Huffman codes

Problem: Huffman Coding

Alphabet A[1…n] with n characters
Relative frequencies F[1…n] of each character

Example:

A	a	b	c	d	e	f	g	h	i	j
F	9	2	5	6	12	3	4	7	8	1

- Assign binary codes C[1…n] to each character (either fixed-length or variable-length codes)
- To encode a message, replace each character A[k] by its corresponding binary code C[k]
- To decode a message, replace each binary code C[k] by its corresponding character A[k]

- We'll use variable-length binary codes to obtain shortest encoded message length
- Larger frequency $\Rightarrow$ shorter binary code, and smaller frequency $\Rightarrow$ longer binary code
- To decode efficiently, the codes must satisfy the prefix property: No character's code C[k] is a prefix of any other character's code C[k']

Goal: Map the characters to variable-length binary codes C[1…n] that satisfy the prefix property and that yield the shortest encoded message lengths

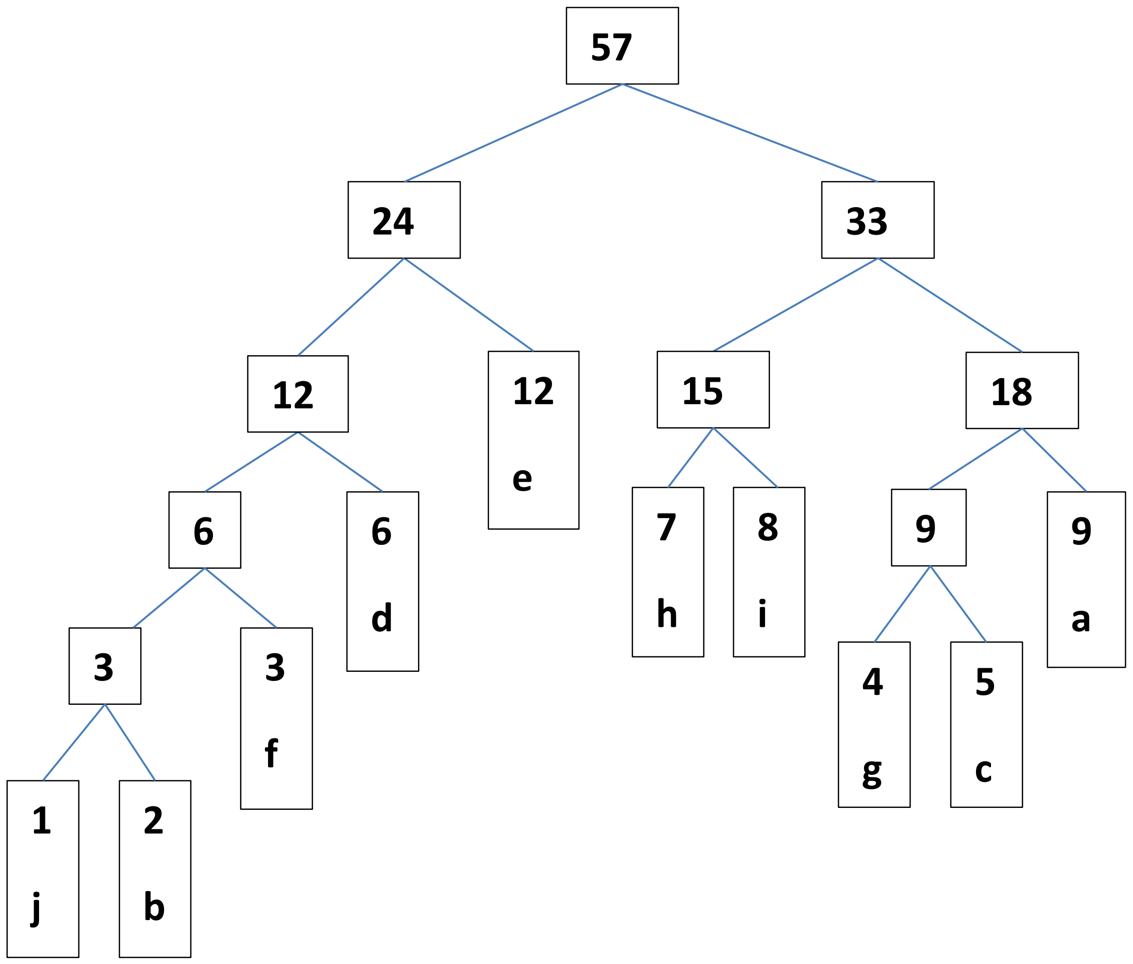
Example:

A	a	b	c	d	e	f	g	h	i	j
F	9	2	5	6	12	3	4	7	8	1
C										

Greedy algorithm for Huffman coding:

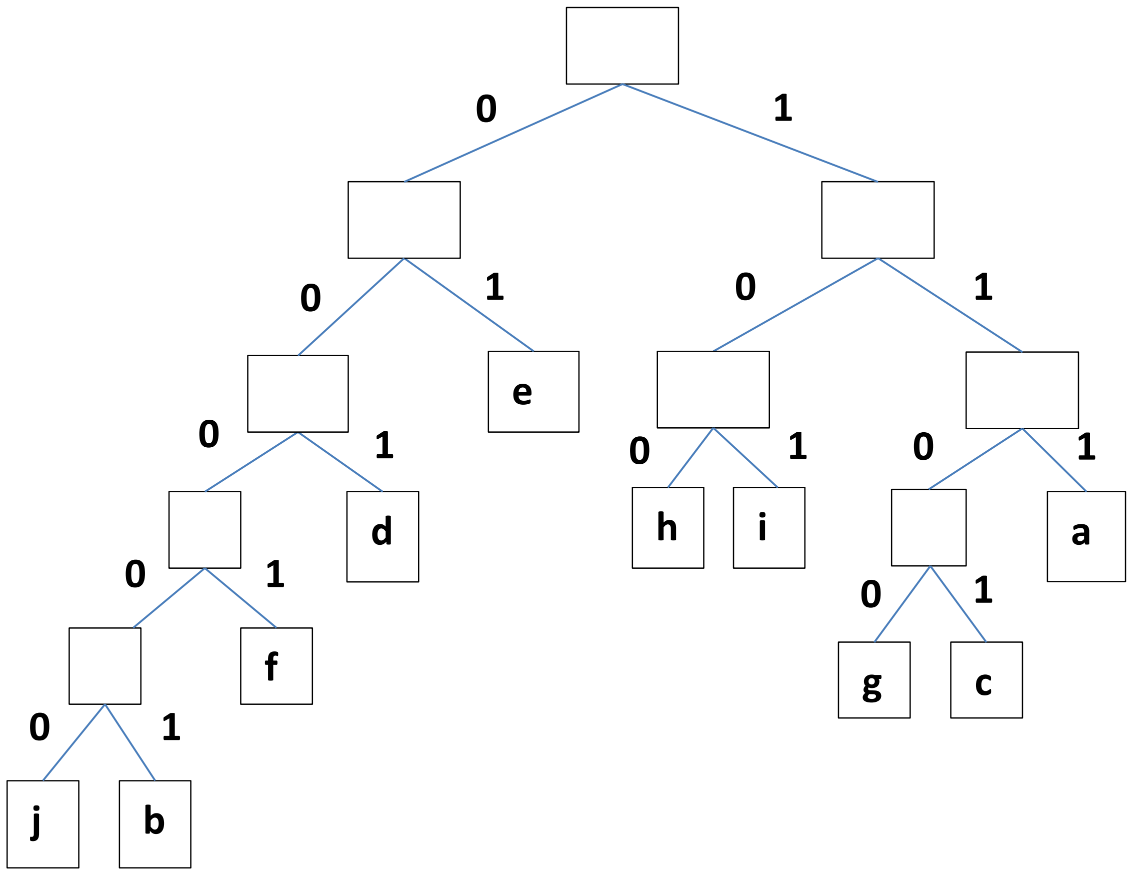
(1) First construct a Huffman tree (binary tree) as follows:

```
H = new Heap( );                // min-ordered heap
for j = 1 to n
    H.insert (new Leaf(F[j], A[j]));          // F[j] is key
while (H.size( ) >= 2) {
    left = H.removeMin( );
    right = H.removeMin( );
    sum = left.key + right.key;
    H.insert (new Node(sum, left, right));  // sum is key
}
root = H.removeMin( );
```



(2) Next assign each character a binary code as follows:

Label each link to left child with 0.
Label each link to right child with 1.
Follow the path from root to each leaf node.
Concatenate the labels to obtain the binary code.



A	a	b	c	d	e	f	g	h	i	j
F	9	2	5	6	12	3	4	7	8	1
C	111	00001	1101	001	01	0001	1100	100	101	00000

Encode a message:

abigchefahead

111 00001 101 1100 1101 100 01 0001 111 100 01 111 001

Decode the message:

1110000110111001101100010001111100011111001

?

Why is prefix property needed to decode the message?

Why does Huffman code always satisfy the prefix property?

Why does Huffman coding yield shorter encoded messages?

- Smaller frequency $\Rightarrow$ added to tree earlier $\Rightarrow$ deeper in tree $\Rightarrow$ longer binary code.
- Larger frequency $\Rightarrow$ added to tree later $\Rightarrow$ shallower in tree $\Rightarrow$ shorter binary code.