Q1:

1. each recursive call is n-1 so each occurrence can be showed as:

$$T(n) = T(n-1) + O(n)$$

We can further say:

$$T(n-1) = T(n-2) + O(n-1)$$

$$T(n-2) = \ldots\ldots + O(n-2)$$ → and so on

so $T(n)$ can be represented as the series:

$$T(n) = O(n) + O(n-1) + \ldots + O(1)$$

and the sum of this series is

$$t(n) = O\left(\frac{n(n+1)}{2}\right)$$ → taken from math stack exchange

$$= O(n^2)$$

Q3:

This program runs on a number of differing size arrays which all incur a worst case scenario for quick sort(that being already sorted arrays).

Q4:

As you can see from the graph below as the number of inputs increase the time taken starts to go up quadratically which fits the previous formula of O(n*n) complexity

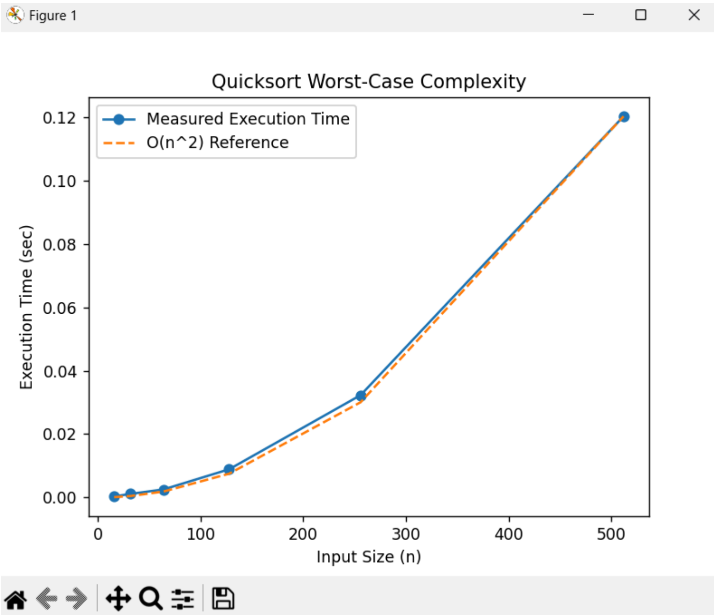
Figure 1
Quicksort Worst-Case Complexity
Measured Execution Time
O(n^2) Reference
Execution Time (sec)
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
100
200
300
400
500
Input Size (n)