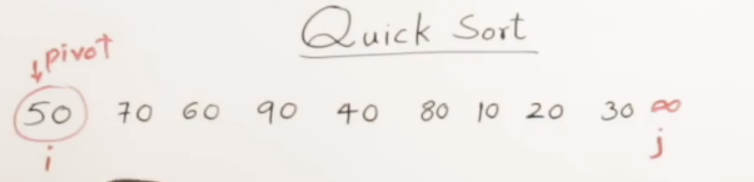
# **Quick Sort | Selection Exchange Sort | Partition Exchange Sort**

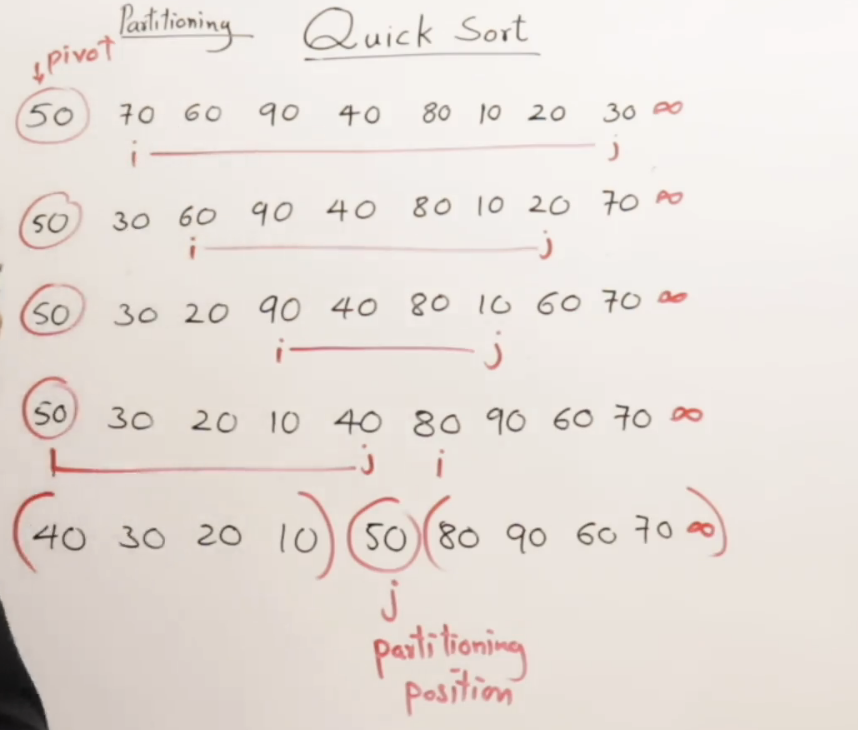
For Sorting, there must at-least 2 elements.  
If 1 element is present, then no need to sort.

