

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 is 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/partitioning step

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

(Chatgpt was used to help construct the above explanation but was significantly modified to provide accurate answers).

2. Array of 16 elements:

[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]

(i) iteration #1:

Partition: [0], [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]

(ii) iteration #2:

Pivot element = 2

Partition: [0,1], [2,3,4,5,6,7,8,9,10,11,12,13,14,15]

Continue this process until your last element is selected as the pivot. This results in partitions where one element is by itself on one side and the rest on the other side.

Unbalanced partitioning like above results in a worst-case time complexity of $O(k^2)$