Linear Search Results (in milliseconds)

N	a	b	c	d
5	0.0006	0.0002	0.0001	0.0001
5^2	0.0001	0.0001	0.0001	0.0001
5^3	0.0003	0.0002	0.0002	0.0002
5^4	0.0002	0.0004	0.0006	0.0005
5^5	0.0008	0.0015	0.0022	0.0025
5^6	0.0035	0.0061	0.01	0.0122
5^7	0.0154	0.0302	0.0499	0.06
5^8	0.0756	0.1494	0.2531	0.3008
5^9	0.3752	0.7481	1.3246	1.9603
5^10	2.1777	4.0052	6.4801	7.5955

Recursive Linear Search Results (in milliseconds)

N	a	b	c	d
5	0.0004	0.0001	0.0001	0.0001
5^2	0.0002	0.0002	0.0001	0.0002
5^3	0.0006	0.0003	0.0001	0.0003
5^4	0.0031	0.0008	0.0003	0.0066
5^5	0.0453	0.0037	0.0013	0.027
5^6	0.2191	0.0148	0.0056	0.1313
5^7	1.0674	0.081	0.026	0.6065
5^8	5.2073	0.6119	0.1418	3.057
5^9	32.6394	14.9828	1.7192	34.0508
5^10	207.482	77.8346	17.2381	199.56

Binary Search Results (in milliseconds)

N	a	b	c	d
5	0.00001	0.000004	0.000008	0.000008
5^2	0.000009	0.000003	0.00001	0.000014
5^3	0.000006	0.000004	0.000017	0.000019
5^4	0.000019	0.000004	0.000025	0.000026
5^5	0.000022	0.000004	0.000029	0.000033
5^6	0.000008	0.000004	0.000034	0.000038
5^7	0.000007	0.000004	0.000038	0.000047
5^8	0.000007	0.000004	0.00004	0.000054
5^9	0.000007	0.000004	0.000037	0.000062
5^10	0.000008	0.000006	0.00004	0.000069

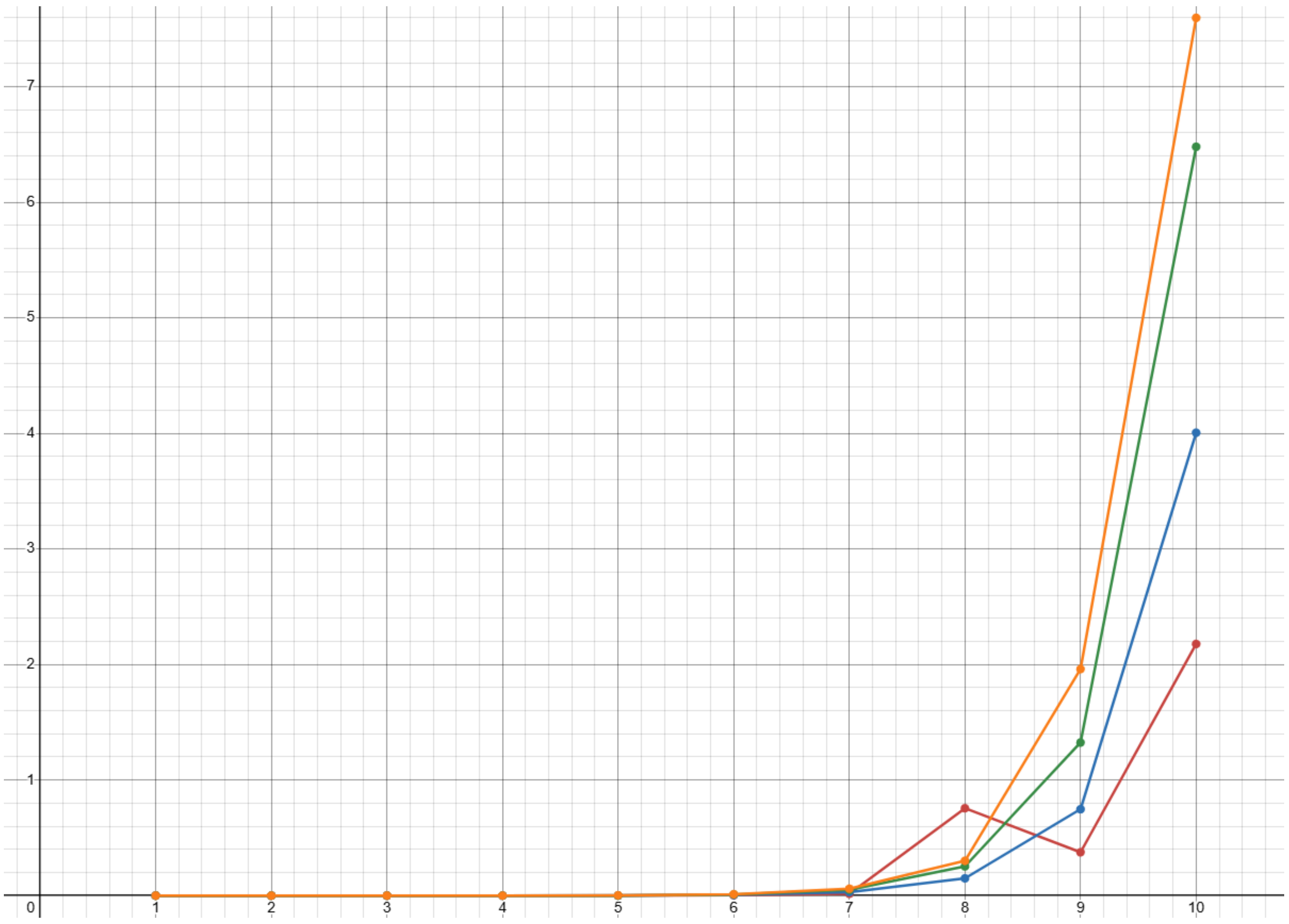
Jump Search Results (in milliseconds)

