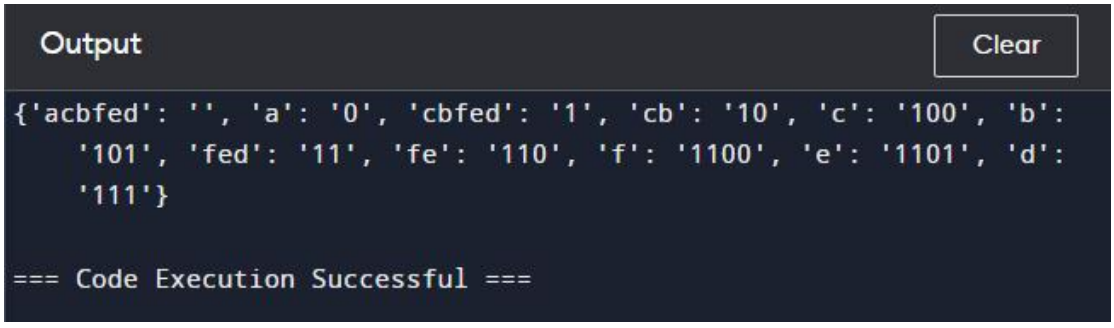
92.Optimal Tree Problem: Huffman Trees

Program:

```
from heapq import heappush, heappop, heapify

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

    def __lt__(self, other):
        return self.freq < other.freq

def build_huffman_tree(freq_dict):
    heap = [Node(freq, symbol) for symbol, freq in freq_dict.items()]
    heapify(heap)

    while len(heap) > 1:
        left = heappop(heap)
        right = heappop(heap)

        merged = Node(left.freq + right.freq, left.symbol + right.symbol)
        merged.left = left
        merged.right = right

        heappush(heap, merged)

    return heap[0]

def huffman_codes(root, current_code="", codes={}):
    if root is None:
        return

    if root.symbol:
        codes[root.symbol] = current_code
    huffman_codes(root.left, current_code + "0", codes)
    huffman_codes(root.right, current_code + "1", codes)
    return codes
freq_dict = {'a': 45, 'b': 13, 'c': 12, 'd': 16, 'e': 9, 'f': 5}
root_node = build_huffman_tree(freq_dict)
huffman_table = huffman_codes(root_node)
print(huffman_table)
```



Time complexity:O(n log n)