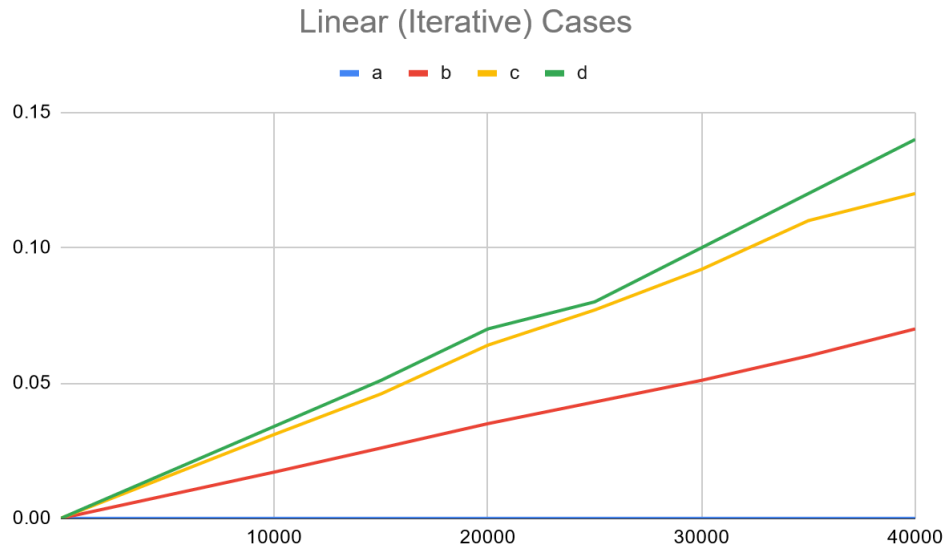
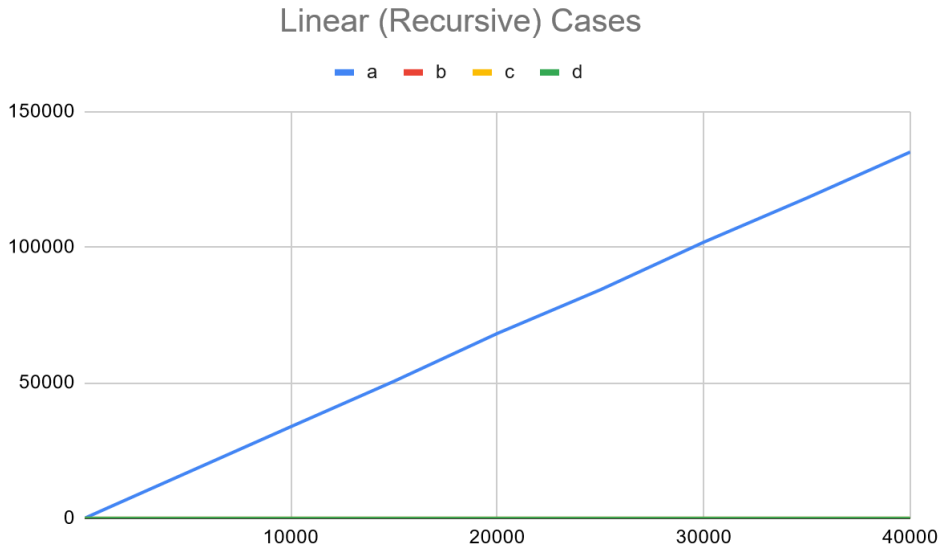
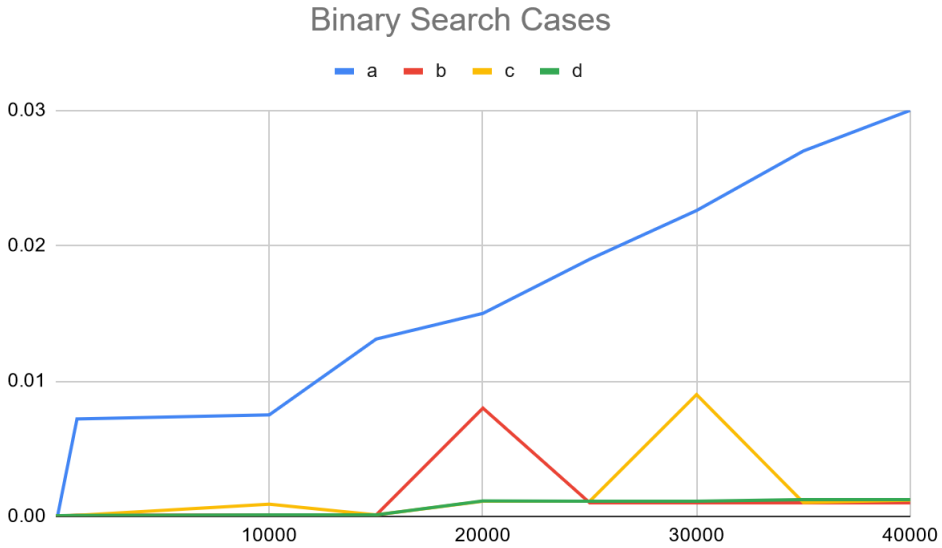
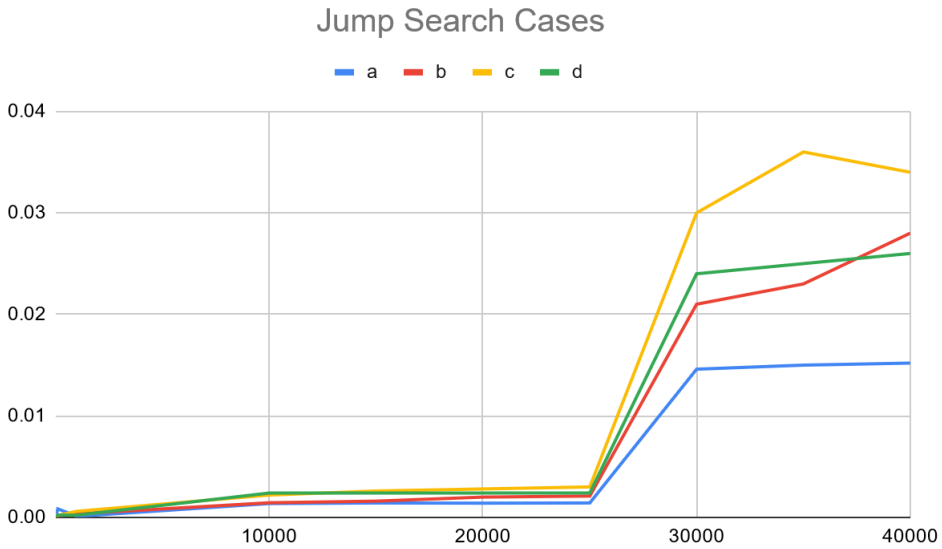
N	Linear (Iterative)				Linear (Recursive)				Binary Search				Jump Search			
	a	b	c	d	a	b	c	d	a	b	c	d	a	b	c	d
10	0.000008	0.00001	0.000026	0.000003	30	0.000003	0.000007	0.00007	0.00003	0.00001	0.000003	0.000026	0.000058	0.000048	0.000082	0.000024
100	0.000008	0.00002	0.000033	0.000003	336	0.000004	0.000007	0.0007	0.0001	0.00001	0.000004	0.000005	0.00084	0.00015	0.0002	0.000022
1000	0.000008	0.00017	0.000031	0.000034	3401	0.000033	0.000007	0.007	0.0072	0.00008	0.000076	0.000074	0.000106	0.00035	0.0006	0.000022
