**PRACTICAL NO. - 2**

import heapq

from collections import Counter, namedtuple

class Node(namedtuple("Node", ["char", "freq", "left", "right"])):

def \_\_lt\_\_(self, other):

return self.freq < other.freq

def build\_huffman\_tree(text):

freq = Counter(text)

heap = [Node(char, f, None, None) for char, f in freq.items()]

heapq.heapify(heap)

while len(heap) > 1:

left = heapq.heappop(heap)

right = heapq.heappop(heap)

merged = Node(None, left.freq + right.freq, left, right)

heapq.heappush(heap, merged)

return heap[0] if heap else None

def build\_codes(node, prefix='', codebook=None):

if codebook is None:

codebook = {}

if node:

if node.char is not None:

codebook[node.char] = prefix

build\_codes(node.left, prefix + '0', codebook)

build\_codes(node.right, prefix + '1', codebook)

return codebook

def huffman\_encoding(text):

root = build\_huffman\_tree(text)

codes = build\_codes(root)

freq = Counter(text)

print(f"{'Char':<6} | {'Freq':<5} | Huffman Code")

print('-' \* 30)

for char, f in freq.items():

print(f"{repr(char):<6} | {f:<5} | {codes[char]}")

return codes

text = "abbaaaeupqt"

huffman\_encoding(text)  
  
OUTPUT:

(base) kjcoemr@kjcoemr-HP-Pro-SFF-280-G9-Desktop-PC:~/Desktop/BE-15$ python prctical2.py

Char | Freq | Huffman Code

------------------------------

'a' | 4 | 11

'b' | 2 | 01

'e' | 1 | 1011

'u' | 1 | 100

'p' | 1 | 001

'q' | 1 | 000

't' | 1 | 1010