1 Conceptual questions about your implementation
10 points

1. Analyze the running time of generating the full Huffman tree for standard Huffman coding.

2. Analyze the running time and space usage of your LZW algorithm. Specifically, think about the size of the symbol table. Could you use another data structure to improve it? If not, say why. State any assumptions you make about the inputs and parameters for the data structures.

3. Compare the compression ratios between Huffman and LZW for a variety of strings. Can you make Huffman perform better than LZW on a specific string? If yes, state the string and parameters used and why they worked. If not, state why this is the case.

4. When might it be better to use Huffman coding instead of LZW Compression coding? Why? (Please give a specific example scenario or use case that is not strictly data dependent like in the above question.)

5. The size of the fixed code length, and therefore the size of the symbol table, has implications on the performance of LZW. If you increase the size of the symbol table, what effect will it have on runtime, space usage, and compression ratio? (Feel free to provide any data you test on) State a scenario where you might want a small symbol table, and a scenario where you would use a large symbol table.

Solution

1. $\Theta(n \log n)$

2. Running Time $O(n)$. Space usage $O(2^w)$ using HashMap where $w$ is the length of the codeword. There can be better complexity with using a Trie in some cases.

3. There are tons of answers here depending on the parameters they use. They could do things like modify the fixed code size of lzw to be huge, or have insights into types of strings like compressing abcd on lzw will probably be worse than huffman, etc.

4. • if you know you want to optimize for compression and you know ahead of time the frequencies of messages that will be sent (for example networking protocols), you may be able to achieve better compression.
   • if you want to optimize for space such as on an embedded device, you don’t have space to store a large symbol table.
   • Similar Examples

5. • Runtime stays the same, space usage is larger
   • Compression ratio should theoretically decrease because you will capture more prefixes
You might want a smaller symbol table for reasons of limited usage (i.e. optimize for space on an embedded device).

Want a large symbol table if you don’t care about space and want better compression.

Suggested rubric:
1. 2 points
2. 2 points
3. 2 points
4. 2 points
5. 2 points

## 2 Compression Ratios for best and worst case
12 points

What is the compression ratio achieved by the following algorithms and inputs? For Huffman and LZW, assume that the input is a sequence of 8-bit characters (R = 256).

1. Run-length coding with 8-bit counts for best-case inputs of N bits.
2. Run-length coding with 8-bit counts for worst-case inputs of N bits.
3. Huffman coding for best-case inputs of N characters.
4. Huffman coding for worst-case inputs of N characters.
5. LZW coding for best-case inputs of N characters using 12-bit codewords. Assume no new codewords are added to the table if the table already has \(2^{12} = 4096\) entries.
6. LZW coding for worst-case inputs of N characters using 12-bit codewords. Assume no new codewords are added to the table if the table already has 4096 entries.

For each of the answers above, give a compression ratio and a 1 or 2 sentence explanation of why that is the case.

### Solution
1. \(\frac{8}{255}\). The best case is an alternating sequence of 255 0s and 255 1s. Each sequence of 255 0s (or 255 1s) is encoded with 8 bits (11111111).
2. \(\frac{8}{7}\). The worst case is an alternating sequence of 0s and 1s. Each bit is encoded with 8 bits (00000001).
3. \(\frac{1}{2}\). The best case is when one character occurs 100% of the time (or all but a constant number of times), in which case it is encoded using 1 bit.
4. \(\frac{8}{5}\). The worst case is when each of the 256 characters occurs with equal frequency. In this case, each character is encoded using 8 bits.
5. \(\frac{12}{8 \times 3840}\). The best case is one 8-bit character, say A, repeated N times. The table contains 12-bit codewords for A, AA, AAA, and so on, all the way up to \(2^{12} - 256 = 3840\) A’s when the table gets full. After this point, each sequence of 3840 consecutive A’s is encoded using only 12 bits.
6. \(\frac{12}{5}\). The worst case is when the codeword table gets filled up with useless codewords and then the rest of the message cannot take advantage of any of the added codewords. The table contains 3840 spots...
all needing to be filled strings of repeating A’s. To add a string of 3840 A’s as a key in the table, a string of 3839 A’s must be seen. Inductively the total number of A’s that must be seen is

\[ (3840 + 3839 + ... + 1) + 1 = 1 + \sum_{i=1}^{3840} i = 7374721 \]

An input of 7374721 A’s followed by \( N - 7374721 \) B’s would have this property. In this case, each B requires a 12-bit codeword.

