INTRODUCTION

This project was made for the Data Structures and Algorithm course for the 2025th Spring semester of 2025 of the Gebze Technical University by Tuana Melisa Aksoi. The main idea of the project was to make spell checker application using Java, focusing on the creating custom data structures such as hash sets and array lists.

The primary function of the spell checker is to determine whether a given word is correctly spelled according to a dictionary and, if not, to provide possible suggestions. The dictionary is provided as a text file named dictionary.txt, which contains a list of words. When the user enters a word, the system checks if the word exists in this dictionary. If it does not, the program generates and returns a list of suggested alternatives that are similar to the input word.

The program uses a custom implementation of a hash set, called GTUHashSet, to store the dictionary words. Hash set allows quick verification of whether a word is present in the dictionary. The suggestion mechanism is based on generating edit-distance variations of the user's input word, specifically variants with one or two character-level edits such as insertion, deletion, substitution, or transposition. Valid suggestions are collected using a custom array list implementation called GTUArrayList, which stores the output in a dynamic list.

Also the project has the performance measurements such as the number of collisions during the dictionary loading and the memory used to load the dictionary.

METHODOLOGY

Entry Class

The entry class represents a single key-value pair. Each entry stores a key, its associated value, and a Boolean flag to indicate whether the entry has been logically deleted. This flag tells the program whether an entry was deleted or not, without actually removing it from the table. This helps the hash table to keep working properly when searching or adding new items, because it won't stop too early if it sees a spot that looks empty but actually had something removed.

GTUHashMap Class

This class is a custom implementation of a hash table using the quadratic probing for collisions.

Constructor

The constructor firstly initializes the hash table with the default capacity of 11. It sets the size to 0 and creates an empty array of Entry. The collision counter is initialized to 0.

Put Method

Put functions purpose is to add the key-value pairs to the hash map. Firstly it checks if the key is null and if so throws an execption. Then calculates the load factor which is current size divided by table length, if the load factor exceeds the threshold which is 0.75, it calls the rehash function to expand the table. The methods computes the hash index of the key and applies quadratic probing to solve the collisions and find good position. If a matching key already exists, its value is updated. If a collision occurs, meaning the index is occupied by a different key, the collision counter is incremented. When an available or deleted index is found, a new entry is inserted, and the size is increased accordingly.

Get Method

The purpose of the get function is to retrieve the value with a specific key from the hash map. Firstly is checks if the given key is null and if so throws a NullPointerException. Then it calculates the hash index using the keys hash code and applies quadratic probing to search through the table. For each probed index, if a null entry is encountered, it means the key does not exist in the map, so the method returns null. If it finds a matching key that is not marked as deleted, it returns the associated value. If the key is not found after probing the entire table, the method returns null, indicating the key does not exist in the map.

Remove Method

The remove function is responsible for deleting the key-value pairs from the hash table by marking them as deleted. It checks if the key is null, then computes its hash index and uses quadratic probing to find the key. If found and not already deleted, it marks the entry as deleted and decreases the size, allowing future insertions in that spot without breaking the probing sequence.

ContainsKey Method

The containsKey method checks if the given key exists in the hash map. It validates the key is not null, calculates its hash index, and uses quadratic probing to search the table. If it finds a non-deleted entry with the matching key, it returns true; otherwise, it continues probing or returns false if the key is not found.

Rehash Method

Rehash function is used for the expanding the hash table when needed. Firstly it calculates the new capacity by doubling the current capacity and finding the next prime number greater than the doubled value using the findNextPrime method. A new table is created with the new capacity, and the old table is stored for rehashing, the function resets the size and collision count to zero as the table is being resized. Then it iterates through the old table and inserts all the not deleted entries to the new table using the put method.

findNextPrime Method

The findNextPrime method calculates the next prime number greater than or equal to the doubled capacity, ensuring that the new table size is optimal for performance.

isPrime Method

The isPrime method checks if a number is prime by testing if it has any divisors other than 1 and itself.

