```
from collections import Counter


class NodeTree(object):
    def __init__(self, left=None, right=None):
        self.left = left
        self.right = right

    def children(self):
        return self.left, self.right

    def __str__(self):
        return self.left, self.right



def huffman_code_tree(node, binString=''):
    '''
    Function to find Huffman Code
    '''
    if type(node) is str:
        return {node: binString}
    (l, r) = node.children()
    d = dict()
    d.update(huffman_code_tree(l, binString + '0'))
    d.update(huffman_code_tree(r, binString + '1'))
    return d
```

```
def make_tree(nodes):
    '''
    Function to make tree
    :param nodes: Nodes
    :return: Root of the tree
    '''
    while len(nodes) > 1:
        (key1, c1) = nodes[-1]
        (key2, c2) = nodes[-2]
        nodes = nodes[:-2]
        node = NodeTree(key1, key2)
        nodes.append((node, c1 + c2))
        nodes = sorted(nodes, key=lambda x: x[1], reverse=True)
    return nodes[0][0]



if __name__ == '__main__':
    string = 'POORNIMAPATHAK'
    freq = dict(Counter(string))
    print("Given Frequency", freq)
    freq = sorted(freq.items(), key=lambda x: x[1], reverse=True)
    print("Sorted Frequency",freq)
    node = make_tree(freq)
```

```
encoding = huffman_code_tree(node)

for i in encoding:

    print(f'{i} : {encoding[i]}')
```