Name : Pranav Nirwane

Roll No. : 21CO086

Code:

```
class Node:
    def __init__(self, freq_, symbol_, left_=None, right_=None):
        self.freq = freq_
        self.symbol = symbol_
        self.left = left_
        self.right = right_
        self.huff = ""  # Will store Huffman code
def print_nodes(node, val=""):
    new_val = val + str(node.huff)
    if node.left:
        print_nodes(node.left, new_val)
    if node.right:
        print_nodes(node.right, new_val)
    if not node.left and not node.right:
        print(f"{node.symbol} -> {new_val}")
def huffman_encoding(chars, freq):
    nodes = [Node(freq[x], chars[x]) for x in range(len(chars))]
    while len(nodes) > 1:
        nodes = sorted(nodes, key=lambda x: x.freq)
        left = nodes[0]
        right = nodes[1]
        left.huff = 0
        right.huff = 1
        new_node = Node(left.freq + right.freq, left.symbol + right.symbol, left, right)

        nodes.remove(left)

        nodes.remove(right)

        nodes.append(new_node)

    # The root of the Huffman tree is now the only element in the nodes list

    print("Characters :", f'[{", ".join(chars)}]')

    print("Frequency :", freq)

    print("\nHuffman Encoding:")

    print_nodes(nodes[0])

# Example usage:

chars = ["a", "b", "c", "d", "e", "f"]

freq = [21, 1, 7, 4, 2, 19]

huffman_encoding(chars, freq)
```

Output:

```
Characters : [a, b, c, d, e, f]

Frequency : [21, 1, 7, 4, 2, 19]

Huffman Encoding:

a -> 0

c -> 100

b -> 10100

e -> 10101

d -> 1011

f -> 11
```