
2

1

1



Let's assume that 8 is our pivot. Then the array should be like this



## Recurrence Formula:

T(N) = T(k) + T(n-k) + O(n)

Here, k symbolizes the position of the pivot.

If pivot is in the middle  $\rightarrow$  k = n/2

T(N) = T(n/2) + T(n-n/2) + O(n)= T(n/2) + T(n/2) + O(n)

This would have a time complexity nlog<sub>2</sub>(n) -> BEST CASE

QuickSort has two types of Time Complexity. If pivot is in the middle, then that is the best case. If pivot is in the start or end index, then that is the worst case.

QuickSort Worst Case