1. Stability in quick sort - we choose not to swap if equal
First partitioning

10 15(x) 5 20 18 15(y)

Curr = 0
10

Curr = 1. No swap if equal to pivot.
10 15(x)

Curr =1
10 15(x)

10 5 15(x)

Curr = 2
10 5 15(x) 20 18

Final : 10 5 ll 15(y) ll 18 20 15(x) -> **Please not that in the class it was swapped incorrectly in the last step. We can see the out-of-order 15 values. This proves the sorting is not stable.**

2. Worst case scenario - time complexity is O(n^2)
Starting array: 10 9 8 7 6 5 4 3 2 1
Partitioning step 1: ___ ll 1 ll 9 8 7 6 5 4 3 2 10 -> 9 -> n - 1
Partitioning step 2: Left arr -> quicksort({ }) and Right arr -> quicksort({ 9, 8, 7 6 5 4 3 2 10 }) -> 8 -> n - 2
Step 3: 7 -> n - 3
Step 4: 6 -> n - 4
…
Last step: 1

Total #comparisons = 1 + 2 + …. + (n -2) + (n - 1)
= n(n-1) / 2 = n^2 / 2 - n / 2 = **O(n^2)**

EXERCISES

1. Come up with an example where this algorithm reverses the order of equal numbers

2. Implement median of 3 partitioning - median (arr[low], arr[(low + high) / 2], arr[high])

3. Implement kth() without recursion