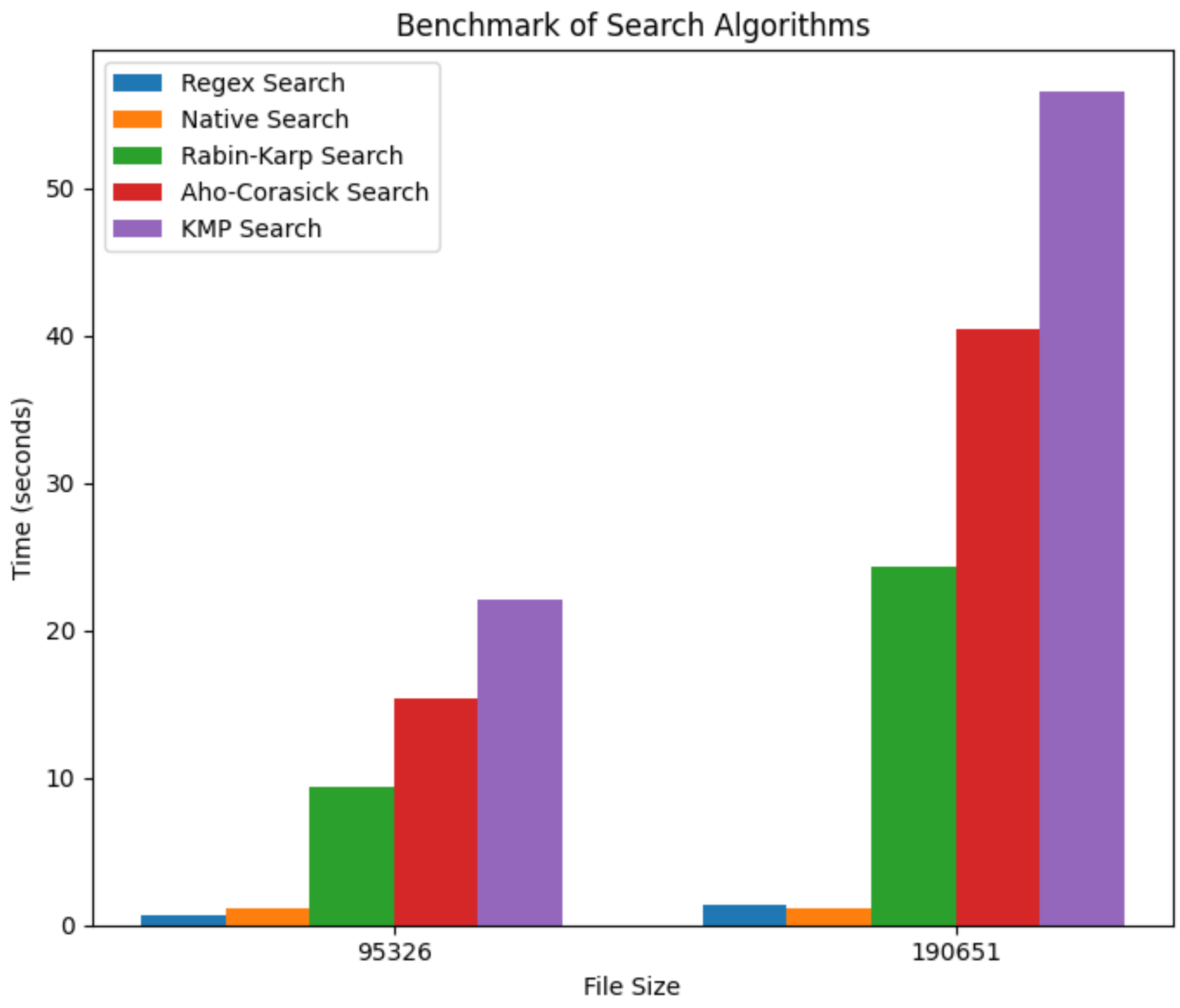
Benchmarking Search Algorithms

Summary

In the world of text processing, efficient search algorithms are essential for applications ranging from simple file searches to complex data mining tasks. This article presents a benchmark report comparing the performance of five popular search algorithms: Regex Search, Native Search, Rabin-Karp Search, KMP Search, and Aho-Corasick Search. The performance of each algorithm was evaluated on files of different different sizes.

Algorithms benchmark

```
Algorithm           95326          190651         Average
-----------------------------------------------------------------
Regex Search        0.637103       1.365456       1.001279 (ms)
Native Search       1.094598       1.148901       1.121749 (ms)
Rabin-Karp Search   9.359143       24.275475      16.817309 (ms)
Aho-Corasick Search 15.387821      40.427823      27.907822 (ms)
KMP Search          22.068264      56.516157      39.292211 (ms)
-----------------------------------------------------------------
```



Speed Test Benchmark

```
Speed Report:
Requests | 1000-kb    | 2000-kb
-------- | ---------- | ----------
10       | 6.7 | 12.2 | 6.1 | 15.6
20       | 6.3 | 14.4 | 6.5 | 13.4
30       | 6.9 | 14.4 | 6.1 | 12.9
40       | 2.1 | 11.1 | 6.9 | 12.5
50       | 5.5 | 11.3 | 7.5 | 12.6
60       | 8.9 | 14.9 | 6.0 | 6.8
70       | 6.9 | 12.0 | 5.1 | 13.8
80       | 7.5 | 13.4 | 7.8 | 9.5
90       | 6.9 | 8.8  | 6.5 | 13.0
```

Legend

The first collumn of each file is measure of time of when RE_READ_ON_QUERY == False, the second collumn is time measure when RE_READ_ON_QUERY == True

Conclusion:

The benchmark results highlight the importance of choosing the right algorithm based on the specific requirements and constraints of the application. For single pattern searches in text files, Regex Search is the clear winner due to its superior performance and efficiency. Native Search, while easy to implement, may not be suitable for larger datasets. Algorithms like Rabin-Karp, KMP, and Aho-Corasick, which are theoretically efficient, may not always offer practical performance benefits due to various overheads.

When designing a search functionality, it is crucial to consider the nature of the search tasks, the size of the data, and the specific performance characteristics of each algorithm. This benchmark provides a foundational understanding to guide such decisions, ensuring optimal performance and efficiency in text search operations.