**Algorithm Analysis: -**

**For the spell checker project, we had two options of algorithm to be used hash map and trie. We chose to do it with trie primarily because of following reason:-**

* **Efficient for operations involving keys with common prefixes**
* **Provides lexographical ordering**
* **Each key (in this case letter) has a unique path in the trie**
* **Also we find it easy to code using trie**

**Trie Implementation**:

* + The code defines a Trie data structure using a class named **Trie**.
  + Each node of the Trie represents a character in the alphabet, and each node has an array of pointers to its children nodes.
  + The **node** class represents a node in the Trie, and it contains an array of pointers to its children nodes (**children**) and a boolean flag (**end\_of\_word**) to mark the end of a valid word.**Insertion into Trie**:
  + The **insert\_word** function in the **Trie** class inserts a word into the Trie.
  + It iterates over each character of the word, converts uppercase letters to lowercase (if necessary), and inserts each character as a node in the Trie.
  + At the end of the word, it marks the last node as the end of a valid word.

**Spell Checking**:

* + The **search** function in the **Trie** class checks if a given word exists in the Trie (i.e., whether it is spelled correctly).
  + It traverses the Trie based on the characters of the input word.
  + If it reaches the end of the word and the corresponding node is marked as the end of a valid word, it returns true; otherwise, it returns false.

**Suggestions for Incorrect Words**:

* + The **suggestCorrections** function in the **Trie** class provides suggestions for incorrect words.
  + It traverses the Trie based on the characters of the incorrect word until it finds a node that doesn't have a corresponding child node (indicating a misspelled portion of the word).
  + Then, it recursively explores all possible paths from that point to generate suggestions for corrections.

**Main Function**:

* + The **main** function:
    - Loads a dictionary of valid words into the Trie data structure.
    - Reads an input text file, checks each word against the dictionary, and identifies misspelled words.
    - For misspelled words, it provides suggestions for corrections based on the Trie structure.
    - It allows the user to choose from suggestions, ignore, or provide custom corrections for each misspelled word.
    - Finally, it prints the corrected paragraph.

Overall, the algorithm efficiently handles spell checking and correction by utilizing the Trie data structure to store the dictionary and providing suggestions based on the Trie's structure.