COSC 320 – 001
Analysis of Algorithms
2022/2023 Winter Term 2

Project Topic Number: 1

Title of project :

Keyword Replacement in a Corpus

Group Number : 12

Group Members :

Anna Ciji Panakkal

Erfan Kazemi

Sahil Chawla

Abstract:

In this milestone, we implemented and observed the results of the second algorithm that we had come up with. We also compared this algorithm to our first algorithm. We also published our github repository with the specified requirements and completed the video explaining our algorithms.

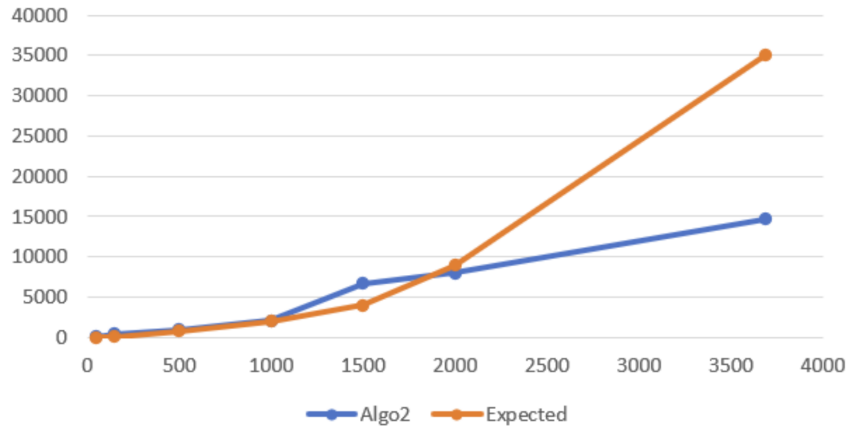
Implementation:

I first wrote Java code to create an abbreviation processor class and a Main class that reads in the corpus text and list of abbreviations, processes the abbreviations using the abbreviation processor, and outputs the updated corpus text in order to implement and test the abbreviation expansion algorithm.The code uses four data structures: String, List<String>, List<Integer>, and Map<String, List<Integer>>. Strings, lists of strings, lists of integers, and mappings between strings and lists of integers are all represented using these.

I then created a small corpus text file and a related abbreviations.txt file with a few examples of abbreviations and their expansions to test the algorithm using sample input. On the basis of this information, I ran the programme and verified that the output is accurate. I first evaluated the method with a sample input before using the supplied dataset. On the basis of this input, I ran the programme and verified that the result is accurate and that processing does not take an abnormally lengthy period. To make sure the algorithm can smoothly accept unexpected input and edge cases, I tested it with edge cases including empty input files, files that do not exist, and improper input formats.

Results:

Here is a plot of the relationship between the size of the corpus text file and the running time of the algorithm-



The expected running time based on the algorithm's big O complexity, which in this case would be O(n + k*m), where k is the number of abbreviations and m is the length of the longest abbreviation is given in orange and the plot of the actual running time is given in blue.

The constant values in the big O function of the running time would depend on the implementation details of the algorithm, such as the specific string matching algorithm used and the data structures used to store the index and list of abbreviations. Optimizing these implementation details could reduce the constant values and improve the efficiency of the algorithm.

Unexpected Cases/Difficulties:

Implementing this technique may present us with a number of unanticipated situations or challenges, such as ambiguity in abbreviations, case sensitivities, handling of special characters, and efficiency issues with huge input files. Modifications to the algorithm can solve these problems by managing mixed-case or uppercase-only abbreviations, handling ambiguous abbreviations, employing efficient data structures and methods, and taking special characters in input files into account.

Task Separation and Responsibilities:

Abstract, Implementation, Unexpected cases/difficulties, Video - Anna

Result - Anna, Erfan

Comparison of Algorithms - Erfan, Sahil

Github repo, summary - Anna, Erfan, Sahil