In this project we implemented two simple search algorithms and tested their respective run times in order to understand whether binary and linear algorithms perform as we would expect them to given their big O time. The two search algorithms we implemented were linear search and binary search. In order to properly test the runtime of linear search and binary search we looked at the runtimes for the worst case scenario possible, that the search algorithm didn’t find the value being searched for. Given that linear search must go through the elements in a list one by one until it finds the value it is looking for, running a linear search code N times will result in a big O time of O(N). Binary search, on the other hand, cuts the list in half each time comparing the value being searched for to the middle value in the list and creating a sublist accordingly. Therefore, when considering the worst case, Binary search is much faster for very large lists, and has a big O notation O(log2N).

In order to test whether our search algorithms’ run times corresponded to what we would expect given big O notation we ran each algorithm through a list of all the words in the English language several times( from 200,000,000 to 2,000,000,000 by intervals of 200,000,000). Our data is shown in Table 1 below.

Table 1: RunTime Data for Linear and Binary Search Algorithms

type N\_millions time

1 Linear 200 1.676

2 Binary 200 0.001

3 Linear 400 5.834

4 Binary 400 0.002

5 Linear 600 8.867

6 Binary 600 0.003

7 Linear 800 9.672

8 Binary 800 0.003

9 Linear 1,000 14.915

10 Binary 1,000 0.004

11 Linear 1,200 11.448

12 Binary 1,200 0.004

13 Linear 1,400 12.825

14 Binary 1,400 0.004

15 Linear 1,600 14.169

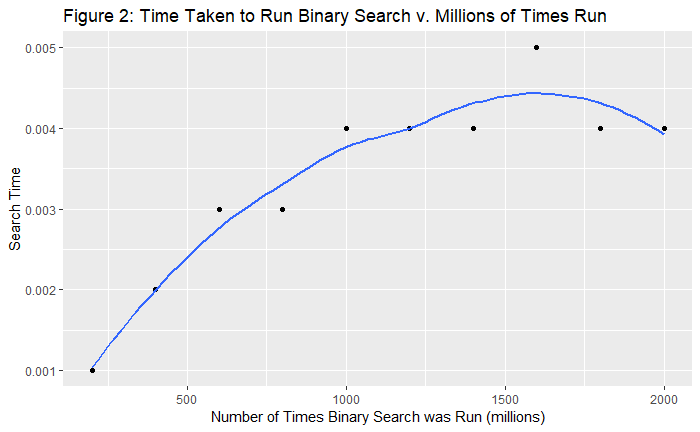
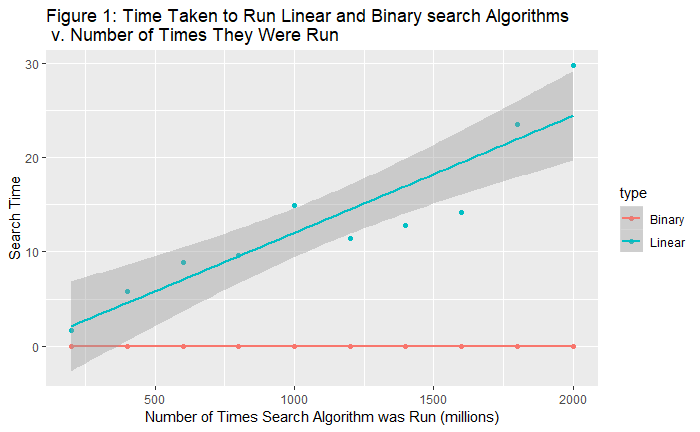
16 Binary 1,600 0.005

17 Linear 1,800 23.541

18 Binary 1,800 0.004

19 Linear 2,000 29.738

20 Binary 2,000 0.004

 After obtaining our data we plotted several graphs comparing the runtimes of the two search algorithms at different numbers of iterations (Figure 1 and Figure 2). As you can see in Figure 1, though there appears to be some autocorrelation within the results for Linear Search, the data points generally appear to follow a linear trend, which corresponds to our expected value of O(N). Though this would not be enough information to confirm our assumption in a statistical study, it is good enough to judge that it generally appears our assumptions about the run time of Linear Search were correct. While Figure 1 shows a good mapping of our results for Linear Search, the runtime for Binary Search increases at such a comparatively shallow rate we cannot judge if it increases at the expected rate of O(log2N) just by looking at Figure 1. However, if we select specifically for Binary Search, as shown in Figure 2, we can see that (excluding the pull from the point at N = 1.6billion ) the runtime for Binary Search does appear to conform to a logged distribution, gradually shallowing out over time. This leads us to believe that our binary search algorithm’s run time corresponds to our expected value of O(log2N).

Overall, after examining our visualizations of the data we collected we can conclude that our linear and binary search algorithms appear to conform to the efficiency levels we expected.