Huffman Encoding

Huffman Encoding is a lossless data compression algorithm that assigns shorter binary codes to more frequent characters and longer codes to less frequent characters.

It is based on the Greedy strategy, meaning it builds the optimal code step by step by always choosing the least frequent nodes first.

Steps in Huffman Encoding:

1. Count frequency of each character in the input.
2. Create a priority queue (min-heap) with each character as a leaf node.
3. Repeat until only one node remains:
    - Remove the two nodes with the smallest frequencies.
    - Create a new internal node with their combined frequency.
    - Insert it back into the heap.
4. The final node is the root of the Huffman Tree.
5. Assign codes:
    - Left edge → 0
    - Right edge → 1

Advantages:

- Reduces the size of data (efficient compression).
- Simple and optimal for known symbol frequencies.
- Used in ZIP, JPEG, and MP3 formats.

Complexity:

- Time Complexity: $O(n\log\ n)$
- Space Complexity: $O(n)$

code

```
import heapq
```

```
# Node class for Huffman Tree
class Node:
    def __init__(self, char, freq):
        self.char = char
        self.freq = freq
        self.left = None
        self.right = None
    def __lt__(self, other):
        return self.freq < other.freq
# Function to build Huffman Tree and generate codes
def huffman_encoding(chars, freqs):
    heap = [Node(chars[i], freqs[i]) for i in range(len(chars))]
    heapq.heapify(heap)
    while len(heap) > 1:
        left = heapq.heappop(heap)
        right = heapq.heappop(heap)
        new_node = Node(None, left.freq + right.freq)
        new_node.left = left
        new_node.right = right
        heapq.heappush(heap, new_node)
    root = heap[0]
    codes = {}
    def generate_codes(node, current_code):
        if node is None:
            return
        if node.char is not None:
            codes[node.char] = current_code
```

```
        generate_codes(node.left, current_code + "0")
        generate_codes(node.right, current_code + "1")
    generate_codes(root, "")
    return codes
# ----- MAIN PROGRAM -----
chars = []
freqs = []
n = int(input("Enter number of characters: "))
for i in range(n):
    ch = input(f"Enter character {i+1}: ")
    freq = int(input(f"Enter frequency of {ch}: "))
    chars.append(ch)
    freqs.append(freq)
codes = huffman_encoding(chars, freqs)
print("\nHuffman Codes:")
for c in codes:
    print(f"{c}: {codes[c]}")
```