```
# -*- coding: utf-8 -*-
"""
Created on Sun Aug 29 19:29:37 2021

@author: 15193
"""


# A Huffman Tree Node
class node:
	def __init__(self, freq, symbol, left=None, right=None):
		# frequency of symbol
		self.freq = freq

		# symbol name (character)
		self.symbol = symbol

		# node left of current node
		self.left = left

		# node right of current node
		self.right = right

		# tree direction (0/1)
		self.huff = ''

# utility function to print huffman
# codes for all symbols in the newly
# created Huffman tree


def printNodes(node, val=''):
	# huffman code for current node
	newVal = val + str(node.huff)

	# if node is not an edge node
	# then traverse inside it
	if(node.left):
		printNodes(node.left, newVal)
	if(node.right):
		printNodes(node.right, newVal)

		# if node is edge node then
		# display its huffman code
	if(not node.left and not node.right):
```

```
            print(f"{node.symbol} -> {newVal}")


# characters for huffman tree
chars = ['a', 'b', 'c', 'd', 'e', 'f','g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w',
'x', 'y', 'z']

# frequency of characters
freq = [84,15,22,42,110,22,20,60,74,1,13,40,24,67,74,19,1,75,62,92,27,9,25,1,20,1]

# list containing unused nodes
nodes = []

# converting ccharacters and frequencies
# into huffman tree nodes
for x in range(len(chars)):
    nodes.append(node(freq[x], chars[x]))

while len(nodes) > 1:
    # sort all the nodes in ascending order
    # based on theri frequency
    nodes = sorted(nodes, key=lambda x: x.freq)

    # pick 2 smallest nodes
    left = nodes[0]
    right = nodes[1]

    # assign directional value to these nodes
    left.huff = 0
    right.huff = 1

    # combine the 2 smallest nodes to create
    # new node as their parent
    newNode = node(left.freq+right.freq, left.symbol+right.symbol, left, right)

    # remove the 2 nodes and add their
    # parent as new node among others
    nodes.remove(left)
    nodes.remove(right)
    nodes.append(newNode)

# Huffman Tree is ready!
printNodes(nodes[0])
```