

```

1 class Node:
2     """A Huffman Tree Node"""
3
4     def __init__(self, freq_, symbol_, left_=None, right_=None):
5         # frequency of symbol
6         self.freq = freq_
7
8         # symbol name (character)
9         self.symbol = symbol_
10
11        # node left of current node
12        self.left = left_
13
14        # node right of current node
15        self.right = right_
16
17        # tree direction (0/1)
18        self.huff = ""
19
20
21 def print_nodes(node, val=""):
22     """Utility function to print huffman codes for all symbols in the newly created
23     Huffman tree"""
24     # huffman code for current node
25     new_val = val + str(node.huff)
26
27     # if node is not an edge node then traverse inside it
28     if node.left:
29         print_nodes(node.left, new_val)
30     if node.right:
31         print_nodes(node.right, new_val)
32
33     # if node is edge node then display its huffman code
34     if not node.left and not node.right:
35         print(f"{node.symbol} → {new_val}")
36
37 # characters for huffman tree
38 chars = ["a", "b", "c", "d", "e", "f"]
39
40 # frequency of characters
41 freq = [5, 9, 12, 13, 16, 45]
42
43 # list containing huffman tree nodes of characters and frequencies
44 nodes = [Node(freq[x], chars[x]) for x in range(len(chars))]
45
46 while len(nodes) > 1:
47     # sort all the nodes in ascending order based on their frequency
48     nodes = sorted(nodes, key=lambda x: x.freq)
49
50     # pick 2 smallest nodes
51     left = nodes[0]
52     right = nodes[1]
53
54     # assign directional value to these nodes
55     left.huff = 0
56     right.huff = 1

```

```
57
58     # combine the 2 smallest nodes to create new node as their parent
59     newNode = Node(left.freq + right.freq, left.symbol + right.symbol, left, right)
60
61     # remove the 2 nodes and add their parent as new node among others
62     nodes.remove(left)
63     nodes.remove(right)
64     nodes.append(newNode)
65
66
67     print("Characters :", f'[{", ".join(chars)}]')
68     print("Frequency :", freq, "\n\nHuffman Encoding:")
69     print_nodes(nodes[0])
70
71     """
72     OUTPUT:
73
74     Characters : [a, b, c, d, e, f]
75     Frequency  : [5, 9, 12, 13, 16, 45]
76
77     Huffman Encoding:
78     f → 0
79     c → 100
80     d → 101
81     a → 1100
82     b → 1101
83     e → 111
84     """
85
```