**Code**

from heapq import heappush, heappop, heapify

class HuffmanNode:

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

def build\_huffman\_tree(data):

heap = [HuffmanNode(char, freq) for char, freq in data.items()]

heapify(heap)

while len(heap) > 1:

left\_child = heappop(heap)

right\_child = heappop(heap)

internal\_node = HuffmanNode(None, left\_child.freq + right\_child.freq)

internal\_node.left = left\_child

internal\_node.right = right\_child

heappush(heap, internal\_node)

return heap[0]

def huffman\_encoding(data):

frequency = {char: freq for char, freq in data.items()}

root = build\_huffman\_tree(frequency)

code\_map = {}

encode\_huffman\_tree(root, "", code\_map)

encoded\_data = "".join(code\_map[char] for char in data)

return encoded\_data, code\_map

def encode\_huffman\_tree(node, code, code\_map):

if node:

if node.char:

code\_map[node.char] = code

encode\_huffman\_tree(node.left, code + "0", code\_map)

encode\_huffman\_tree(node.right, code + "1", code\_map)

if \_name\_ == "\_\_main\_\_":

data = {}

num\_chars = int(input("Enter the number of characters: "))

for i in range(num\_chars):

char = input("Enter the character: ")

freq = int(input(f"Enter the frequency of {char}: "))

data[char] = freq

encoded\_data, code\_map = huffman\_encoding(data)

print("Character frequencies:", data)

print("Huffman codes:", code\_map)

print("Encoded data:", encoded\_data)

**Output**

Enter the number of characters: 6

Enter the character: a

Enter the frequency of a: 6

Enter the character: b

Enter the frequency of b: 9

Enter the character: c

Enter the frequency of c: 12

Enter the character: d

Enter the frequency of d: 13

Enter the character: e

Enter the frequency of e: 16

Enter the character: f

Enter the frequency of f: 45

Character frequencies: {'a': 6, 'b': 9, 'c': 12, 'd': 13, 'e': 16, 'f': 45}

Huffman codes: {'f': '0', 'c': '100', 'd': '101', 'a': '1100', 'b': '1101', 'e': '111'}

Encoded data: 110011011001011110