GTUHashSet Class

The GTUHashSet class is a custom implementation of a set, utilizing a GTUHashMap<E, Object> as the underlying data structure. It stores elements as keys with a constant WORD object as the value to ensure uniqueness. The class provides methods to add elements (add()), remove elements (remove()), check if an element exists (contains()), get the size of the set (size()), and track the number of collisions that have occurred in the hash map (getCollisionCount()).

GTUArrayList Class

The GTUArrayList class is a custom implementation of a dynamic array that stores String elements. It initializes with a default capacity of 10 and resizes automatically when the number of elements exceeds the current capacity. The class provides methods to add elements (add()), retrieve the size of the list (size()), and access elements by index (get()). When the array reaches its capacity, the extendCapacity() method doubles the size of the underlying array and copies the existing elements to the new array. This ensures that the list can grow dynamically as more elements are added.

SpellChecker Class

initializeDictionary Method

As parameter this function takes a path to the dictionary file that contains a list of words.It first measures the memory usage before loading the dictionary using Runtime.getRuntime(). Then it reads the each line from the file, loads the words, measures memory usage before and after loading, and stores the words in a custom hash set. It performs garbage collection before and after loading the dictionary to measure the memory usage accurately. The memory used for loading the dictionary is stored in dictionaryMemoryUsage.

findSuggestions Method

The purpose is to find and returns a list of suggestions for a given word based on the edit distance. The parameter of the function is a String type, the word to check and generate suggestions. Firstly the function converts the input to lowercase. Initializes a GTUHashSet to keep track of already-seen variants and a GTUArrayList to store suggestions. If the word exists in the dictionary, it returns an empty suggestion list (the word is correct). Generates all possible variants of the word that are one edit (insert, delete, substitute, or transpose) away. Checks each edit-distance-1 variant, If it exists in the dictionary and hasn't been seen yet, it's added to the suggestions. Repeats the process for distance-2 variants, using the results from distance-1 variants as a base. And in the end returns the list of valid suggestions.

generateEditDistance1 Method

The main goal is to generate all possible variations of a word that are one edit away. As parameter this function takes the original word for which single edit variants will be generated. Firstly it creates an empty GTUArrayList to store the results, calculates the word's length, converts the word into a character array for easy manipulation, defines the English alphabet as a character array to use for substitutions and insertions, and stores the alphabet's length to optimize looping operations. The function has 4 ways to generate the possible variants. First one is transposition, this method about the swapping each pair of adjacent characters in the word. Second one is substitution, creating new words by replacing each character in the original word with every other letter of the alphabet. Third one is deletion, generates variants by removing one character at a time from the word. And the last one is insertion produces new words by inserting every letter of the alphabet at every possible position within the word. In the end the method is returns the variants ArrayList.

generateEditDistance2 Method

The generateEditDistance2 method creates a list of words that are two character edits away from the input word. It starts by generating all words that are one edit away (using generateEditDistance1). Then, for each of those distance-1 words, it generates their own distance-1 variants—effectively producing distance-2 variants from the original word. It filters out variants that differ in length from the original by more than two characters to avoid overly dissimilar suggestions. A GTUHashSet named seen ensures that only unique variants are added to the final GTUArrayList, which is then returned.