10000	0.000008	0.00171	0.0031	0.0034	33798	0.0034	0.0007	0.074	0.0075	0.0001	0.0009	0.0011	0.00136	0.00144	0.0022	0.0024
15000	0.000008	0.0026	0.0046	0.0051	50551	0.0054	0.0011	0.131	0.0131	0.0001	0.000112	0.000112	0.00142	0.0016	0.0026	0.0024
20000	0.000008	0.0035	0.0064	0.007	68180	0.0074	0.0014	0.15	0.015	0.008	0.000112	0.000114	0.00140	0.002	0.0028	0.0024
25000	0.000008	0.0043	0.0077	0.008	84309	0.1	0.02	0.19	0.019	0.001	0.000112	0.000112	0.00142	0.0021	0.003	0.0024
30000	0.000008	0.0051	0.092	0.1	101840	0.12	0.021	0.225	0.0226	0.001	0.0009	0.000112	0.0146	0.021	0.03	0.024
35000	0.000008	0.06	0.11	0.12	118155	0.13	0.021	0.27	0.027	0.001	0.000102	0.000124	0.015	0.023	0.036	0.025
40000	0.000008	0.07	0.12	0.14	135086	0.15	0.022	0.3	0.03	0.001	0.00012	0.000124	0.0152	0.028	0.034	0.026

