DATA STRUCTURES AND ITS APPLICATIONS (UE22CS252A)

Mini Project

TITLE: Huffman Encoding Visualizer

Team Members (SRN Name)

1. PES2UG22CS351 Neville Joseph Sabu
2. PES2UG22CS355 Nihal Satish
3. PES2UG22CS352 Nevin Marthandan
4. PES2UG22CS315 Mohammed Arfa

SYNOPSIS

HUFT is a project designed to showcase the implementation of lossless text compression through the utilization of the Huffman encoding technique. By implementing the Huffman coding algorithm, the system dynamically formulates a Huffman tree using information theory concepts. Each input string token is then encoded based on this dynamically generated Huffman tree. Our project has a graphical interface, which generates the Huffman tree procedurally, providing users with a visual representation of the compression process. The interface also offers in-depth analysis of each node within the Huffman tree, giving insights into the compression mechanism.

ADT

1. A Binary tree is used to represent the huffman tree which stores the character and frequency of each node.
2. We also a modified implementation of a priority queue for real time updation of the tree.
3. Both the Huffman tree and the priority queue is nested within a tree structure which keeps track of the current Huffman encoding instance.
4. createNode, mergeNode, freeNodes, treeInit and freeTree are used to manipulate the structure of the tree ie; both the huffman binary tree and priority queue by dynamic memory allocation and deallocation.
5. treeStateNext is used to change the state of the tree real time whenever called by modifying the tree and the priority queue.
6. treeInit initializes the tree along by taking the input string and generating character:frequency correlated nodes into the priority queue.
7. generateHuffmanCodes is a function used to recursively generate huffman codes for the current instance of encoding by inorder traversal within the tree.

Contribution of each Team Member

1. Renderer setup and drawNode - Nihal
2. NodeOverlay and state management - Nevin
3. createNode, mergeNode and treeInit - Arfa
4. Project structure - Neville

Learning Outcomes

1. We were able to understand and apply the concept of binary trees by applying the Huffman encoding Theory.
2. The concept of “Information Theory” was thoroughly studied by us, enabling a comprehensive understanding of lossless compression and its history.
3. GUI implementation using the raylib library provided deep insights into graphics programming.
4. Build tools like Cmake was used to automate builds, running tests, packaging etc. which gave us familiarity into working with large codebases and industry toolsets.
5. Collaborative workflows tools like git was used which helped us understand distributed version control systems on collaborative level.