Data Structure and Algorithms

Laboratory 5

Submitted by:
Centina
Delera
Discutido
Gangoso
Laguting
Purcia

November, 2023

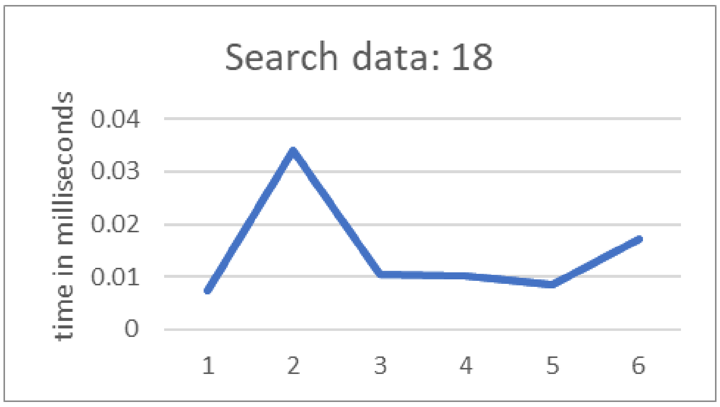
Search Algorithms Analysis Worksheet

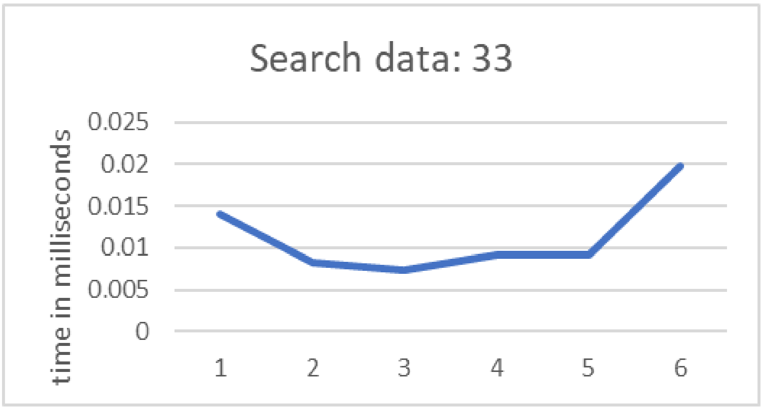
		Linear	Binary	Ternary	Exponential	Interpolation	Jump
Target set	**Search data**	**Time in Milliseconds**					
100	**18**	0.00740	0.0340	0.0106	0.0101	0.0084	0.0172
	33	0.0140	0.00830	0.0074	0.0092	0.0092	0.0198
	36	0.0089	0.04470	0.00840	0.0109	0.0060	0.0107
	46	0.0091	0.00740	0.00910	0.0103	0.0068	0.0128
	80	0.0099	0.00800	0.0147	0.0095	0.0071	0.0293
1,000	**118**	0.0191	0.0072	0.0097	0.0112	0.0061	0.0117
	428	0.0291	0.0469	0.0159	0.0138	0.0118	0.0152
	440	0.1793	0.0227	0.0758	0.0652	0.0457	0.0212
	608	0.1623	0.0159	0.0205	0.0246	0.0095	0.0223
	663	0.1695	0.0144	0.0224	0.0309	0.0114	0.0321
10,000	**710**	0.1779	0.0214	0.1318	0.0627	0.0614	0.0842
	1266	0.3119	0.0196	0.0288	0.0262	0.0106	0.0429
	3341	0.8222	0.0198	0.0341	0.0283	0.0089	0.0438
	5725	1.4357	0.0181	0.0222	0.0317	0.0118	0.0580
	7253	1.7634	0.0175	0.0296	0.0400	0.0094	0.0669

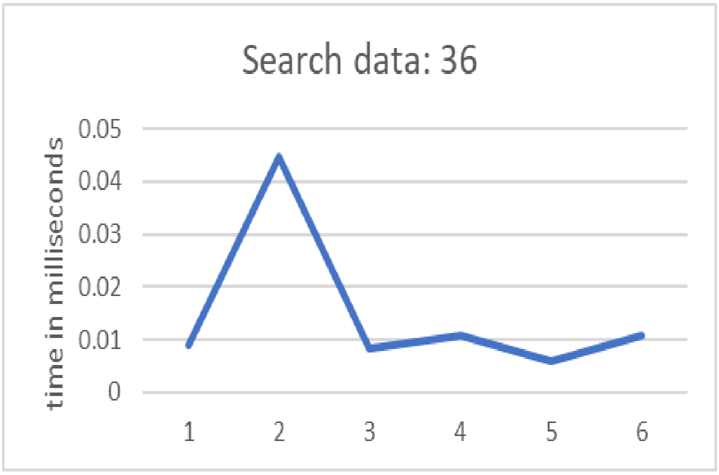
1 - Linear
2 - Binary
3 - Ternary
4 - Exponential
5 - Interpolation
6 - Jump