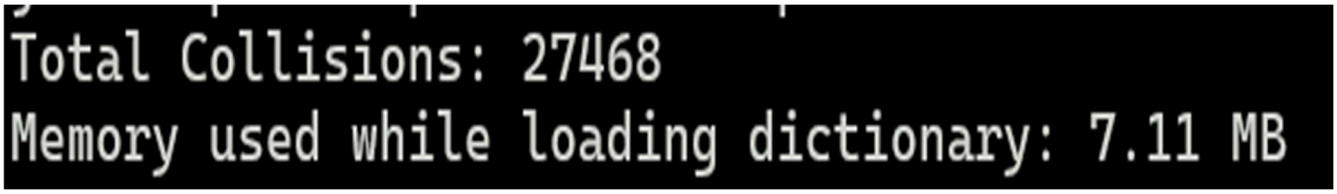
Main Method

This main method begins by loading a dictionary from a file using initializeDictionary, which also calculates memory usage and collisions in the hash table. It then enters an infinite loop where it repeatedly prompts the user to input a word. If the word is found in the dictionary, it prints “Correct.” If not, it prints “Incorrect.” and provides suggestions for possible correct words by calling findSuggestions. It formats the suggestions as a comma-separated list or informs the user if no suggestions are found. The method also measures and displays the time taken for each lookup and suggestion process in milliseconds.

RESULTS AND DISCUSSION

Results

The program was tested on two different computers with varying hardware capabilities. The screenshots below from my computer which is older and slower than the other computer sometimes for the different words it gives more than 100ms suggestion time but when the same word entered a couple times to the program the suggestion time to down each time and in the end it is less than 100ms.




```
Enter a word: kebra
Incorrect.
Suggestions: debra, verba, kerbs, kerb, debar, kebab, cobra, cera, libra, membra,
sera, terra, tetra, umbra, vera, zebras, kenya, kerry, kerr, ezra, ebba, kebob, kerf,
bra, era
Lookup and suggestion took 81.35 ms
Enter a word: kebra
Incorrect.
Suggestions: debra, verba, kerbs, kerb, debar, kebab, cobra, cera, libra, membra,
sera, terra, tetra, umbra, vera, zebras, kenya, kerry, kerr, ezra, ebba, kebob, kerf,
bra, era
Lookup and suggestion took 75.43 ms
Enter a word: kebra
Incorrect.
Suggestions: debra, verba, kerbs, kerb, debar, kebab, cobra, cera, libra, membra,
sera, terra, tetra, umbra, vera, zebras, kenya, kerry, kerr, ezra, ebba, kebob, kerf,
bra, era
Lookup and suggestion took 65.39 ms
Enter a word: kebra
Incorrect.
Suggestions: debra, verba, kerbs, kerb, debar, kebab, cobra, cera, libra, membra,
sera, terra, tetra, umbra, vera, zebras, kenya, kerry, kerr, ezra, ebba, kebob, kerf,
bra, era
Lookup and suggestion took 59.52 ms
```

```
Enter a word: testt
Incorrect.
Suggestions: teste, tests, testy, test, beset, bests, best, festa, jests, jest, lest, nests, nest, pests, pest, resat, reset, resit, rests, rest, sestet, vests, vest, west, zesty, zest, taste, tasty, teats, teat, teeth, tempt, tenet, tenth, tents, tent, texts, text, tess, tested, tester, testes, testis, est, tes, testate
Lookup and suggestion took 67.23 ms
Enter a word: testt
Incorrect.
Suggestions: teste, tests, testy, test, beset, bests, best, festa, jests, jest, lest, nests, nest, pests, pest, resat, reset, resit, rests, rest, sestet, vests, vest, west, zesty, zest, taste, tasty, teats, teat, teeth, tempt, tenet, tenth, tents, tent, texts, text, tess, tested, tester, testes, testis, est, tes, testate
Lookup and suggestion took 105.03 ms
Enter a word: testt
Incorrect.
Suggestions: teste, tests, testy, test, beset, bests, best, festa, jests, jest, lest, nests, nest, pests, pest, resat, reset, resit, rests, rest, sestet, vests, vest, west, zesty, zest, taste, tasty, teats, teat, teeth, tempt, tenet, tenth, tents, tent, texts, text, tess, tested, tester, testes, testis, est, tes, testate
Lookup and suggestion took 69.90 ms
Enter a word: testt
Incorrect.
Suggestions: teste, tests, testy, test, beset, bests, best, festa, jests, jest, lest, nests, nest, pests, pest, resat, reset, resit, rests, rest, sestet, vests, vest, west, zesty, zest, taste, tasty, teats, teat, teeth, tempt, tenet, tenth, tents, tent, texts, text, tess, tested, tester, testes, testis, est, tes, testate
Lookup and suggestion took 62.62 ms
Enter a word: testt
Incorrect.
Suggestions: teste, tests, testy, test, beset, bests, best, festa, jests, jest, lest, nests, nest, pests, pest, resat, reset, resit, rests, rest, sestet, vests, vest, west, zesty, zest, taste, tasty, teats, teat, teeth, tempt, tenet, tenth, tents, tent, texts, text, tess, tested, tester, testes, testis, est, tes, testate
Lookup and suggestion took 51.41 ms
```

