Assignment 3

Please read this document very carefully. Follow instructions exactly. If you have any questions please post them to MS Teams or ask during office hours.

This assignment is due Dec. 8th, by 11:59pm

In this assignment you are responsible for implementing a custom data structure called a "Word Tree". The description of such a data-structure is given later in this document. You are given two .java files as starter code.

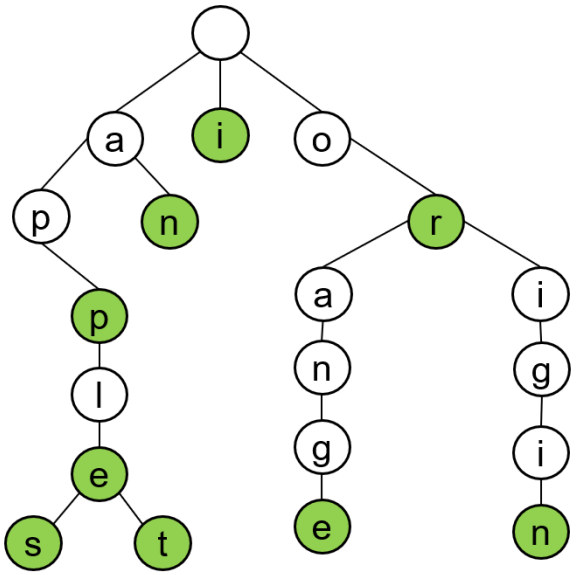
- `TreeNode.java`
- `WordTree.java`

You are responsible for submitting one .java file after making the appropriate additions.

- `WordTree.java`

WordTree

A word tree is a data-structure used for storing strings. Within the tree, each node represents a specific character. Unlike a binary tree where each node has at most two children, nodes in a Word Tree can have up to 26 children – one for each letter of the alphabet. These children are stored in an array. Index 0 represents the child for the letter "a", and index 25 represents the child for the letter "z". See the figure below, where this Word Tree stores the Strings: an, app, apple, apples, applet, I, or, orange, origin.



Note, in the above Word Tree "a" is not a word. You are responsible for implementing four methods in WordTree.java.

- contains(String word): Returns true if and only if word is in the Word Tree. In our example, contains("orange") should return true; contains("oranges") should return false.

- addToEach(char c, ArrayList words): Returns a version of the ArrayList words, where c has been added to the beginning of each element of words. For example:
  - addToEach("a", ["pp", "pple", "n"]) should return an ArrayList which only contains: "app", "apple", and "an".
  - addToEach("o", [", "range"]) should return an ArrayList which only contains: "o", and "orange".
  - addToEach("f", []) should return an empty ArrayList.
- suggestAutoCompletes(String prefix): Returns an ArrayList which contains (and only contains) all Strings in the Word Tree which begin with prefix. In our example:
  - suggestAutoCompletes("app") should return an ArrayList which only contains: "app", "apple", "apples", "applet".
  - suggestAutoCompletes("ora") should return an ArrayList which only contains: "oranges".
  - suggestAutoCompletes("be") should return an empty ArrayList.
- suggestCorrections(String word, int offBy): Returns an ArrayList of words contained within the Word Tree, which are the same length as *word*, and differ by at most offBy characters. In In our example:
  - suggestCorrections("appler", 1) should return an ArrayList which only contains: "apples", and "applet".
  - suggestCorrections("apllet", 1) should return an ArrayList which only contains: "applet".
  - suggestCorrections("xxxxxx", 6) should return an ArrayList which only contains: "apples", "applet", "orange", and "origin".
  - suggestCorrections("xxxxxx", 0) should return an empty ArrayList.

Additional Comments

- This assignment uses ArrayLists for some of its methods. Take a look at what built-in functionality Java has for ArrayLists. The method addAll() may be particularly useful.
- If you are struggling with recursion this assignment will be difficult. If you do not know where to begin, review the lectures on recursion, especially the one where we implemented power set. Once you begin to understand recursion this assignment will become clearer.
- There are two private helper methods letterToInt() and intToLetter() – use them.
- Take a look at the methods which are already implemented. Specifically, insertWord() and numOfWords(). Understanding these methods will have you implement the rest.
- For all methods dealing with Strings, you may assume these Strings only contain lowercase letters.
- When returning an ArrayList, the order in which certain Strings appear does not matter. Instead, ensure the following two points are met:
  1. All Strings that should be in the ArrayList are in the ArrayList.
  2. No String which should not be in the ArrayList is in the ArrayList.
- Do not add any other import statements.
- Do not change the method headers for the methods you are responsible for implementing
- You may add other methods to aid in your implementation (in fact, I suggest it).

Submitting and Grading

This assignment will be submitted electronically via Avenue. This assignment is worth 15% of your final grade. Grading is done automatically. That is, a program calls your method, passes it certain arguments, and checks to see if it returns the expected output. The assignment is broken down as follows:

- contains: 15%
- addToEach: 15%
- suggestCorrections: 35%
- suggestAutoCompletes: 35%

Good luck!

Academic Dishonesty Disclaimer

All of the work you submit must be done by you, and your work must not be submitted by someone else. Plagiarism is academic fraud and is taken very seriously. The department uses software that compares programs for evidence of similar code.

Please don't copy. The TAs and I want you to succeed and are here to help. Here are a couple of general guidelines to help you avoid plagiarism:

Never look at another assignment solution, whether it is on paper or on the computer screen. Never show another student your assignment solution. This applies to all drafts of a solution and to incomplete solutions. If you find code on the web that solves part or all of an assignment, do not use or submit any part of it! A large percentage of the academic offenses in involve students who have never met, and who just happened to find the same solution online. If you find a solution, someone else will too.