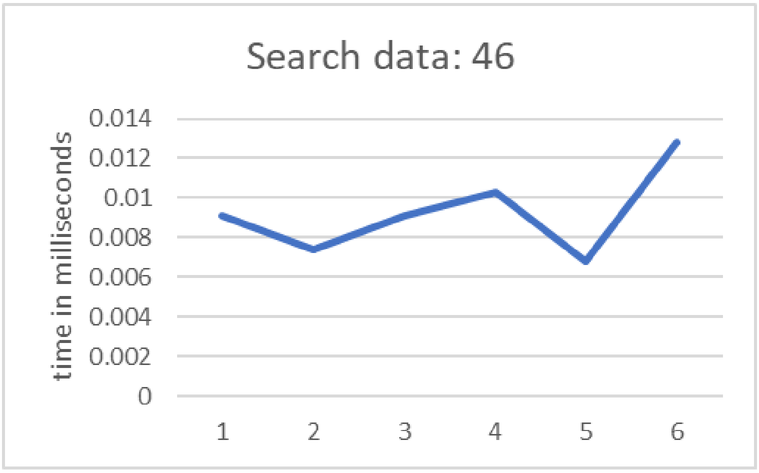
Search Algorithms Graphs

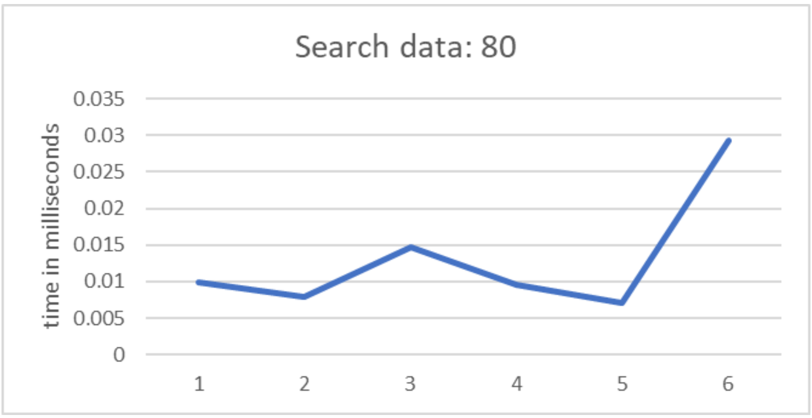
1-100



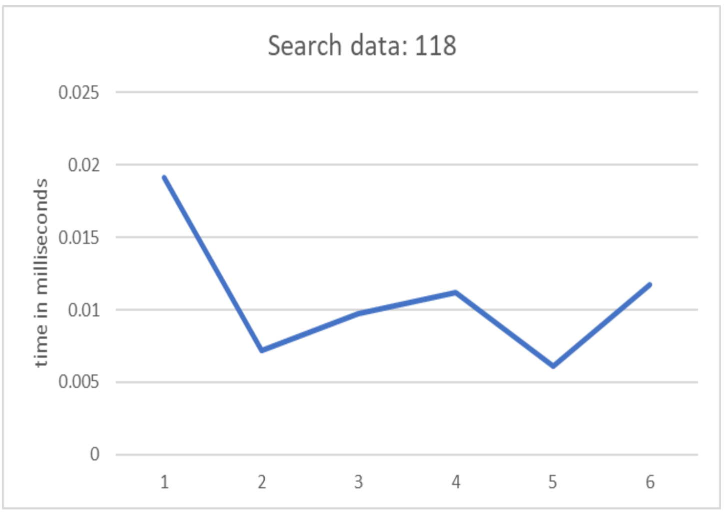


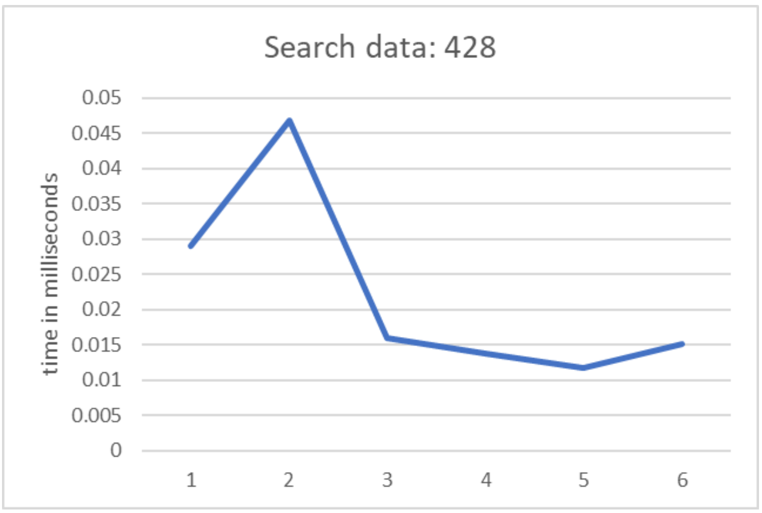


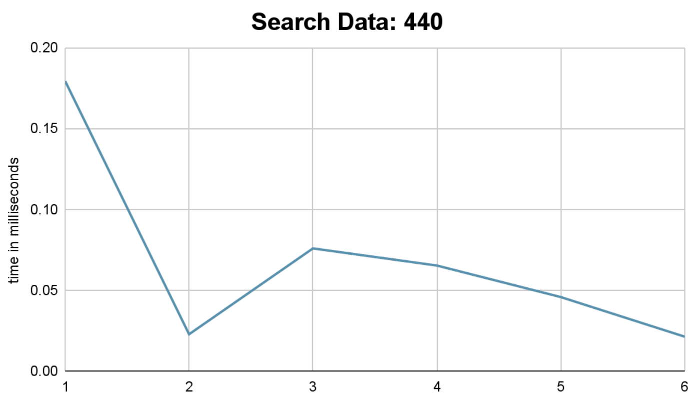


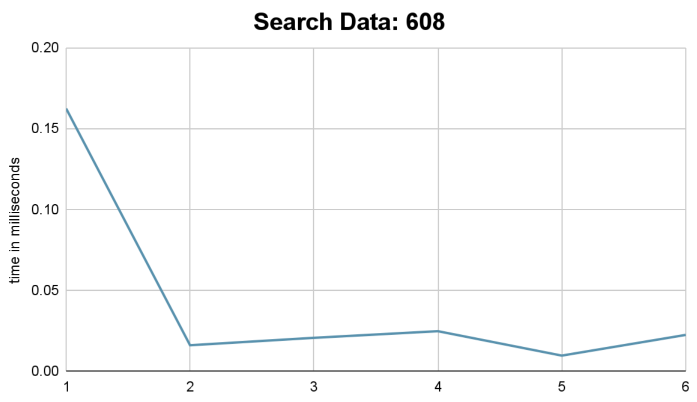


