Project 2 Report Da Cheng id:0028653515

The second project is Huffman coding which consist of encoding and decoding part.

Encode:

The encoded file should have at least two parts: head information and encoded information from original texts. For head information, it has different versions such as the combination from post-order traversal of a tree and the size of the original text, the combination from pre-order traversal of a tree and the size of the original text. However, my algorithm is different. I first write the size of original text. The reason is that I need a reference to tell me when to stop decode in the decompression program. Then, I write the total number of characters appeared in the text without repetition, which remind me when to stop read the head information in the decompression program. After these two numbers, I write all characters appeared in the text followed by its frequencies. These characters and frequencies combinations are separated by the letter'\n'. Finally, I write all my encoded information derived from original text to the compression file. In my opinion, this is the easiest method for me to restore a tree from the head information in the decompression program because I just need to reuse the same tree-built function in the compression program to build a tree in the decompression program. For the actual encoding part, I use the dequeue and enqueue operation to build a Huffman tree. What's more, I make the unique binary key for each character appeared in the text by going through the whole Huffman tree under the" left 0 right 1 " rule, and I build a 2Darray to store them. To encode every character in the text, every time I

read a character from the text, I would write its corresponding binary key to the compression file by an 8 slots ' buffer.

Decode

I first read the head information by using the number of characters appeared in the original text without repetition as a stop reading point, which I write at the beginning of the compression file. Then I build the Huffman tree by the head information which is the frequency of all the characters appeared in the original text. Finally, I read encoded information and use them as a reference to traversal the Huffman tree. Each time I reach to the leaf node, I write the character stored in the leaf node and reset my position to the root of the tree. Keep doing the previous steps until all the characters are written. The time to stop decode could be decided by the first number in the compression file.

file name	original size(byte)	compressed size(byte)	compression ratio
text0.txt	3	6	2
text1.txt	7	27	3.9
text2.txt	155	176	1.1
text3.txt	3150000	2312703	0.734
text4.txt	6300000	4624673	0.734

As shown in the table above, the compression ratio is better for big size file. The reason is that the size of my head information is too large.

Reference

LU, Y. (2017). *INTERMEDIATE C PROGRAMMING*. S.l.: CRC PRESS.

S. (n.d.). Sparida/ECE-368-Data-Structures-Fall-2015-. Retrieved from https://github.com/sparida/ECE-368-Data-Structures-Fall-2015-/blob/master/ECE368/project2/huff.c