Milestone 4 - COSC 320 Term Project

Aidan Elliott, Eddie Zhang, Brandon Mack

For this project our group took two different approaches to create an efficient algorithm to replace keywords in text. Our first approach involved using hashmaps and the second was using a trie data structure. While both approaches have a worst-case time complexity of O(m) where m is the number of characters, the trie data structure in practice is quicker by a constant factor.

As previously mentioned, the first approach involves using hashmaps. When given a string A, the algorithm will create an array B that will split every string by " " in the text. It will then iterate through B and look up the value of B; 'i' into a hashmap C. If 'i' is a key of C then 'i' will be replaced with the value found in the hashmap. B will then get outputted with all abbreviations converted into their full meaning. We were able to prove the algorithm's correctness using a loop invariant and inductive hypothesis. Some difficulties we encountered included how it will be capable of handling an empty input as well as there being limitations in the pre-processing of the data due to it taking a raw string for input.

The second approach mentioned involved using a trie which had potential to significantly reduce the search time. Our algorithm creates a trie by adding every keyword we want to check for to it. At the "leaf " of every abbreviation, the extended phrase that we want to replace the abbreviation with is stored. Now to do our keyword replacement, we can search our trie for every word in the text, and replace those words when the search returns true. Because the search will cancel as soon as a character does not match anything in the trie, the actual runtime of the algorithm should be faster than the worst case on average. We used proof by induction to prove our algorithm's correctness. We also had some difficulties with this algorithm such as how it can tell when a keyword is completed for example whether the user is typing H=Hydrogen or if there is more to the keyword HD=High Definition.

In conclusion, the trie data structure does have an improvement in search time over the hashmap implementation. This improvement is from the trie being able to stop searching once the prefix does not match anything in the trie while the hashmap must always process the entire word no matter what. This result is reflected in our graphs found in our code. Both structures run in linear time although the trie is quicker by a constant factor. Each has their advantages and disadvantages but the trie data structure proved itself as the more efficient algorithm for keyword replacement in text.

Link to Datasets used:

https://ubcca-my.sharepoint.com/:u:/r/personal/fatemeh_fard_ubc_ca/Documents/Courses/COSC%20320-2022-2023%20(Jan%202023)/Project-Dataset/2.8M%20App%20Reviews%20-%203686%20Top%20Paid%20Apps%20-%20998.8MB%20on%20Disk.7z?csf=1&web=1&e=CjlzCP