Pivot is initialized to 1st element



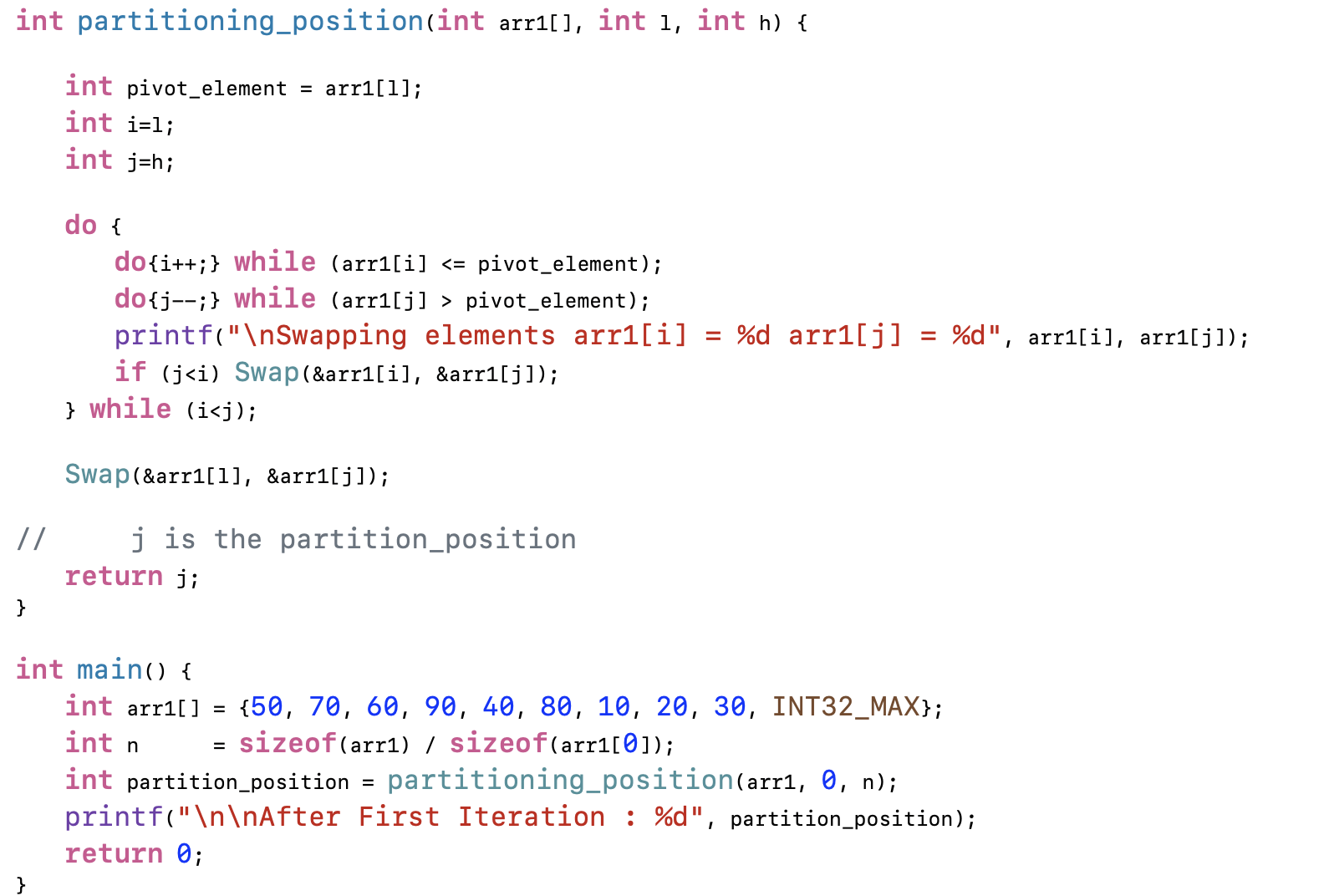
i will be moving for > pivot. Until i moves  
j will be moving for <= pivot. Until j moves

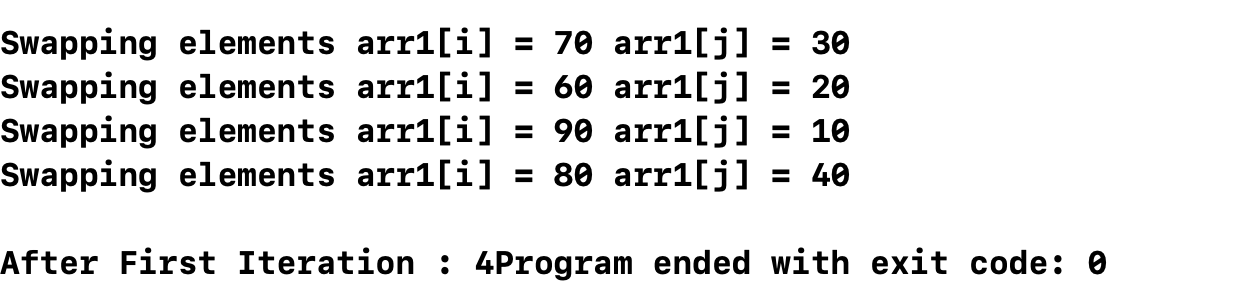
Then Swap(i, j)

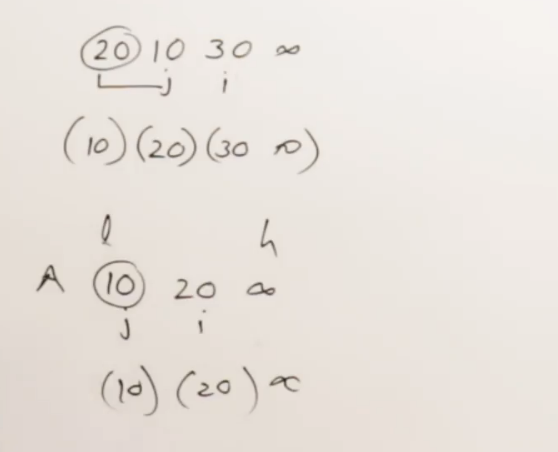


Perform Quick Sort on both LHS and RHS recursively.

