**COSC 320 – 001**

***Analysis of Algorithms***

2022/2023 Winter Term 2

**Project Topic Number: 2**

**Title of project: Plagiarism Detector**

**Group Members:**

**Divyajot Kaur Dadiala**

**Jusnoor Kaur Sachdeva**

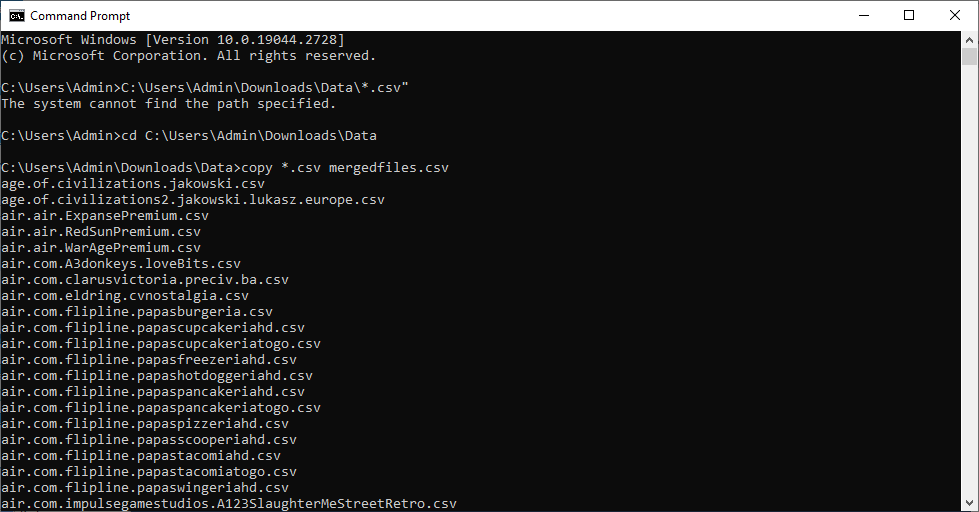
**Robert Barnstead**

Working with Dataset

We used the dataset provided by the professor. We merged the csv files into a single csv file using command prompt.

Script to combine csv files

copy “C:\Users\Admin\Downloads\Data\\*.csv” mergedfiles.csv



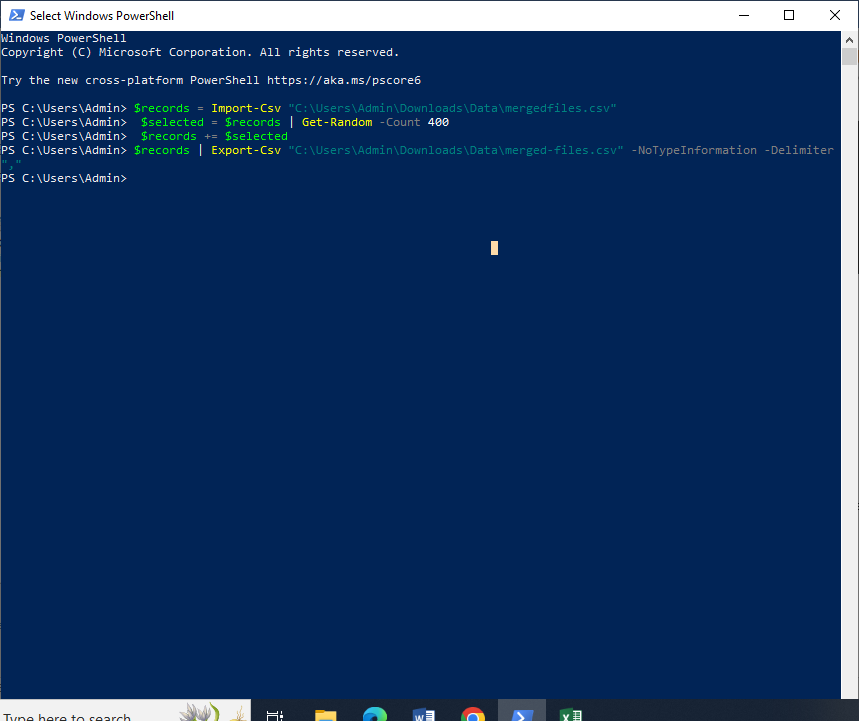
**Creating Plagiarized DataSet**

$records = Import-Csv "C:\Users\Admin\Downloads\Data\mergedfiles.csv"

$selected = $records | Get-Random -Count 400

$records += $selected

$records | Export-Csv "C:\Users\Admin\Downloads\Data\mergedfilesplagiariseddata.csv" -NoTypeInformation -Delimiter ","



Implementation

The KMP Algorithim has been designed to take two strings a text string and a pattern string and determine whether the pattern appears in the text. To implement this we took the dataset provided and read the CSV file using openCSV in Java to create one giant string of text. To define our pattern we randomly took one of the entries from the text that was found while reading the CSV file and stored it as a string. Then we called the search function using both the text and pattern. In the search function we call the LPS function in order to create the LPS array that assists us in finding the pattern in the string. This returns us the Char range where the pattern is found, or in our case where plagiarism has been found. A major issue that we ran into was the time it takes to create the string.

The github link is as follows

https://github.com/rbarnstead/cosc320-milestone3

Results

* For analyzing the performance of the KMP algorithm, we plotted Input Size and Time

**Input Size and Time:**

Time taken to read the input file increases exponentially as the number of lines in the input file increases. The time to read a single line is constant but as the number of line increases, the time taken to read the file increases exponentially.

KMP algorithm has a worst-case time complexity of O(n)(where n is the length of the text). The length of the text increases as the number of lines read from CSV file increases and the time taken to perform string matching increases linearly.

The time complexity of the KMP algorithm is O(n+m), where n is the length of the text and m is the length of the pattern. As ‘n’(i.e., the length of the text) grows, the running time of the algorithm will keep increasing. KMP algorithm has a linear time complexity, so the graph of the running time will also be a linear graph. Therefore, the running time graph is straight line passing through the origin.

It is important to choose the right data structures, use efficient algorithms and functions, and apply optimization techniques to minimize the constant factors in the running time. KMP algorithm uses an array to store the longest proper prefix that is also a suffix of each substring of the pattern. Depending on the implementation of this array , the constant factors may differ. The KMP algorithm also uses a function to compute the LPS array, which is used for the string matching, implementation of this function can affect the constant factors in the running time of the algorithm.

**Unexpected Cases/Difficulties.**

**Large File Size** – We faced difficulty in uploading and downloading the Dataset files. It could be due to Internet connectivity.

**Creating plagiarized Dataset and Implementation**: We wrote a script. We should have taken TA’s help to have more clarity on implementation.

**Team Coordination** – We think that working together in a Lab would have helped and we could have done the task faster.

**Task Separation and Responsibilities**

Working on Dataset – Writing scripts for Dataset – Divyajot

Writing the Code – Robert Barnstead

Result Analysis – Jusnoor and Divyajot

Coordination and Final Submission – Divyajot