**ADAPTIVE HUFFMAN COMPRESSOR**

**Overview**

This program takes input in the form of text file and encodes/ decodes as per the constraints provided by user.

**Files and external data**

Users need to provide a text file with extension which should be same directory location as the program file. No database is used by the program since the file is processed on the go. Output file is generated in the current directory as per name provided.

**Data structures and their relations to each other**

Program uses a Node structure to handle the characters iterated over string content of file. HashMap find its key usage in this program. Node class manages characters in the form of tree where elements are present on the leaf. Additionally, TreeSet is also used to manage sorted frequency set while rebuilding the tree. There is custom comparator defined for elements managed in treeset.

To iterate over the data structures, it uses for loop, while loop and iterators.

**Assumptions**

* Input file is already having required read permissions.
* Input file is resting on the same location as .java file.
* There are only 10 levels possible ranging from 0-9.
* For level input as 0, file content is to be rebuilt for every character.
* When level is 0 and after last character, end-of-file character represented as “EOF” (in program) is incremented for its frequency and then the tree is forwarded for rebuilding.

**Key algorithms and design elements**

User is prompted for inputting key information and constraints for encoding and decoding procedure.

Encoding is done in two phases. In first phase, characters are inserted on rightmost end of tree recursively. This tree is fed to rebuilder method which manages a sorted frequency for every element of node. These sorted elements are picked one by one and merged to form a new tree. Merged node elements are stored in a node dictionary map to be used in next round of iteration of rebuilding process.

Rebuilt node is sent back to pending character’s insertion in sequence. Frequencies of node elements are reset to 0 if reset flag is on, prior to every round of rebuilding operation.

**Limitations**

* Characters are stored and processed in form of Strings rather than in ‘Character’ object.
* Frequencies of characters don’t stay updated all the lifetime of program. They are managed separately.
* Program is currently unable to decode given Huffman encoded file.

**Citations**

**1.** Baeldung. (2019). *Implementing a Binary Tree in Java | Baeldung*. [online] Available at: https://www.baeldung.com/java-binary-tree [Accessed 28 Jan. 2019].

**2.** Java - How to create new Entry (key, v., Leschinski, E., Filotto, N. and Beckman, N. (2019). *Java - How to create new Entry (key, value)*. [online] Stack Overflow. Available at: https://stackoverflow.com/questions/3110547/java-how-to-create-new-entry-key-value [Accessed 28 Jan. 2019].

**3.** JournalDev. (2019). *java.util.ConcurrentModificationException - JournalDev*. [online] Available at: https://www.journaldev.com/378/java-util-concurrentmodificationexception [Accessed 28 Jan. 2019].

**4.** GeeksforGeeks. (2019). *Huffman Coding | Greedy Algo-3 - GeeksforGeeks*. [online] Available at: https://www.geeksforgeeks.org/huffman-coding-greedy-algo-3/ [Accessed 28 Jan. 2019].

**5.** Baeldung. (2019). *Count Occurrences of a Char in a String | Baeldung*. [online] Available at: https://www.baeldung.com/java-count-chars [Accessed 28 Jan. 2019].

**6.** Leaf, H. and Wills, S. (2019). *Huffman Coding - Get Path to Leaf*. [online] Stack Overflow. Available at: https://stackoverflow.com/questions/33464778/huffman-coding-get-path-to-leaf [Accessed 28 Jan. 2019].

**7.** Examples Java Code Geeks. (2019). *Java.util.TreeSet Example*. [online] Available at: https://examples.javacodegeeks.com/core-java/util/treeset/java-util-treeset-example/ [Accessed 28 Jan. 2019].