N	a	b	c	d
5	0.0006	0.0002	0.0002	0.0002
5^2	0.0001	0.0002	0.0002	0.0003
5^3	0.0003	0.0005	0.0006	0.0005
5^4	0.0005	0.0007	0.001	0.0006
5^5	0.0006	0.0008	0.0009	0.0008
5^6	0.0016	0.0017	0.0026	0.002
5^7	0.0026	0.0019	0.0035	0.0035
5^8	0.0036	0.0055	0.0078	0.0073
5^9	0.0129	0.0088	0.0139	0.0214
5^10	0.0187	0.0259	0.0382	0.0449

Randomized Linear Search Results (in milliseconds)

N	a	b	c	d
5	0.0019	0.0014	0.001	0.0009
5^2	0.001	0.0012	0.0015	0.0009
5^3	0.0014	0.002	0.0015	0.0027
5^4	0.0073	0.0067	0.0052	0.0067
5^5	0.0379	0.0209	0.0135	0.0316
5^6	0.1713	0.0932	0.1	0.1727
5^7	0.3191	1.0572	0.3132	0.8421
5^8	5.0484	2.7787	1.4242	5.2453
5^9	22.3895	16.7893	7.3928	29.6046
5^10	129.547	83.7909	38.1889	149.932

Plot for Linear Search



y-axis --> Running time in milliseconds

x-axis --> N values in a way that the values correspond to the power of 5; for instance, the value 2 represents the N value 5^2

