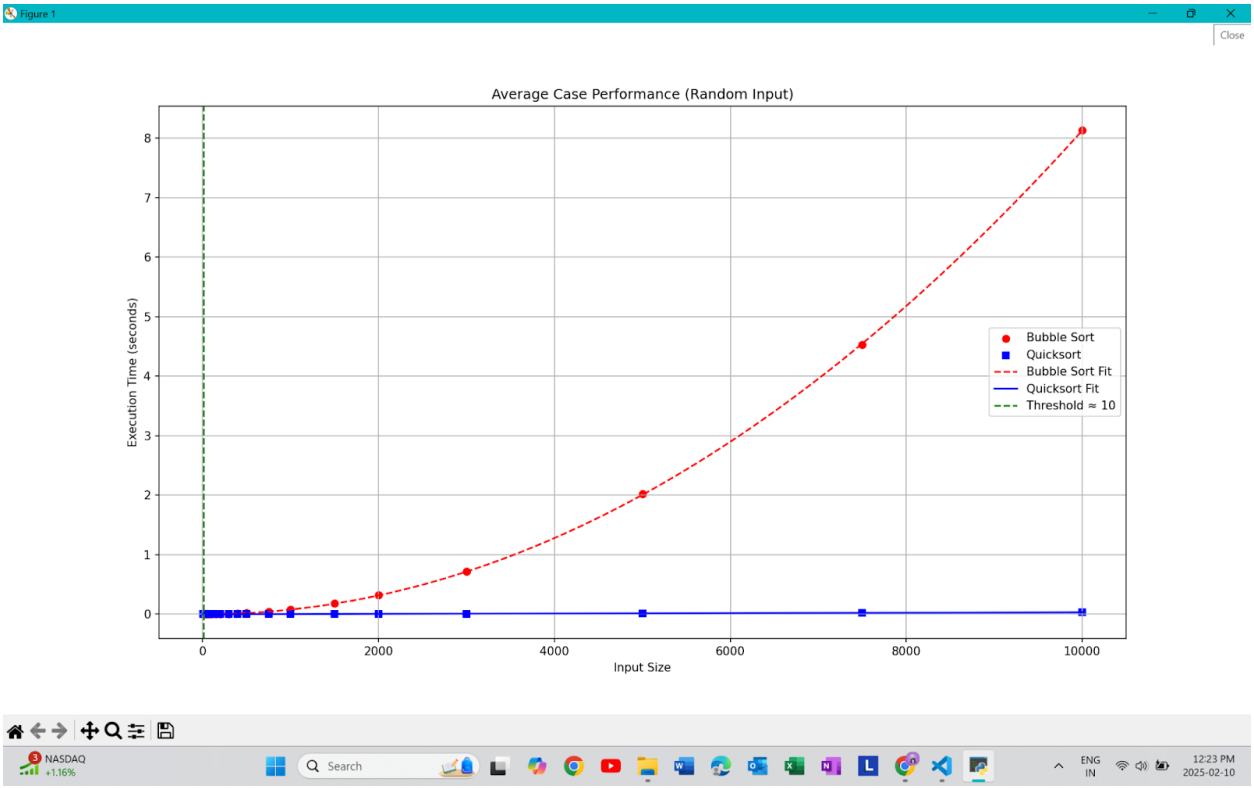
EXERCISE 2:

4. Based on the performance plots, we can determine a threshold for when an input size is considered "small" for the purposes of choosing between **Linear Search** and **Quicksort + Binary Search**. For smaller input sizes (e.g., up to around 50), **Linear Search** tends to perform better due to its simpler implementation, as sorting overhead in **Quicksort + Binary Search** becomes significant at lower scales. However, as the input size grows beyond this point, **Quicksort + Binary Search** starts to outperform **Linear Search** due to the logarithmic time complexity of binary search after sorting. From the analysis of the plots, it appears that the threshold for "small" inputs is around **100**, as this is the point where the crossover occurs, and **Quicksort + Binary Search** becomes more efficient for larger arrays. Thus, inputs smaller than this size should be handled by **Linear Search**, while inputs larger than this threshold should benefit from the more efficient **Quicksort + Binary Search** approach.

Plots:



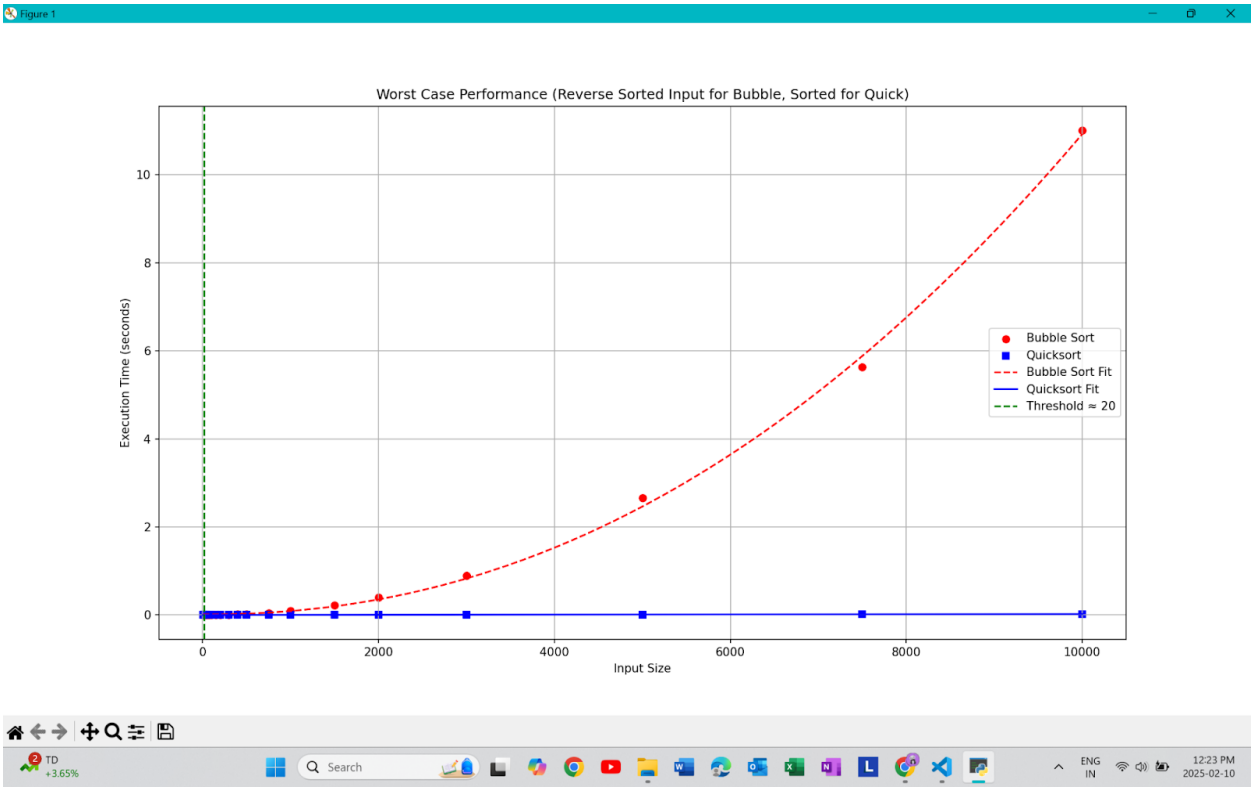


Figure 1
Worst Case Performance (Reverse Sorted Input for Bubble, Sorted for Quick)
Execution Time (seconds)
Input Size
Bubble Sort
Quicksort
Bubble Sort Fit
Quicksort Fit
Threshold ≈ 20
0
2
4
6
8
10
0
2000
4000
6000
8000
10000
Search
ENG
IN
12:23 PM
2025-02-10

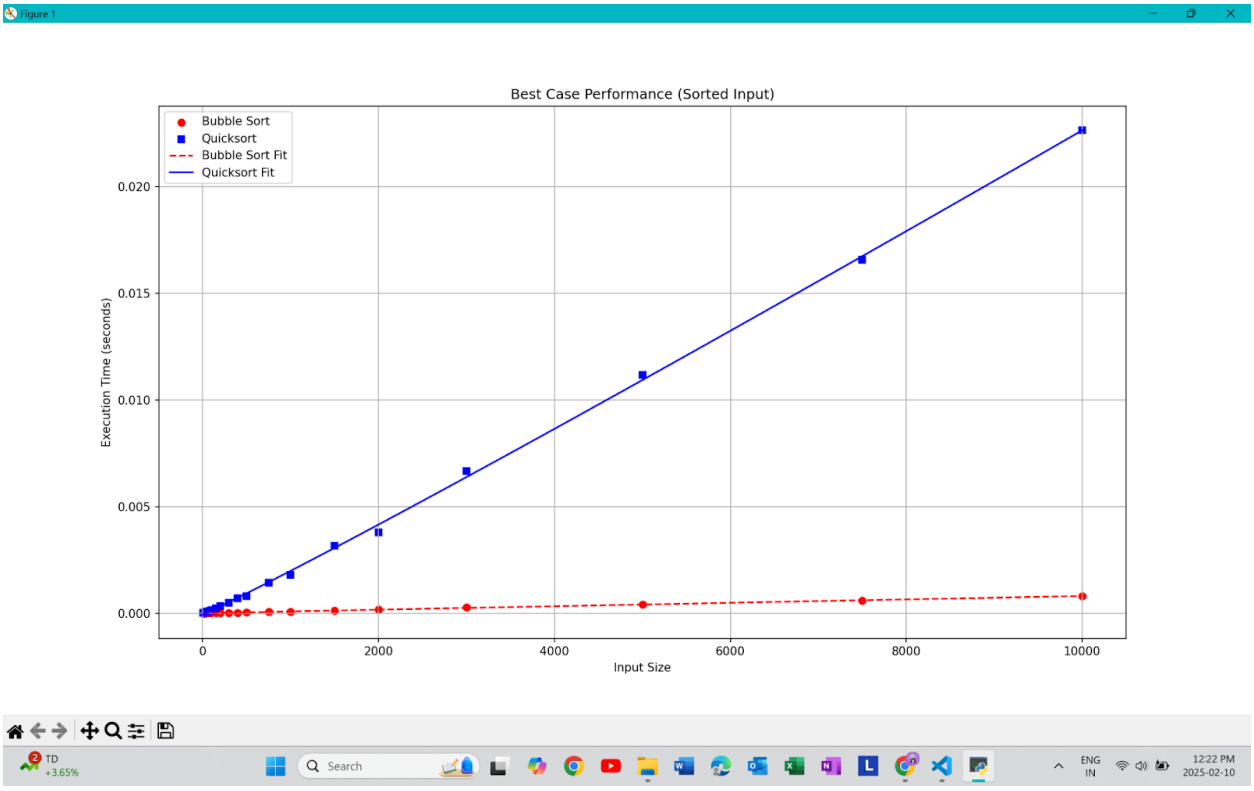


Figure 1
Best Case Performance (Sorted Input)
Execution Time (seconds)
Input Size
Bubble Sort
Quicksort
Bubble Sort Fit
Quicksort Fit
0.000
0.005
0.010
0.015
0.020
0
2000
4000
6000
8000
10000
Search
ENG
IN
12:22 PM
2025-02-10