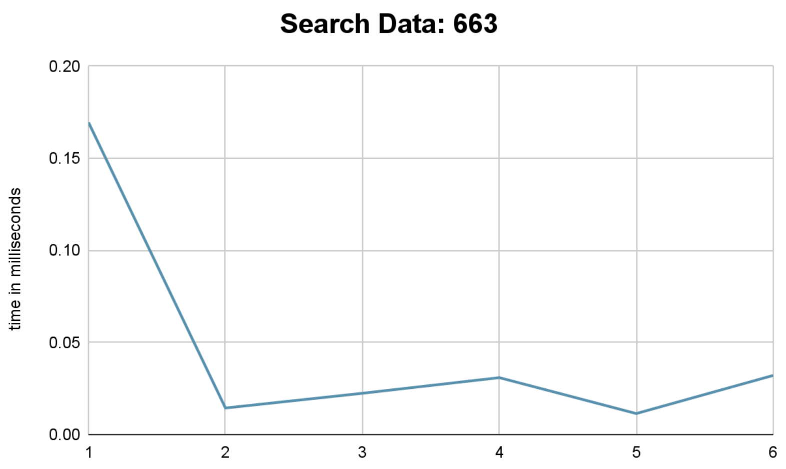
1-1000


Search data: 118
time in milliseconds
0.025
0.02
0.015
0.01
0.005
0
1
2
3
4
5
6


Search data: 428
time in milliseconds
0.05
0.045
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
1
2
3
4
5
6