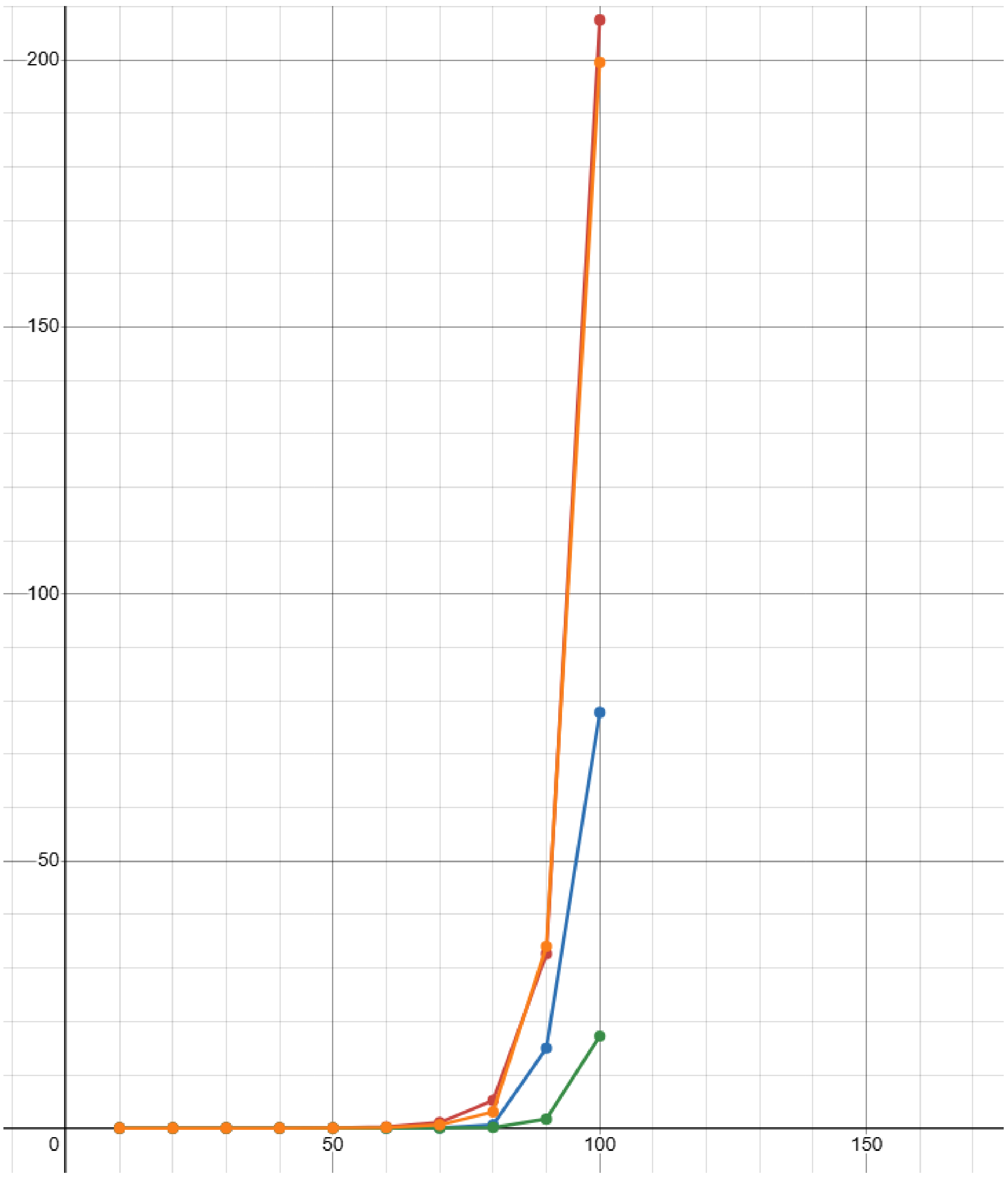
red line --> a

blue line --> b

green line --> c

orange line --> d

Plot for Recursive Linear Search



y-axis --> Running time in milliseconds

x-axis --> N values in a way that the values divided by 10 correspond to the power of 5; for instance, the value 20 represents the N value 5^2

