Keefer Rourke
Mr. Fryer-Davis

19 May 2015
ICS4UI

Text Compression and Decompression – Requirements

Text compression is a technique used to save disk space by representing strings of text in a binary file, which can be significantly smaller than the original plain text file. Huffman coding is a technique that uses a binary tree to store all the letters in a text file or string, along with their frequencies.

Reading from the top or root of the tree, each letter is associated with a unique variable length prefix code, where going left on the tree generates a zero, and going right on a tree generates a one. The letters are sorted by frequency, where the most frequent letter or character, has the smallest value in the tree.

Once the tree is constructed, it can be traced through to generate a binary number string for each character. These strings are then combined in order (as occurring in the original string), split into 8-bit chunks, and written to the disk as numbers. The tree is also written to the disk.

As far as decompression goes, the binary file will contain a representation of the original tree, and the numbers which represent the encoded text. The tree will need to be reconstructed perfectly, and the encoded text changed back into a binary string of ones and zeroes. The string will then be read digit by digit, where a zero means that the program should go left in the tree, and a one means that the program should go right in the tree. Once a leaf node has been found, the letter it contains will be added to another string of characters, until the binary digit string has been completely decoded. The new string will then be written to a new plain-text file.

This project will consist of five major parts:

- parsing the string and building the Huffman tree
- reading the Huffman tree to create the encoded string
- writing both the Huffman tree and encoded string to the disk
- reading the encoded file, to rebuild the tree
- decoding the string using the rebuilt tree

Keefer Rourke
Mr. Fryer-Davis

19 May 2015
ICS4UI

Approximate Schedule:

May 19 – May 22	Finish planning	• Finish planning and design documentation • Look at adding command line arguments for the program (for instance, `-c <filename>` to compress, and `-x <filename>` to decompress, similar to GNU tar)
May 23 – May 29	Implement string/file parsing	• Implement a Huffman tree library • Parse the strings from a file, and build a binary tree from the frequency of each character
May 30 – June 5	Generate prefix codes	• Read binary tree and generate variable length codes for each character, save to a string • Break up string into 8-bit binary numbers
June 6 – June 9	Write/read binary files	• Write Huffman tree and encoded string data to a file • Read encoded string data and Huffman tree from file
June 10 – June 17	Decode text	• Rebuild Huffman tree from binary file • Read encoded binary string, and get plain-text letters from the tree • Output decoded text to new file