Title: "Word Corrector and Suggester using Levenstein's Distance Algorithm in C++”

Subtitle: "Data Structure and Algorithm Project Report”

Group Members:

Anas Khan - Student ID: 22k4483

Umar Aurangzeb - Student ID: 22k4139

Azif Hussain - Student ID: 22k4379

**Introduction**

Our project focuses on developing a Word Corrector in C++ that utilises the Levenstein algorithm. The primary objective is to provide users with suggestions for misspelled words in each paragraph. We achieve this through a combination of data structures, including a hash-map for efficient word storage, a linked list for tracking the original paragraph, and a jagged array in the Levenstein algorithm for calculating word distances.

Data Structures Used

Hash-map with Chaining:

To efficiently store and retrieve words from a dictionary file, we implement a hash-map with a chaining mechanism. This data structure allows for quick access to words during the correction process.

Linked List:

A linked list is employed to store the original paragraph and the corresponding incorrect words. This facilitates easy traversal and replacement of words during the correction process.

Jagged Array for Levenstein Algorithm:

The Levenstein algorithm, a crucial component of our word correction logic, utilises a jagged array for efficient computation of distances between words. This enables the identification of close matches for misspelled words.

Implementation Details

Dictionary Loading:

The project begins by loading a comprehensive dictionary from a text file into a hash-map. Chaining is employed to handle collisions, ensuring that the dictionary is efficiently organised for quick word retrieval.

Paragraph Checking:

User input paragraphs are processed, and incorrect words are highlighted in the console. The original paragraph is maintained in a linked list for reference during the correction process.

Levenstein and Hash-map Integration:

The Levenstein algorithm is seamlessly integrated with the HashMap to find the closest words with distances ranging from 1 to 6. This combination enhances the accuracy of word suggestions and contributes to the effectiveness of our correction logic.

User Interaction and Conclusion

User Interaction:

Users interact with the correction system by selecting from an array of suggested words for each highlighted incorrect word. This interactive process allows users to choose the most appropriate replacement or retain the original word.

Conclusion:

In conclusion, our Word Corrector project successfully implements data structures like hash-map, linked list, and jagged array to provide an effective solution for identifying and correcting misspelled words. While the system is robust, there is always room for improvement, and further enhancements could be explored to refine the correction process.