Ihsan Dogramaci Bilkent University

CS Department

CS201 Sec-01

Homework Assignment 2

Alper Kandemir

21703062

10.04.21

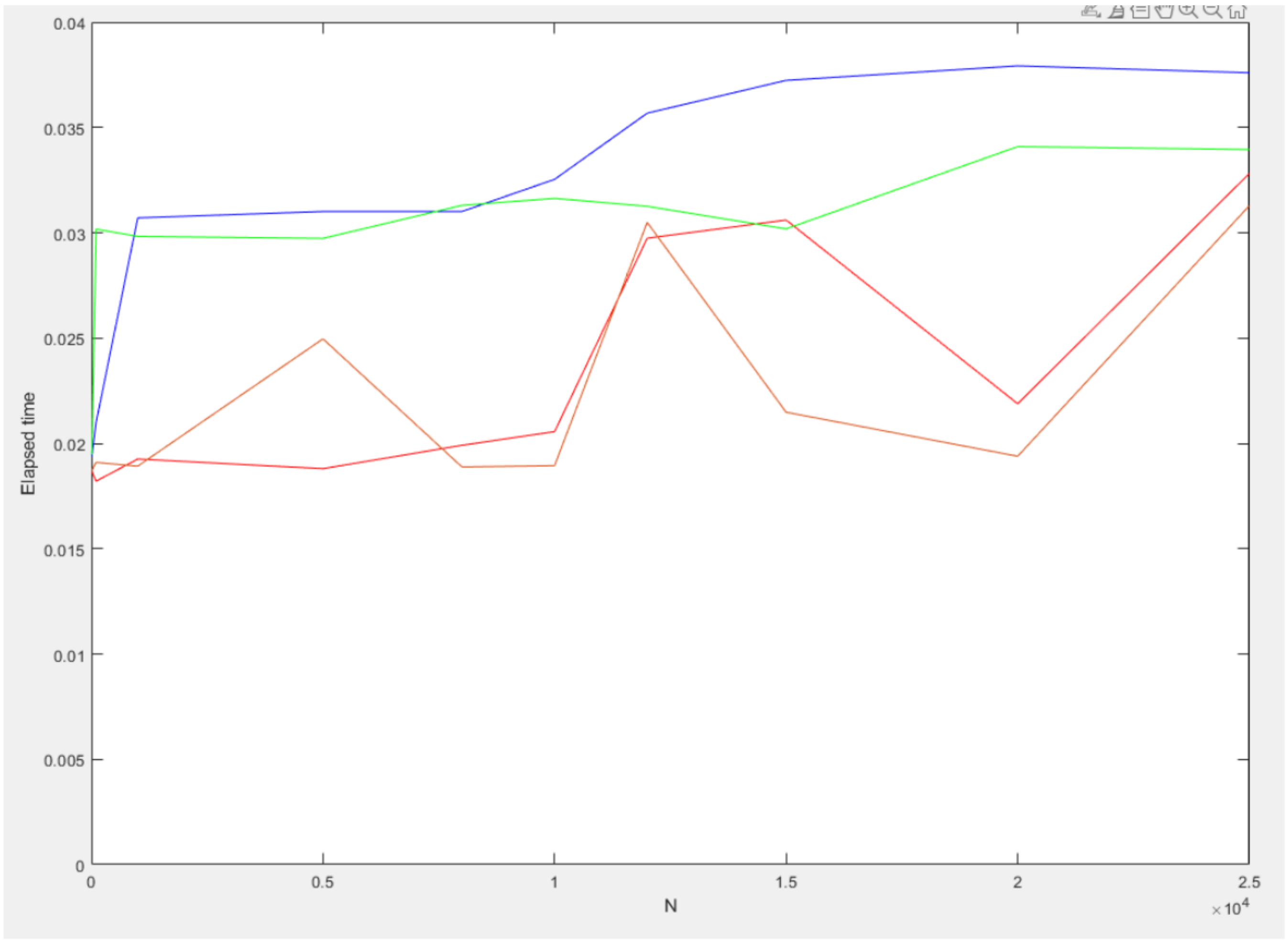
3)

N	Linear (Iterative)				Linear (Recursive)			
	a	b	c	d	a	b	c	d
10	0.018669	0.029462	0.030462	0.031765	0.018707	0.031163	0.030723	0.032442
100	0.018206	0.030645	0.031613	0.032203	0.019096	0.033253	0.031092	0.03198
1000	0.019258	0.031753	0.031688	0.035179	0.018919	0.03172	0.040238	0.034928
5000	0.018799	0.0333	0.034143	0.038781	0.024962	0.03324	0.038523	0.038747
8000	0.019909	0.03708	0.04173	0.041945	0.01888	0.03483	0.040542	0.043315
10000	0.020559	0.03753	0.044908	0.04551	0.018942	0.037786	0.049179	0.044685
12000	0.029741	0.037626	0.04542	0.047705	0.030492	0.040986	0.046536	0.04751
15000	0.030605	0.041484	0.049796	0.050310	0.021478	0.040875	0.049344	0.051285
20000	0.02188	0.04481	0.058957	0.059777	0.019389	0.046398	0.06561	0.06027
25000	0.03281	0.049016	0.063075	0.089467	0.031283	0.04787	0.063834	0.065275

