NAME

fileCompressor - compresses .txt files and decompresses .hcz files using Huffman trees.

SYNOPSIS

fileCompressor |RECURSIVE| [MODE] [PATH] |CODEBOOK|

DESCRIPTION

fileCompressor has 3 main functions, building codebook, compressing files, and decompressing .hcz files. It will do so either on a single file or recursively on a directory.

-R

The program will find all files and subdirectories in the given path

-b

Will build a codebook from the given file(s).

-c

Compresses given file(s) using the given codebook.

-d

Decompresses given file(s) using the given codebook. Will also skip all non .hcz files

Author

Written by Andrew Park and Ryan Davis.

Reporting Bugs

Email rgd51@scarletmail.rutgers.edu or ap1614@scarletmail.rutgers.edu

SYNOPSIS

The project consists of mainly 5 files: **fileCompressor.c**, **fileCompressor.h**, **heapSort.c**, **heapSort.h**, **Makefile**

- **heapSort.c** contains the code for the heap sort algorithm, compiled as a library
- **heapSort.h** contains the function headers for heapSort
- **fileCompressor.c** contains the bulk of the code, compressing, decompressing functions, and building codebook
- **fileCompressor.h** contains the function headers for fileCompressor
- **Makefile** will automatically clean and compile fileCompressor

DESIGN

int main(int argc, char** argv)

- Main function of the program, handles recursive mode and figures out building codebook, compressing and decompressing files.

Node* tokenizeDict(int fd)

- Takes a file descriptor for a codebook file and will tokenize it and turn it into a Huffman tree.

int readFile(int fd, Node*** arr, int size)

- Takes a file descriptor for a file and will read all strings in the file and stores it in the array passed.

int decompressFile(Node* tree, int ofd, int nfd)

- Will take 2 file descriptors, the compressed file file descriptor, and the file descriptor for the decompressed file. Uses the given Huffman tree to decompress the file

int compressFile(cbLL* codes, int ofd, int nfd)

- Will take 2 file descriptors, the original file, and the file descriptor for the compressed file. Uses a linked list of the codes and the strings related to them to compress the file.

void createDictionary(Node* tree, int fd)

- Takes the Huffman tree and stores it in the file descriptor given.

File* recurseFiles(char* path)

- Takes the path for a directory and recursively searches for all files in subdirectories and returns a linked list of File struct, which contains the file descriptor of a file and its path.

Complexity

Time complexity for fileCompressor is at worst $O(n^2)$. This is due to building the codebook being done in $O(n^2)$ time, though this only occurs at the worst case when each string is unique. Otherwise, building the codebook would occur near O(nlogn) time.