DAA - 2

class Node:

"""A Huffman Tree 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 = ""

def print\_nodes(node, val=""):

"""Utility function to print huffman codes for all symbols in the newly created Huffman tree"""

# huffman code for current node

new\_val = val + str(node.huff)

# if node is not an edge node then traverse inside it

if node.left:

print\_nodes(node.left, new\_val)

if node.right:

print\_nodes(node.right, new\_val)

# if node is edge node then display its huffman code

if not node.left and not node.right:

print(f"{node.symbol} -> {new\_val}")

# characters for huffman tree

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

# frequency of characters

freq = [5, 9, 12, 13, 16, 45]

# list containing huffman tree nodes of characters and frequencies

nodes = [Node(freq[x], chars[x]) for x in range(len(chars))]

while len(nodes) > 1:

# sort all the nodes in ascending order based on their 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)

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

print("Frequency :", freq, "\n\nHuffman Encoding:")

print\_nodes(nodes[0])

"""

OUTPUT:

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

Frequency : [5, 9, 12, 13, 16, 45]

Huffman Encoding:

f -> 0

c -> 100

d -> 101

a -> 1100

b -> 1101

e -> 111

"""