**Suggested rubric:** 2 points for each item; one for ratio, one for explanation.

### 3 Huffman Codes

#### 8 points

Suppose that all of the symbol frequencies are equal. Describe the Huffman code.

**Solution**

If all characters occurred with equal frequency, we would have a balanced trie where all codes were roughly equal in length.

In such a situation there is no data compression, because the number of bits required to encode each unique character is the same as for the original encoding (e.g. 7-bit ASCII encodes 128 characters; if the Huffman trie to encode them is balanced, then each character will require a \( \log_2 128 = 7 \) bit code. This makes intuitive sense, as the situation with equal-frequency characters corresponds to a random sequence, which we know is not compressible.

**Suggested rubric:** 4 points for saying trie is balanced. 4 points for explanation.

### 4 Huffman Trie

#### 10 points

Consider this Huffman Trie for the alphabet A, B, C, D, E generated from the frequency of the characters appearing in a given message:
For each statement below say whether it is:

- True for all messages
- False for all messages
- Depends on the message

1. The frequency of A is strictly less than the frequency of B.
2. The frequency of C is greater than or equal to the frequency of A.
3. The frequency of D is strictly greater than the frequency of A.
4. The frequency of D is greater than or equal to that of A, B, and C combined.
5. The frequency of E is strictly less than that of A, B, and C combined.

Solution

1. Depends on the message. The frequency of A could be less than or equal to the frequency of B.
2. True for all messages. The frequency of C is strictly greater than the frequency of A, because we know that \( f(A) + f(B) \leq f(C) \), since C is on the right-hand side of that subtree. Since \( f(B) \geq 1 \), it must be the case that \( f(A) < f(C) \). Since \( f(A) < f(C) \), it must be the case that \( f(A) \leq f(C) \).
3. True for all messages. We just established that \( f(C) > f(A) \), and D is added to the trie after C.
4. Depends on the message. \( f(D) \leq f(A) + f(B) + f(C) \), since it is on the left-hand side of that subtree. If \( f(A) = f(B) = f(C) = 1 \), then \( f(D) \) could be 2 or 3 and get the same tree. Since
\[ f(D) < f(A) + f(B) + f(C) \] and \[ f(D) = f(A) + f(B) + f(C) \] both yield the same tree, the statement is true sometimes depending on the message.

5. False for all messages. \( f(E) \geq f(A) + f(B) + f(C) \). We could get this resulting tree if \( f(D) = f(E) = f(A) + f(B) + f(C) \), or also if \( f(E) > f(A) + f(B) + f(C) \).

**Suggested rubric:** 2 points each (1 for correct answer, one for explanation).

## 5 LZW Decoding

10 points

Decode each of the following LZW-encoded messages or explain briefly why it is not a valid LZW-encoded message (Recall that codeword 80 is reserved to signify the end of file).

Example: If the encoded message is 41 42 43 44 80, then you would write ‘ABCD’.

1. 42 41 4E 82 41 80
2. 42 41 83 80
3. 41 42 81 82 80
4. 41 42 81 83 80
5. 42 41 4E 44 41 4E 41 80

For reference, below is the hexadecimal-to-ASCII conversion table from the textbook.
Solution
1. B A N A N A
2. Not possible since 83 would not yet be in the table.
3. A B A B B A
4. A B A B A B A (see p. 843 of textbook)
5. Not possible because of second occurrence of 41 4E.
Suggested rubric: 2 points for each correct answer