CSE4014 – Data Structures & Algorithms
Lab Project

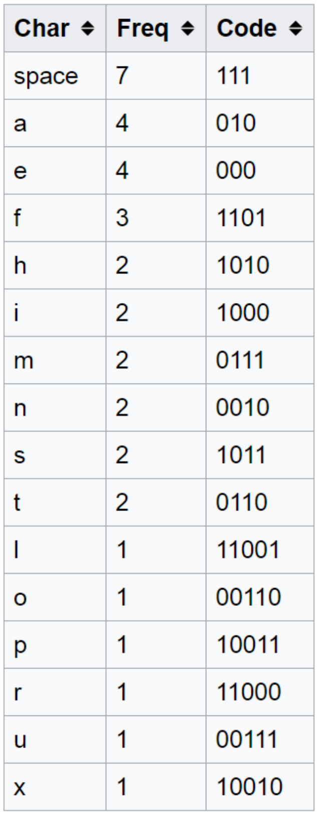
⟨ 2018 - 2019, Spring ⟩

April 16, 2019

Due: April 29 at 23:55

Huffman Coding

In this project, our aim is to implement Huffman coding (see [1]) which is commonly used for lossless compression. Think about a text file consisting of 36 characters. In ASCII, this file will occupy 324 bits, since each character is encoded by 8 bits. The idea behind Huffman coding is encoding the characters with variable length bit sequences, instead of fixed 8-bit ones, such that the more often a character is observed, the shorter the encoder bit sequence becomes. Assume the characters and their frequencies given in Fig. 2 are corresponding to the ones in our 36-character file. They can be encoded as in the last column by applying Huffman's algorithm, and so that one can represent the same file by only using 135 bits.



Char	Freq	Code
space	7	111
a	4	010
e	4	000
f	3	1101
h	2	1010
i	2	1000
m	2	0111
n	2	0010
s	2	1011
t	2	0110
l	1	11001
o	1	00110
p	1	10011
r	1	11000
u	1	00111
x	1	10010

Figure 1: Taken from [1].

Huffman's algorithm encodes by using a binary tree (aka Huffman tree). It stores the characters in the leaf nodes along with their frequencies. Each internal node holds the total frequency of the characters that are stored in the leaf nodes of its own subtree. Therefore, the root node holds the total number of characters. Additionally, the edges (links) between the nodes are weighted, either with 0 or with 1. While the edges linking a parent to its left child are 0, the ones that link a parent to its rights child are 1. For the full explanation of the algorithm, you're referred to the following links:

See "Basic Technique" section in the following wiki article:
`https://en.wikipedia.org/wiki/Huffman_coding`
See this gif:
`https://en.wikipedia.org/wiki/File:Huffman_huff_demo.gif`
See this video:
`https://www.youtube.com/watch?v=dM6us854Jk0`

Here is what you are expected to do:

- Read the input text either from a file or directly from the standard input (it's up to you but it seems better to play with text files.)
- Compute the character frequencies of the input text. Store them in a data structure (you may use the ones that you already know such as array, linked list, etc. or you may learn about priority queues).
- Build a Huffman tree according to the frequencies.

- Once the tree is constructed, traverse it and build a lookup table containing the codes for all characters in the input text.
- Print the uncompressed and compressed versions of the input text, as the streams of bits, to a file or to the standard output.

Consider the following sample for an illustration:

input.txt:

```
Hey, Jude, don’t make it bad
Take a sad song and make it better
Remember to let her into your heart
Then you can start to make it better
```

ascii.txt:

```
01001000 01100101 01111001 00101100 00100000 01001010 01110101 01100100 01100101 00101100 00100000
01100100 01101111 01101110 00100111 01110100 00100000 01101101 01100001 01101011 01100101 00100000
01101001 01110100 00100000 01100010 01100001 01100100 00001101 00001010 01010100 01100001 01101011
01100101 00100000 01100001 00100000 01110011 01100001 01100100 00100000 01110011 01101111 01101110
01100111 00100000 01100001 01101110 01100100 00100000 01101101 01100001 01101011 01100101 00100000
01101001 01110100 00100000 01100010 01100101 01110100 01110100 01100101 01110010 00001101 00001010
01010010 01100101 01101101 01100101 01101101 01100010 01100101 01110010 00100000 01110100 01101111
00100000 01101100 01100101 01110100 00100000 01101000 01100101 01110010 00100000 01101001 01101110
01110100 01101111 00100000 01111001 01101111 01110101 01110010 00100000 01101000 01100101 01100001
01110010 01110100 00001101 00001010 01010100 01101000 01100101 01101110 00100000 01111001 01101111
01110101 00100000 01100011 01100001 01101110 00100000 01110011 01110100 01100001 01110010 01110100
00100000 01110100 01101111 00100000 01101101 01100001 01101011 01100101 00100000 01101001 01110100
00100000 01100010 01100101 01110100 01110100 01100101 01110010
```

