1. Derive the formula for worst-case complexity

Quicksort's worst-case time complexity occurs when the pivot selection leads to the most unbalanced partitions. This happens when the pivot is either the smallest or largest element in the partition, resulting in one subarray of size (n-1) and the other subarray of size 0.

Recurrence relation:

In the worst case, the partitioning step takes (O(n)) time, and the algorithm recursively processes a subarray of size (n-1). This gives the recurrence relation:

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

Expanding the recurrence:
T(n) = T(n-1) + n

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

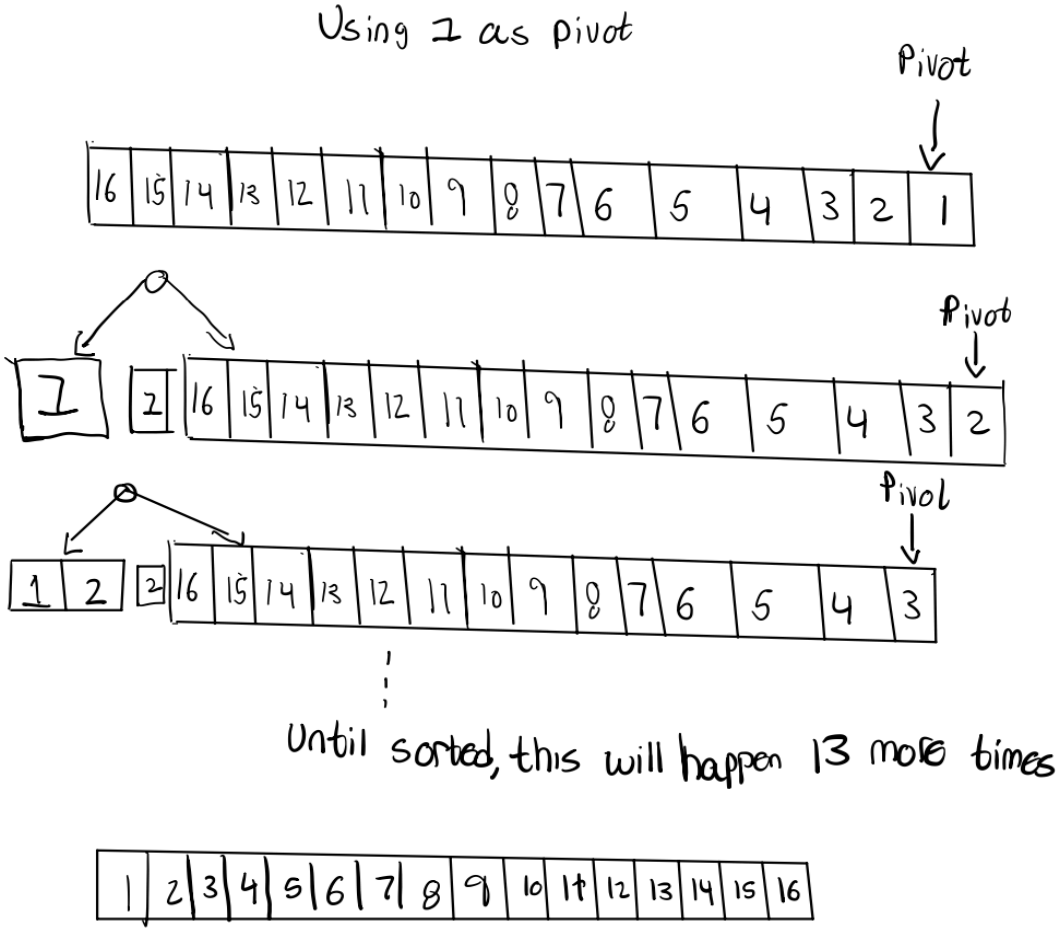
T(n) = T(n-3) + (n-2) + (n-1) + n

T(n) = T(0) + 1 + 2 + ….. + n

The sum of the first (n) integers is $\frac{n(n+1)}{2}$ so: T(n) = $O(n^2)$

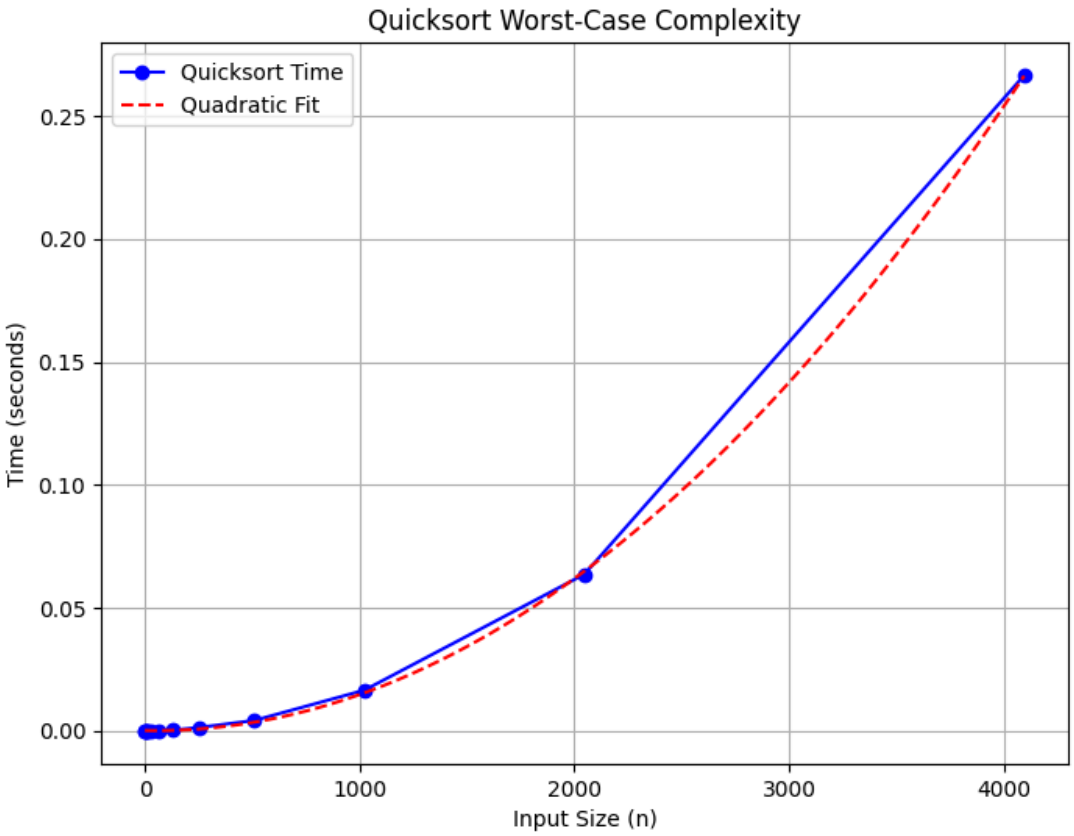
Thus, the worst-case time complexity of quicksort is T(n) = $O(n^2)$

2. Vector of 16 elements that incurs worst-case complexity



At each step, the pivot is the smallest element, resulting in one empty partition and one partition of size (n-1). This continues until the array is fully sorted.

4. Plot the results and discuss



The plot illustrates that the running time increases quadratically with input size, aligning with the $O(n^2)$ worst-case complexity. The quadratic fit closely follows the observed running times, reinforcing the theoretical analysis. This confirms that Quicksort performs inefficiently on already-sorted