University of Nevada, Las Vegas Computer Science 302 Fall 2018

Assignment 6: Huffman Coding. Due November 9, 2018

1. Write a C++ program which uses Huffman's algorithm to create an optimum prefix-free code for the Roman alphabet, together with a symbol for space, using the frequencies of letters used in English. Since I want everyone's code to be the same, you must use the list of frequencies given in the file.
2. Write a C++ program which encodes a file of plaintext into a string of bits, using that Huffman code.
3. Write a C++ program which decodes a string of bits into plaintext.
4. You can combine all three tasks into one program. The program should first create the code, then prompt the user to either encode or decode. In either case, the user enters the name of the file (plaintext or code) to read, and the name of the file (code or plaintext) to write. You could, instead, write the Huffman code into a file, then read that file to either encode or decode. I do not suggest this, since it is better to store the Huffman code as a data structure to make encoding and decoding easier.
5. Use your program to encode text files and to decode files consisting of zeros and ones.

This assignment is more complex than any previous programming assignment. I will describe my implementation, but you may do it differently.

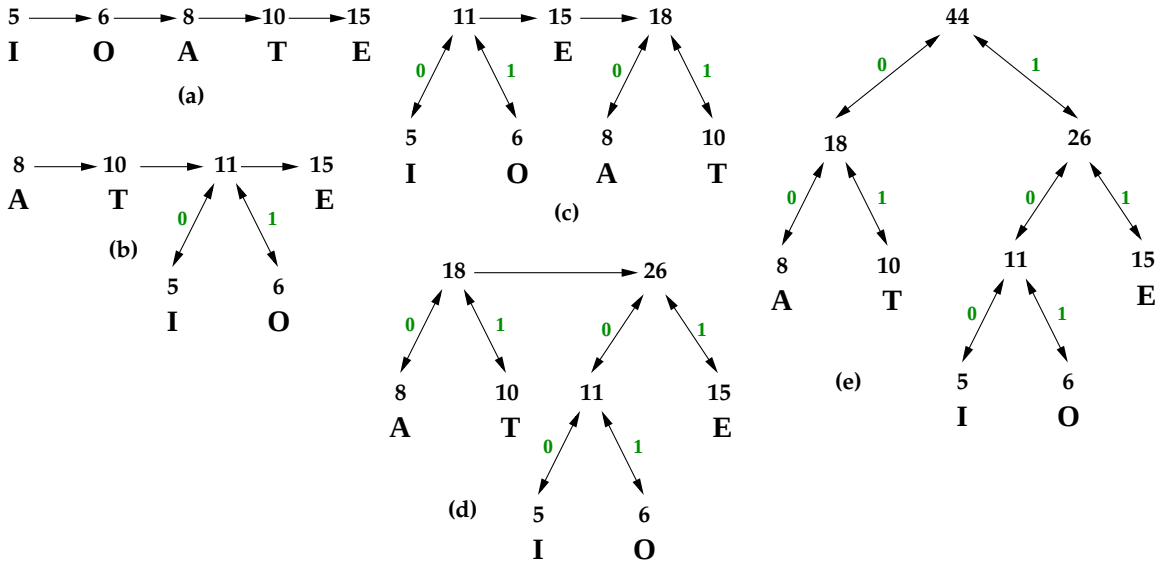
6. If the symbols are sorted by frequency, the Huffman code can be constructed in linear time. On the internet, a linear time algorithm which uses a stack and a queue is described. I don't recommend this algorithm, since it requires a confusing merge step. Instead, I suggest creating a linked structure, as follows.
6. Initially, each node contains one symbol, either a space or a letter, together with the frequency of that symbol. Thus, there are 27 nodes initially.
7. Each node has a pointer to its successor, also a node. The nodes and pointers from a linked list, but that list changes at each step of the algorithm.
8. The first step is to load the data on the file of frequencies into the linked list. We will call this list "treelist." That list will be altered at every step of Huffman's algorithm.
9. The second step is to sort treelist by frequency. I recommend a dynamic form of mergesort, as described on the internet.
10. Eventually, there will be 53 nodes, one for each node of the Huffman tree. The leaves of the tree are the original 27 nodes.
11. Nodes are combined in pairs. At each combination step, two nodes are the first two nodes of treelist are combined and deleted from treelist and become the children of the a node, which is then inserted into treelist. Thus, the length of treelist decreases by 1 at each combination step. The list consists of those nodes which are still roots. Eventually, treelist will have length 1, and its sole node will be the root node of the Huffman tree.

12. Each node has a weight. If it is a leaf, its weight is the frequency of its symbol; otherwise its weight is the sum of the weights of its children. After every step, the nodes of treelist are ordered by weight. The last node created is the root node, whose weight is the sum of all the frequencies.

13. In summary, each node has four pointers, any of which could be NULL, namely its parent pointer, its two child pointers, and a pointer to its successor in treelist. The first three of those are permanent once assigned, but the successor pointer can change several times.

14. If you draw a picture of the Huffman tree with the parent and child pointers shown as arrows, you will note that the path from each leaf to the root is a doubly linked list.

15. Using that path as a bottom-up linked list, we can compute the codon for any symbol. Using that path as a top-down linked list, we can compute the symbol for any codon.



The steps of Huffman's algorithm, given a distribution on an alphabet of size 5. Figure (a) shows **treelist** after sorting. Figures (b) through (e) show the the combine steps. Note that **treelist** shrinks, while the subtrees get larger. In Figure (e), **treelist** has length 1, and the sole subtree in that list is the final Huffman tree.