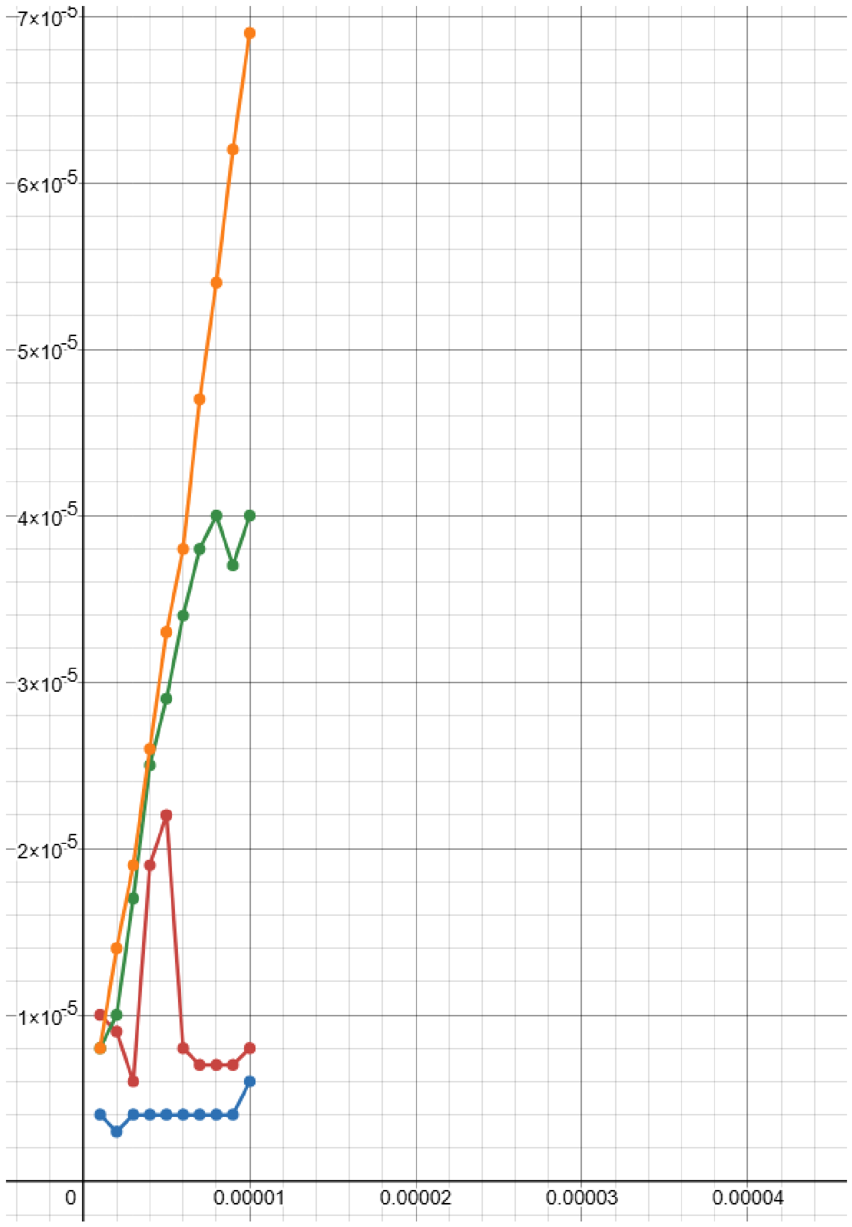
red line --> a

blue line --> b

green line --> c

orange line --> d

Plot for Binary Search



y-axis --> Running time in milliseconds

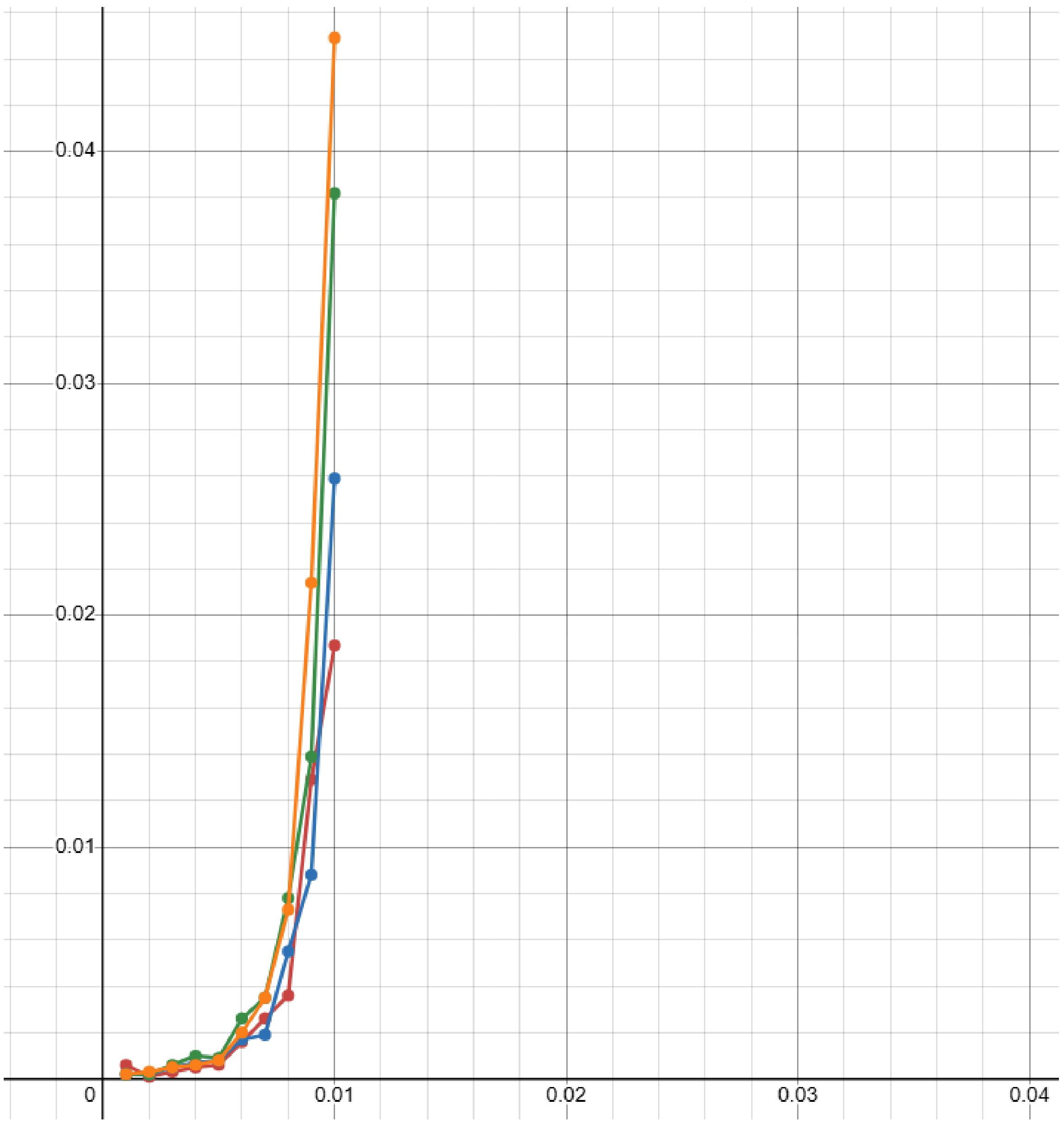
x-axis --> N values in a way that the values multiplied by 1000000 correspond to the power of 5; for instance, the value 0.000002 represents the N value 5^2 red line --> a

blue line --> b

green line --> c

orange line --> d

Plot for Jump Search



y-axis --> Running time in milliseconds

x-axis --> N values in a way that the values multiplied by 1000 correspond to the power of 5; for instance, the value 0.002 represents the N value 5^2

