Time Complexities of Binary and Linear Search Algorithms

Ege Ozan Özyedek, 21703374, CS201 – 02

Introduction

The purpose of this homework is to learn how the time complexities of algorithms are computed and described with notions such as big-Oh and to observe how similar these algorithms are to actual real-life tests. To reach this goal, first a summary of the theoretical analysis of time complexities were made, then by writing a driver function and testing multiple different array sizes the operation times of thwo different search algorithms, comparisons were made between the theoretical and the real-life case.

Theoretical Analysis of Algorithm Time Complexities

Algorithms, in computer science, are described as a set of rules to be followed for problem solving purposes. Algorithms are, perhaps, one of the most important aspects of computer science since the focus of computer science is to solve problems in the most optimized way possible. This homework emphasizes the optimization, or the efficiency, of algorithms and in particular, their time efficiency. The time it takes for an algorithm to do its job in the generalized form is called the time complexity of that algorithm. There are three notations which describe this time complexity, but the most popular one is called the “big-Oh” notation. This describes the time complexity as a simple function of the size of iterations (or in the algorithms that this assignment requires, it’s the iterations of the for loop).

There some rules to finding the time complexity of an algorithm, these are

- Simple operations such as assignment or arithmetic operations are assumed to take 1-unit time
- The time complexity does not include the time it takes to read the input, and also assumes infinite memory
- The run time of a loop is the operations inside times the number of iterations of said loop.
- Similarly, for nested loops, the run time is the operations in the inner most loop times the iteration number of the inner loop, plus the operations on the outer loop times the iterations of the outer loop.
- For an if/else statement, the run time is never more than the maximum time of the statements in the if/else operations plus the conditioning statement.

Time complexities of algorithms are represented by best, average and worst cases. For a search algorithm, these cases change according to the index of the searched value. These will be explained in the linear search example a bit more clearly.

As an example, and also for the comparisons in this report, the complexity of linear search and binary search will be found.

For the Linear search algorithm, there exists a single for-loop, which has N iterations, and as explained above, by multiplying this with the time of operations inside it the time complexity can be found. However, looking only at the size of the array creates the assumption that this algorithm will always run for N iteration but obviously this is wrong. There exists best, average and worst cases which are dependent on the index value. For example, for index i = 0, the first element will be returned, and the total number of iterations will be O(1) (or a constant, but in notation constants are represented as 1). This is the best case for the linear search algorithm. The average and worst cases can also be found easily, as because the average case is just the middle index of the array (i = (N-1)/2) and the worst case is iterating through the whole array and not finding the index.

For Binary search, the analysis is different since binary search uses a mare complex approach to searching. It requires a sorted array, since it compares values. There exists one while loop, which ends if the search has ended (if high is smaller than low meaning the two “arrows” that search the array have crossed paths). The best case in this algorithm is when the index of the searched value is equal to the middle index value (i = (N-1)/2) and has complexity O(1) since only one iteration will be covered. The avg case and worst case can be found by an easy analysis of an array.

Time Complexities of Binary and Linear Search Algorithms

Ege Ozan Özyedek, 21703374, CS201 – 02

1	2	3	4	5	6	7	8	9

Assume the above table represents the array at hand and the values inside the boxes are the values which correspond to the array values. Assume that the function is inputted 6 as the value to be searched (has index i = 5). The operations of this process can be observed below (int variables are rounded down).

$$1^{st} \rightarrow \text{mid} = 4, \text{arr[mid]} = 5$$
$$2^{nd} \rightarrow \text{mid} = \frac{8+5}{2} \cong 6\,, \text{arr[mid]} = 7$$
$$3^{rd} \rightarrow \text{mid} = \frac{5+5}{2} \cong 5, \text{arr[mid]} = 6$$

This case it took 3 iterations for an N = 9 array. It can be observed that traces similar to above give the result that the average and worst-case complexity of the binary search is $\log_2 N$. It is true for both the average and worst case, since they are not far apart in terms of the iteration number. This gives us the below theoretical table, and plot which shall be used as a guide for the real computer test.

