Tutorial for Huffman code generation

Jay Vikrant EE22BTECH11025

```
import heapq
#Jay Vikrant
#EE23210, 21 october 2023

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 = ''
    #arranging the frequency in ascending order
        def __lt__(self, nxt):
                return self.freq < nxt.freq


# function to allot and print code for each
    character in the tree
def printNodes(node, val=''):

        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 give
        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']

# frequency of characters
freq = [1/2,1/4,1/8,1/16,1/32,1/64,1/128,1/128]

# empty nodes
nodes = []
# converting characters and frequencies into
    huffman tree nodes
for x in range(len(chars)):
        heapq.heappush(nodes, node(freq[x],
            chars[x]))

while len(nodes) > 1:

        # sort all the nodes in ascending order
        left = heapq.heappop(nodes)
        right = heapq.heappop(nodes)

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

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

#printing the huffman code
printNodes(nodes[0])
```

1) define all the parameter for the huffman tree i.e **class node:**. Also rearrange the frequency in ascending order
2) Using a function **printNodes(node,val='')** to create the tree print the huffman code.
3) **while len(nodes)¿1** is loop to assign values to the node from the previous step

4) finally printing the code.