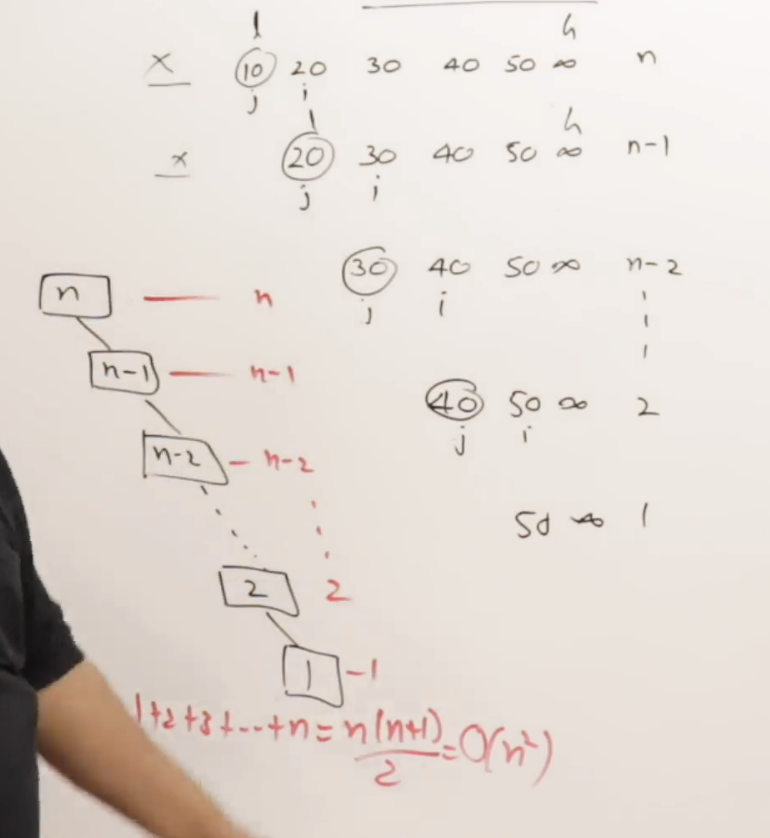
In RHS, there is infinity.  
In LHS, this sorted 50 act as infinity.