4.




Linear (Recursive) Cases
a
b
c
d
150000
100000
50000
0
10000
20000
30000
40000


Binary Search Cases
a
b
c
d
0.03
0.02
0.01
0.00
10000
20000
30000
40000


Jump Search Cases
a
b
c
d
0.04
0.03
0.02
0.01
0.00
10000
20000
30000
40000

5.1 Computer Specifications

Processor: Intel(R) Core(TM) i7-2630QM CPU @ 2.00GHz 2.00 GHz

RAM: 8 GB

OS: Microsoft Windows 10 Home

5.2 Theoritical Scenarios for Linear (Iterative)

Worst case: O(n)

Average case: O(n/2)

Best case: O(1)

Expected output for A is O (1) because the searched element is at the beginning. For B, it's O (n / 2) because the searched element is in the middle. O (n) for C and D because the searched element is at the end or does not exist.

Theoritical Scenarios for Linear (Recursive)

Worst case: O(n)

Average case: O(n/2)

Best case: O(1)

Expected output for D is O (1) because the searched element is at the end. For C, it's O (n / 2) because the searched element is in the middle. O (n) for A and B because the searched element is at the beginning or does not exist.

Theoritical Scenarios for Binary Search

Worst case: O(logN)

Average case: O(logN)

Best case: O(1)

For A, C, and D, the output is log (N) because the searched element does not have a leading end. The algorithm starts searching from the middle and proceeds accordingly, so the output for B is O (1).

Theoritical Scenarios for Jump Search

Worst case: O(√ n)

Average case: O(√ n)

Best case: O(1)

O (1) for A because the searched element is at the beginning. For B, C, and D, O(√ n) because the searched element is in the middle, the probe or does not exist and the algorithm finds it by processing approximately the square root of N.

5.3 Experimental Scenarios for Linear (Iterative)

Worst case: (d) the key does not exist in the collection

Average case: (b) the key is around the middle

Best case: (a) the key is close to the beginning

Experimental Scenarios for Linear (Recursive)

Worst case: (a) the key is close to the beginning

Average case: (d) the key does not exist inthe collection

Best case: (d) the key does not exist in the collection

Experimental Scenarios for Binary Search

Worst case: (a) the key is close to the beginning

Average case: (c) the key is close to the end

Best case: (d) the key does not exist in the collection

Experimental Scenarios for Jump Search

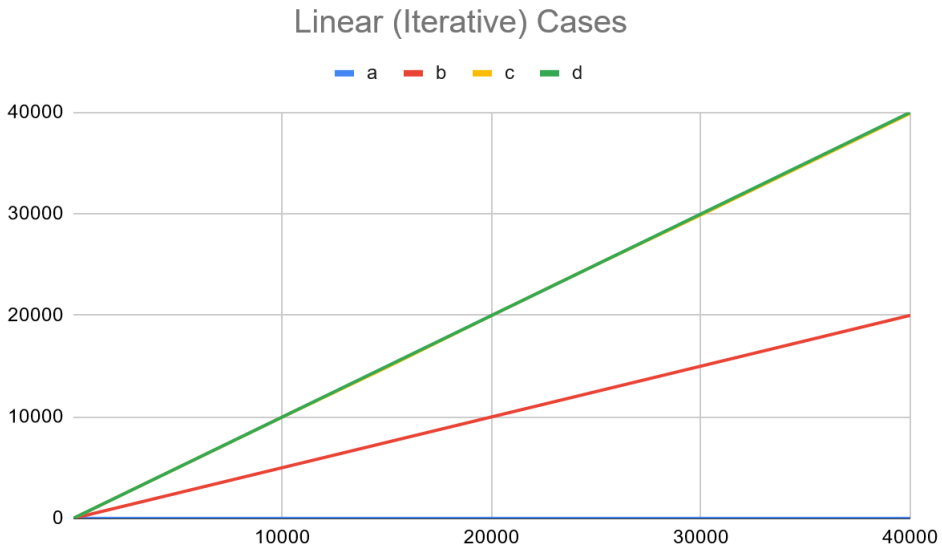
Worst case: (c) the key is close to the end

