Samantha Lee (sll155), Matthew Notaro (myn7)
Professor Francisco
Systems Programing
6 April 2020

Assignment 1

Description:
FileCompressor is a program that reads and writes files using Huffman Encoding.

How to Use:
FileCompressor can build a Huffman codebook, compress files, or decompress files, indicated by the flags -b, -c, and -d. An -R flag included alongside any one of these actions indicates that the action must be done recursively to a directory. To build, a user can enter a command-line prompt that follows the following format:

./fileCompressor -b <path or file>

To compress, a user must provide that follows the following format:

./fileCompressor -c <path or file> <codebook>

To decompress, a user can enter a command-line prompt that follows the following format:

./fileCompressor -u <path or file> <codebook>

Design & Implementation:
FileCompression largely relies on a binary search tree and a heap as its data structures.

When a codebook is built, the content of the book is read into a string. The program loops through the string, extracting tokens and delimiters and inserting them into a BST. During insertion, if the token is found in the tree, its frequency increases by one. If it is not found, it is inserted as a new BSTNode. Then, the BST is heapified — each node is converted to a heapNode, and are sorted so that the nodes with the smallest frequencies are deleted first. Then, huffEncode() follows the encoding algorithm and builds a Huffman Tree with the BSTNodes.

When a file is compressed, the contents of the file as well as the provided codebook are read into strings. Then, the program loops through each character of the file until a delimiter is found or end of time is reached and extracts each token, getting its corresponding Huffman code via the helper function getCodeFromBook(). The code is then appended to a new file. If it exists, the delimiter is also extracted and appended.

The function getCodeFromBook() accepts a token and a string of the codebook as parameters. If the token is a delimiter, the escape character is extracted from the codebook, used to update the token to represent the delimiter; otherwise, the token is unchanged. A pointer is set to the location at which the token is found in the codebook and it backtracks in order to read the corresponding code and return it.

When a file is decompressed, the contents of the file are read into a string, and the provided codebook is passed to the function bookToBST() to build a Huffman Tree. A pointer is set to point at the root of the tree, and the program loops through each bit of the file string, traversing down the tree accordingly. Once a leaf node is reached, the program checks if its token is a representation of a delimiter (in which case it appends the true delimiter to the new file). If it is not, it appends the token to the new file.

If the recursive flag is set, recursion() descends into the directories and calls doOp() for each regular file it finds, effectively applying the action to each one.

Time-Space Analysis:

We deal with data structures when applying the following actions:

Build

- BST Insertion: Average O(log(n))
- Heapify: O(n*log(n))

Decompress

- bookToBST() —> Average O(log(n))
- Search —> Average O(log(n))