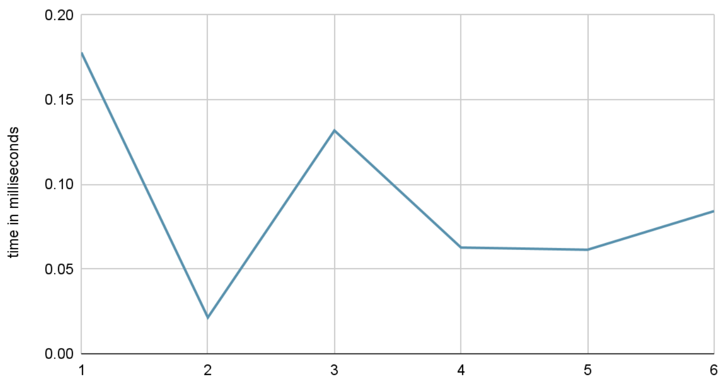

Search Data: 440
time in milliseconds
0.20
0.15
0.10
0.05
0.00
1
2
3
4
5
6


Search Data: 608
time in milliseconds
0.20
0.15
0.10
0.05
0.00
1
2
3
4
5
6

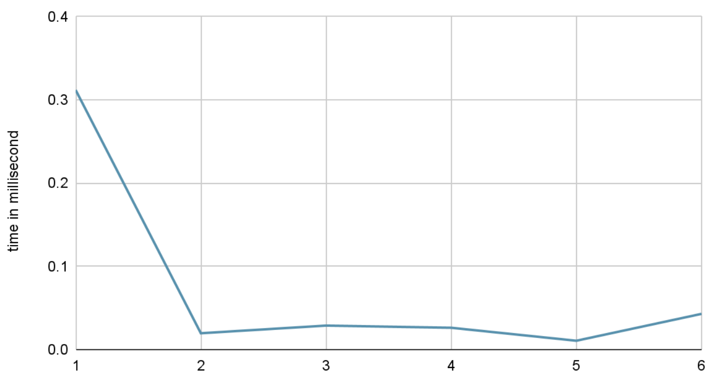

Search Data: 663
time in milliseconds
0.20
0.15
0.10
0.05
0.00
1
2
3
4
5
6