	Binary Search				Jump Search			
N	a	b	c	d	a	b	c	d
10	0.019382	0.019924	0.030483	0.032325	0.019470	0.0192	0.02988	0.032442
100	0.030978	0.030023	0.031075	0.031968	0.030173	0.031003	0.030698	0.032173
1000	0.019707	0.030535	0.036633	0.032115	0.029829	0.03186	0.030882	0.032268
5000	0.030008	0.019044	0.031242	0.032112	0.029736	0.031743	0.032425	0.031692
8000	0.03101	0.030198	0.032363	0.03406	0.031303	0.033915	0.031892	0.032342
10000	0.030535	0.020172	0.035203	0.035901	0.031633	0.032090	0.032175	0.032003
12000	0.031378	0.020195	0.031103	0.035210	0.031258	0.031305	0.032017	0.033125
15000	0.024236	0.018576	0.03333	0.033366	0.030185	0.033105	0.032296	0.032087
20000	0.034342	0.020602	0.03224	0.034645	0.034093	0.039105	0.035027	0.035508
25000	0.03297	0.019574	0.034125	0.035392	0.033953	0.03664	0.042192	0.043226

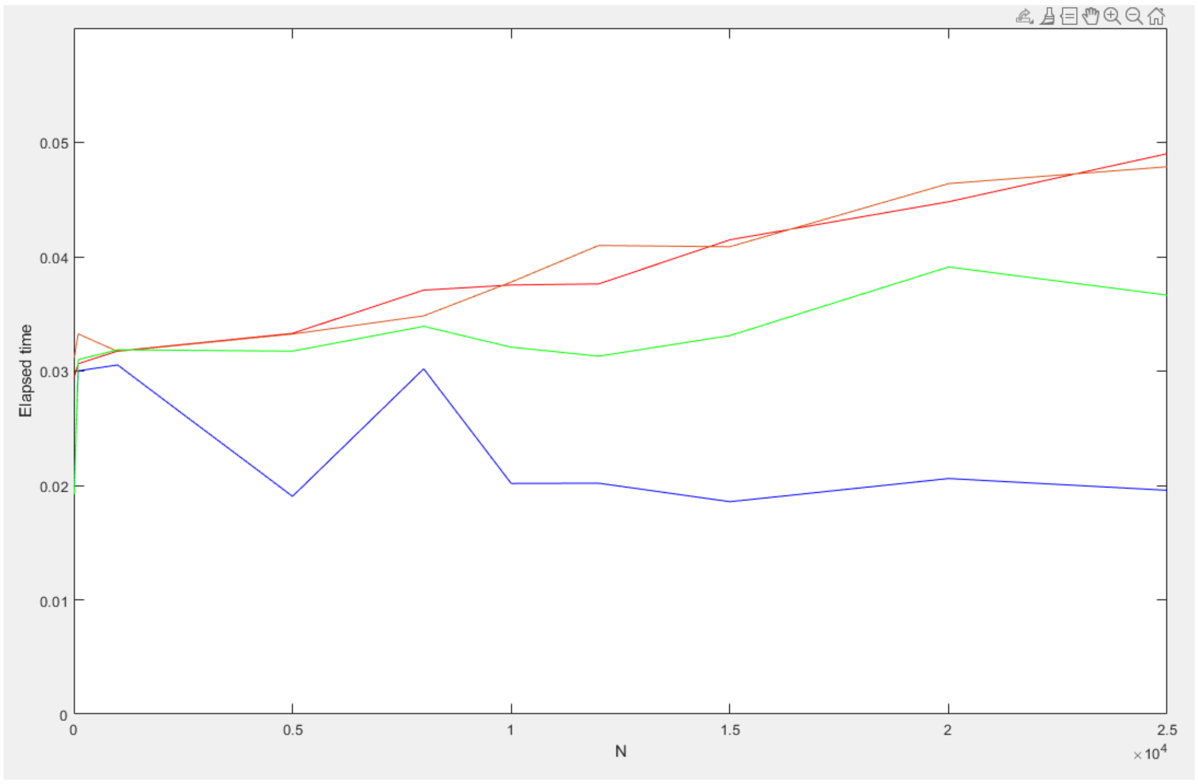
4)

Scenario a:



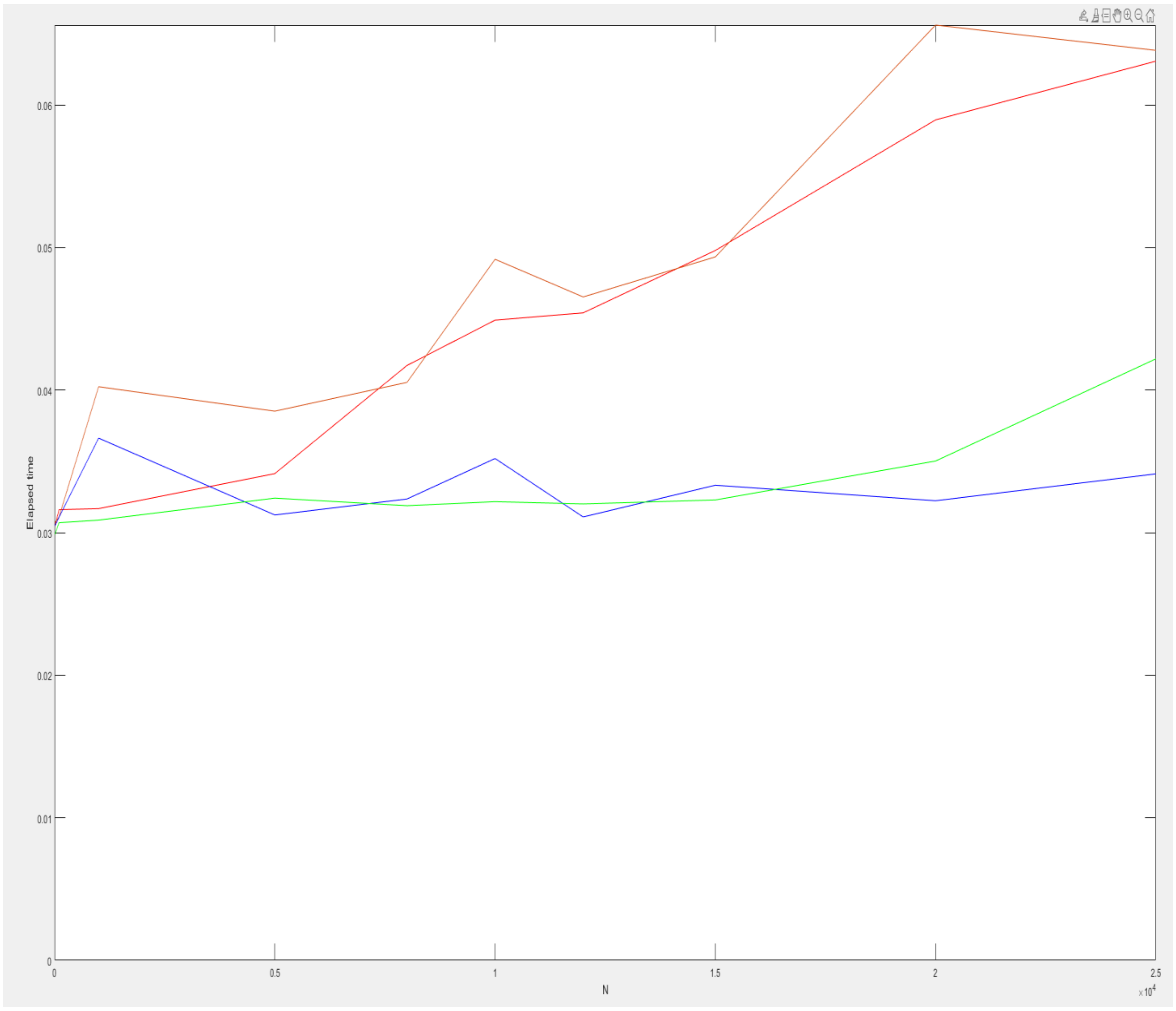
-Algorithm1: Red, -Algorithm2: Orange, -Algorithm3: Blue, -Algorithm4: Green

Scenario b:



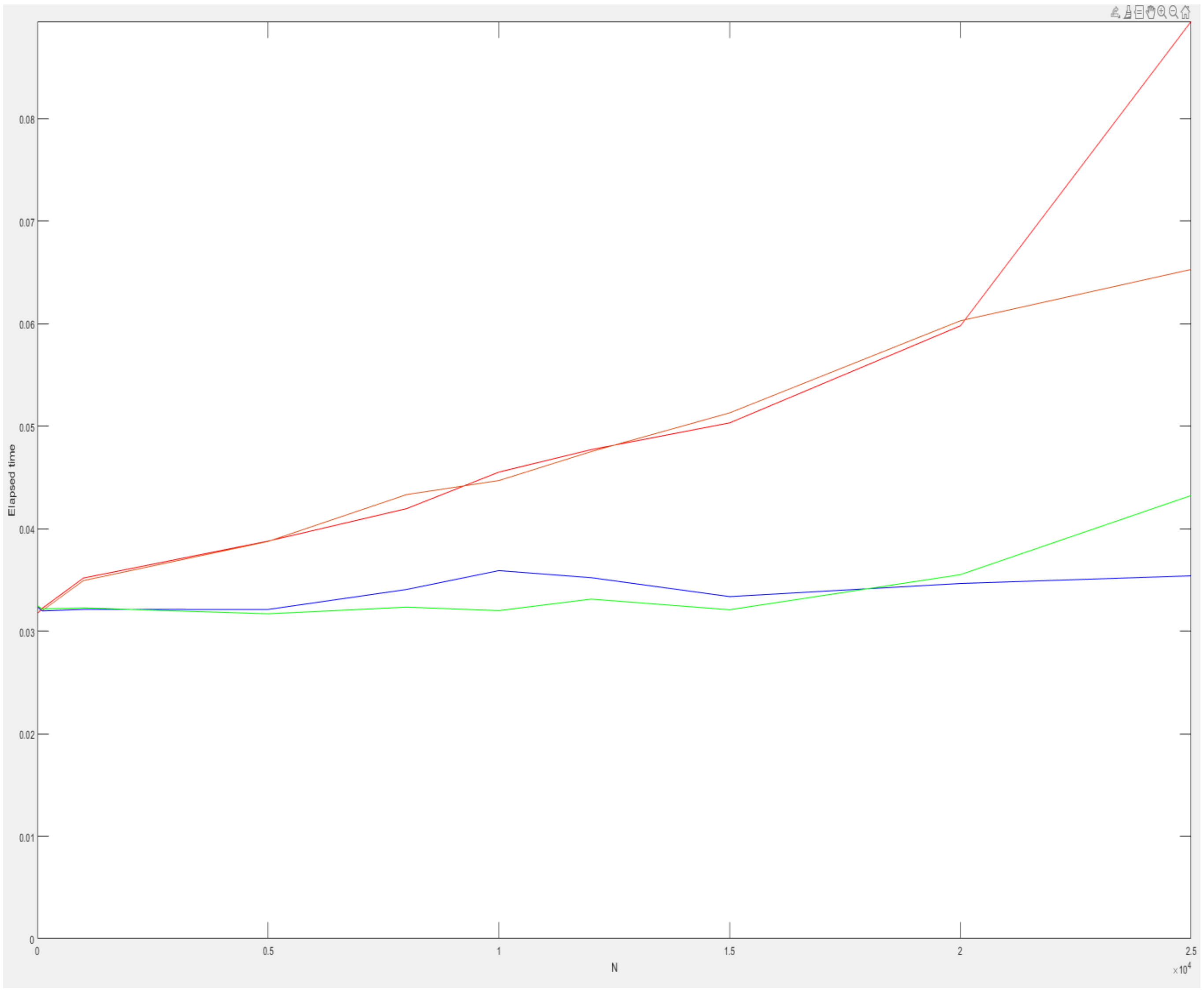
-Algorithm1: Red, -Algorithm2: Orange, -Algorithm3: Blue, -Algorithm4: Green

Scenario c:



-Algorithm1: Red, -Algorithm2: Orange, -Algorithm3: Blue, -Algorithm4: Green

Scenario d:



-Algorithm1: Red, -Algorithm2: Orange, -Algorithm3: Blue, -Algorithm4: Green

5)

5.1) Computer specs:

OS: Windows 10 x64

CPU: Intel core i9 9900K 4.70GHz

Mem: 32 GB DDR4 RAM

5.2) Theoretical worst, average, and best cases for each algorithm and expected running times for each scenario for each algorithm.

Algorithm1:

Best case is the key value is at first index. Worst case is key value is at last index or not in the list. Average case is the key value at middle. Expected running times: Scenario a< Scenario b< Scenario c< Scenario d

Worst case time complexity O(N)

Average case time complexity O(N)

Best case time complexity O(1)

For an unsuccessful search: Worst-case, best-case, and average-case are all the same → O(N)

Algorithm 2:

Best case is the key value is at first index. Worst case is key value is at last index or not in the list. Average case is the key value at middle.

Expected running times: Scenario a < Scenario b < Scenario c < Scenario d

Worst case time complexity O(N)

Average case time complexity O(N)

Best case time complexity O(1)

For an unsuccessful search: Worst-case, best-case, and average-case are all the same → O(N)

Algorithm 3:

Best case is the key value at the middle. Worst case is the key value at the beginning or end, or not in the list. Average case is the key value at somewhere between the middle and the beginning, or somewhere between the middle and the end.

Expected running times: Scenario b < Scenario a <=Scenario c < Scenario d

Worst case time complexity O(log N)

Average case time complexity O(log N)

Best case time complexity O(1)

For an unsuccessful search: Worst-case, best-case, and average-case are all the same → O(log N)

Algorithm 4:

Best case: if the key value in the first block that the algorithm searches for key value. Worst case: if the key value in the last block. Average case: somewhere in the middle blocks.

Expected running times: Scenario a < Scenario b <Scenario c < Scenario d

Worst case time complexity O(√N)

Average case time complexity O(√N)

Best case time complexity O(1)

For an unsuccessful search: Worst-case, best-case, and average-case are all the same → O(√N)

5.3

Algorithm 1:

Scenario a is the best case for this algorithm, so we can expect that the running time should be the same at each run or very close to each other and the same for all sizes. Also, the running time of scenario a should be the shortest. The running times should be the same or very close to each other, but sometimes the difference is high. It probably caused by the compiler or the drops at CPU’s GHz. Although there are different running times, the times are very close to each other, and the times are the shortest times of the runs. Therefore, theoretical and observed results are similar.