Table 1: Time Complexity of Search Algorithms

Algorithm	Best	Average	Worst
Linear	O(1)	O(N)	O(N)
Binary	O(1)	O(logN)	O(logN)

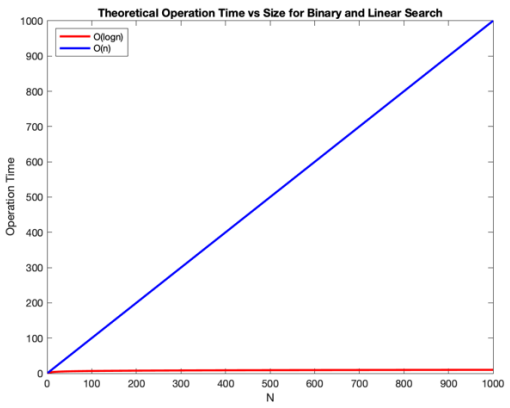

Figure 1: The Operation Time (or the complexity function) with respect to N

Analysis of Algorithm Time Complexities on a Real Computer System

The hardware specifications of the computer used to test the operation times of these two algorithms can be observed below:

- Computer: MacBook Pro 2018
- OS: macOS Catalina (version 10.15.1)
- CPU: 2.6 GHz 6-Core Intel Core i7
- Memory: 16 GB 2400 MHz DDR4
- GPU: Radeon Pro 560X 4 GB

For the testing of the algorithms, first a class which contains the search algorithms were created. To capture the operation time of the function, the ctime library was used. At both functions, the clock starts right at the begging after all input operations are made and ends right before the functions returns. Pass by reference was used in order to obtain the operation time from the function.

Another function was the generateArray function which generated a sorted array that assigned the index value to the array value of that index.

In the driver function (or main), first an array that contains the sample points were created. Sample points were taken to be multiples of 2. There was a total of 30 sample points (or array sizes, N) up to and including the number 1073741824 (which is around one billion, the max size the assignment requires). After the array has been constructed, for each array size, the search functions were called 100 times for 4 different values of search (the i,ii,iii and iv cases in the assignment). preciseness. The average of those 100 calls were assigned to be the operation time at that size. All the operation times were written to a file.

After the files were filled out and the C++ program came to an end, the files were used to create MATLAB arrays in order to then use the data inside them. Using these arrays, three operation time graphs were plotted, with one containing all data, one containing only the 4 index searches from the binary search and the other containing the other 4 index searches from the linear search. The plots and the data table which contains all obtained data can be observed below. All data below are measured in miliseconds.

Time Complexities of Binary and Linear Search Algorithms

Ege Ozan Özyedek, 21703374, CS201 – 02

Table 2: Linear Search Algorithm Operation Times

N	i = 0	i = (N - 1)/2	i = N - 1	i = -1
2	0.0007	0.00071	0.0008	0.0007
4	0.0007	0.00062	0.0007	0.0007
8	0.0006	0.00068	0.0007	0.0007
16	0.0007	0.00069	0.0007	0.0007
32	0.0006	0.00057	0.0006	0.0006
64	0.0005	0.00068	0.0007	0.001
128	0.0007	0.00082	0.0009	0.0007
256	0.0006	0.00091	0.0014	0.001
512	0.0005	0.00092	0.0015	0.0014
1024	0.0006	0.00139	0.0024	0.0023
2048	0.0005	0.00238	0.0043	0.0043
4096	0.0008	0.00416	0.009	0.0084
8192	0.0004	0.00813	0.0162	0.0167
16384	0.0005	0.01611	0.0306	0.0308
32768	0.0005	0.03189	0.0591	0.0592
65536	0.0005	0.07125	0.1331	0.1267
131072	0.0006	0.11302	0.2193	0.2188
262144	0.0004	0.21736	0.4383	0.4357
524288	0.0004	0.42755	0.8746	0.8964
1048576	0.0004	0.87627	1.7632	1.7746
2097152	0.0005	1.80278	3.5447	3.5494
4194304	0.0005	3.48641	7.1131	7.1421
8388608	0.0004	7.1	14.158	14.207
16777216	0.0004	14.2854	28.088	28.337
33554432	0.0005	28.2798	54.912	54.798
67108864	0.0004	55.1113	108.82	110.41
134217728	0.0004	111.863	224.39	220.02
268435456	0.0004	222.377	438.05	445.23
536870912	0.0004	445.482	886.25	893.17
1.074E+09	0.0004	903.045	1786.8	1770.8