1-10000

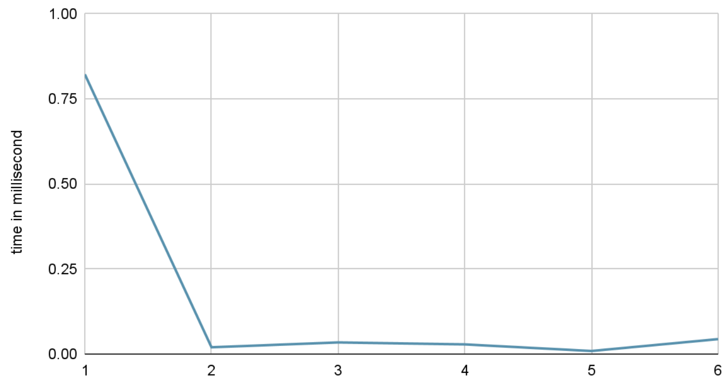
Search Data: 710



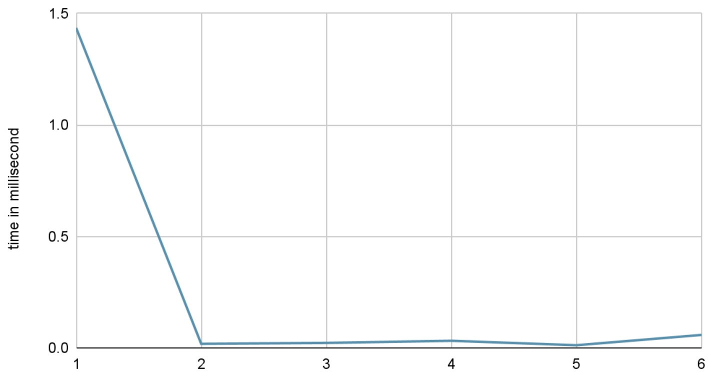
Search Data: 1266



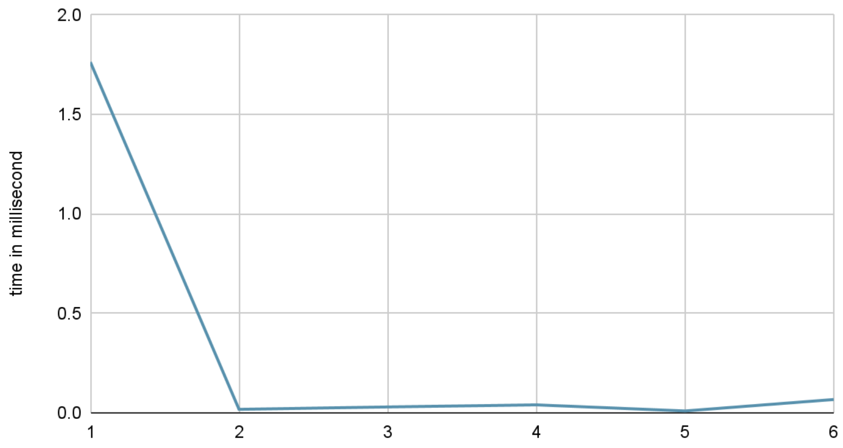
Search Data: 3341



Search Data: 5725



Search Data: 7253

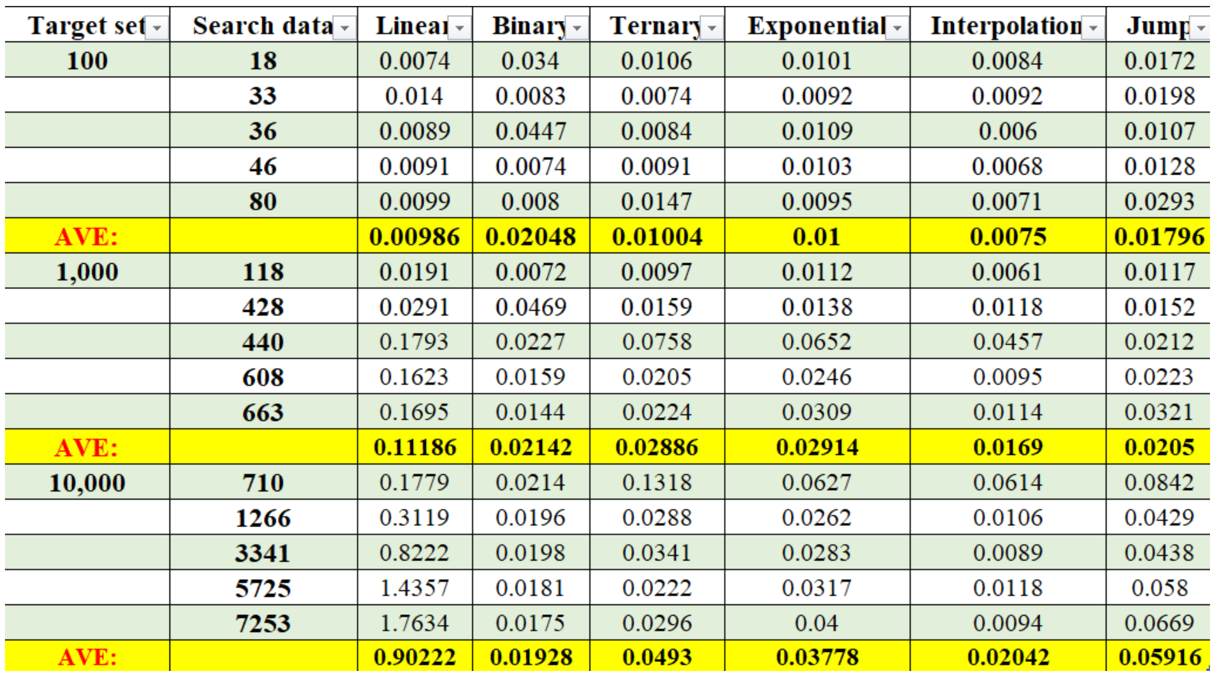


Github Link of the Dataset Generator:

https://github.com/peewweee/Search-Algorithms

Analysis and Conclusion

A. Which search algorithm performed the best overall?



Target set	Search data	Linear	Binary	Ternary	Exponential	Interpolation	Jump
100	18	0.0074	0.034	0.0106	0.0101	0.0084	0.0172
	33	0.014	0.0083	0.0074	0.0092	0.0092	0.0198
	36	0.0089	0.0447	0.0084	0.0109	0.006	0.0107
	46	0.0091	0.0074	0.0091	0.0103	0.0068	0.0128
	80	0.0099	0.008	0.0147	0.0095	0.0071	0.0293
AVE:		0.00986	0.02048	0.01004	0.01	0.0075	0.01796
1,000	118	0.0191	0.0072	0.0097	0.0112	0.0061	0.0117
	428	0.0291	0.0469	0.0159	0.0138	0.0118	0.0152
	440	0.1793	0.0227	0.0758	0.0652	0.0457	0.0212
	608	0.1623	0.0159	0.0205	0.0246	0.0095	0.0223
	663	0.1695	0.0144	0.0224	0.0309	0.0114	0.0321
AVE:		0.11186	0.02142	0.02886	0.02914	0.0169	0.0205
10,000	710	0.1779	0.0214	0.1318	0.0627	0.0614	0.0842
	1266	0.3119	0.0196	0.0288	0.0262	0.0106	0.0429
	3341	0.8222	0.0198	0.0341	0.0283	0.0089	0.0438
	5725	1.4357	0.0181	0.0222	0.0317	0.0118	0.058
	7253	1.7634	0.0175	0.0296	0.04	0.0094	0.0669
AVE:		0.90222	0.01928	0.0493	0.03778	0.02042	0.05916

Table 1: Search Algorithms Analysis Worksheet Average

The following search algorithms are represented in the table: Linear, Binary, Ternary, Exponential, Interpolation, and Jump. The data displayed in the table represents the average search results for each of the three different target set sizes: 100, 1,000, and 10,000. The computational operations or comparisons that need to be performed by each algorithm in order to identify a particular target within the dataset are represented by the search data. Consistent lower search data values imply a higher level of efficiency since they suggest fewer actions were required, on average, to locate the target.

Upon closer inspection of the table's calculated averages, it becomes clear that the **Binary Search Algorithm** demonstrates the most consistent lowest average search data throughout the specified target set sizes. To be more specific:

- At Target 100, the average search data produced by Binary Search, despite it being the slowest at all of the search algorithm, it still considered performing a better performance obtaining 0.2048 milliseconds.

- At Target 1,000, the average search data obtained using Binary Search is 0.02142ms, which is a value that is lower than the values obtained through Linear, Ternary, and Exponential searches at this target size. Though interpolation and jump searches were much lower in the second target set, it is still not enough proof to consider it with the best performance since they are not consistent with other target sizes.
- For Target 10,000, the average search data obtained using Binary Search is lower (0.01928ms) than the averages obtained using Linear Search, Ternary Search, Exponential Search, Interpolation Search, and Jump Search for this target size.

Based on a thorough analysis of the numbers, The Binary Search algorithm always keeps one of the consistent lowest average search data across the board for any target set sizes that are provided. The other search algorithm on the other hand, can not be considered as having a greater performance since it is proven that they are not consistent with its average value and it gives a significantly large time interval in comparison to its performance to each target. The consistency of Binary Search Algorithm illustrates that the method is excellent in effectively locating targets inside datasets, particularly when compared to the performance of other algorithms that are presented in the dataset.

Therefore, on the basis of the data that was provided as well as the consistent presentation of lower average search data across different kinds of target set sizes, Binary Search has emerged as the most effective algorithm among those that were listed. The reason for this is that Binary Search, on average, takes less processing power than the other algorithms to locate the targets inside the datasets.

B. Did any search algorithms perform better on specific data sets?

The average time for each search algorithm was determined to assess which algorithm performed better on specific data sets.

Target Set 100

	Linear	**Binary**	**Ternary**	**Exponential**	**Interpolation**	**Jump**
Average time (ms)	0.00986	0.02048	0.01004	0.01	0.0075	0.01796

Table 2: Search Algorithms Averages for Target of 100

At a target size of 100, interpolation outperformed the other search algorithms with a time of 0.0075 ms, followed by linear with 0.00986 ms, exponential with 0.01 ms, ternary with 0.01004 ms, jump with 0.01796 ms, and binary with 0.02048 ms.

Target Set 1,000

	Linear	Binary	Ternary	Exponential	Interpolation	Jump
Average time (ms)	0.11186	0.02142	0.02886	0.02914	0.0169	0.0205

Table 3: Search Algorithms Averages for Target of 1,000

At a target size of 1000, interpolation again performed better than the other search algorithms with a time of 0.0169 ms, followed by jump with 0.0205 ms, binary with 0.02142 ms, ternary with 0.02886 ms, linear with 0.11186 ms, and exponential with 0.2914 ms.

