Information:

Giray berk Kuşhan-Begüm Şara Ünal

ID:10889878942-10096206194

Section 4

Algorithm 2 - Homework 4

Question 1

Problem Statement and Code Design:

The question is about implementing a Trie structure and apply operations on this structure. Count prefix, Boolean search, and reverse find. This code occurred with Main class, TrieNode class and Trie class as static. In additional occurs several methods. Aim of this code is shortly, designing allows separating and reusability in Trie class by using several operations on the data structure as Trie.

Implementation and Functionality:

Main, TrieNode, Trie classes handle executing operations and inputs on Trie. Additionally, these classes use a lot of different methods. Shortly, these methods and classes do these operations:

TrieNode: it shows a node Trie data structure, and it has these methods and fields:

wordsEnd: This is a flag which has Boolean type. Indicates node marks if it is in the end.

children: it is used for character mapping to TrieNodes. It represents childNode.

Trie: It provide Trie data structure operations like TrieNode.

insOrderWrds: Stores string list that are inserted into the Trie.

rootAsTrie: Related with root Node of the Trie.

add (String word): Used for inserting operation into Trie word by traversing characters and update Trie.

lookup (String word): it controls whether given word is existed or not in the Trie by traversing the characters. Also, these returns true if it reaches last valid node of word.

PrefixCounter (): used for counting for number of occurrences of all prefixes. It does this iterating with inOrderWrds. Then prints occurrences number by using count.

reverseFind (String suffix): This method is using for finding and printing all words by lexicographic order by iterating insOrderWrds. Also, it adds it is using for adding matching words to a LinkedList and it sorts and print results.

Main: It is main class, and it is using in order to enter input. Also executes operations related with Trie.

Testing:

In testing part, we created a tester class which is generating extra test cases. These cases cover several ways of the solution. They will contain input/output provided and extra cases in order to validate response of program. They include several scenarios and provide for ensuring whether program handles various inputs truly or not some of them are looking for these scenarios:

Reverse finding, prefix counting, word searching.

Final Assessments:

In this question Trouble point is clearly understanding the question and thinking how we use Trie algorithm in this question. Also, we can say that Trie data structure implementation correctly and optimizing efficient data structures solution is most challenging part. we liked this assignment because we have learned many things about Tries data structure and string operations efficiently.

Question 2

Problem Statement and Code Design:

We applied a String sorting algorithm to solve the question. The code we created first takes two separate string arrays of different lengths as input. After taking these inputs, our code finds the distance between the arrays in the same directory. Then our code checks if the calculated distance value is odd or even. If the distance is even, the result of the string sorting algorithm we created by applying the given rules is returned. If the distance is odd, our code applies lexicographically in ascending order. Finally, the resulting sequence is written.

The code is split into two classes to solve the question:

Driver Class: oversees controlling the main execution flow, handling user input, calling stringSort class methods, and printing the sorted array.

stringSort Class: The class stringSort manages the utility methods and sorting algorithms to sort the arrays.

Implementation and Functionality:

Driver Class:

- Reads arrays of strings from the input. After that, creates an object to call and use the stringSorts' class method. After that, prints the sorted array according to the formulas.

- **stringSort Class:**
  - takeInput (): Readslines using Scanner and **nextLine ().** After that, splits the input and returns the split array.
  - **calculateString ():** Gets the length of the string to iterate over each character in the string. Using the formula calculates the value of the character. Add all calculated values in the integer sum and return this value.
  - **sortStringAlgorithm ():** By calling the *'calculateString'* method, calculate the distance between string values. If this result is even called the *'sorting'* method. If it is not called the *'alfabeticOrder'* method.

- **sorting ():** Inserts characters from string 'sequence' in order. Using the left. indexOf (String.valueOf(ch)) method gets the index of the character in the left string. Until no more indexes are found appends, the character deletes the character at the index from the left StringBuilder and gets the index of the character. After that, adds the remaining characters not in 'sequence'. Finally, returns the result.
- **alfabeticOrder ():** Method converts input string to char array. After that, sorts the array in lexically order. Returns sorted array.

Testing:

We tested whether our code worked correctly by creating new scenarios. In our test, we check the input and output states with different inputs, such as arrays of different sizes and characters. We also tested whether our code produced any bugs.

Final Assessments:

The troubling point on this assignment for question two was the calculation of string values according to the formula.
The most challenging part is question two is understanding and implementing coding requirements. In question two, I liked to use what I learn in lectures, and I learned how I use the string sort algorithm.