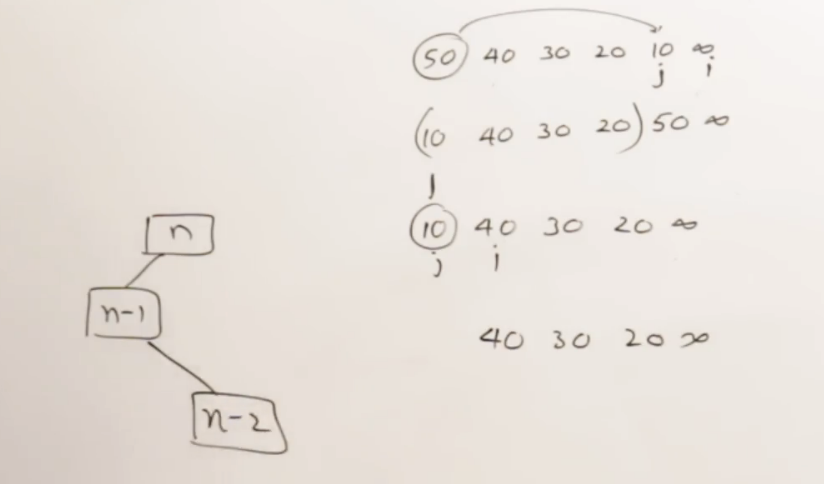




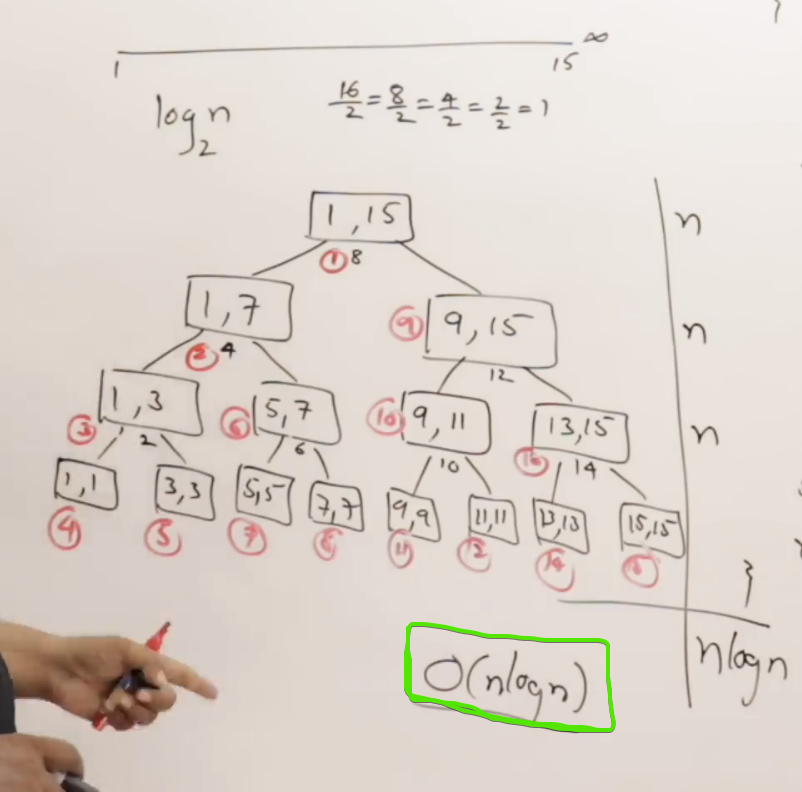


Pivot element swapped with j 🡪 Then it is an Sorted Array

Already Sorted in Ascending Order  


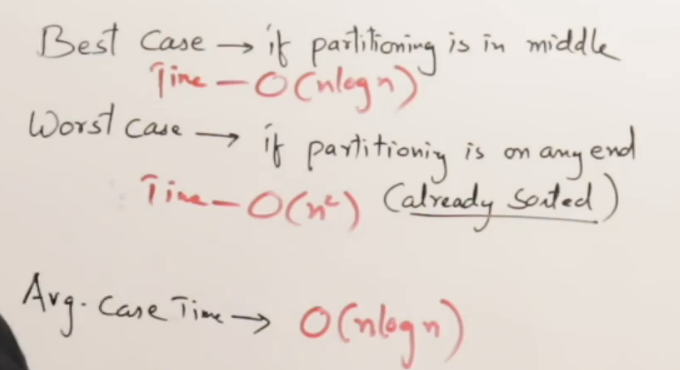
Already Sorted in Descending Order  


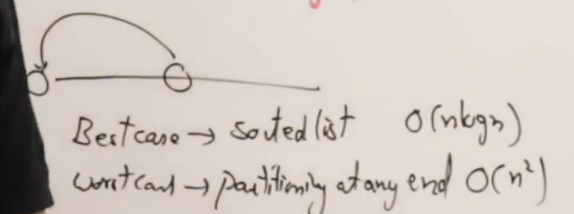
If partitioning takes place at left-most position (or) right-most position 🡪 O(n^2)

Assume the list is portioned at the middle.  


Successive Divisions 🡪 every time divided by 2 🡪 log base-2 (n)

First element is selected as pivot



Middle element is selected as pivot  


Tried to change the pivot element position, But also no improvement in time complexity.

Instead of selecting a first element (or) middle element as pivot. Selecting a random element as pivot 🡪 Randomized Quick Sort

Selection Sort 🡪 Select a position and find-out the element for that position.  
Quick Sort 🡪 Select a element and find-out the position for that element.