Table 3: Binary Search Algorithm Operation Times

N	i = 0	i = (N - 1)/2	i = N - 1	i = -1
2	0.0007	0.00068	0.0007	0.00069
4	0.0007	0.00063	0.0007	0.00061
8	0.0007	0.00063	0.0006	0.00073
16	0.0007	0.00067	0.0007	0.00077
32	0.0006	0.00052	0.0006	0.00061
64	0.0007	0.00061	0.0005	0.00056
128	0.0007	0.00052	0.0007	0.00058
256	0.0006	0.00058	0.0008	0.00063
512	0.0007	0.00051	0.0006	0.00076
1024	0.0006	0.00055	0.0006	0.00062
2048	0.0007	0.00051	0.0006	0.00069
4096	0.0004	0.00051	0.0007	0.0005
8192	0.0007	0.00098	0.0007	0.00075
16384	0.0006	0.00065	0.0007	0.00071
32768	0.0006	0.00036	0.0007	0.00064
65536	0.0006	0.00053	0.0006	0.00059
131072	0.0005	0.00044	0.0005	0.00044
262144	0.0006	0.00039	0.0005	0.00054
524288	0.0006	0.00044	0.0006	0.00067
1048576	0.0006	0.00043	0.0006	0.00062
2097152	0.0011	0.00047	0.0007	0.0008
4194304	0.0006	0.00047	0.0008	0.00092
8388608	0.0007	0.0006	0.0007	0.00108
16777216	0.0006	0.00051	0.0009	0.00104
33554432	0.0007	0.00049	0.0008	0.00109
67108864	0.0007	0.00055	0.0008	0.00119
134217728	0.0007	0.00034	0.0009	0.00127
268435456	0.0007	0.0004	0.0011	0.00137
536870912	0.0008	0.00061	0.0012	0.00151
1.074E+09	0.0007	0.0005	0.0011	0.0015

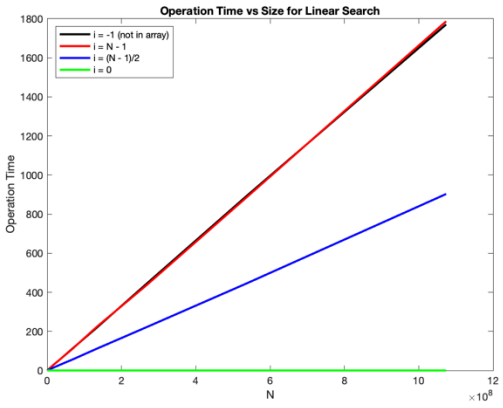

Figure 2: Operation time plot for Linear Search

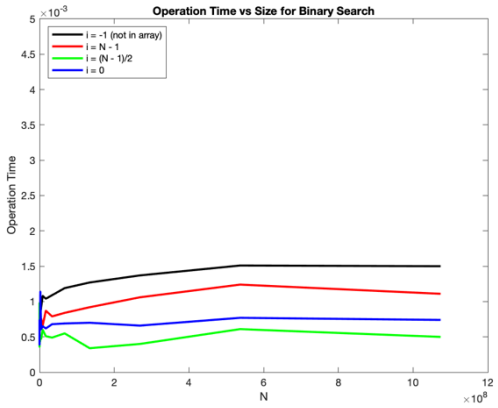

Figure 3: Operation time plot for Binary Search

Time Complexities of Binary and Linear Search Algorithms

Ege Ozan Özyedek, 21703374, CS201 – 02

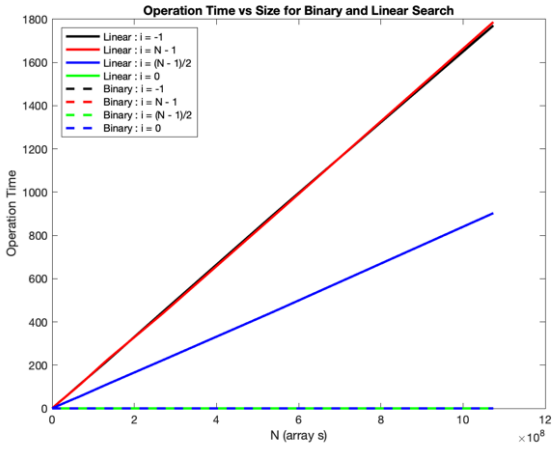

Figure 4: Operation time plots of both algorithms on the same graph

Observing the above figures and tables show that the obtained data was as expected. Linear search gets slower as N increases with high acceleration while binary search barely increases in terms of operation time. Another thing to note is how the best-case operation times are constant, as it was expected. It can also be seen that the best case for the binary search algorithm was choosing the middle index (the value (N – 1)/2 in our case, since the arrays have the same value as their indexes). Overall, it can be seen that all observations made as well the expected time complexity of each algorithm were true to the theoretical analysis.

Conclusion

The homework assignment was successfully completed by the given methods in the report. The report first introduced the derivation of time complexities of different algorithms and exemplified this process for two different algorithms, linear and binary search. After this summary, the methodology of the testing was explained. The testing of the operation time was done for 30 samples, each sample for 100 times and for four different search values. The testing was done for the linear and binary search algorithms and the results were compared to the theoretical derivation of the time complexities.