COSC 320 – 001

Analysis of Algorithms

2022/2023 Winter Term 2

Project Topic Number: #1

Title of project:
Not Your Grandma's Review Analyzer

Group Lead: Connor Doman

Group Members: Quinn Marshall
Archita Gattani
Connor Doman

Abstract:

Within Milestone 3, we've implemented and tested the algorithm we had described about in Milestone 1. As a reminder, our algorithm should be able to find the abbreviations and replace them with their keywords. For example, with a medical data set: "BMI" should be replaced with "body mass index".

We have included a description of the data set, the plots and our interpretation of the plots as input grows. There is also a comparison done to the big O function of the running time to show how the implementation of our algorithm might have affected the constant values and how the choice of data structure might have affected this result. Lastly, we added the difficulties we came across while completing milestone 3, and our task separation within the completed milestone.

Dataset:

The dataset, Amazon product data, has a collection of reviews for different products sold on Amazon.com, released by the University of California, San Diego.

The dataset contains reviews from May 1996 to July 2014 and is available in multiple formats, including text files and SQLite databases. It covers a wide range of product categories, such as books, electronics, home and kitchen, and sports and outdoors, among others.

There are different versions of the dataset with varying sizes, ranging from a few gigabytes to several terabytes. The dataset also includes metadata for each product, such as the product ID, category, and price, as well as information about the reviewers, such as their ID and location.

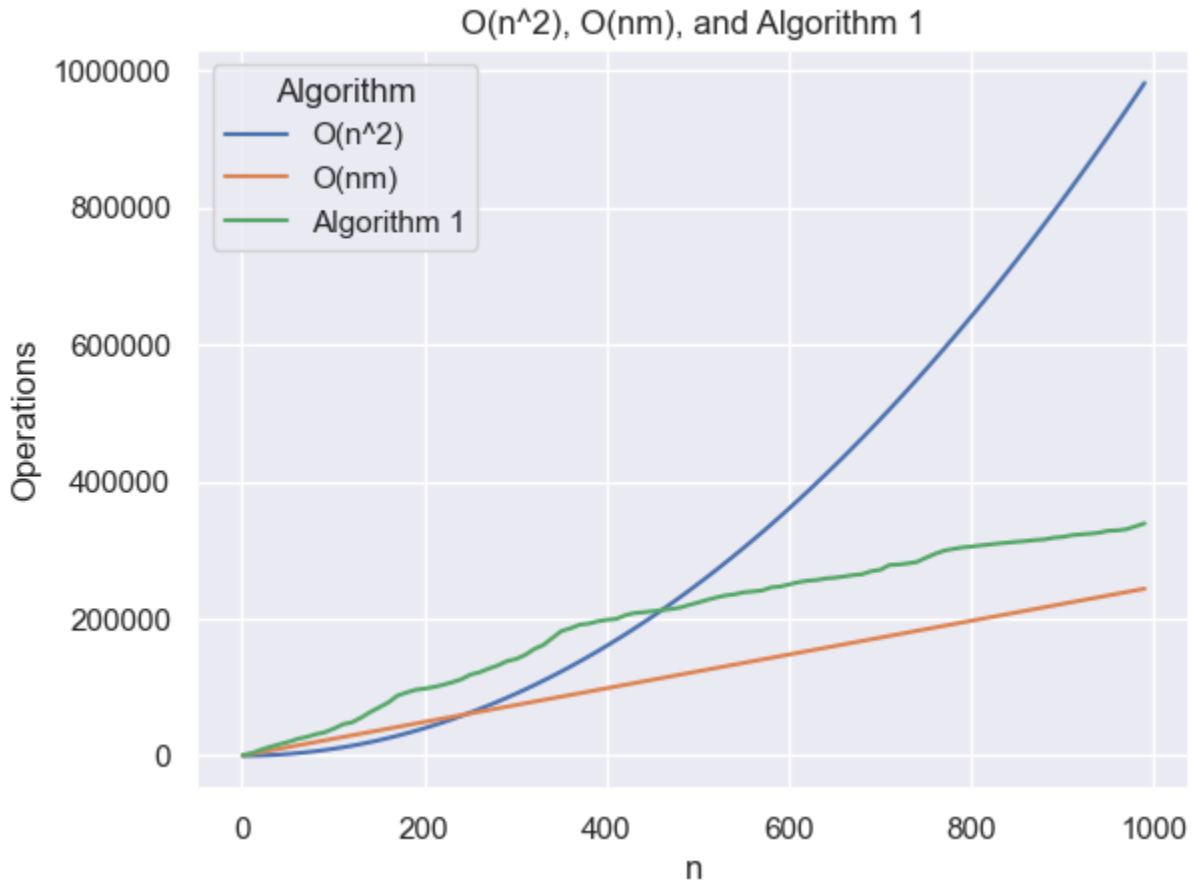
Implementation:

After loading the dataset from JSON we processed it to contain only the body of each review. A million of these reviews were then saved into a list.

Our first algorithm was implemented essentially as-is from the pseudocode in our first milestone. The only alteration we made was returning a count of every operation made by each step. We created a function that repeatedly runs our algorithm using $n = 1..N$ in increments of 10 and then saves the number of operations performed per value of n to a dataframe. As we expected our algorithm to be $O(nm)$, we performed this same process for a simple $O(nm)$ and $O(n^2)$ function to generate relevant comparison data.

The input n in our testing function determines how many reviews to use as the searchable body of text in our algorithm. Each review is concatenated into a very large string that is then passed into the algorithm. As such, the m factor of our O(nm) function was set to the average length of a review, in words, as determined at ingest time.

Results:



We expected our algorithm to have a running time of O(nm). When graphing our algorithm compared to O(n^2) and O(nm), we can see that our algorithm very closely resembles the runtime of O(nm), except for the Fast that our function is slightly slower than the nm function. Some factors that might have caused the differences between the two functions are the n in the nm function is constant, however, in our function n represents the length of the review text which varies. Furthermore, the number of operations in our algorithm also accounts for splitting and recombining our string which would increase the runtime by a constant. Lastly, the length of our dictionary m is the number of words in our dictionary, which as the size of the dictionary increases, the algorithm will take longer to run since it must iterate that many more times.

Unexpected Cases/Difficulties:

- When trying to import our dataset, we had trouble loading in our data set from a JSON file. We managed to figure out a python command to be able to load in our data set and run it through our algorithm.
- The very large dataset was slow to ingest and difficult to edit.
- Determining appropriate test cases between example algorithms and our own was challenging.
- Timing our function in seconds proved ineffective compared to timing based on operations.

Task Separation and Responsibilities:

Connor: prepare test cases
Quinn: plot test cases
Archita: implement test cases