huffman.txt:

```
11011011 100 110111 101001 111 0101011 110100 10110 100 101001 111 10110 0100 0001 11011010 011 111
10111 1100 00111 100 111 10101 011 111 01011 1100 10110 00001 110101 010100 1100 00111 100 111 1100
111 00110 1100 10110 111 00110 0100 0001 1101100 111 1100 0001 10110 111 10111 1100 00111 100 111
10101 011 111 01011 100 011 011 100 0010 00001 110101 0101010 100 10111 100 10111 01011 100 0010 111
011 0100 111 1010000 100 011 111 00000 100 0010 111 10101 0001 011 0100 111 110111 0100 110100 0010
111 00000 100 1100 0010 011 00001 110101 010100 00000 100 0001 111 110111 0100 110100 111 1010001
1100 0001 111 00110 011 1100 0010 011 111 011 0100 111 10111 1100 00111 100 111 10101 011 111 01011
100 011 011 100 0010
```

Remarks

1. Note that this project is worth 5%.
2. You are allowed to use/modify the helper source codes that are shared in labs and/or present on internet by giving references to them. You’re free to use whichever data structures your want. Also you’re allowed to use STL.
3. **Write a short report (1-2 paragraphs only)** explaining your algorithm in terms of the data structures you used, run time and space complexity; as well as anything that I may need while testing your program. You can add anything you think is important and feel free about the report format.
4. **You will also do a demo in the classroom**, in order to run your program, show us under which conditions your program is working properly, and answer the questions about your project.

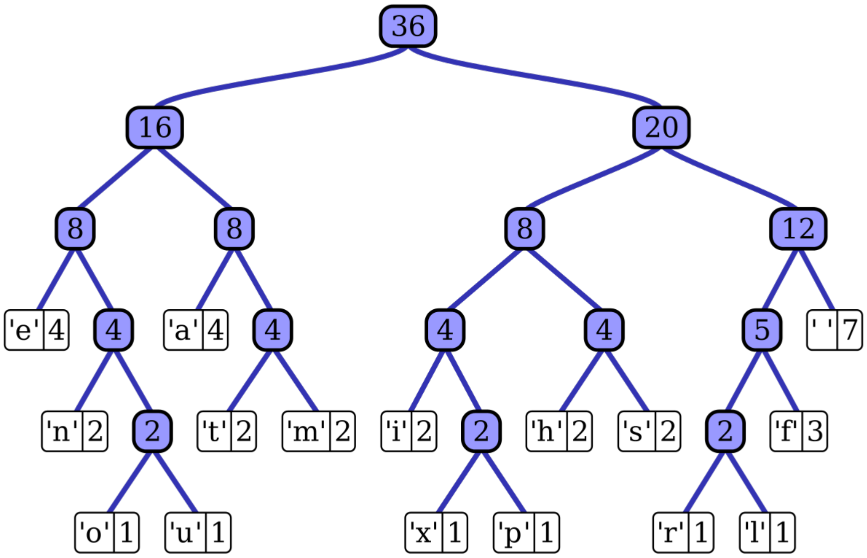


Figure 2: An example Huffman tree [1].

5. **The projects without the report and the demo will not be graded.**
6. Late giving policy is strict. You may not submit after the deadline.
7. Demo sessions will be held on April 30 (Tuesday) during the lab sessions. But since we may need more time, I'll try to arrange one more session on the same day between 11:00-13:00. It will be announced later on.

Submission:

You should submit all source and header files in a folder named as SID_NameSurname (e.g. 0901020003_BjarneStroustrup). The project report should also be included. The folder should be zipped and uploaded from CATS (Assignments section).

References

[1] https://en.wikipedia.org/wiki/Huffman_coding

Good luck!
TA: Ezgi Demircan-Tureyen