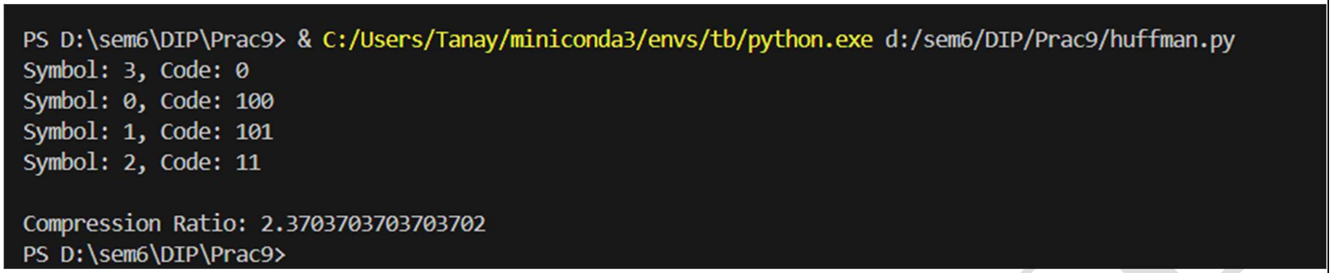
Code :

```
import heapq
from collections import defaultdict, Counter

class Node:
    def __init__(self, freq, symbol=None, 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 build_huffman_tree(frequencies):
    heap = [Node(freq, sym) for sym, freq in frequencies.items()]
    heapq.heapify(heap)

    while len(heap) > 1:
        left = heapq.heappop(heap)
        right = heapq.heappop(heap)
        parent = Node(left.freq + right.freq, left=left, right=right)
        heapq.heappush(heap, parent)

    return heap[0]

def generate_huffman_codes(root):
    codes = {}

    def traverse(node, code):
        if node is None:
            return
        if node.symbol is not None:
            codes[node.symbol] = code
            print(f"Symbol: {node.symbol}, Code: {code}")
        traverse(node.left, code + '0')
        traverse(node.right, code + '1')

    traverse(root, '')
    return codes

def calculate_original_size(data):
    original_bits = len(data) * len(data[0]) * 4  # Assuming each symbol is 4 bits
    return original_bits

def calculate_encoded_size(data, codes):
    encoded_data = ''
    for row in data:
        for symbol in row:
            encoded_data += codes[symbol]
    return len(encoded_data)

def huffman_encode(data):
    flat_data = [symbol for row in data for symbol in row]
    frequencies = Counter(flat_data)
    root = build_huffman_tree(frequencies)
    codes = generate_huffman_codes(root)
    return codes

# Input data
data = [
    [3, 3, 3, 3],
    [2, 3, 3, 3],
    [3, 2, 2, 2],
    [2, 1, 1, 0]
]

# Calculate original size
original_size = calculate_original_size(data)

# Encode the data using Huffman coding and calculate the size of the encoded data
codes = huffman_encode(data)
encoded_size = calculate_encoded_size(data, codes)

# Calculate compression ratio
compression_ratio = original_size / encoded_size

# Display compression ratio
print("\nCompression Ratio:", compression_ratio)
```

Output:



```
PS D:\sem6\DIP\Prac9> & C:/Users/Tanay/miniconda3/envs/tb/python.exe d:/sem6/DIP/Prac9/huffman.py
Symbol: 3, Code: 0
Symbol: 0, Code: 100
Symbol: 1, Code: 101
Symbol: 2, Code: 11

Compression Ratio: 2.3703703703703702
PS D:\sem6\DIP\Prac9>
```