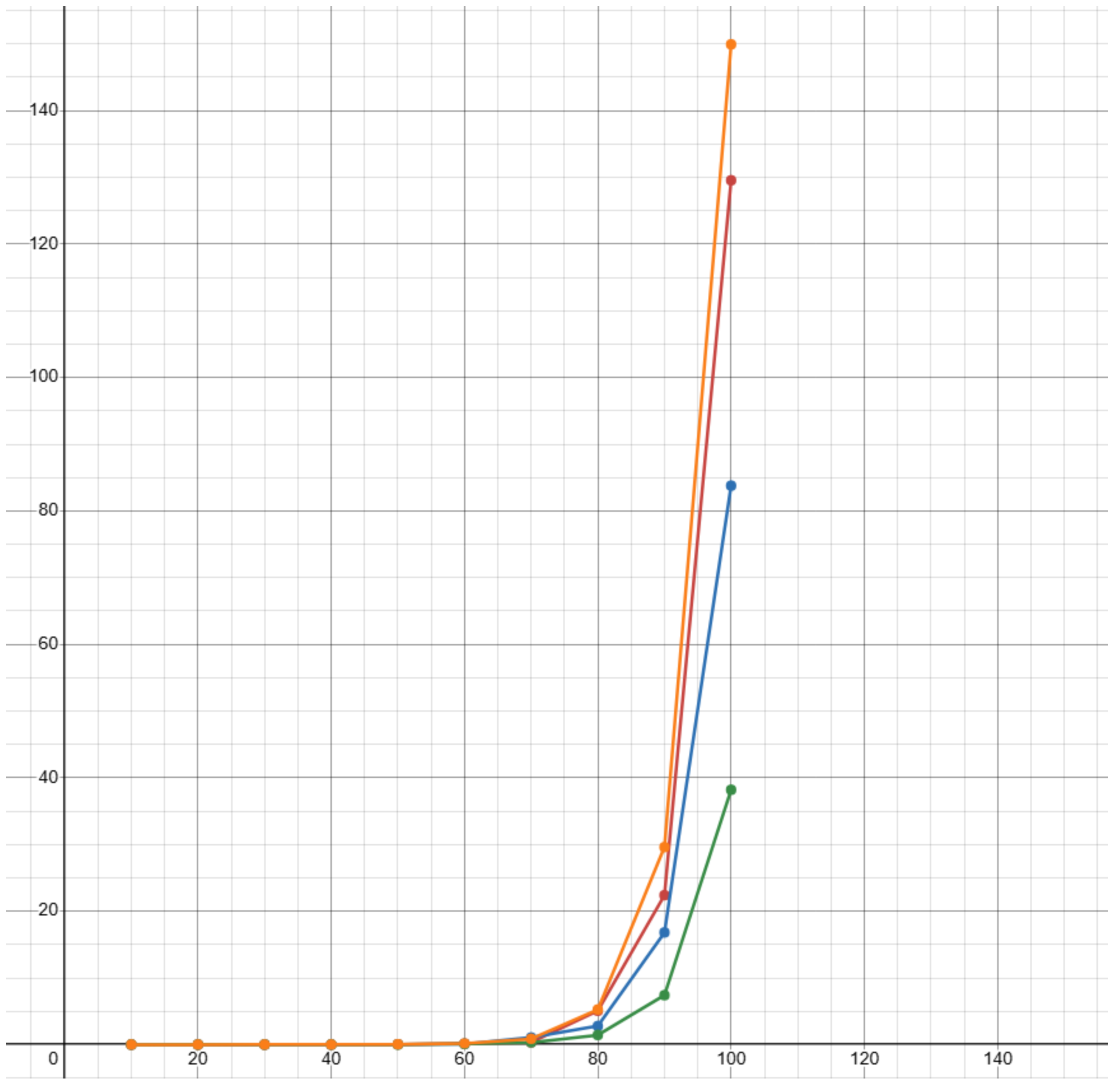
red line --> a

blue line --> b

green line --> c

orange line --> d

Plot for Randomized Linear Search



y-axis --> Running time in milliseconds

x-axis --> N values in a way that the values divided by 10 correspond to the power of 5; for instance, the value 20 represents the N value 5^2

red line --> a

blue line --> b

green line --> c

orange line --> d

Comments on My Results

The specifications of my computer are as in the following:

Processor --> AMD Ryzen 5 7500F

Motherboard --> Asus Prime A620M-E CSM

Graphics card --> Asus Dual GeForce RTX 4070

RAM --> Corsair Vengeance 16 GB (2x8GB) DDR5

SSD --> Adata Legend 800 500GB PCle Gen4 x4 M.2 SSD

Operating System --> Windows 11

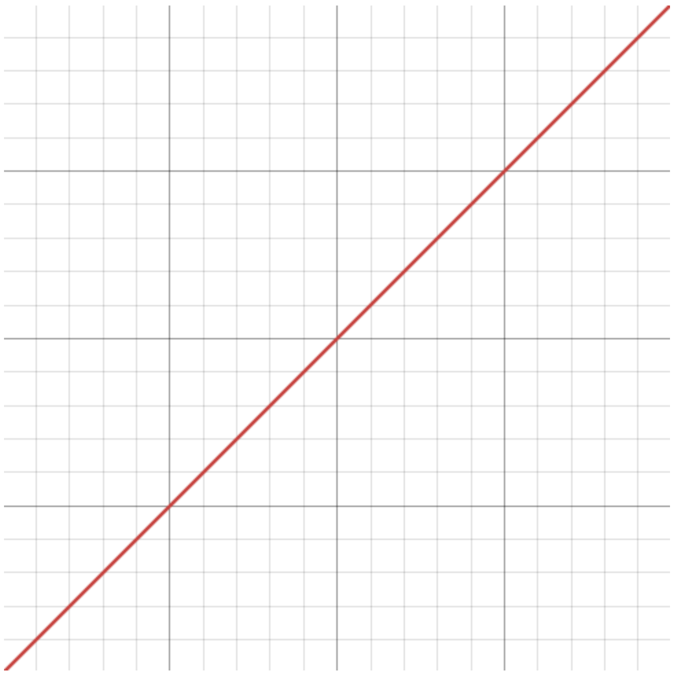
The theoretical worst, average and best cases for each algorithm is as in the following:

Search Algorithms	Worst-case	Best-case	Average-case
Linear	O(n)	O(1)	O(n)
Recursive Linear	O(n)	O(1)	O(n)
Binary	O(logn)	O(1)	O(logn)
Jump	O(√n)	O(1)	O(√n)
Random Linear	O(n)	O(1)	O(n)

As it can be seen from the table, best case scenario for all of them has a constant running time. For the linear, recursive linear and random linear search; the worst and average running times of them increases in a linear way as the input size increases. For the binary search, the worst and average running times increase significantly slower linear search algorithms. For an input size "n", the rate of increase in the running time is "log(n)". For the jump search, the worst and average running times increase faster than the binary search but slower than the linear search algorithms. For an input size "n", the rate of increase in the running time is "√n".

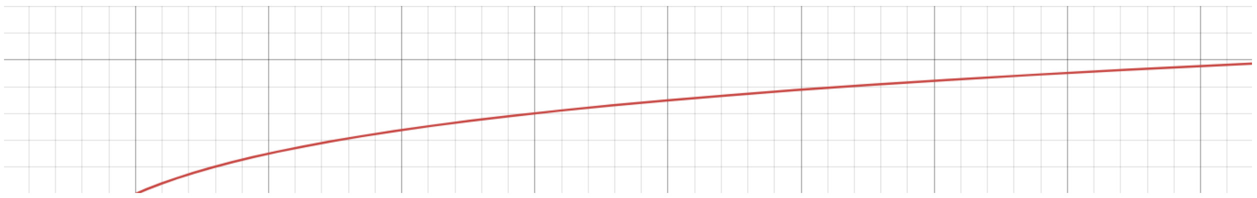
For the linear and recursive linear search algorithms, the rate of increase starts to get linear for the sizes of inputs greater than 5^4. The reason why the rate of increase is inconsistent with the input sizes less than 5^4 is because optimization done by the compiler plays a more significant role in determining what the running time is going to be. Since the difference in running time caused by this optimization is small, it cannot be seen when the input sizes get greater. The same situation applies for the randomized linear search for the sizes of inputs greater than 5^3. For the binary search, there is no consistency even as the size of inputs gets close to 5^10. The reason for this is because the efficiency of this algorithm causes the running time to be genuinely small even for the big input sizes. Since the optimization done by the compiler plays a more significant role in determining the small running times, there is no consistency throughout the algorithm up to input sizes 5^10. For the jump search, the rate of increase starts to get "√n" for the sizes of inputs greater than 5^3. The reason for the inconsistency for the sizes of inputs less than 5^3 is again caused by the optimization done by the compiler as explained before.

The plot for worst and average running times of linear, recursive linear and randomized linear search algorithms is as in the following:



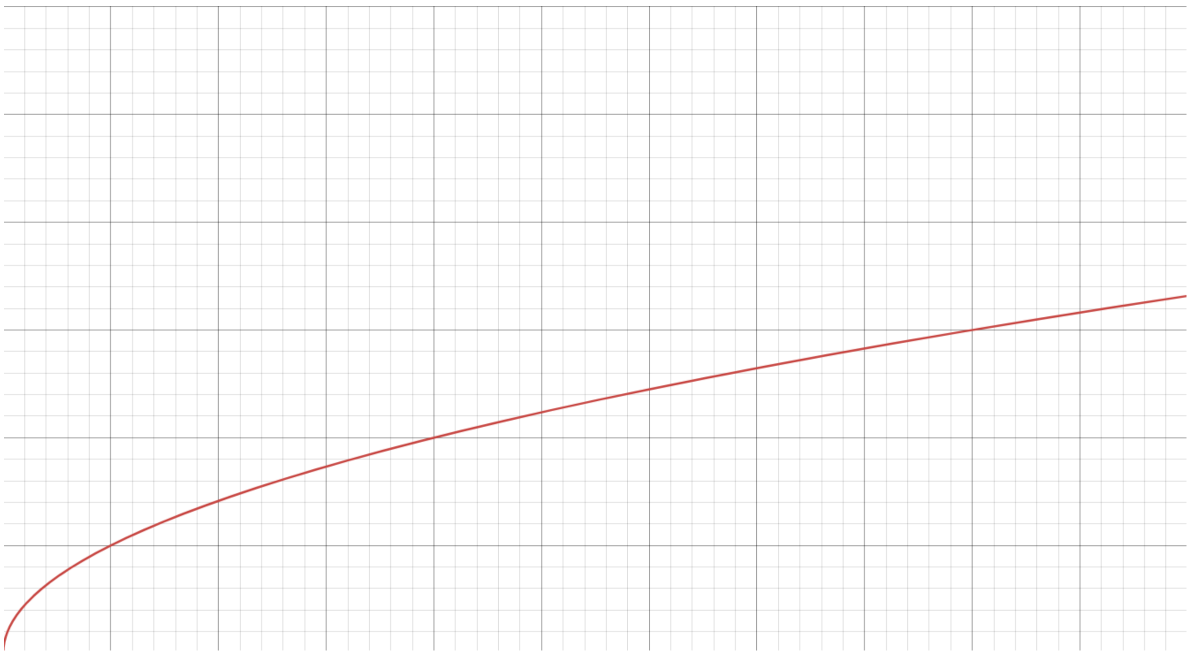
It can be seen from the plots done using the data acquired that these plots start to get consistent after input sizes greater than 5^4 for the linear and recursive search algorithms. It starts to get consistent after input sizes greater than 5^3 for the randomized linear search algorithm. The reason for the inconsistencies is caused by the optimization done by the compiler as explained before.

The plot for the worst and average running time of binary search is as in the following:



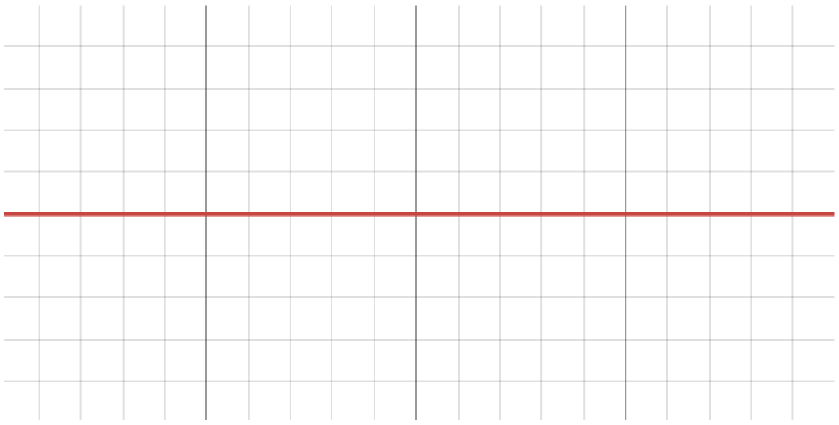
There is no consistency for the binary search algorithm in the plots done by using the data acquired. It is not possible to deduce a rate of increase of "log(n)" from the plots for any of the input sizes up to 5^10. This is again caused by the optimization done by the compiler and the running times being too small even for the input size 5^10 as explained before.

The plot for the worst and average running time of jump search is as in the following:



There is consistency in the plots done by using the data acquired for the input sizes greater than 5^3. The inconsistency is again caused by the optimization done by the compiler as explained before.

The plot for best running time of all these five algorithms is as in the following:



Since there is no perfectly flat line for any of the plots that are done by using the data acquired, there is no way for me to compare this theoretical plot for the best running time with my own plots. Thus, there is no comment I can provide.