Title: CSE4ALG Assignment 2 Repost

Student_ID: 19190687

Student_Name: Prasad Belhe

Section 1:

WordMatcher :-

Brief Explanation:

- Main class
- Accepts input arguments
- threadloader() - Starts a thread for loading information from data and pattern file.
- Triggers method for generation sorting and finds neighboring elements for lexicon.
- Triggers method to find matching patterns.
- threadstorer() - Stores Lexicon and matching pattern result using thread.

Variables:

- wordList : ArrayList<Word>
- patternList : ArrayList<String>
- patternOut : ArrayList<String>
- inputFile1 : String
- inputFile2 : String
- outputFile1 : String
- outputFile2 : String

Methods:

- main(String[])
- threadloader()
- threadstorer()
- loadData(String)
- loadPattern(String)
- writeData(String)

Word :

Brief Explanantion:

- This class implements comparable, to compare objects of this class.
- Stores Word Details.
- Includes word, Frequency count, Neighboring words and their getter and setters.
- Overriten toString() method for formatted displaying data.

Variables:

- word : String
- frequency : int
- neighbourList : ArrayList<String>
- path : List<String>
- lastWord : String
- length : int

Methods:

- Word(String, int)
- compareTo(Word)
- getWord()
- setLength(int)
- getFrequency()
- setFrequency()
- addNeighbour(String)
- getNeighbors()
- toString()

Utility:

Brief Explanation

- A special class without variables, only used for methods.
- Finds matching pattern
- Compares neighboring element
- Writes data to file

Methods

- removeSpecialCharacters(String, ArrayList<Word>)
- neighbourFoundCode(String, String)
- findMatchingPatternWords(String, ArrayList<Word>, ArrayList<String>)
- writeTofile(ArrayList, String)

SortNSearch:

Brief Explanation

- Used for sorting and searching
- Uses quick sort algorithm for sorting word lexicon as per dictionary order.
- Searches neighboring element using findNeighbouringElements method.

Methods

- shellShellSort(List<E>)
- getShellGapSequence(int)
- shellSort(List<E>, int[])
- hSort(List<E>, int)
- sortGappedSubArray(List<E>, int, int)
- insertElements(List<E>, int, int, int)
- quickSort(List<E>)
- quickSort(List<E>, int, int)
- partition(List<E>, int, int)
- swap(List<E>, int, int)
- findNeighbouringElements(List<Word>)

Section 2:

In this section,

(a) Working on huge files are challenge for sorting and searching lexicon as this process requires lot of processing time in comparison. Optimal hardware usage and suitable algorithm helps in improving results.

(b) After trying and implementing multiple algorithm helped me calculate time complexity.

Quick sort gives better results. Understanding of the dataflow helped in implementing threading technique. This process improved processing power or hardware cores and waiting times.

Section 3:

- The functional correctness of the program is checked by identifying whether the neighbour stored in the first output file are of same length and differ by one word.
- The neighbor is also present in each other’s list.
- The functional correctness is checked by observing whether the specification of the program is satisfied.

Carefully considering documentation, I decided some test cases based on following factors:

- Case sensitivity of lexicon should be small case.
- Alphabetical order of lexicon should be dictionary order.
- Frequency count of word should match real count.
- Neighboring word should differ by 1 word and of same length.
- Regex pattern should collect respective lexicons.
- File should read and write data.
- Functional correctness should meet programing satisfaction.

Section 4:

- Programing efficiency is calculated based on both functional and non-functional requirements.
- Software program should meet documented functionalities.
- Ie. It should give accurate lexicon word with its’s frequency and neighboring element.
- It is also expected that the program should sort and search the result as quick as possible.
- We can increase the efficiency by using suitable algorithm for data and improving hardware usage to optimal level.

TASK4

Given :

M = 21

N = $2K^{H}$- 1

Solution:

K=sqrt(21/2) = sqrt(10.5) = 11

N+1=$2K^{H}$

(N+1)/2 = $2K^{H}$

H = $\log_{11}$((N+1)/2)

H = $\log_{11}$((1000000+1)/2)

H= $\log_{11}$(1000001/2)

<u>H=5.47</u>