Overlapping Words - A4

Due: Nov 22, 2022 at 11:59PM

1 Overlapping Words - A4 Specification

1.1 Assignment Instructions

- This is an *individual* assignment.
- Make sure to read through all of the specifications so your submission is complete.
- Follow all the submission steps in the Setup document by the assignment deadline.

1.2 Assignment Link

The skeleton code for the assignment is available at `https://classroom.github.com/a/c5uWnVFK`.

2 Implementation

The goal of this assignment is to find the 10 longest words that are found in every file in a set of ASCII files. Use your knowledge of data structures to implement a solution. Create any other Java classes you would like. Do not change the method signature of `FindLongestCommonWords`.

2.1 Detailed Rules

Definition of a word:

- A word is a sequence of alphabetic characters, upper or lower case
- A word has no punctuation in the middle: 'long-term' would count as two separate words
- A word has a minimum of 8 characters to be considered
- A word has a maximum of 50 characters to be considered
- Words should all be converted to lower case before analysis – 'expression' and 'Expression' and 'EXPRESSION' are all the same word

Rules for output:

- The `FindLongestCommonWords` method should return an array of Strings **in sorted order**
- The array should contain the 10 longest words that appear in every input file
- If there are fewer than 10 overlapping words, the array should contain all overlapping words
- If you put an overlapping word into the array, put all overlapping words of that length in the list. This may cause the array may contain more than 10 words if there are many overlapping words of the same length
- The array should be appropriately sized to hold its words

2.2 Coding

Break the problem down into small testable steps: read the files, extract the words, store the words in some data structure, analyze the word overlaps, and compile the results. For input and data structures, import and use any Java libraries you find helpful. Throughout the semester, we have learned about many data structures and this is your opportunity to select the structure or structures that you think are best to efficiently solve the problem.

2.3 Testing

Incrementally test your program by printing out results (or setting breakpoints in the debugger and confirming expected values) each step of the way. Can you read the files? Can you parse the words? Can you store them and retrieve them? Can you compare word sets between text files?

Make sure that you get proper results for all the output cases: fewer than 10 overlapping words, exactly 10, or more than 10.

I have included some JUnit tests to test simple files, which may be useful for testing. I have also supplied some large text files if you are looking for longer texts to work with.

2.4 Competition

There are many ways of solving this problem, some more efficient than others. I will give 10 extra credit points to the submission that run the fastest on a group of large files that I have not supplied. In addition to that, I will give 5 extra credit points to everyone whose submission runs faster than mine. Only correct solutions will be considered for extra credit points.

3 Double Check:

- Have you implemented the `TODO` method?
- Have you added/committed/pushed your code from **all** the files you modified?

4 Grading

Parse words from text file: +20%

Store words in a reasonable data structure: +20%

Compare words between different files: +20%

Generate longest-ten-words list: +40%

Top speed: +10% (bonus)

Faster than Frank's implementation: +5% (bonus)