Assignment 4

Dr. S. Antoun

Aim

The aim of this assignment is to practice building and using Hash tables (chaining and linear probing), Rabin-Karp pattern matching, and resizable arrays.

Whitelisted libraries

The allowed libraries for this assignment are "iostream", "cassert", "iomanip", "fstream", "chrono", "cstdlib", "ctype", and "(c)string", STL containers, STL iterators, no STL algorithms.

Task

The task involves testing pattern matching algorithms using a stream dataset. The provided dataset is a text file named "A Scandal In Bohemia.txt", which contains twelve works by Sir Arthur Conan Doyle. The assignment requires developing a suitable hash-table implementation with open hashing (chaining) and Linear probing to store the contents of the text file. Specifically, works I through VI should be stored using open hashing, while works VII to XII should be stored using Linear Probing. The data should be read only once as a stream, i.e., one string at a time.

Once the data is processed, the program should display and record to a file a list of word occurrences from highest to lowest and vice versa of the 80 most and least repeated words (and their count) in Conan Doyle's works. It's important to note that capitalization should not matter (i.e., "Watson" and "watson" should be counted as the same word), but hyphenation is critical, and double hyphens "–" should be discarded.

As the program starts processing work IX "The Adventure of the Engineer's Thumb", it should prompt the user for search keys (up to 8, delimited by "@@@" if less than 8) and display the position of each key's occurrence in the text (i.e., word count position). It's essential not to assume that any of the keys sought are unique, so each occurrence of each key must be accounted for. Rabin-Karp pattern matching algorithm and Horner's rule for the rolling hash should be utilized for this purpose.

Key Reports

The assignment requires reporting on several key aspects:

1. The occupancy ratio to be used in linear probing. This involves experimenting with different values such as 50%, 70%, and 80%, and reporting runtimes in nanoseconds.
2. Optimizing chain length in open hashing. At least three experiments should be conducted, and runtimes in nanoseconds should be reported.
3. Experimentation with different hash functions. A simple function such as $f(r) = r\%hsize$ should be the initial attempt.
4. Handling collisions in the table for linear probing. The collision resolution method implemented must be described, with research and inclusion of a method described in the lecture.
5. The necessity of an interface file (a ".h" file) for the functions implemented.
6. Writing a function to prompt a user for a word, display the number of occurrences of this word in the text, and the locations of said occurrences in "The Adventure of the Engineer's Thumb".
7. Implementing a function to output a list of the 80 least frequently occurring words in the text.
8. Implementing a function to output a list of the 80 most frequently occurring words in the text.

9. A function to output the number of sentences in the text.
10. Reporting the runtime for each task, while keeping reusability in mind.

Implementation

The main file, `FinalAssignment.cpp`, should be created with the appropriate entry in the makefile. This program, when built, will expect a command-line argument consisting of an input text file and an output text file name. It will read the data in the input file (strings or cstrings) and write to the output file the outcomes of all tasks listed above, with clearly labeled sections and runtimes for each task. The output strings should all be in lowercase.

Testing

The functionality of the program should be tested by augmenting the driver program from previous steps in `FinalAssignment.cpp` to form a menu driver to test all tasks. A "zero" menu option request should terminate the program.

Final Report

A final report in LaTeX format should outline empirical findings and offer insightful views on the implemented algorithms. This report will be required for the next theory assignment.

Submission

All code files including makefile should be submitted via D2L.