Scenario b is the average case for this algorithm, so the running time of scenario b should be greater than scenario a and less than scenario c/d. We can see the relation on the table. Also, the time values should be increased as the size increased like O(N). This condition is also met; therefore, theoretical and observed results are similar.

Scenario c is the worst case for a successful search, so the running time of scenario c should be greater than scenario a and b. We can see the relation on the table. Also, the time should be increased as the size increased like O(N). this condition is also met; therefore, theoretical and observed results are similar.

Scenario d is the worst case for unsuccessful search, so the running time of scenario d should be greater than scenario a and b, and close to scenario c. We can see the relation on the table. Also, the time should be increased as the size increased like O(N). This condition is also met; therefore, theoretical and observed results are similar.

Algorithm 2:

I expect similar results to the first algorithm, and algorithm 2 gave almost identical results as to the first algorithm. However, sometimes algorithm 2 times are increased more than algorithm 1 times. It probably caused by the compiler or the drops at CPU’s GHz or recursive and iterative differences of the compiler. Therefore, theoretical and observed results are similar to in the first algorithm.

Algorithm 3:

Scenario a is one of the worst cases for this algorithm, so we can expect that the running time of scenario a should be greater than scenario b, and similar to c. According to the table, the conditions is mostly met except for some values. We have close scenario a and scenario c values, and they are greater than scenario a values. Also, the running time is increased with respect to n size very slowly, like the log(n) graph. So, theoretical and observed results are similar.

Scenario b is the best case for this algorithm so, we can expect that the running time should be the smallest. The table meets the condition. However, some N sizes do not give the correct result. The running times should be the same or very close to each other, but sometimes the difference is high. It probably caused by the compiler or the drops at CPU’s GHz. Although there are different running times, theoretical and observed results are similar.

Scenario c is one of the worst cases for this algorithm, so we can expect that the running time of scenario c should be greater than scenario b, and similar to a. The table meets the condition except for small differences. Also, the running time is increased with respect to n size very slowly like the log(n) graph. So, theoretical and observed results are similar.

Scenario d is the worst case for unsuccessful search, so we can expect that the running time of scenario d is the greatest time. The scenario d’s time is always the greatest, so theoretical and observed results are similar. Also, we can see the running time is increased with respect to n size very slowly, like log(n) graph.

Algorithm 4:

Scenario a is the best case for this algorithm, so we can expect that the running time should be the smallest, and we can expect that the running time should be the same at each run or very close to each other, and the same for all sizes. However, sometimes the difference between values is high. It probably caused by the compiler or the drops at CPU’s GHz. The general structure is met, so theoretical and observed results are similar.

Scenario b is the average case for this algorithm, so the running time of scenario b should be greater than scenario a and less than scenario c/d. According to the table, the running times of scenario b is greater than scenario a, but they are not less than c/d times all the time. It probably caused by the compiler or because of the small collection sizes so, the times are not precise enough to observe that condition. The time should be increased as the size increased like $O(\sqrt{N})$. According to the table, this condition seems also met. Therefore, theoretical and observed results are similar.

Scenario c is the worst case for a successful search, so the running time of scenario c should be greater than scenario a and b. We cannot see the relation on the table until the collection size is big enough. We need bigger sizes to observe that condition. Also, the time should be increased as the size increased like $O(\sqrt{N})$. this condition is met. Therefore, theoretical and observed results are similar if the sizes would be bigger.

