CS6140: Advanced Programming Lab
Assignment 7: Huffman Coding
An Exercise in Greedy Algorithm Design.

Lead TA: Preethi.

September 25, 2014.

Goal of the assignment

The goal of this assignment is to implement a coding mechanism which will allow us to compress a given text file. We will use the well-known Huffman codes in order to achieve compression. The implementation of the Huffman coding algorithm will also require you to make use of Binary Trees and Priority Queues that you have already implemented. Feel free to reuse parts of your code submitted for earlier assignments as appropriate – alternatively feel free to implement Binary Trees and Priority Queues from scratch if that seems more convenient.

Requirements

Implement a Huffman coding and decoding program which supports the following options. Assume your program is called HuffCode.

- ./HuffCode -c *inputfile* -m *mapfile* -o *outputfile*
  You may assume that all characters in the file *inputfile* are printable however, do not assume any other restrictions on the set of characters or the length of the file. Given such a text file your program should output the following:

  1. A file *mapfile* containing for every character its frequency in the input file and the corresponding Huffman code generated by your algorithm. Sort the output based on frequency of the character. Your file should contain lines of this format:

character	frequency	code
A	45	0
H	9	1101
?	5	1100

  2. A file *outputfile* containing the Huffman code for each character in the input file. For example, if the input file contains "AH?" and the Huffman code for the characters is as shown above, then the *outputfile* contains "011011100".

  Note that if you directly compare the sizes of *inputfile* and *outputfile*, it is very likely that *outputfile* has a size larger than *inputfile*.

- ./HuffCode -d *codefile* -m *mapfile* -o *decodedfile*
  Given a codefile generated by your program (which contains the ASCII code written as stated above) and the mapfile denoting the code you should decode the file and reconstruct the original inputfile. The output generated by your program should be written to file named *decodedfile*.

- **Optional** ./HuffCode -c *inputfile* -m *mapfile* -b *binfile*
  If you are enthusiastic, you can implement a function which will actually write a binary file named *binfile* wherein each character in the original inputfile is replaced with its Huffman code as bits (not printable digits). Note that this file will not be human readable however is likely to be smaller in size than the original input text file, essentially achieving compression. To read the file (that is, to decompress it), you will have to implement another function that would read this bit pattern from *binfile* and interpret it as the appropriate character depending upon the Huffman tree. This should ensure that you have retrieved the original file intact without any data loss. This part of the assignment is only for those interested and will not have any grading associated.

Huffman Coding algorithm

The Huffman Coding algorithm is a well-known greedy algorithm and its description can be found in both CLRS [1] and DPV [2]. Refer any of the above sources for getting a complete description of the Huffman coding algorithm.

Submission Guidelines

Implement all your classes in files having appropriate names and use .h and .cpp files to separate class definitions from functions. Use a makefile to clearly indicate dependencies. Finally, tar all your sources and make a zip file named *roll-number*.tar.gz to upload it to moodle.

References

[1] T. H. Cormen and C. E. Leiserson and R. L. Rivest and C. Stein, *Introduction to Algorithms*, MIT Press, 2009.

[2] Sanjoy Dasgupta and Christos H. Papadimitriou and Umesh V. Vazirani, *Algorithms*, McGraw-Hill, 2008.