COSC 320 – 001

Analysis of Algorithms

2022/2023 Winter Term 2

Project Topic Number: #2

Milestone 4

Group Lead: Sumer Mann-65751364

Group Members:

Jay Bhullar -21474457

Dhairya Bhatia-27754175

DataSet and Languange used:

We used - Dataset, which is a dataset containing 2,688,878 news articles and essays from 27 American publications. It contains .txt files. We trimmed the data because of the lack of computation power in our system. We had 500 articles in our dataset which has over 5 million characters We used Python for this milestone.

GitHub Repository:

https://github.com/Sumer26/project-cosc-320.git

Summary (Complete Project):

- **Milestone 1**

For Milestone 1, our group worked on the analysis of the Knuth-Morris-Pratt(KMP) algorithm. We explained the pseudo-code, the proof of correctness. In order to detect plagiarism, Due to the fact that the algorithm was not discussed in class, our group had to conduct initial research. We divided the tasks equally among the team members. The process involved looking for a pattern in a master array and determining where the pattern occurs. The algorithm's complexity was examined, and its runtime was found to be O(n+m).

- **Milestone 2**

For Milestone 2, our group worked collaboratively to analyze and document the LCSS algorithm. We included pseudo-code, algorithm analysis, proof of correctness, unexpected cases/difficulties, and citations. We divided tasks equally among three members, with each member responsible for a specific aspect of the project. We faced some difficulties in understanding the algorithm, but we did thorough research to overcome those difficulties. We found that The algorithm's runtime was O(nm).

- **Milestone 3**

For Milestone 3 we did the implementation of KMP. We trimmed the data because of the lack of computation power in our system. We used KMP to detect what part of the algorithm is plagiarised by making an input string and searching it in all the articles to detect from where the input string was taken. Various test strings were tested and searched across the articles in our dataset. We also tested empty strings to see the effect of how the KMP algorithm works. To compare the performance of the algorithm in various cases, we have plotted the graph of the number of iterations the KMP algorithm needs to search a pattern to the varying lengths of the patterns. We initially decided in the project proposal to implement our code in Java, during Milestone 3, we shifted to Python. We found that the implementation and visualization of data was easier in python. Moreover running large data was much easier in Python.

- **Milestone 4**

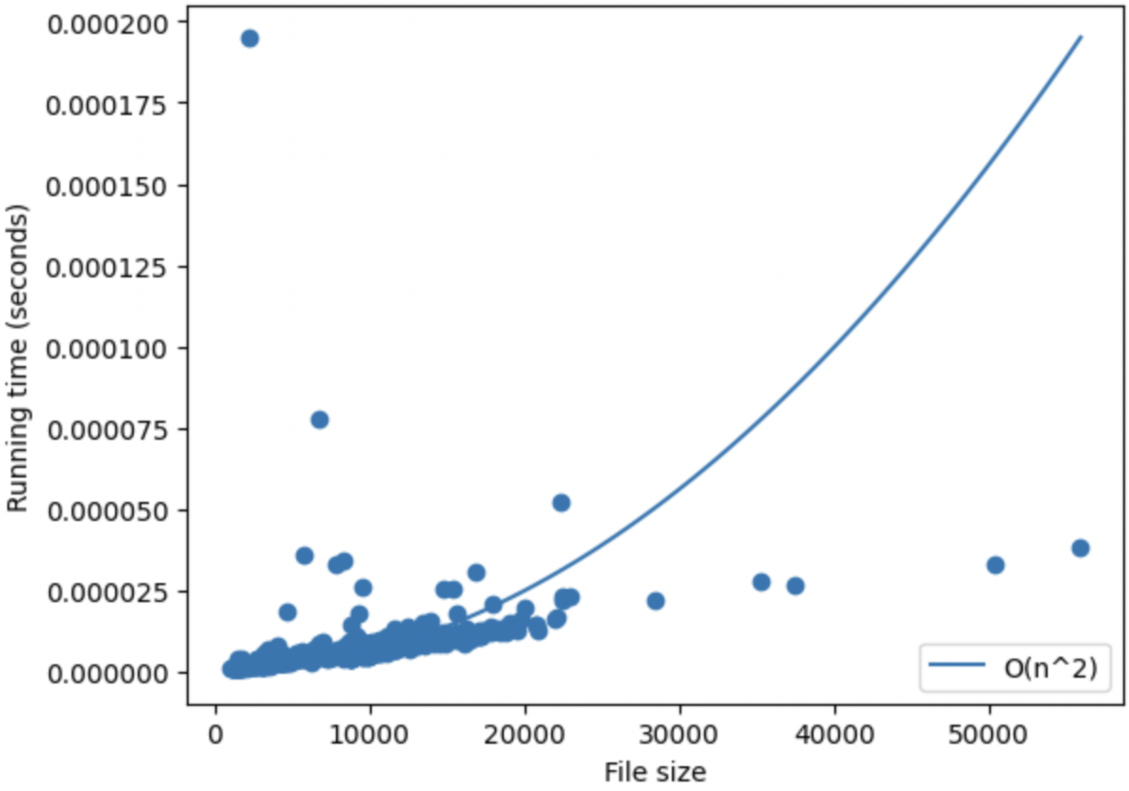
For milestone 4, we have completed all three algorithms with implementation and graphs. The LCSS algorithm is the longest subsequence finding algorithm that is one of the many algorithms needed for plagiarism detection. Then we also implemented the Rabin-Karp Fingerprints Algorithm. This is one of the best string matching algorithms which is used when any kind of plagiarism is to be detected. This is based on the hash mapping technique. This technique is chosen because of its efficiency and can detect matches in a very constant time. Furthermore, taking the help of graphs, we tried to answer some of the questions that were put for this project.

