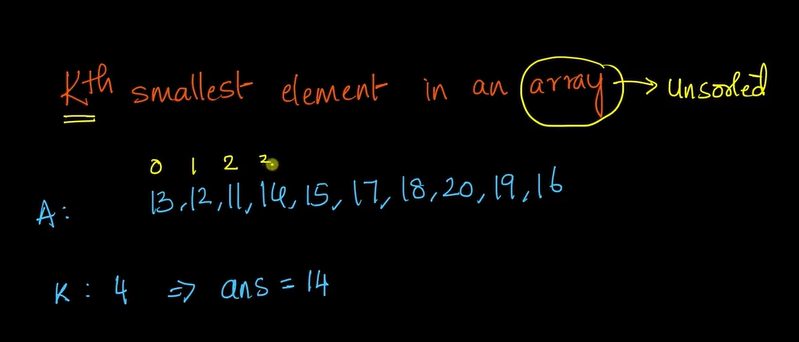
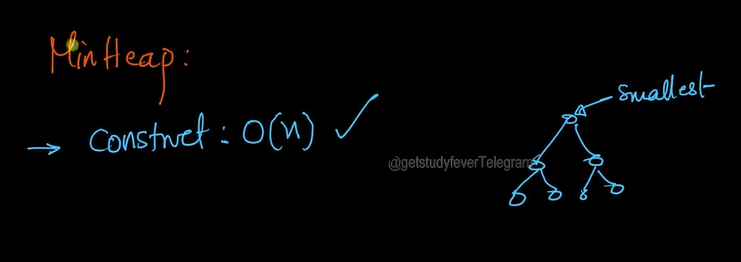
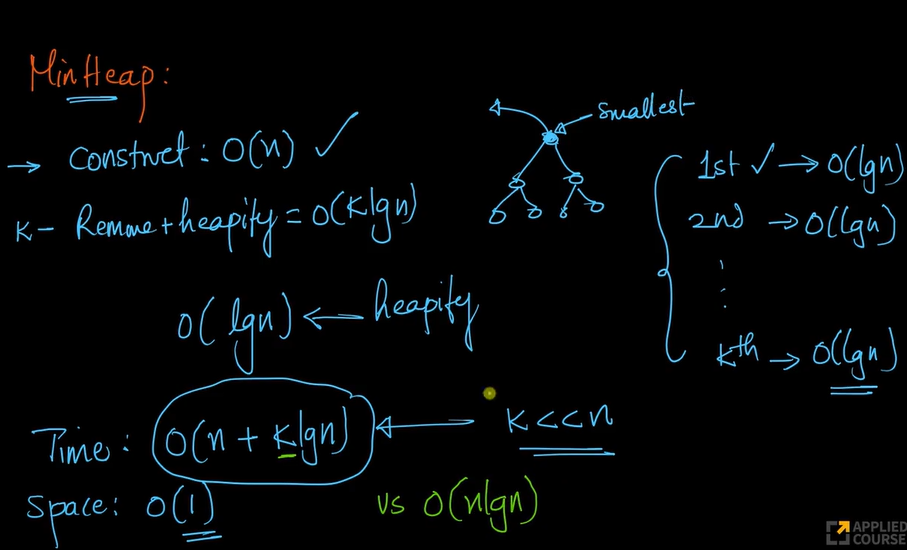
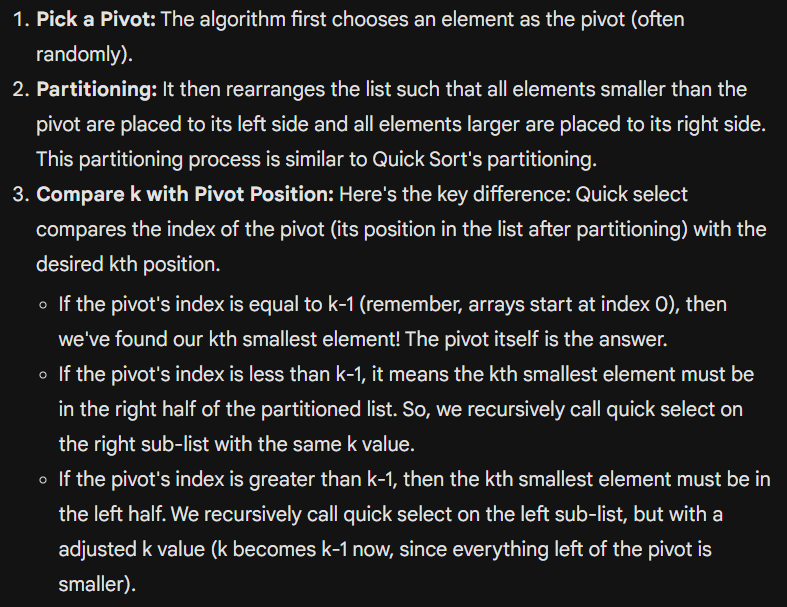
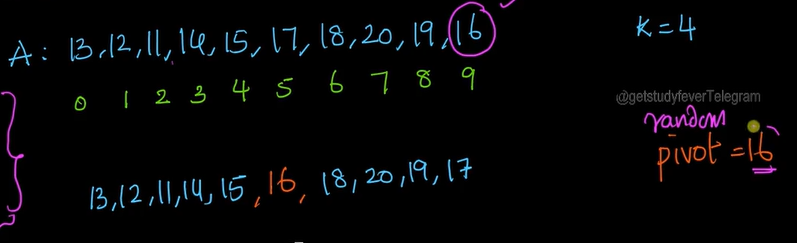
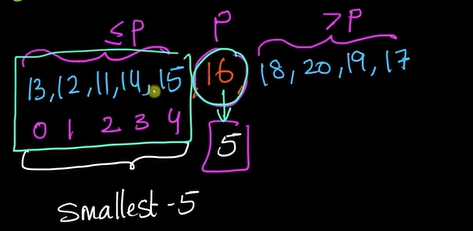
[215. Kth Largest Element in an Array](https://leetcode.com/problems/kth-largest-element-in-an-array/) - Similar problem on Leetcode



1. So here if we use any comparison sorting algo’s…then it will take around O(nlogn) time complexity
2. To solve this problem …without sorting..we are going to use heap and its properties
3. MinHeap
4. In MinHeap the root is always the smallest
5. ANd if we remove the smallest element …again we have to heapify and make the root element as smallest
6. So to get the fourth element..we’ll remove root element 4 times and we haepify again 4 times..which takes O(klogn) time complexity.
7. 
8. If k is smaller than n..then this approach has better time complexity than O(nlogn)
9. But if k = n..then time complexity is O(nlogn)..
10. Can we do better than this?
11. So here we’ll be using quick select algorithm to even reduce the time complexity
12. 
13. Lets assume we picked 16 as random pivot element 
14. Now after partitioning the left,pivot and right we get this
15. Now all the smallest 5 elements are on the left of pivot
16. We can completely ignore the right part and can now focus on left part …it is similar to the binary seach strategy
17. So if we implement this strategy…the avg TC will be O(n) = T(n)....