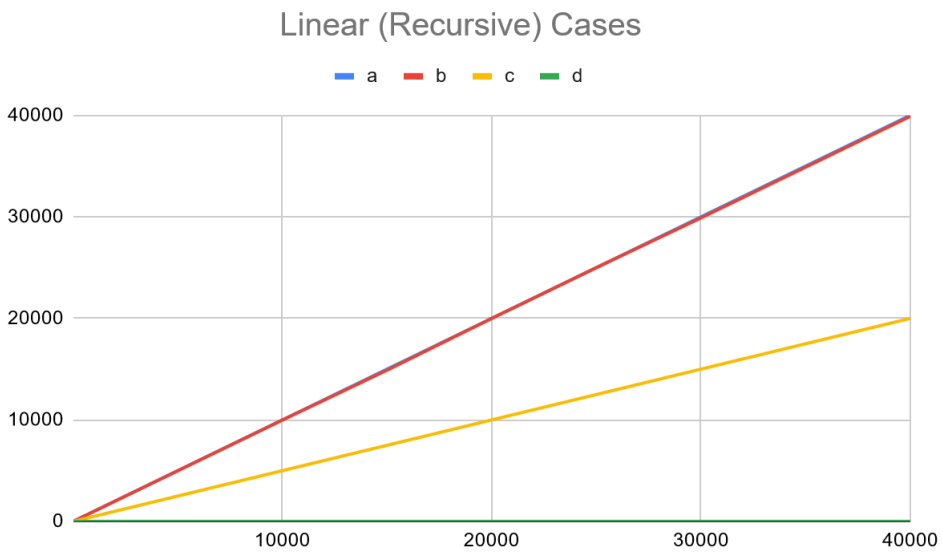
Average case: (d) the key does not exist in the collection

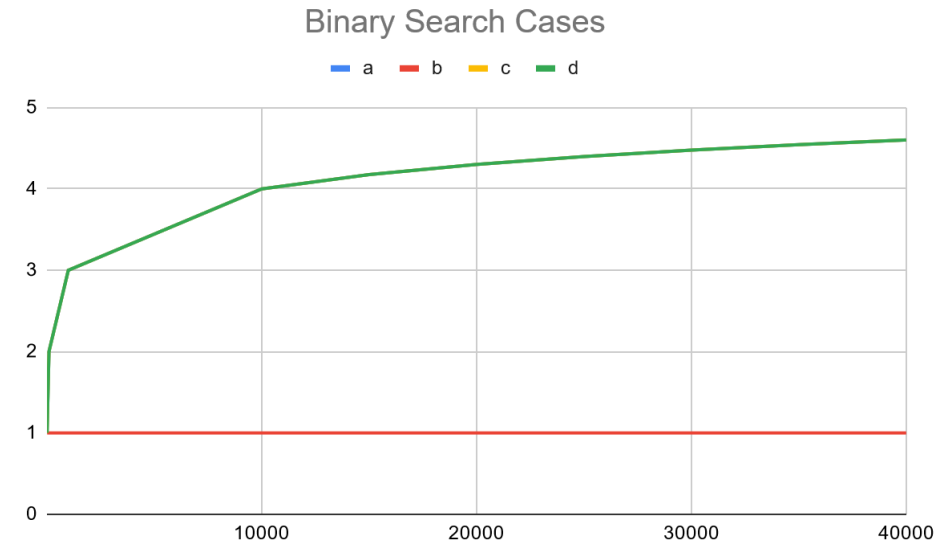
Best case: (a) the key is close to the beginning

My experimental data did not completely match the theoretical data. There were overlapping or approaching situations. However, it did not completely overlap. I think this is due to the hardware the machine has, such as RAM, and that a small number of inputs manipulate the values even if we multiply them in for loop. The change in the location of the key in the cases also contributes to the manipulation of the expected situation. Random selection of values also causes differences in the expected array table and manipulations in run time.

5.4







Linear algorithms are very close to their theoretical values. However, sufficient logarithmic increase is not seen in binary search algorithm. In the jump search algorithm, the initial values of the graph do not match. However, it later progressed towards a similar appearance. I think that the inconsistencies that occur here are due to the probability of the same number and the insufficient randomness of the numbers, the hardware of the computer, the change in the location of the key value in situations.