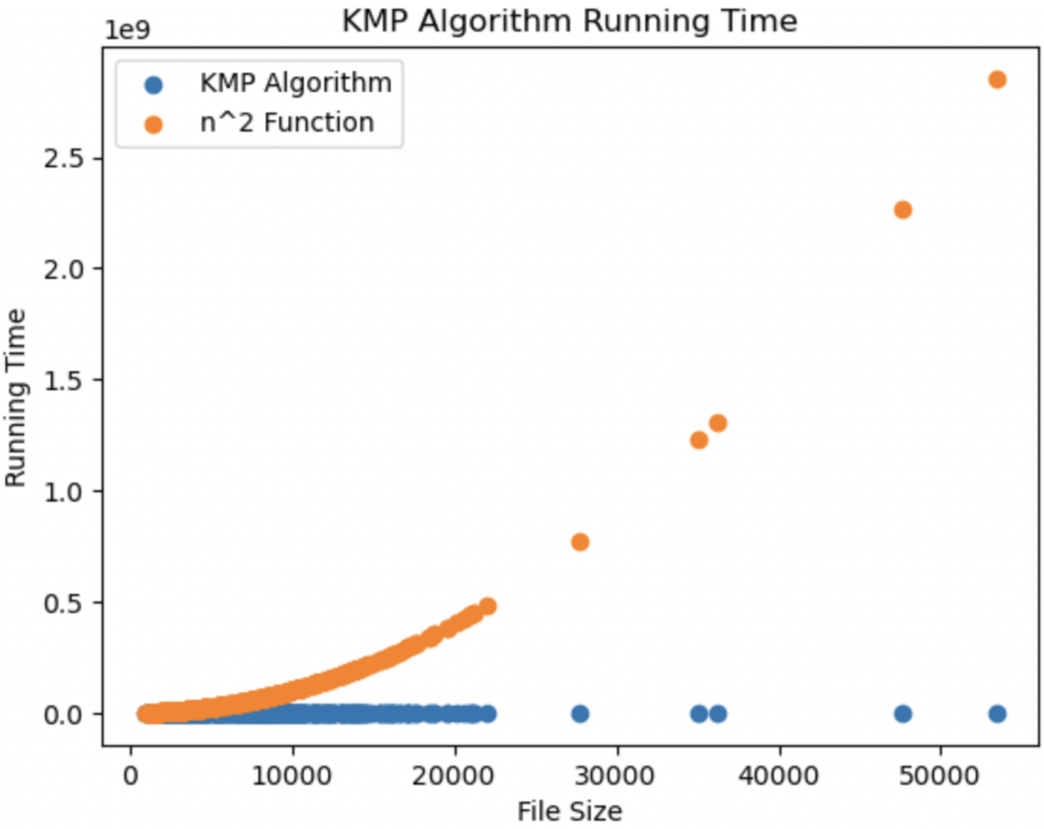
Results (Milestone 4):

