TUSHAR KOUSHIK

PROFESSOR MARK LEBLANC

AI/ML

PROJECT 1

Table 1 presents the execution times (in microseconds) for three different search algorithms—Linear Search, Binary Search, and Fibonacci Search—across varying list sizes. The performance metrics recorded include the minimum, average, and maximum execution times for each algorithm at different input sizes (ranging from 1,000 to 1,000,000 elements).

List Size	Linear Min (µs)	Linear Avg (µs)	Linear Max (µs)	Binary Min (µs)	Binary Avg (µs)	Binary Max (µs)	Fibonacci Min (µs)	Fibonacci Avg (µs)	Fibonacci Max (µs)
1000	1.66E+01	2.62E+01	2.78E+02	2.13E+00	2.27E+00	1.05E+01	2.88E+00	5.13E+00	2.41E+01
10000	1.11E+02	1.37E+02	4.77E+02	8.75E-01	9.72E-01	2.92E+00	2.46E+00	2.62E+00	1.76E+01
100000	1.10E+03	1.18E+03	2.61E+03	1.04E+00	1.14E+00	5.13E+00	3.04E+00	3.24E+00	9.38E+00
1000000	5.11E+03	5.19E+03	5.67E+03	1.33E+00	1.47E+00	6.79E+00	4.00E+00	4.21E+00	1.94E+01

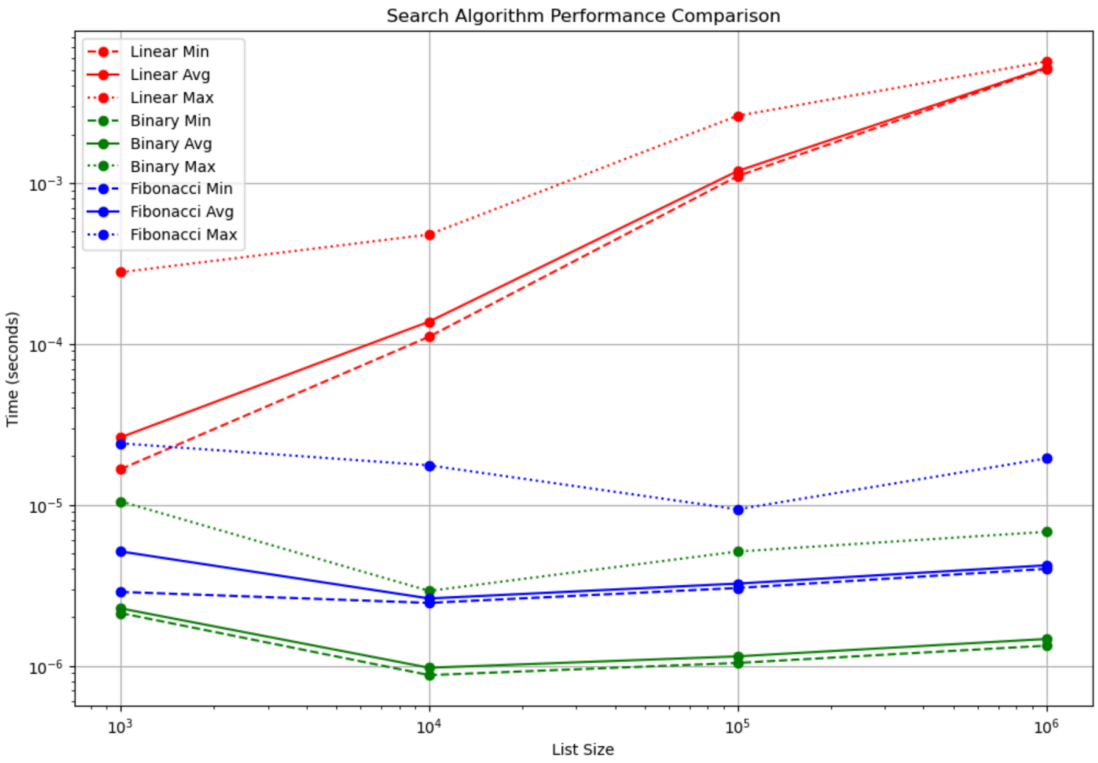
Table 1: Performance Comparison of Search Algorithms

Analysis of Table 1:

1. Linear Search Performance:
    - The execution time for Linear Search increases significantly as the list size grows, following an expected O(n) complexity.
    - For 1,000,000 elements, the max time reaches 5.67×10^3 µs, confirming that linear search scales poorly with larger datasets.
2. Binary Search Performance:
    - The Binary Search execution times remain significantly lower than those of Linear Search, especially at larger list sizes.
    - The max execution time for 1,000,000 elements is 6.79×10^2 µs, showcasing an O(log n) complexity, which is much more efficient.

3. Fibonacci Search Performance:
    - Fibonacci Search exhibits similar performance trends to Binary Search, confirming its logarithmic complexity.
    - However, its execution times are slightly higher than Binary Search, suggesting additional computational overhead due to Fibonacci number calculations.

Graph 1 illustrates the execution time of three different search algorithms—Linear Search, Binary Search, and Fibonacci Search—as the input list size increases. The x-axis represents the list size, while the y-axis represents the execution time in seconds, plotted on a logarithmic scale to better visualize performance differences.



Graph

Graph 1: Search Algorithm Performance Comparison

Outcome:

Binary Search outperforms Linear Search across all list sizes, as expected from theoretical complexity analysis.

Fibonacci Search performs similarly to Binary Search but may introduce additional overhead, making it slightly slower in practice.

Linear Search becomes increasingly inefficient as input size grows, confirming that it is best suited for small or unsorted datasets.

Where is Most of the Runtime Spent?

The cumulative execution time for Linear Search is significantly higher compared to Binary and Fibonacci Search, highlighting its inefficiency when applied to large datasets. This observation aligns with the theoretical O(n) complexity of Linear Search, where execution time increases proportionally with the size of the input. In contrast, Binary Search and Fibonacci Search demonstrate superior performance, benefiting from their O(log n) complexity, which allows them to scale more efficiently as the dataset grows. The profiling results confirm that Linear Search is the primary performance bottleneck, reinforcing the necessity of using more optimized search algorithms for large-scale data processing.

Citation:
OpenAI. (2025). ChatGPT (Feb 7 Version) [Large language model]. Retrieved from https://chat.openai.com/
Stack Overflow. Available at: https://stackoverflow.com/questions/18010660/binary-search-implementation-with-python