201) Given a Huffman Tree and a Huffman encoded string, decode the string to get the original message.

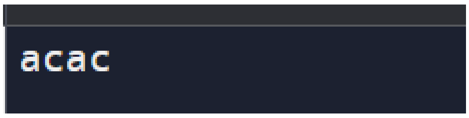
Test Case 1:
Input:
n = 4
characters = ['a', 'b', 'c', 'd']
frequencies = [5, 9, 12, 13]
encoded_string = '1101100111110'
Output: "abacd"
Test Case 2:
Input:
n = 6
characters = ['f', 'e', 'd', 'c', 'b', 'a']
frequencies = [5, 9, 12, 13, 16, 45]
encoded_string = '110011011100101111001011'
Output: "fcbade"

AIM: To write a python program for the Huffman encoded string, decode the string to get the original message.

PROGRAM:

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

def decode_huffman(root, encoded_string):
    decoded_string = ""
    current = root

    for bit in encoded_string:
        if bit == '0':
            current = current.left
        else:
            current = current.right

        if current.left is None and current.right is None:  # It's a leaf node
            decoded_string += current.char
            current = root

    return decoded_string
```

```
root.left = Node('a')
root.right = Node()
root.right.left = Node('b')
root.right.right = Node('c')

encoded_string = "0110111"
print(decode_huffman(root, encoded_string))
```

OUTPUT: 

```
acac
```

TIME COMPLEXITY: O(1)