Search Test Lab Report

Names: Kejian Tong

1. Linear Search

We know from class that the theoretical time complexity of linear search over *unordered lists* is:

Best Case	**Worst Case**	**Average Case**
1	*N*	*N/2*

Q1: Increasing the number of trials and the value of N

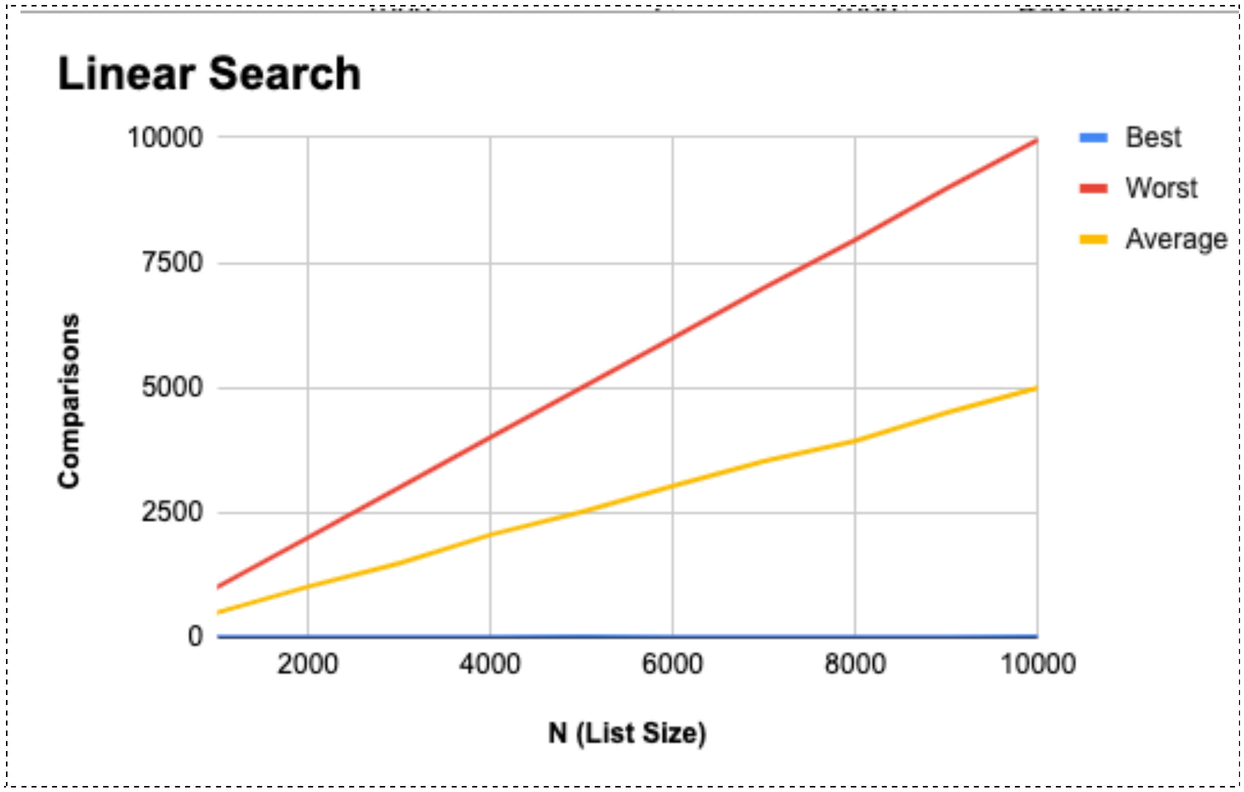
A. Run experiments with an increasing value of N (from 1000 to 10,000). Does increasing N affect how many trials you have to run to get accurate results? Explain.

   Yes, according with my experiments if the list of values increases, we need to run more trials to get the similar accurate results. When the size of the input increases there are a lot more possibilities in the order of the numbers we receive, so we need to test more sample cases to determine the best, worst and average case. This affects mainly best results.

B. Write down the number of trials that seem to have worked well for N=10,000.

Number of Trials
5000

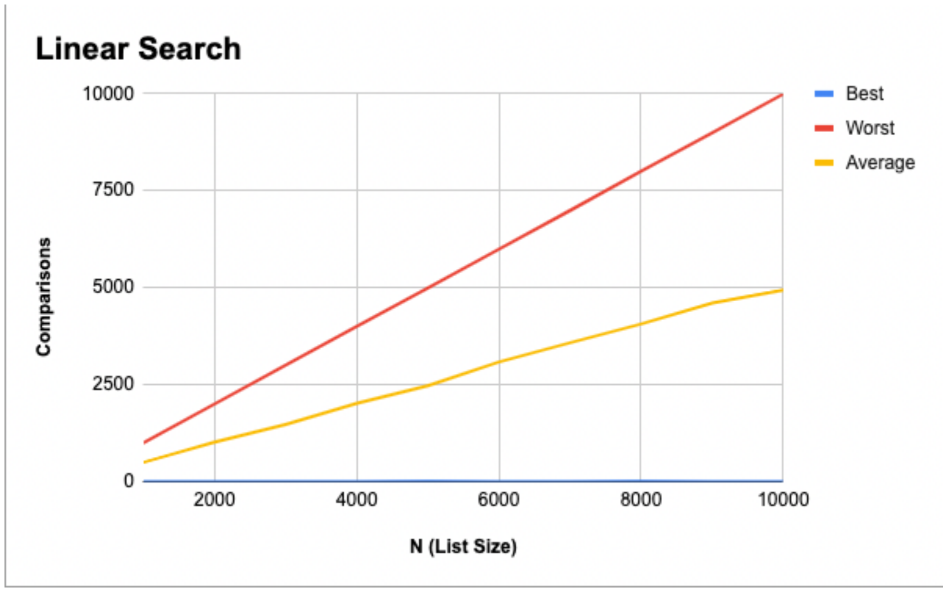
Q2: Linear Search Time Complexity Plot (Unordered List)



