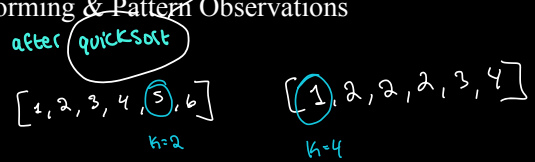
Problem: Given unsorted array, return kth largest element. Not kth distinct element.

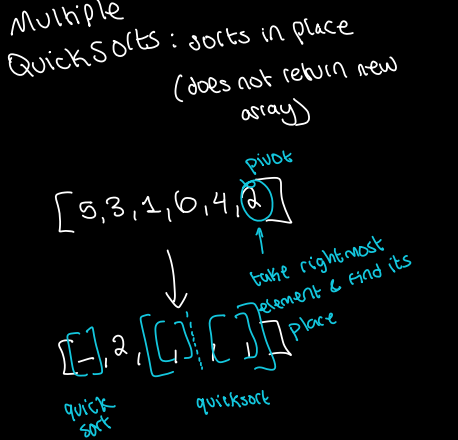
k = 2,
[5, 3, 1, 6, 4, 2] then [1, 2, 3, 4, 5, 6] ⇒ 5

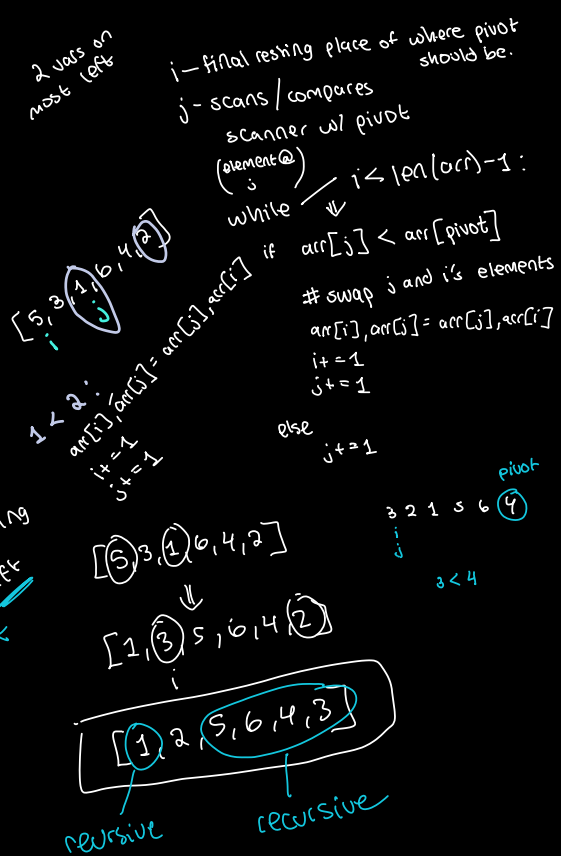
k = 4
[2, 3, 1, 2, 4, 2] then [1, 2, 2, 2, 3, 4] ⇒ 1

Intro:
- Verify Constraints
  - k larger than len(arr)? no
- Create Testcases
  - [5, 3, 1, 6, 4, 2] and k = 2 -> 5
  - [2, 3, 1, 2, 4, 2] and k = 4 -> 1
  - [3] and k = 1 -> 3

Brute Force:
- Brainstorming & Pattern Observations







- Pseudocode
- Write code
- Run through testcases
- Analyze time and space complexity
    Time: O(nlogn): how long does it take to find the partition index? looks thru n times and then splitting tree 2x
    Space: O(logn): split tree into two parts every time (2 recursive

<u>Optimal:</u>
- Brainstorming & Pattern Observations
    Using HOARE'S QUICKSELECT ALGORITHM
- Pseudocode
- Write code
- Run through testcases
- Analyze time and space complexity
    Time: partition function: O(n)
        quickSelect:
            best: O(n) + O(n/2)+ O(n/4) + … = O(n)
            worst: array sorted in descending order:
                the array does not get split in half, always split on one huge array
                O(n^2)
    Space: reduced to tail recursion O(1): we're not waiting on both the left and right branch recursion