1. Quicksort is a divide and conquer algorithm. It works by picking a pivot element from an array and then dividing
the remaining elements into two smaller arrays (sub-arrays) based on whether each element in greater or less than the
pivot element. After the creation of the two sub arrays, the arrays are then sorted in a recursive manner.
The worst-case scenario for the quicksort method occurs when the pivot element is chosen as the smallest or largest
element in the array. In this case, the algorithm will make n-1 recursive calls due to the unbalanced partitions, and each
sub-array will be of size n-1. Consequently there is a total of n-1 levels of recursion, and each sub-array will be of
size n-1. Thus, this leads to a worst-case complexity of O(n^2)

Let X(k) be the worst-case time complexity of a quicksort algorithm for an array of size k.

Then,

X(k) = X(k-1)+X(0)+O(k) where X(k-1) is the time taken to divide the array into smaller sub-arrays
X(0) is the time taken to divide the array with zero elements and,
O(k) is the time taken for the dividing/partioning step

By solving this, it is determined that the quicksort time complexity is O(k^2).

2. Array of 16 elements:
[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]

(i) Choose an element within the array to be your pivot.
We chose element 0 as our pivot element

(ii) Divide the array into 2 sub-arrays.
[0] and [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]

(iii) Once you have 2 sub-arrays, recursively sort them.