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

Project Topic Number: #1

Keyword Replacement

Group Lead:

Sanjith Senthil

Group Members:

Cole Van Steinburg, Issa Hashim, Sanjith Senthil

Abstract

In this milestone, we have successfully implemented the first algorithm - a naive approach to the keyword replacement problem. We tested the algorithm across a range of test cases, varying the number of inputs from 1000 records to millions. We generated plots depicting the algorithm's runtime and analyzed how it scales with the input size. In addition, we compared the runtime to its corresponding big O function.

Dataset

The dataset contains approximately 2.8 million records with a total size of 998.8 MB, based on reviews from the Google Play Store. It includes separate csv files, each about one application. The dataset consists of 10 columns, namely: “reviewId”, “userName”, ”userImage”, “content”, “score”, “thumbsUpCount”, “reviewCreatedVersion”, “at”, “replyContent” and “repliedAt”. We worked with the column named “content” as this is the app reviews written by the users. We also manually compiled and created a csv file of commonly used abbreviations, along with their corresponding full forms.

Implementation

For each csv file in the dataset, our algorithm reads in all the reviews to a list. Once the list is read in, it loops through each review. For every word in the review, we first strip and save any punctuation to be preserved after doing word replacement. After stripping punctuation, we loop through the replacement word list doing a comparison to the current word for each. If a match is found, the word is replaced. If a replacement is made, we add back the punctuation that was stripped earlier. Once the whole file has had words appropriately replaced, we write the results to a new csv file in a new directory.

Results

Since the runtime of the algorithm is O(n * m), we made different graphs to show what happens as ‘n’ increases with ‘m’ held constant and vice versa. Here, ‘n’ denotes the number of words in the paragraph list and ‘m’ denotes the number of words in the replacement list.

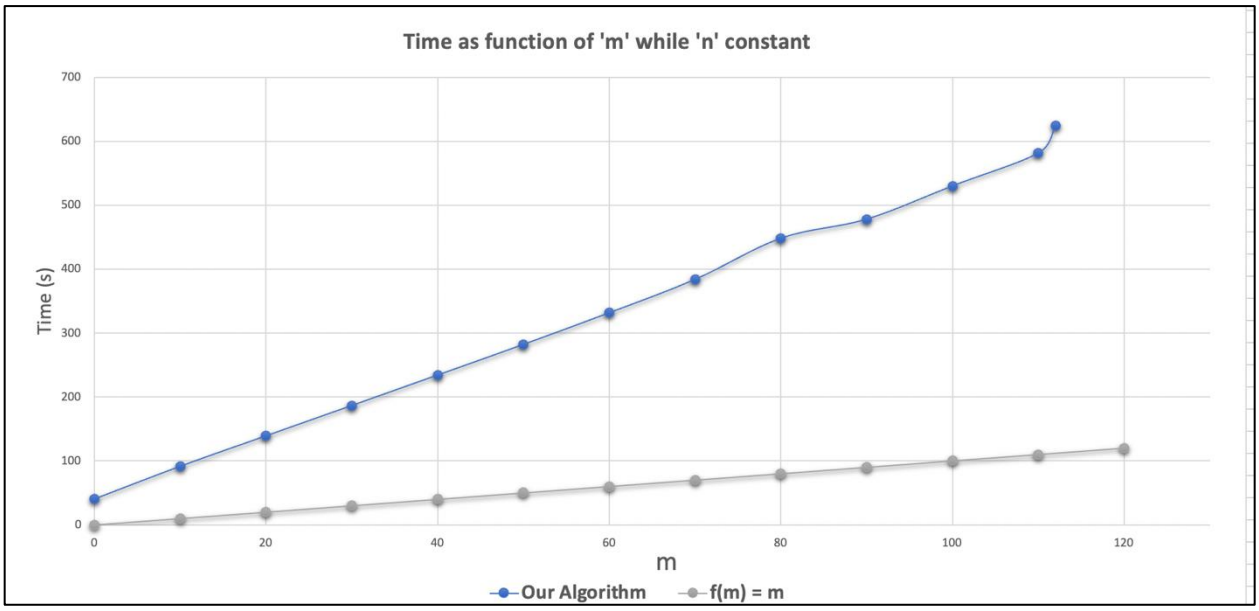
The data obtained and presented below maintains a constant value of ‘n’ while increasing the value of ‘m’.