Q3: Does the order of the data in the list affect the number of comparisons? In the table below, guess the time complexity of Linear Search on an *Ordered List.*

Best Case	Worst Case	Average Case
1	N	N/2

Linear Search Time Complexity Plot (Ordered List)



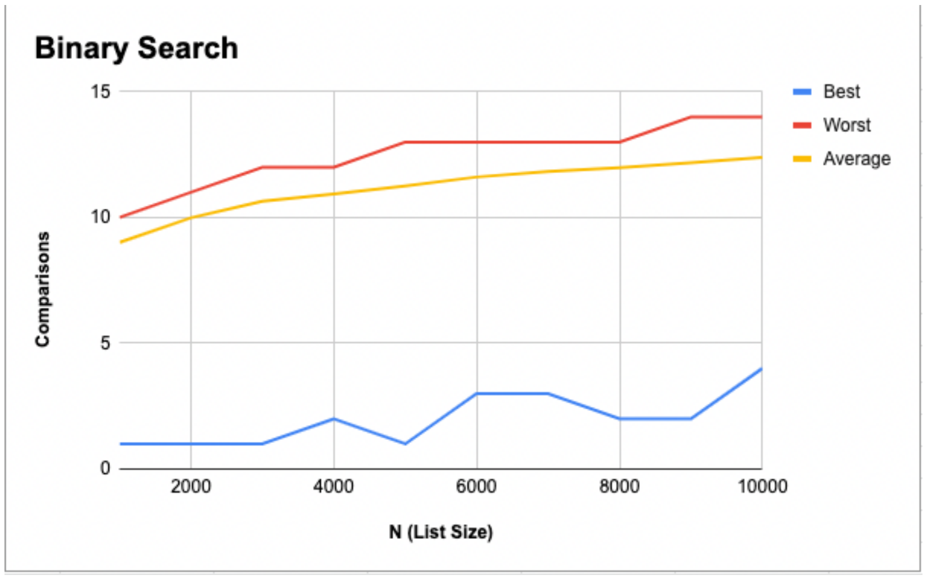
Conclusion:
Linear search has the same time complexity, regardless of whether the data is ordered or unordered. As the size increases, the trend of change is almost the same.

2. Binary Search

We know from class that the theoretical time complexity of binary search over *ordered lists* are:

Best Case	**Worst Case**	**Average Case**
1	*log_2(N)*	*log_2(N)*

Q4: Binary Search Time Complexity Plot



Conclusion: What do your results tell you about the average-case complexity of Binary Search?

The average case time complexity change trend is almost the same as the worst case change trend. Both are log_2(N).

3. Median

Q5: We hypothesize that the time complexity of find_median is:

Best Case	Worst Case	Average Case
N	N*N	N*N/2

Justification:

A. Best case scenario:

In this algorithm we start evaluating every element in the list and counting how many elements are greater or less than it, if the number of greater elements is equal to the number of less than elements then we conclude that the element is the median and we stop evaluating the next elements. If the median is the first element in the list, we only need to compare it with the whole list and then we can stop without evaluating the rest of the elements.

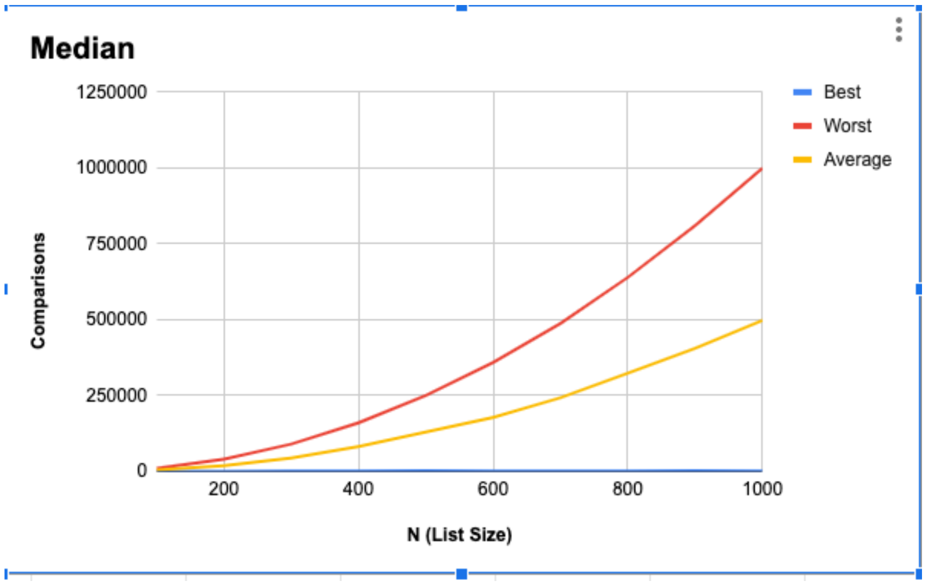
B. Worst case scenario:

We need to iterate the length data of the inside for loop list and all the data lengths of the outside loop list, so the time complexity is the product of these two lengths of list, assuming that the size of the two lists is N, the time complexity is N*N.

C. Average case scenario:

According to my experimental results, the average case time complexity is N*N/2. But in general, the coefficient, can also be ignored, so this case is similar to the worst case if the size is very large.

Find_median Time Complexity Plot



Conclusion: Did your results support your hypothesis? If not, why not, and how does it change your original hypothesis?

Yes, the results of my experiment are consistent with my hypothesis. The change trend of the plot drawn by the test result is consistent with the hypothesis.