Orry Butler
919288162

Program 1 Report

The purpose of this assignment was to analyze the runtime of three different searching algorithms: linear search, binary search, and recursive binary search. These three algorithms have very different applications and different strengths. For example, linear search is good for smaller amounts of data to search through, while binary search is good for quickly searching through large sets of data that are already sorted. Below are five different sets of data and the runtimes of each of these algorithms for the given data, with the books representing a value n and the requests representing a value m.

Linear Search	Trial 1 Runtime	Trial 2 Runtime	Trial 3 Runtime	Average Runtime
1000 books 10 requests	1434.91	1376.53	1372.34	1394.59
1000 books 100 requests	6019.62	6538.23	6092.77	6216.87
1000 books 1000 requests	20913.2	20270.8	20,939.6	20707.87
100 books 1000 requests	6264.63	6363.78	6602.86	6410.42
10 books 1000 requests	1466.56	1439.43	1454.88	1452.62

Binary Search	Trial 1 Runtime	Trial 2 Runtime	Trial 3 Runtime	Average Runtime
1000 books 10 requests	2274.45	2322.81	2272.68	2289.98
1000 books 100 requests	2597.72	2622.16	2616.68	2612.19
1000 books 1000 requests	5160	5795.44	5583.96	5513.13
100 books 1000 requests	2750.48	2739.53	2766.53	2752.18

10 books 1000 requests	1627.14	1639.8	1614.95	1627.30

Recursive Binary Search	Trial 1 Runtime	Trial 2 Runtime	Trial 3 Runtime	Average Runtime
1000 books 10 requests	10,368.3	10,349.4	10474.2	10397.3
1000 books 100 requests	58258	59847.1	58815.7	58973.6
1000 books 1000 requests	541032	541372	542005	541469.67
100 books 1000 requests	32,155.4	32550.4	32568.9	32424.9
10 books 1000 requests	6262.22	6674.94	6138.64	6,358.6

The information in these tables displays the trend of how runtimes of each of the algorithms is affected by a different set of data. One of the most interesting trends is that, with binary search and recursive search, large n values result in a worse runtime than the runtime that results from the algorithm being used on the same amount of total data but with a larger m value than n value. This trend occurs because, in order to use binary search, you must first sort the list of books of data size n. The larger the n value, the longer the list takes to be sorted–so that sorting time adds to the total runtime. Then, we can see that recursive binary search runs slower in all cases than binary search. This is a byproduct of using a recursive algorithm; it runs slightly slower but more efficiently than the quicker binary search.

In comparing the linear search algorithm and the binary search algorithm, the linear search for small values of total data has a faster runtime than binary search. Linear search has a total runtime of O(n*m) because it iterates through nested for loops in order to find all of the matching data in the two vectors. In contrast, the larger data algorithm of binary search runs in O(mlog(n)) time because each time the vector is checked for books, the program cuts the data in half. However, for smaller sets of data, this method is less effective than linear search due to the cost of first sorting the list, adding up the runtime in those cases. In the end, these algorithms are all strong for different sets of data, and any can be used efficiently in the right situation.