High-level algorithm (encoding)

1. **Count frequencies** of every symbol in the input text.
2. **Create leaf nodes** for every symbol with its frequency.
3. **Insert** all leaf nodes into a **min-heap** (priority queue) keyed by frequency.
4. **While** the heap has more than one node:
    - Pop the two nodes with smallest frequency (call them L and R).
    - Create a new internal node P with freq = L.freq + R.freq, left = L, right = R.
    - Push P back into the heap.
5. The remaining node in the heap is the **root** of the Huffman tree.
6. **Generate codes** by traversing from root to each leaf: append 0 for left and 1 for right. The path to each leaf is the symbol's code.
7. **Encode** the message by replacing each symbol with its code and concatenating bits.

High-level algorithm (decoding)

1. Start at the **root** of the Huffman tree.
2. Read encoded bits left to right:
    - For bit 0 → move to current.left.
    - For bit 1 → move to current.right.
3. If current is a **leaf node**, output its symbol, and reset current to the root.
4. Repeat until all bits are consumed.

Because codes are prefix-free, traversal always reaches a leaf at symbol boundaries.