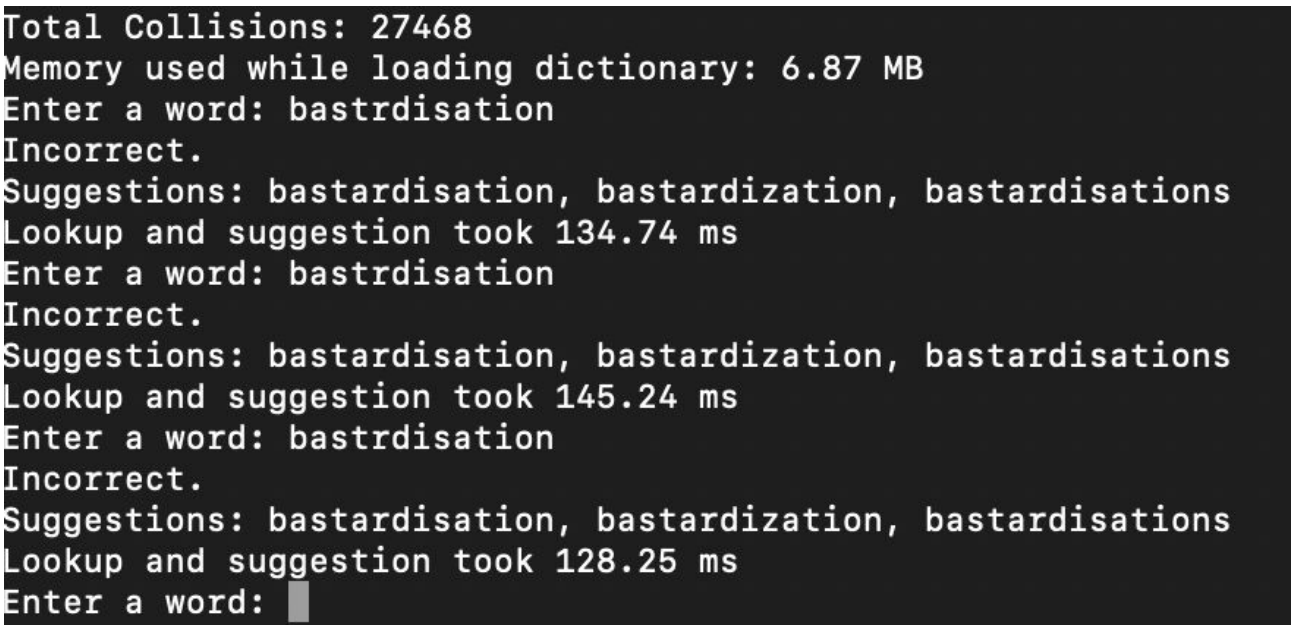
```
Enter a word: hello
Correct.
Lookup and suggestion took 0.20 ms
Enter a word: eagle
Correct.
Lookup and suggestion took 0.09 ms
Enter a word: slim
Correct.
Lookup and suggestion took 4.56 ms
Enter a word: abandonment
Correct.
Lookup and suggestion took 0.26 ms
```