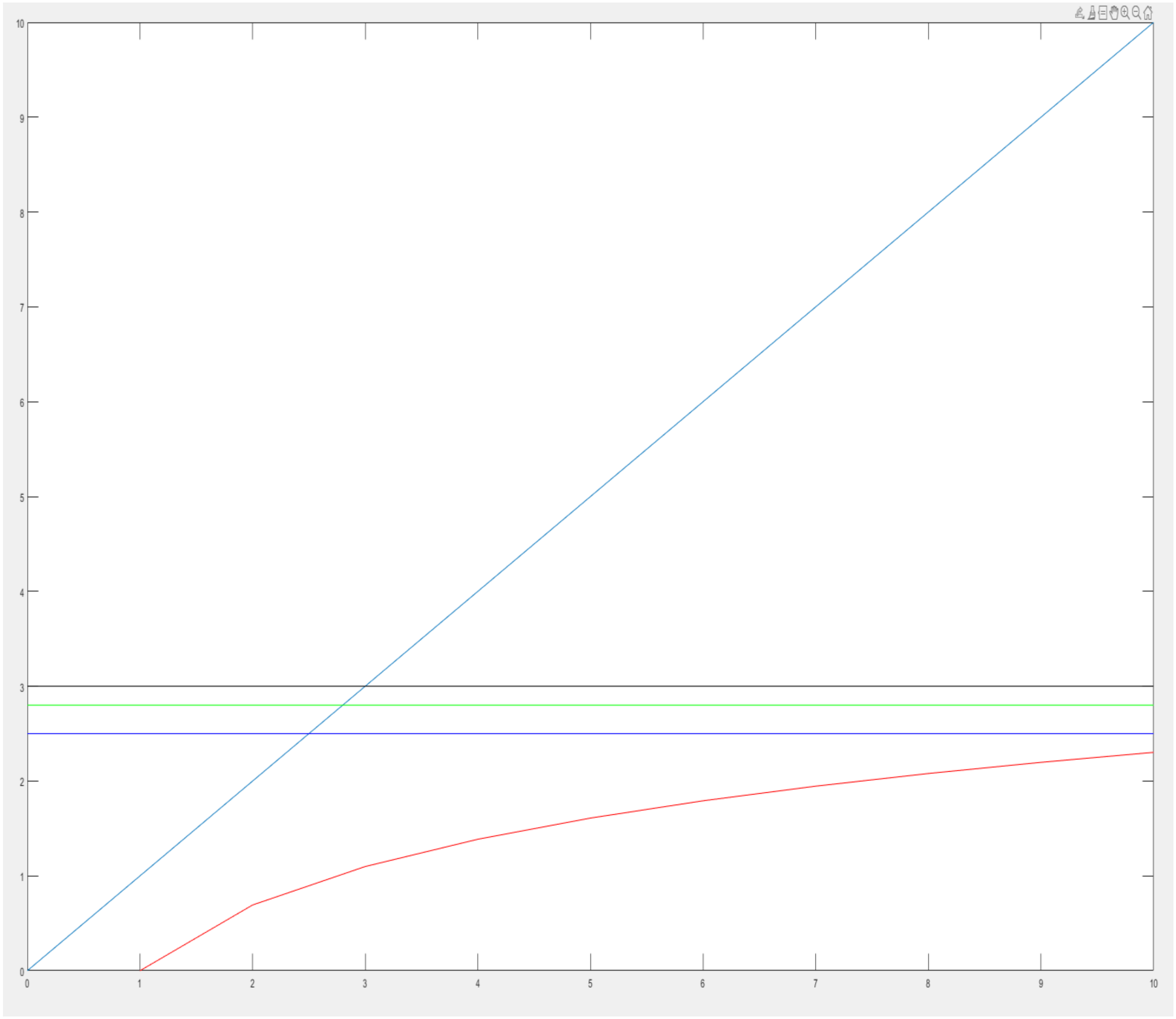
Scenario d is the worst case for unsuccessful search, so the running time of scenario d should be greater than scenario a and b, and close to scenario c. We cannot see the relation on the table until the collection size is big enough.

We need bigger sizes to observe that condition. Also, the time should be increased as the size increased like O(√N). This condition is met. Therefore, theoretical and observed results are similar if the sizes would be bigger.

5.4)

Theoretical

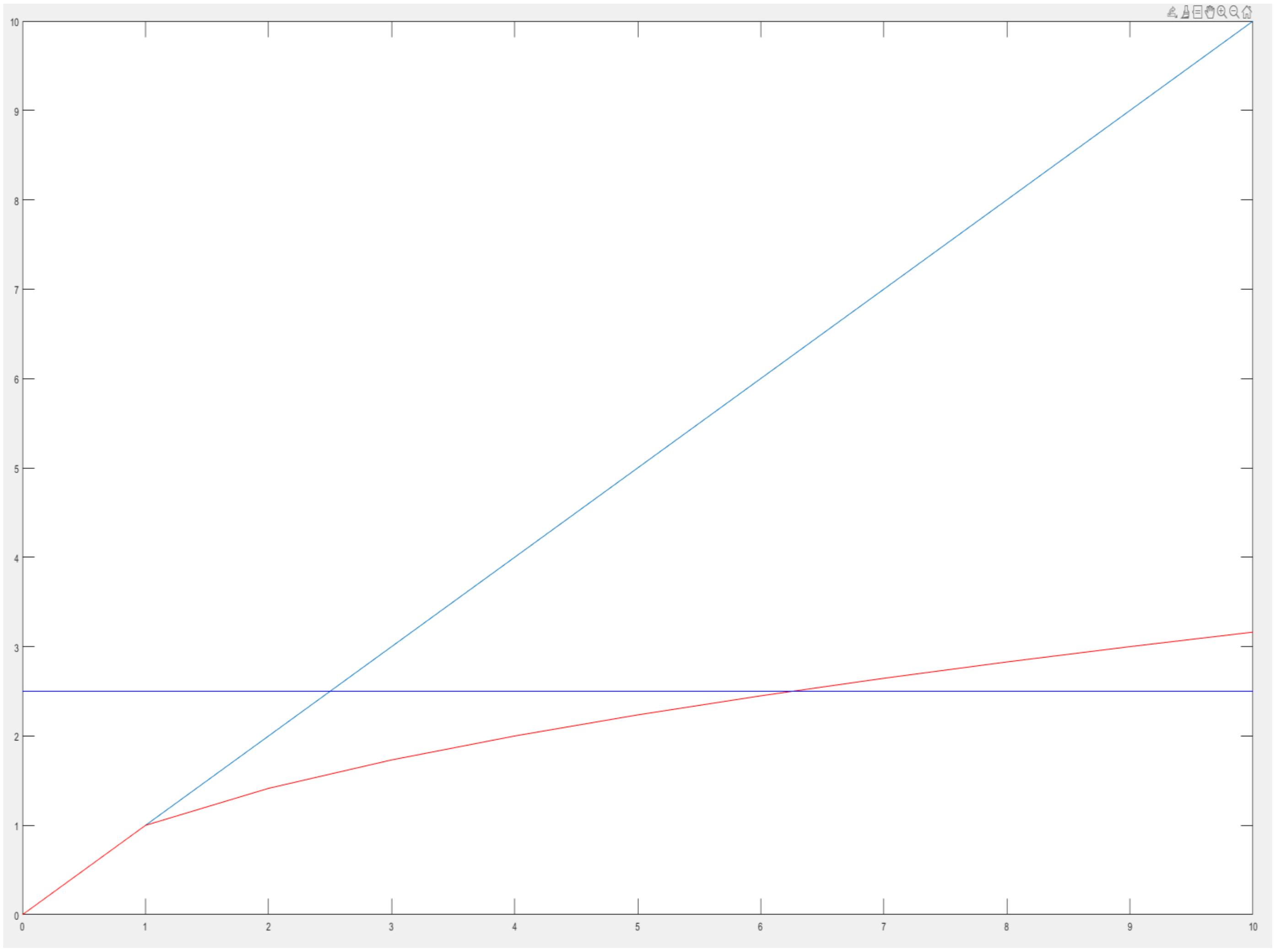
Scenario a:



Blue: O(N) for comparison, red: O(logN), others: O(1)

Algo 1,2,3,4: O(1), O(1), O(logN), O(1)

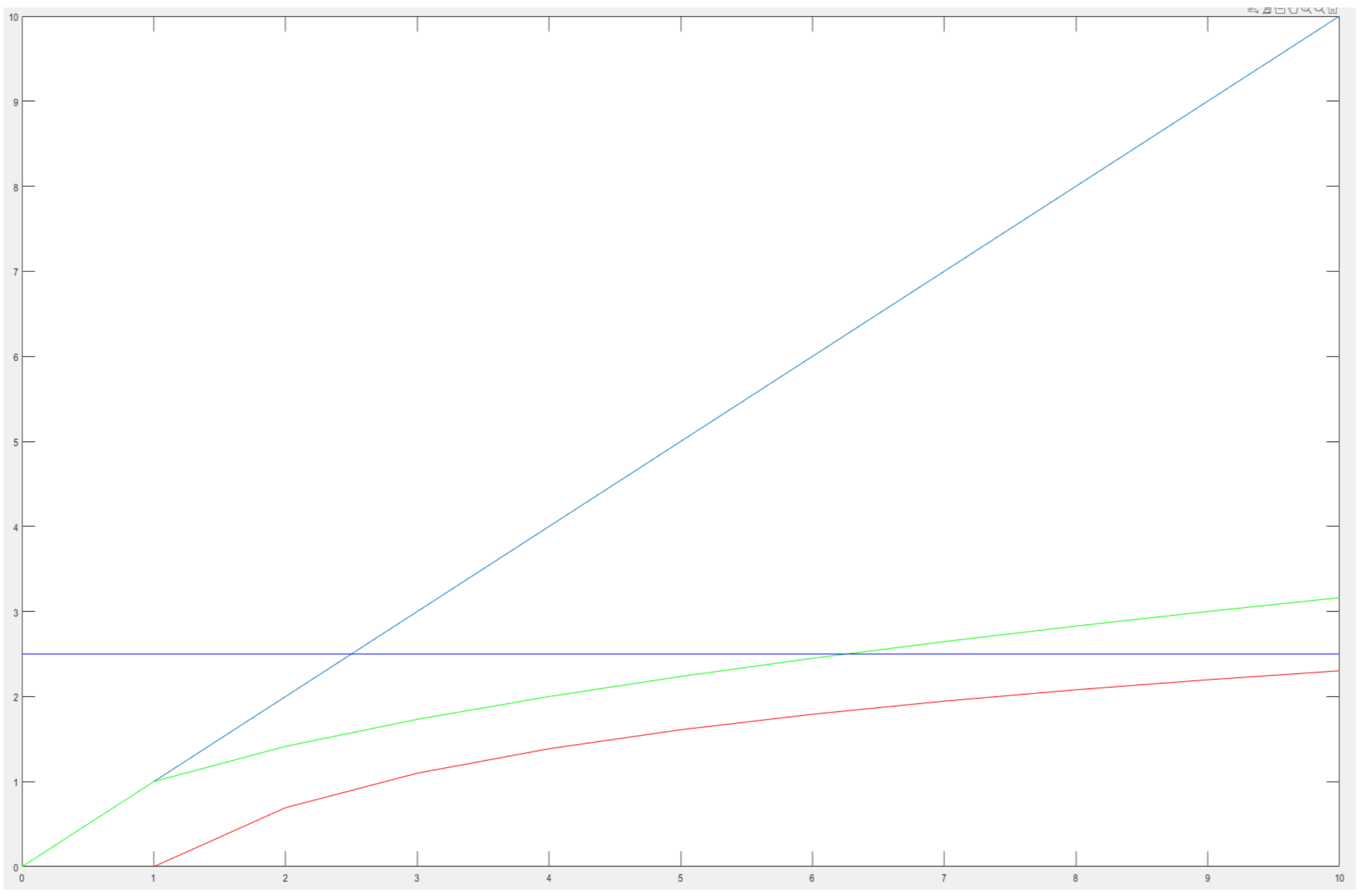
Scenario b:



Red: O(√N), purple: O(1), others: O(N)

Algo 1,2,3,4: O(N), O(N), O(1), O(√N)

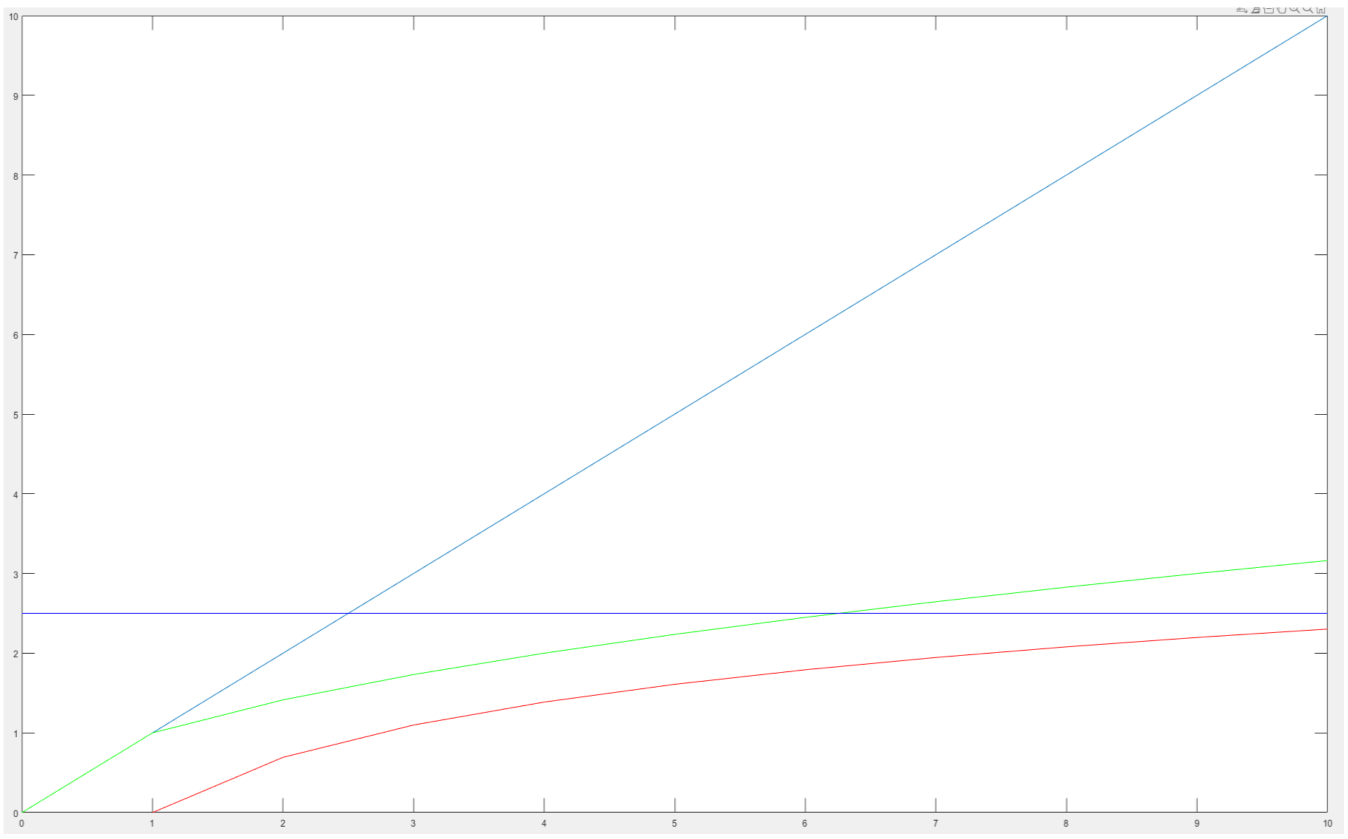
Scenario c:



Red: O(logN), green: O(√N), blue: O(N), purple: O(1)

Algo 1,2,3,4: O(N), O(N), O(logN), O(√N)

Scenario d:



Red: O(logN), green: O(√N), blue: O(N), purple: O(1)

Algo 1,2,3,4: O(N), O(N), O(logN), O(√N)

These plots should have the same organization as plots in step 4 however, the compiler that I used does not work with bigger N sizes. I could not use very big collection sizes. So, my plots are very small pieces of certain areas of the theoretical plots. I can understand it from the increment ratio. For instance, the o(N) is increased faster than log N and sqrt N and I can see that the relation on my plots. Also, we see drastic changes sometimes because of the small number of N sizes. If I could use bigger N sizes, my plots will be closer to theoretical plots.