Data Structures C & D
FAST-NU, Lahore, Fall 2016

Homework 6

Compression using Huffman's Algorithm

Due: Tuesday October 18 11:55PM Marked out of 100 points.

In this assignment you will write two programs: compression and decompression, which will follow the algorithms described in class. I shall not be explaining the algorithm in this document, but I'm going to provide other information about the programs, their inputs/outputs and how they should work.

1. **Compression program and the .8b file**

   You program will accept a file of type **.8b.** This is basically a text file. You can open it in notepad, and you'll see that it contains 0's and 1's. Each of these 0's and 1's is a character. You can be sure than the total number of 0's and 1's in this file (let it be called N) is a multiple of 8. Consider the following content; assume that this is the content of a file called **abc.8b**

   0000101100101101000000000000101100001011000000000000000000001011

   In this case N = 64. In a .8b file each group of 8 characters beginning from the start of the file is considered a **symbol.** So this file contains 8 symbols but when you see carefully you will notice only three of these symbols are unique. These unique symbols are: 00001011, 00101101 and 00000000, as shown below in red, blue and orange respectively.

   00001011001011010000000000001011000010110000000000000000000001011

   As you know, a combination of 8 0's and 1's can create 256 unique symbols (00000000 = 0, 00000001 = 1, …, 11111110 = 254, 11111111=255). Each of these combinations can be thought of as representing a number (the decimal equivalent of that combination).

   Your compression program will do the following:

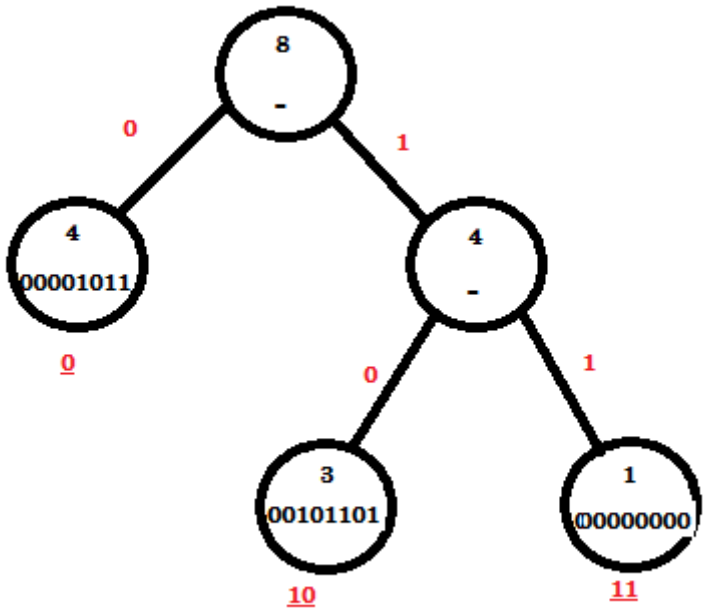
   1. Convert an 8-charater combination into its decimal equivalent number. You can do this by using a function you'll write, let's say, int getNumber(string comb). If the input to the function is 00000000 it returns 0, if the input is 01000000 it

returns 128, if input is 11111111 it returns 255 and so on (simple binary to decimal conversion).

2. Create an array int freq_table[256], and initialize it with zeros.
3. Now go through the file abc.8b and count the frequencies of each 8-character combination. This can be done by reading an 8-character string, let's say, s, and then making the following increment: **freq_table[getNumber(s)]++; This will be done in a loop for each 8-character combination read from the file.**
4. Now, using this freq_table you will apply the algorithm describe in class, and let's say, this algorithm returns the following codes:

Symbol	Code
00000000	11
00101101	10
00001011	0

The algorithm also produces a compression tree, let's call it T. In this case T looks like this (each node contains count and symbol):



Now, the compressed version of abc.8b file is also a text file. You can open and examine its contents in notepad, and it contains 0's and 1's where each 0 and 1 will be a character. For the file abc.8b whose content is given above, the compressed file **abc.cmp** will contain the tree T in the first line, stored in a way decided by you; and this will be followed by the content (use color codes to see what was compressed to what):

010110011110

The total number of 0's and 1's in this file is n = 12

2. **Compression Ratio**

The compression ratio, r, is computed by the formula: $r = \left(1 - \frac{n}{N}\right)$

So for the example above, $r = \left(1 - \frac{12}{64}\right)$ = 0.8125, in other words, 81.25%

You compression program should print this percentage on the screen, as well as save the abc.cmp file in the current directory.

3. **Decompression program**
   i. Read the input file of type .cmp, for example, abc.cmp: read the tree from the first line and reconstruct the tree T
   ii. Using the sequence of 0's and 1's in the file and the tree T, apply the decompression process as discussed in class and recreate the abc.8b file.
   iii. Save this output file in the current directory.

4. **Program interface**

We should be able to run your programs directly from the console by writing lines like the following and pressing enter:

> compress abc.8b

Output on screen: compression completed, p = 81.25%

>decompress abc.cmp

Output on screen: decompression completed

5. **Extra Credit (optional)**

Generalize the programs to work on all text files of type .txt. To do this you'll have to compress the files to the bit level. You can discuss this with me.

That's it!