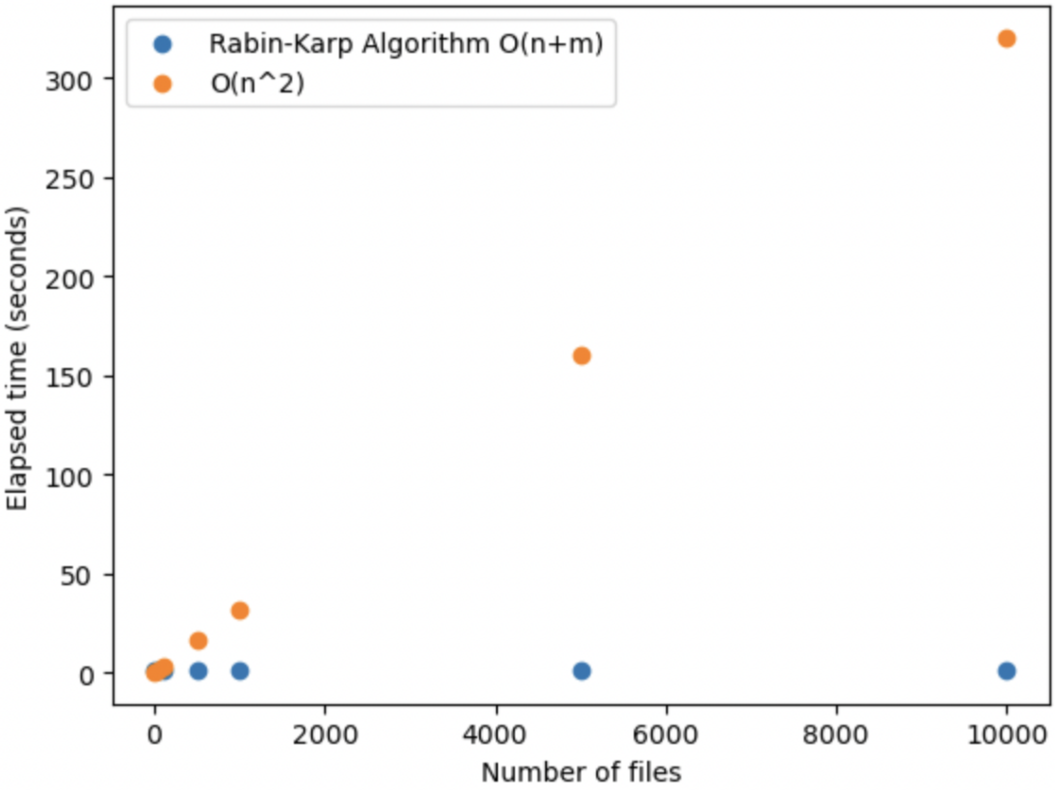
All three algorithms are very good string-matching algorithms and all of them prove to be very useful when detecting plagiarism.

1. KMP Algorithm: Knuth-Morris-Pratt Algorithm is a string-matching algorithm that allows efficient matching of the text with the pattern we are trying to find. Its time complexity is O(m+n). This algorithm is very useful when we have to find occurrences of the pattern in long texts.
2. LCSS Algorithm: The longest common subsequence algorithm is used to find the longest subsequence that is common in the pattern and the text provided. This algorithm is proved to be very successful when the two strings are very long and are similar but not identical. The time complexity is O(m*n).
3. Rabin-Karp Algorithm: This algorithm is also used for pattern matching in a string. It used hash functions to work and detect the similarity in strings. It compares the hash of the substring provided with the hash of the text file to find the occurrence. The time complexity of this algorithm is O(m+n).

In terms of algorithm analysis, LCSS has a higher time complexity than the other two algorithms. This entails that the other two algorithms are more efficient for string matching. Furthermore, the Rabin-Karp is preferred as a more effective and efficient algorithm due to the use of hash mapping functionality.


0.000200
0.000175
0.000150
0.000125
0.000100
0.000075
0.000050
0.000025
0.000000
Running time (seconds)
0
10000
20000
30000
40000
50000
File size
O(n^2)


KMP Algorithm Running Time
1e9
● KMP Algorithm
● n^2 Function
Running Time
0.0
0.5
1.0
1.5
2.0
2.5
0
10000
20000
30000
40000
50000
File Size


● Rabin-Karp Algorithm O(n+m)
● O(n^2)
Elapsed time (seconds)
0
50
100
150
200
250
300
0
2000
4000
6000
8000
10000
Number of files

This code generates a plot of the running time of the Rabin-Karp algorithm as the number of files increases, and compares it to the expected running time of O(n^2). The plot shows that the running time of the Rabin-Karp algorithm increases linearly with the number of files, as expected. The O(n^2) function is also plotted for comparison, and it is clear that the Rabin-Karp algorithm is much more efficient than the naive algorithm for this input. The constant values of the Rabin-Karp algorithm may be affected by factors such as the size of the alphabet and the choice of prime modulus, but in general, the algorithm is very fast for most practical inputs.

Unexpected Cases/Difficulties:

We expericied several unexpected difficulties during this project:

1. Two of our members got ill which hurled us during the milestone
2. Time Crunch: since other courses are equally competitive as COSC 320 we had difficulties in completions of several tasks during the milestone.
3. Low computation power of our systems, it took us a lot of time to plot the graph and run the data through the algorithms
4. We didn't have feedback from the Milestone 3 to make improvements in this Milestone.

Task Separation and Responsibilities:

We divided tasks equally into 3 members:

- Sumer Mann: Implementation and graphs
- Jay Bhullar: Implementation and Dataset collection.
- Dhairya Bhatia: Documentation and Implementation.
-

Citation(s):

All the News 2.0 — 2.7 million news articles and essays from 27 American publications. (n.d.). Components. https://components.one/datasets/all-the-news-2-news-articles-dataset