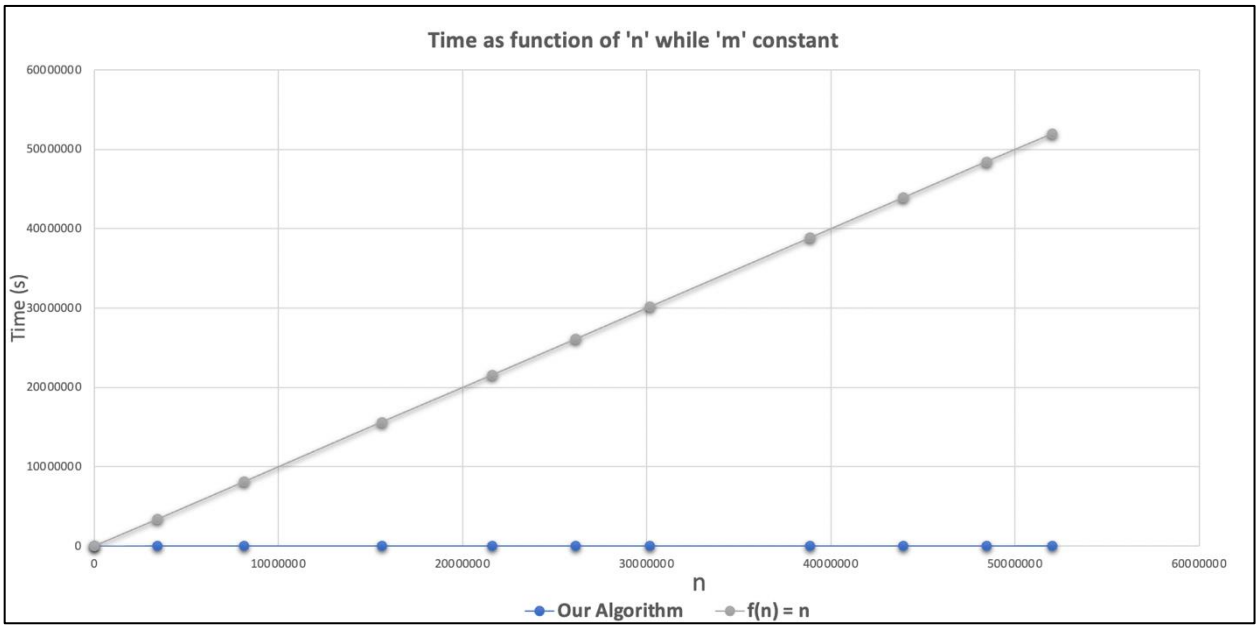
n	52012294	52012294	52012294	52012294	52012294	52012294	52012294	52012294	52012294	52012294	52012294	52012294	52012294
m	0	10	20	30	40	50	60	70	80	90	100	110	112
Time (s)	39.82083	91.01449	138.6669	186.145	233.8054	281.6957	331.2654	383.4449	447.8163	477.6728	529.7837	581.4052	623.7937

The data obtained and presented below maintains a constant value of ‘m’ while increasing the value of ‘n’.

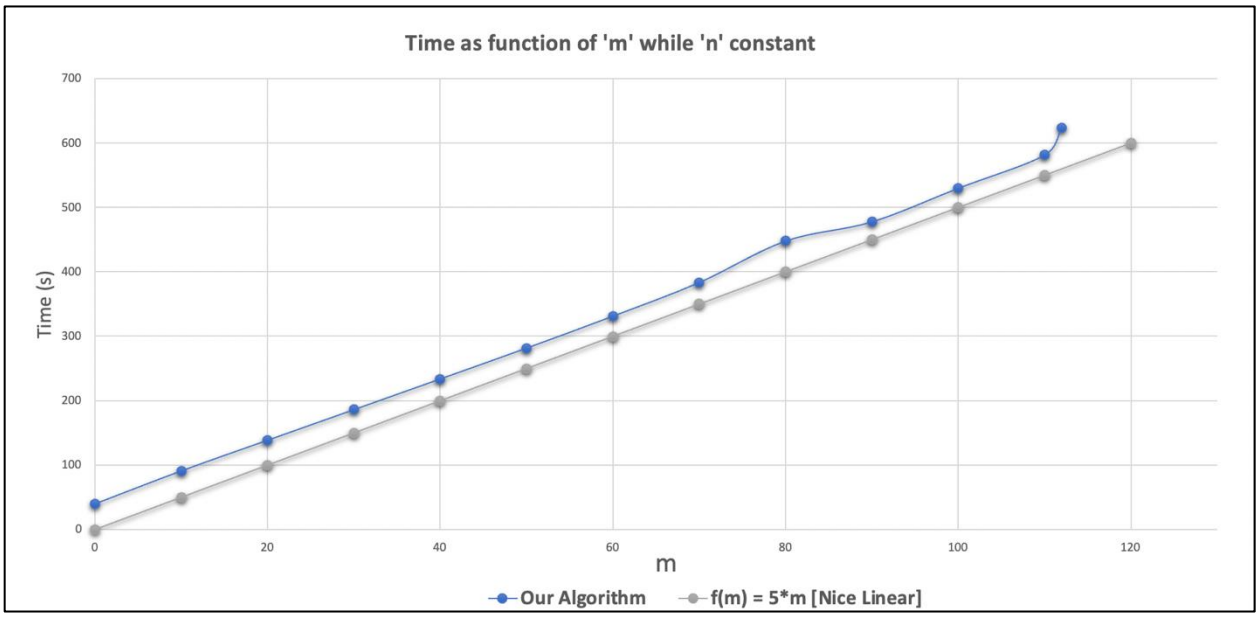
n	0	3414544	8143360	15630428	21592267	26126390	30158570	38859042	43940140	48457311	52012294
m	112	112	112	112	112	112	112	112	112	112	112
Time (s)	0.00784	40.76588	94.13427	182.4663	247.95	298.3001	344.6238	444.2251	504.8494	575.3879	608.5458