Sometimes some long inputs can take more than 100ms like the word “bastrdisation”.

```
Total Collisions: 27468
Memory used while loading dictionary: 7.11 MB
Enter a word: bastrdisation
Incorrect.
Suggestions: bastardisation, bastardization, bastardisations
Lookup and suggestion took 342.81 ms
Enter a word: bastrdisation
Incorrect.
Suggestions: bastardisation, bastardization, bastardisations
Lookup and suggestion took 283.55 ms
Enter a word: bastrdisation
Incorrect.
Suggestions: bastardisation, bastardization, bastardisations
Lookup and suggestion took 387.41 ms
```

I tried the running the same code on the different computer more new and more faster than mine. And the inputs that took about 100ms on my computer took less than 50ms on the my friends device.

```
Total Collisions: 27468
Memory used while loading dictionary: 6.87 MB
Enter a word: testt
Incorrect.
Suggestions: teste, tests, testy, test, beset, bests, best, festa, jests,
jest, lest, nests, nest, pests, pest, resat, reset, resit, rests, rest,
sestet, vests, vest, west, zesty, zest, taste, tasty, teats, teat, teeth,
tempt, tenet, tenth, tents, tent, texts, text, tess, tested, tester, testes,
testis, est, tes, testate
Lookup and suggestion took 46.06 ms
Enter a word: testt
Incorrect.
Suggestions: teste, tests, testy, test, beset, bests, best, festa, jests,
jest, lest, nests, nest, pests, pest, resat, reset, resit, rests, rest,
sestet, vests, vest, west, zesty, zest, taste, tasty, teats, teat, teeth,
tempt, tenet, tenth, tents, tent, texts, text, tess, tested, tester, testes,
testis, est, tes, testate
Lookup and suggestion took 37.56 ms
Enter a word: testt
Incorrect.
Suggestions: teste, tests, testy, test, beset, bests, best, festa, jests,
jest, lest, nests, nest, pests, pest, resat, reset, resit, rests, rest,
sestet, vests, vest, west, zesty, zest, taste, tasty, teats, teat, teeth,
tempt, tenet, tenth, tents, tent, texts, text, tess, tested, tester, testes,
testis, est, tes, testate
Lookup and suggestion took 31.64 ms
Enter a word: testt
Incorrect.
Suggestions: teste, tests, testy, test, beset, bests, best, festa, jests,
jest, lest, nests, nest, pests, pest, resat, reset, resit, rests, rest,
sestet, vests, vest, west, zesty, zest, taste, tasty, teats, teat, teeth,
tempt, tenet, tenth, tents, tent, texts, text, tess, tested, tester, testes,
testis, est, tes, testate
Lookup and suggestion took 30.41 ms
```



Discussion

The program produced correct suggestions and ran without any issues. On my computer, some inputs took around 100 ms to process. When I tested it on a newer and faster machine, the same inputs were processed a lot faster. This shows that the program performs better with more powerful hardware.

Conclusion

The purpose of this project was to implement a spell-checker that finds incorrect words and offers suggestions using edit-distance techniques. Through the implementation and testing of custom data structures and edit-distance algorithms, it was confirmed that the system correctly identifies misspellings and provides valid alternatives. The application demonstrated consistent and efficient performance across various test cases. Overall, the spell-checking system functions as intended and achieves its goal of helping users identify and correct simple spelling errors.

References

https://www.geeksforgeeks.org/substring-in-java/

https://www.youtube.com/watch?v=QvHBHuuddYk&t=777s

https://www.youtube.com/watch?v=H62Jfv1DJlU

https://www.geeksforgeeks.org/java-util-hashmap-in-java-with-examples/

https://www.geeksforgeeks.org/hashset-in-java/

https://www.vogella.com/tutorials/JavaDatastructureList/article.html

https://www.geeksforgeeks.org/measure-time-taken-function-java/

https://www.geeksforgeeks.org/garbage-collection-java/