26 Lecture - CS301

Important Subjective

1. **What is Huffman encoding and how does it work?**
   Huffman encoding is a lossless data compression algorithm that uses variable-length codes to represent characters. It works by assigning shorter codes to more frequently occurring characters and longer codes to less frequently occurring characters.

2. **What is the difference between Huffman coding and Shannon-Fano coding?**
   Huffman coding is a bottom-up approach while Shannon-Fano coding is a top-down approach. Huffman coding is more efficient than Shannon-Fano coding in terms of the length of the code words.

3. **What is the optimal prefix property in Huffman coding?**
   The optimal prefix property states that no codeword is a prefix of another codeword.

4. **How do you construct a Huffman tree?**
   To construct a Huffman tree, you start by creating a leaf node for each character and assigning a weight to each node. Then you repeatedly merge the two nodes with the smallest weights into a new parent node until there is only one node left, which is the root of the Huffman tree.

5. **What is the advantage of using Huffman encoding over other compression algorithms?**
   Huffman encoding is very efficient in terms of compression ratio because it assigns shorter codes to more frequently occurring characters. It also preserves the original data without any loss.

6. **What is the role of a Huffman table in data compression?**
   A Huffman table is a lookup table that maps each character to its corresponding Huffman code. It is used to encode and decode data during compression and decompression.

7. **Can Huffman encoding be used for lossy data compression?**
   No, Huffman encoding is a lossless data compression algorithm and cannot be used for lossy data compression.

8. **How does the size of the input data affect the compression ratio in Huffman encoding?**
   The compression ratio in Huffman encoding depends on the frequency of occurrence of each character in the input data. The larger the input data, the more accurate the frequency count, and the better the compression ratio.

9. **How does the order of the input data affect the Huffman tree construction?**
   The order of the input data does not affect the Huffman tree construction, as the algorithm only looks at the frequency of occurrence of each character.

10. **How do you decode a Huffman-encoded message?**
    To decode a Huffman-encoded message, you start at the root of the Huffman tree and follow the path corresponding to each bit in the encoded message until you reach a leaf node, which represents a character. Repeat this process until you have decoded the entire message.