# ALGORITHM SPEED REPORT

## TABLE OF CONTENT

[ALGORITHM SPEED REPORT 1](#_Toc569153972)

[TABLE OF CONTENT 1](#_Toc272429806)

[SUMMARY 2](#_Toc398477165)

[INTRODUCTON 3](#_Toc639076698)

[METHOD AND REPORT 3](#_Toc1012301621)

[TABLE SHOWING RESULTS 4](#_Toc1961622416)

[CHART SHOWING RESULTS 4](#_Toc1909655590)

[CONCLUSION 5](#_Toc220046381)

[RECOMENDATION 5](#_Toc1747761069)

## SUMMARY

This report shows how it collected the data from the program with each step well explained. The results are then showcased with the use of a table and a bar chart which compares the performance of all the eight algorithms tested. In the report, the best performing algorithm is recommended with the reasons explained.

## INTRODUCTON

Time being an important requirement of the project, the speed of the search for a string in the given file was recorded for every algorithm used. The process was to help in getting the algorithm with the fastest search time for both an existing string and non-existing string. The speeds were recorded in the logging file and can be referred to in the future for reference. The results and conclusions are recorded below.

## METHOD AND REPORT

The method used in the collection of data for this report was by capturing the time the user input was recorded and the search function called with the user input as the parameter. The time was recorded in a variable start time. The program then recorded the time a response was sent to the client and labeled it as stop time. The program then derived the execution time from negating the stop time from the start time. The time was then logged to the file server.log.

With the execution time being recorded, the server was started by running the command python3 server.py on the terminal. This was followed by running the client file with pthon3 client.py on the terminal. A copy of a string from the first, middle, and last line of the file was used to test the time it took for a response to be returned. The next input was a random sting which was not in the file. This was to test the time it took to for a response to be got from a non-existing string. This process was then done for each of the algorithms.

The algorithms that were tested in the program are all included in the list below.

1. Linear search
2. Naïve string search
3. Boyer-Moore algorithm
4. Mmap
5. Rabin-Karp algorithm
6. Knuth-Morris-Paritt algorithm
7. Binary search algorithm
8. Bitap algorithm

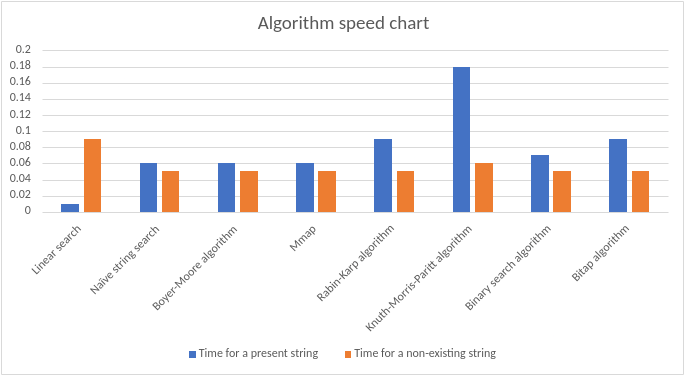
All the algorithm above were implemented in their own function in the file file\_searcher.py. The module was then imported and the functions executed and the time recorded. The table below shows the algorithms and their recorded time.

## TABLE SHOWING RESULTS

|  |  |  |
| --- | --- | --- |
| **Algorithm** | **Time to get a present string** | **Time for a non-existing string** |
| Linear search | 0.01 | 0.09 |
| Naïve string search | 0.06 | 0.05 |
| Boyer-Moore algorithm | 0.06 | 0.05 |
| Mmap | 0.06 | 0.05 |
| Rabin-Karp algorithm | 0.09 | 0.05 |
| Knuth-Morris-Paritt algorithm | 0.18 | 0.06 |
| Binary search algorithm | 0.07 | 0.05 |
| Bitap algorithm | 0.09 | 0.05 |

To compare the algorithms performance with each other, the below bar chart was used. The chart shows the time taken for both, a string present in the file and that of a string which does not exist. This is done for all the algorithm listed.

## CHART SHOWING RESULTS



## CONCLUSION

From the table the report makes a few conclusions. For the linear search algorithm, the report concludes that it has the shortest time to get a string which is present in the file. However, it also takes the longest time to respond for a non-existing string. Naïve string search, Boyer-Moore algorithm, Mmap, and Binary search algorithm all showed approximately the same results for both strings tested while Rabin-Karp algorithm and Bitap algorithm also show similar results. Knuth-Morris-Paritt algorithm shows the worst time to get a sting present in the file of all algorithms although the time for a non-existing string compared to that of liner search is less.

## RECOMENDATION

From the results and conclusion, the report recommends the use of the linear search algorithm. Although it has the worst time while responding to a string not precent, the linear search gets the string if precent the fastest.