5. Optimal Tree Problem: Huffman Trees and Codes

Code:

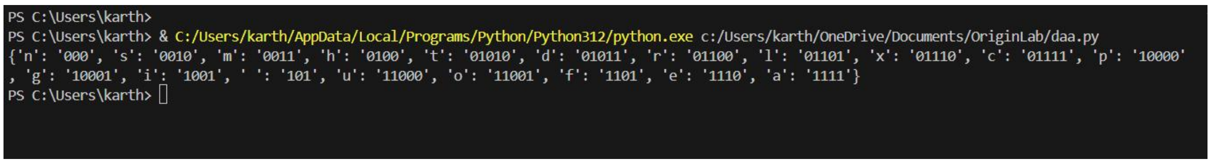
```
import heapq

from collections import defaultdict, Counter

class Node:

    def __init__(self, freq, symbol, left=None, right=None):

        self.freq = freq

        self.symbol = symbol

        self.left = left

        self.right = right

    def __lt__(self, other):

        return self.freq < other.freq

def huffman_coding(frequencies):

    heap = [Node(freq, symbol) for symbol, freq in frequencies.items()]

    heapq.heapify(heap)

    while len(heap) > 1:

        left = heapq.heappop(heap)

        right = heapq.heappop(heap)

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

        heapq.heappush(heap, merged)

    root = heap[0]

    huffman_codes = {}

    def generate_codes(node, current_code=""):

        if node is not None:

            if node.symbol is not None:

                huffman_codes[node.symbol] = current_code

            generate_codes(node.left, current_code + "0")

            generate_codes(node.right, current_code + "1")

    generate_codes(root)

    return huffman_codes

data = "this is an example for huffman encoding"
```

frequencies = Counter(data)

huffman_codes = huffman_coding(frequencies)

print(huffman_codes)

output:



```
PS C:\Users\karth>
PS C:\Users\karth> & C:/Users/karth/AppData/Local/Programs/Python/Python312/python.exe c:/Users/karth/OneDrive/Documents/OriginLab/daa.py
{'n': '000', 's': '0010', 'm': '0011', 'h': '0100', 't': '01010', 'd': '01011', 'r': '01100', 'l': '01101', 'x': '01110', 'c': '01111', 'p': '10000'
, 'g': '10001', 'i': '1001', ' ': '101', 'u': '11000', 'o': '11001', 'f': '1101', 'e': '1110', 'a': '1111'}
PS C:\Users\karth>
```

Time complexity:

F(n)=o(nlogn)