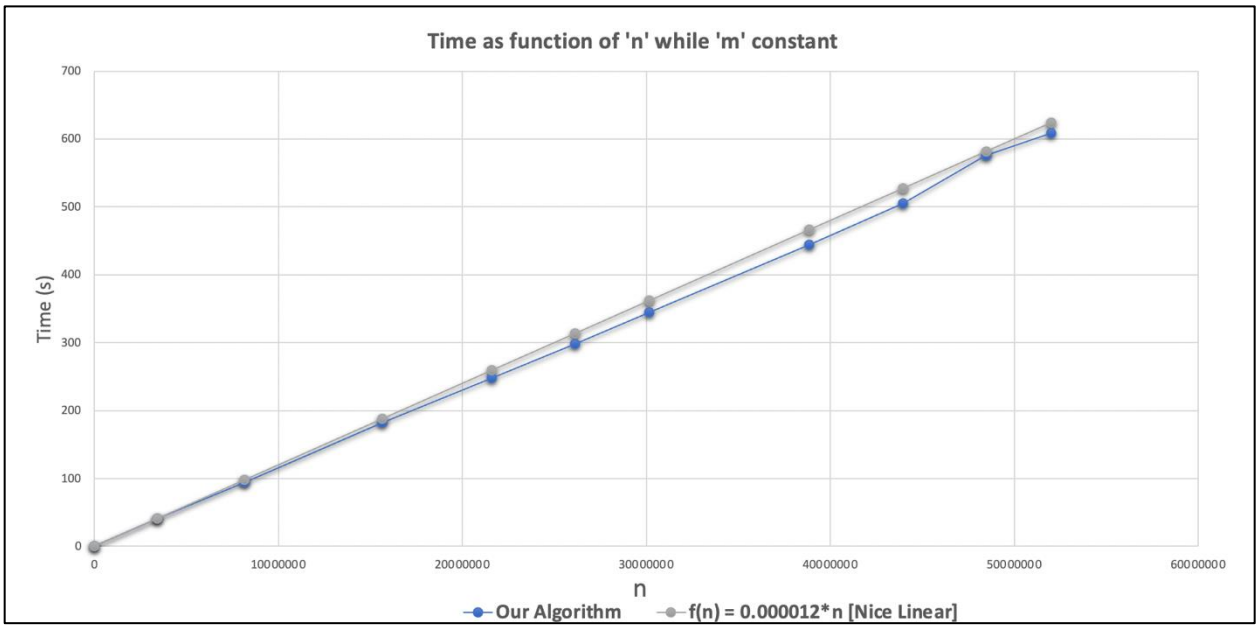
Generating the plots of the algorithm’s runtime and comparing it with the big O function f(x) = x:





Generating the plots of the algorithm's runtime and comparing it with a linear function of matching slope:





As expected, when one variable is held constant, the time taken to run varies linearly with the adjustment of the other variable.

The implementation of the algorithm has affected the constant values. Note that when we hold 'n' constant and set 'm' to 0 our algorithm does not take 0 time to run, even though the runtime is O(n * m). This is because there are lower order

terms / constants pertaining to ‘n’ to read the files in, as well as stripping the punctuation from each word. These steps happen even if there are no words in the replacement list.

We use the naive approach of looping through a list of tuples, which is why our algorithm runs in O(n * m). No additional data structure has been used.

Unexpected Cases/Difficulties

The dataset contained partial corruption in certain places, leading to null bytes that caused exceptions during processing. To address this issue, we replaced all null bytes with empty spaces. This allowed us to work with the dataset without encountering errors and ensured that our analyses were based on reliable and error free data.

In addition to partial corruption and null bytes, we also encountered some unexpected characters that turned out to be emojis. These were not able to be read by default. To tackle this issue, we converted the reader to take in UTF-8 format.

Task Separation and Responsibilities

We divided the work among us equally, meaning that all members of our group contributed to every task of the milestone together. We collaboratively worked on the algorithm implementation, testing, data collection, plot generation, interpretation, and analysis. In addition, Cole and Issa were responsible for setting up the group meeting, and Sanjith was responsible for communicating with the instructor and TA’s.