Exercise – 4

1) Quicksort is a sorting algorithm that follows divide and conquer method. In the worst-case scenario, one pivot is chosen at each step. The chosen one might be the smallest or largest in the current subarray, The result relation can be expressed as follows:

$$T(n)=T(n-1)+T(0)+\Theta(n)$$

We can solve this recurrence relation for the worst-case time complexity of quicksort like,

$T(n)=T(n-1)+\Theta(n)$
$=(T(n-2)+\Theta(n-1))+\Theta(n)$
$=T(n-2)+2\Theta(n-1)+\Theta(n)$

Then after k iterations,

$T(n)=T(n-k)+k\Theta(n-(k-1))$

This continues until n=k, then the recurrence becomes,

$T(n)=T(0)+n\Theta(1)$
$T(n)=\Theta(n)$ //since, T(0) is a constant and $\Theta(1)$ is also a constant.

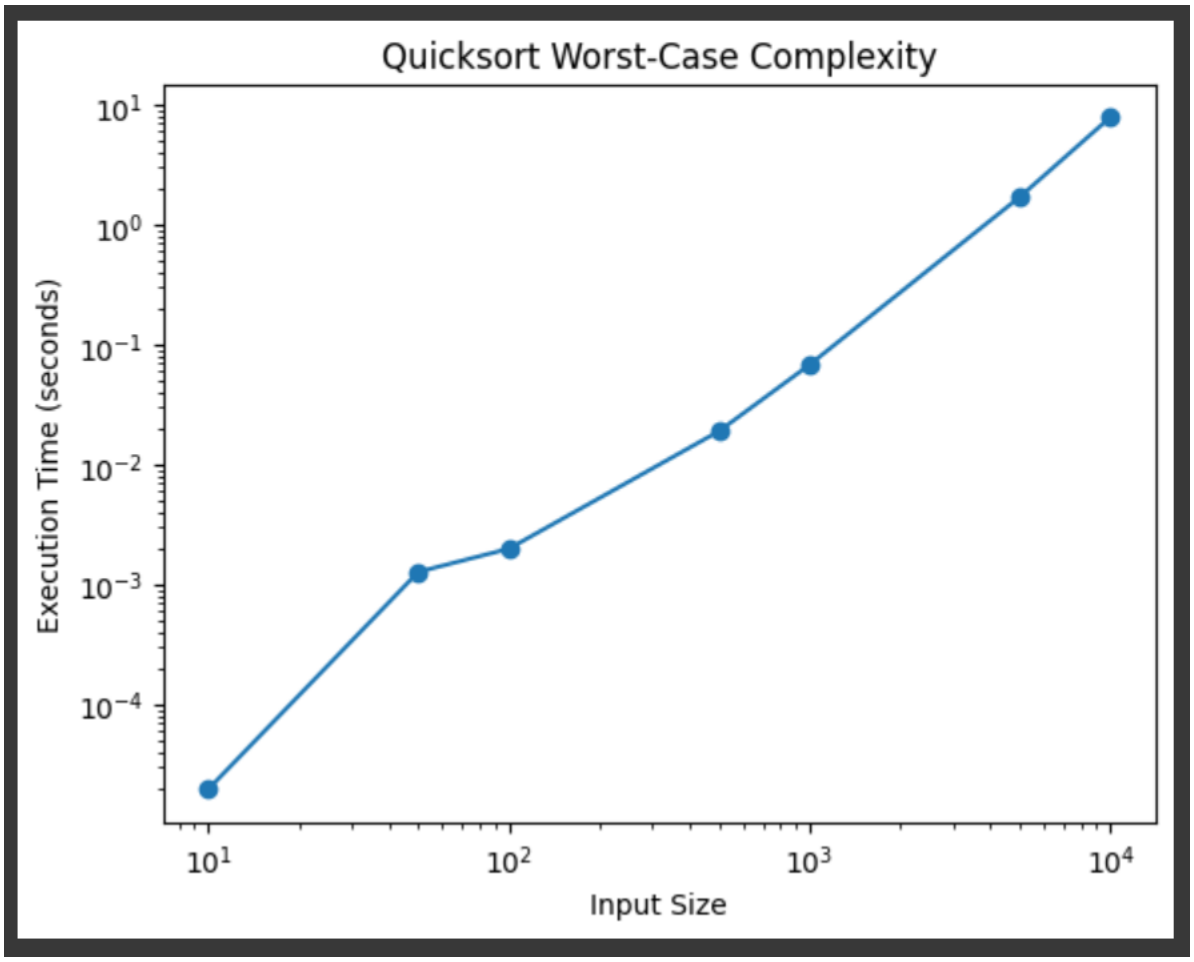
Thus, the worst-case time complexity of quicksort is O(n2)

2) Let's take a vector of 16 elements,

[16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1]

In the first step of quicksort we take the initial array and choose a pivot. Suppose the last one from the array which is 1. Then we get two arrays one consisting of [1] and another one consisting of the other elements. Then we keep repeating the same process every time and then by recursive calls we keep sorting the whole array one by one by choosing one pivot. This process continues until we reach subarrays of size 1, which are already sorted. The overall process involves many redundant portioning steps, leading to a worst-case time complexity of O(n2).

4)



The plot increases rapidly as the input size grows. This aligns with our complexity analysis of worst-case time complexity of quicksort algorithm.

The algorithm takes more time, as the input size is also increasing and it takes more steps for the comparisons and swaps. The bigger the array is the more time it will take to sort the array. So, yes we can say that the plot carries the same theme as our observation.