# ECS36C Program 01 Report

918378167 Pavlovich, Nicole

Given a list of the books and a list of requests, the program searches for matches. Time results for linear search shown in the first chart, time results for binary search shown in the second chart. The binary search time does not include the time to sort.

| **Linear Search** | | | | | |
|---|---|---|---|---|---|
| **Books** | **Request** | **Run A** | **Run B** | **Run C** | **Average** |
| 10 | 1 | 1.501 | 2.659 | 1.190 | 1.783 |
| 100 | 1 | 2.883 | 5.336 | 1.683 | 3.301 |
| 1000 | 1 | 16.675 | 11.096 | 11.318 | 13.030 |
| 100 | 10 | 46.432 | 24.228 | 14.809 | 28.490 |
| 1000 | 100 | 3300.450 | 2285.800 | 2675.680 | 2753.977 |
| 10000 | 100 | 7610.680 | 15341.800 | 14762.500 | 12571.660 |
| 10000 | 1000 | 19268.600 | 17618.100 | 13104.700 | 16663.800 |
| | | | | | |
| **Binary Search** | | | | | |
| **Books** | **Request** | **Run A** | **Run B** | **Run C** | **Average** |
| 10 | 1 | 1.062 | .829 | .337 | .743 |
| 100 | 1 | .750 | 1.579 | 2.125 | 1.485 |
| 1000 | 1 | 3.223 | 3.417 | 2.610 | 3.083 |
| 100 | 10 | 10.893 | 12.892 | 11.172 | 11.652 |
| 1000 | 100 | 46.243 | 58.962 | 76.580 | 60.595 |
| 10000 | 100 | 38.644 | 39.876 | 60.734 | 46.418 |
| 10000 | 1000 | 58.373 | 40.979 | 51.400 | 50.251 |

In general it seems as if binary search is more effective than linear search. We already knew this, because the binary search has O(logn) whereas the linear search has O(n). At very small values though, linear search may beat binary search depending on the location of the requested data. However, you'll notice that by the time the dataset had increased to 10^3, there was already a noticeable difference between the two algorithms, with linear taking approximately 13 microseconds and binary taking just 3, a four-fold difference.

One downside of binary search is that the list must be sorted before it can be searched. Although I did not account for this in my data collection, it's easy to understand that as the dataset gets large, the sorting time will also get noticeably larger. However, I decided to discount this because at the largest size, 10000 books by 1000 requests, binary search was more than 330 times faster than linear search, which I believe would be enough of a margin that the sorting would be irrelevant.

I was not able to determine if there exists a ration of requests to data that makes either solution the best option, but I did notice one strange line in my dataset. At 1,000 data points by100 requests, the binary search performed 45 times better than linear search, but worse than it performed at 10,000 data points by 1,000 requests. I thought this was odd because it was the same ratio of data to requests (10:1), but binary search consistently did worse with the smaller set. I also noticed that for 10,000 data points, the number of requests did not significantly impact the runtime of binary search. With 100 requests, it ran only 4 microseconds faster than at 1000 requests, an increase of 10x in the requests but only 1.08x times slower. On the other hand, the same 10x increase in requests for linear search caused a 1.33x increase in speed for linear search.

In conclusion, the data shows that binary search is almost always faster than linear search if not accounting for the sort time, and that even at large data sizes, the ratio of data to requests does not seem to change this statement.