CS 214: Systems Programming, Spring 2019

Devvrat Patel dap299 Section 08

Shubham Mittal sm1841 Section 02

Project 2: File Compression

Step1. Design and Implementation1.

In this project, the first thing I did is decompressing files.

I used opendir(), readdir() functions to look into the file system.

There I can get list of files and to open certain file, I used open() function.

By using IO functions, I got information from files and generated Huffman codes.

Then, according to those Huffman codes, I did compress and decompressing files.

In helper.h file, you can see these functions - declaration:

*void compress(int fd, char * input, int length, char ** cod, char ** tokens, int size);*

*void decompress(int fd, char * input, int length, char ** cod, char ** tokens, int size);*

In tokenizer.c file, you can see definition of those functions.

Step2. Design and Implementation2.

To generated codebook or Huffman codes, I used token and binary tree.

In helper.h file, you can see structures:

struct nodeLi, struct nodeT, struct tree.

In huffman.c, you can see the functions:

createNode, createTree, nodeSwap and so on.

fileCompressor.c is main c file.

There I used IO functions to analyze file structure.

I used token there.

So the program flow order is fileCompressor.c, tokenizer.c and huffman.c.

It shows file system managing, using token to generate codebook and implementation of Huffman code.

Implementation : fileCompressor.c

*unsigned int tokenize(char * input); //searches and finds total number of tokens in the book*

*void pArray(char ** cod, char ** tokens, char * input, int length); //goes through the codebook*

*int recFcn(int fd, char * file, char flag, char ** cod, char ** tokens, int size); // recursive function for everything*

*int main(int argc, char ** argv); //main funtion to take the input*

tokenizer.c

*int codebookTotal(char * input, int length); // gets the total lines in codebook*

*int search(char ** arr, int size, char * string); // looks for the string*

*void compress(int fd, char * input, int length, char ** cod, char ** tokens, int size); // creates the hcz file using the codebook*

*void decompress(int fd, char * input, int length, char ** cod, char ** tokens, int size); // checks the codebook and the hcz file to decompress and create/overwrite a new file*

Huffman.c

*nodeT * createNode(char * token, unsigned int oft) // creates the node for huffman tree*

*tree * createTree(unsigned int mSize) { // starts the tree*

*void nodeSwap(nodeT ** first, nodeT ** second) //updates the node based on the hierarchy*

*void heapify(tree * root, int index) // heap function from data structurers*

*nodeT * lastNode(tree * root); // provides the last node from the tree*

*void inNode(tree * root, nodeT * node); // inserts a node to the tree*

*void makeTree(tree * root); // makes the tree in the correct order*

*tree * enTree(unsigned int size, nodeLi * node); //makes the entire tree with all the nodes*

*nodeT * createHuff(unsigned int size, nodeLi * node); // creates the huffman tree using the ndoes and the root*

*void setCod(nodeT * node, unsigned short code_arr[], int parent, int fd); // left is 0 and right is 1 for the leaves*

And as you can see in fileCompressor.c and other functions, the runtime is:

O(lg n).

Thanks.