Huffman Encoding - By Aryan Garg

What is compression?

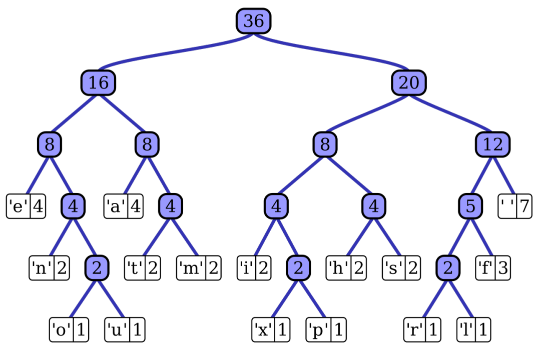
- Data compression tries to encode information in such a way that it takes up less space / bits
- There is Lossless Compression, where all the data is preserved (useful for example when sending zip files)
- Then there is Lossy Compression, where some data is thrown away in an effort to make the file size smaller (ex. In image compression)

Structure of Program

- The program works by first reading in a file and storing all the characters in a Doubly-Linked List and sorting each element as a character "weight" (# of occurrences) is incremented
- The program then starts from the back of the linked list and assembles the two least common nodes into a node and resorts the linked list
  - Carry this on until the linked list is only composed of one node, which contains a tree with all of the characters
- After we get the Huffman Tree, it is just a matter of traversing the tree to decode / encode letters
  - In my code, to speed things up I create a map which links all of the characters to their corresponding Huffman Code

Huffman Encoding

- Huffman Encoding is a form of Lossless compression
- Huffman Encoding works by encoding commonly found characters in less bits (ex. A = 010) and less common letters in more bits (ex. Z = 0111010)
- To find the optimal encoding format, we need to build a huffman tree by arranging all the letters in descending order of how common they appear
  - Then you assemble the tree by going from the least common letters and combining them



Sample Output:

MENU OPTIONS:

1. Print Character Weights
2. Print Tree
3. Print Huffman Code for All Character
4. Print Huffman Code for a Character
5. Print Huffman Code for a Word
6. Decode An Encoded Word
7. Encode File
8. Decode File
9. Exit

```
Option 1
' ' (DEC32): 3
'!' (DEC33): 1
'C' (DEC67): 1
'I' (DEC73): 1
'L' (DEC76): 1
'P' (DEC80): 1
'a' (DEC97): 1
'e' (DEC101): 1
'g' (DEC103): 2
'i' (DEC105): 1
'm' (DEC109): 2
'n' (DEC110): 1
'o' (DEC111): 2
'r' (DEC114): 2
'v' (DEC118): 1
```

```
Option 3
' ' (DEC32): 010
'!' (DEC33): 10110
'C' (DEC67): 00110
'I' (DEC73): 00010
'L' (DEC76): 00011
'P' (DEC80): 00111
'a' (DEC97): 00100
'e' (DEC101): 00001
'g' (DEC103): 100
'i' (DEC105): 00101
'm' (DEC109): 111
'n' (DEC110): 1010
'o' (DEC111): 011
'r' (DEC114): 110
'v' (DEC118): 00000
```

```
Option 5
Enter a word: CProg!
ASCII REPERSENETATION:
0100001101010000011100100110111101100111001
00001
HUFFMAN CODE: 0011000111110011110010110
```

```
Option 6
Enter an encoded word in binary:
0011000111110011110010110
CProg!
```