Target Set 10,000

	Linear	Binary	Ternary	Exponential	Interpolation	Jump
Average time (ms)	0.90222	0.01928	0.0493	0.03778	0.02042	0.05916

Table 4: Search Algorithms Averages for Target of 10,000

At a target size of 10000, binary outperformed the other search algorithms with a time of 0.01928 ms, followed by interpolation with 0.02042 ms, exponential with 0.03778 ms, ternary with 0.0493 ms, jump with 0.05916 ms, and linear with 0.90222 ms.

In summary, the performance ranking of the search algorithms varies depending on the target size. Interpolation consistently performs well for smaller target sizes, while binary performs better for larger target sizes. The choice of the most suitable algorithm may depend on the specific characteristics of the data set and the size of the target.

C. How did the size of the data set affect the performance of the search algorithms?

After analyzing the effect of the size of the data set on the performance of search algorithms, several key insights were revealed. The relationship of the size of the data set and the performance of search can be examined from various perspectives.

Linear

Target set	100	1,000	10,000
Average time (ms)	0.00986	0.11186	0.90222

Table 5: Average time of Linear Search Algorithm

As the dataset size increased, linear search exhibited a linear increase in the time of search to find a specific element. Larger data sets resulted in longer search times. This is because this algorithm iterates through each element until a match is found. This can make linear search less efficient for larger-scale datasets.

Binary

Target set	100	1,000	10,000
Average time (ms)	0.02048	0.02142	0.01928

Table 6: Average time of Binary Search Algorithm

Based on the average search results in binary search, the algorithm had a better performance compared to linear search for larger data set sizes. As the data set size increased, the algorithm made a relatively small impact on search time. Binary search is particularly advantageous for large data sets, making it suitable for handling large amounts of data.

Ternary

Target set	100	1,000	10,000
Average time (ms)	0.01004	0.02886	0.0493

Table 7: Average time of Ternary Search Algorithm

Similar to binary search, ternary search is efficient for larger size of data set. The performance was better than linear search but may be slightly slower than binary search. Since this search algorithm divides the dataset into three parts instead of two, it might exhibit slightly higher time complexity for extremely large datasets due to the extra divisions.

Exponential

Target set	100	1,000	10,000
Average time (ms)	0.01	0.02914	0.03778

Table 8: Average time of Exponential Search Algorithm

In exponential search, the times from the average search results demonstrated that as the data set size increases, the time taken by exponential search also increases. However, the observed increase is not strictly proportional to the data set size. This suggests that exponential search can also be efficient for handling larger data sets.

Interpolation

Target set	100	1,000	10,000
Average time (ms)	0.0075	0.0169	0.02042

Table 9: Average time of Interpolation Search Algorithm

From the average search results, the performance of interpolation search, as the data set size increased, also increased but at a lower rate. This algorithm may remain efficient, but the time complexity may lead to a moderate increase in search time for larger datasets.

Jump

Target set	100	1,000	10,000
Average time (ms)	0.01796	0.0205	0.05916

Table 10: Average time of Jump Search Algorithm

As the data set size increased, jump search showed a low increase in time search. It strikes a balance between linear and binary search, making it suitable for moderately large data sets.

Overall, the size of the data set affected the performance of search algorithms in a way that as the size of the data set increased, search algorithms generally experienced increased search time and execution. This is because larger data sets have more elements to be searched for. Moreover, larger size of data sets may also lead to increased memory requirements, affecting the execution time of search algorithms.

D. Write a brief conclusion summarizing your findings

Based on the data from the tests conducted, it shows that the efficiency of different search algorithms varies heavily on the target sizes of the data. For smaller target sizes, interpolation performs better than the rest of the search algorithm, followed by linear, exponential, ternary, jump, and binary respectively. This makes it useful for quick searching on a small data set. On the other hand, binary searching comes out as the most efficient search algorithm for larger data sets, followed by interpolation, exponential, ternary, jump, and linear respectively. As for other search algorithms, Linear search shows a linear increase in time while the data set becomes bigger and bigger making it less efficient for larger datasets. Ternary search is also better for a larger size of dataset but is slightly slower than the binary search. While exponential search exhibits an increase in time as the size grows, it is still a possible option for a larger dataset, just slower than binary and ternary search. To summarize, Interpolation works best for smaller-scale datasets, while Binary search shows the most efficiency when it comes to handling larger and extensive datasets.