What is a greedy algorithm, and how does it function?

A greedy algorithm makes the best possible choice at each step to achieve an optimal solution locally. It does not backtrack or reconsider choices, aiming for a top-down approach to solve problems.

List two advantages and one disadvantage of the greedy approach.

Advantages: (1) Easier to describe and implement, (2) Can perform better than other algorithms for some cases. Disadvantage: May not always provide the optimal solution.

Define Huffman Encoding and its primary purpose.

Huffman Encoding is a technique for compressing data without losing any information. It uses variable-length codes for each character, with more frequent characters assigned shorter codes.

Explain why Huffman Encoding is considered a greedy algorithm.

Huffman Encoding uses a greedy approach by always combining the two nodes with the smallest frequencies, thereby achieving optimal compression without needing to reconsider previous steps.

What is a prefix code, and why is it important in Huffman Encoding?

A prefix code is a type of code where no code assigned to a character is a prefix of another code. This rule prevents ambiguity in decoding, ensuring unique decodability.

Describe the major steps in building a Huffman Tree.

Steps include calculating character frequencies, creating nodes, combining nodes with minimum frequencies, assigning codes, and repeating until a single root node (tree) remains.

What role does frequency play in Huffman Encoding?

Frequency determines the length of the code assigned to each character; more frequent characters get shorter codes, and less frequent characters get longer ones.

Explain how Huffman Encoding is lossless.

Huffman Encoding is lossless because it maintains all information by encoding data without omitting any part. Decoding reverses the process accurately using the Huffman tree.

What is the time complexity of Huffman Encoding, and why?

The time complexity of Huffman Encoding is O(nlogn) due to the use of min-heap operations, which involve repeatedly extracting minimum values for merging nodes.

Why is a priority queue used in Huffman Encoding?

A priority queue helps efficiently manage and retrieve nodes with minimum frequencies, which are required at each step to build the Huffman Tree.

What would happen if Huffman Encoding did not follow the prefix rule?

Without the prefix rule, decoding could become ambiguous, as a shorter code could potentially be misinterpreted as the beginning of a longer code.

How are binary codes assigned to each character in the Huffman Tree?

Codes are assigned by traversing the tree: moving left adds a '0', and moving right adds a '1'. Each character receives a unique binary code based on its path.

What is the benefit of using Huffman Encoding in data compression?

Huffman Encoding reduces the total number of bits required to represent data, thereby saving storage space and reducing transmission time for data.

Give an example where Huffman Encoding may be less effective.

Huffman Encoding may be less effective with data that has uniform frequency for all characters, as the variable-length encoding won't significantly reduce the overall size.

What are leaf nodes in the context of a Huffman Tree?

Leaf nodes are the nodes representing individual characters in the Huffman Tree, each assigned a unique binary code based on its frequency.

Why is decoding straightforward with a Huffman Tree?

Decoding is straightforward because the tree structure and prefix codes ensure that each code uniquely maps to a character, enabling unambiguous decoding from root to leaf.

What is the difference between fixed-length encoding and variable-length encoding?

Fixed-length encoding assigns the same number of bits to each character, while variable-length encoding assigns shorter codes to more frequent characters, as in Huffman Encoding.

How do you calculate the total bit usage after Huffman Encoding?

Calculate the bit usage by multiplying each character's frequency by the length of its Huffman code, then summing these products for all characters.

What is the purpose of using a min-heap in the implementation of Huffman Encoding?

A min-heap is used to efficiently access the two nodes with the smallest frequencies at each step, necessary for constructing the Huffman Tree.

How would you explain Huffman Encoding to someone new to data structures?

Huffman Encoding compresses data by building a tree based on character frequencies, assigning shorter codes to more common characters, thus saving space while preserving all original information.