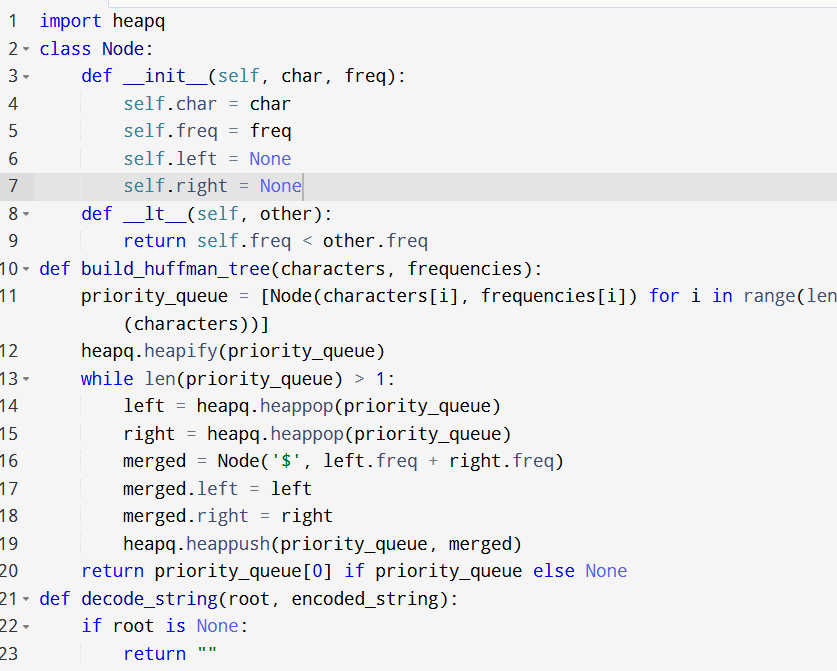
**5.8 Huffman Coding Decoder**

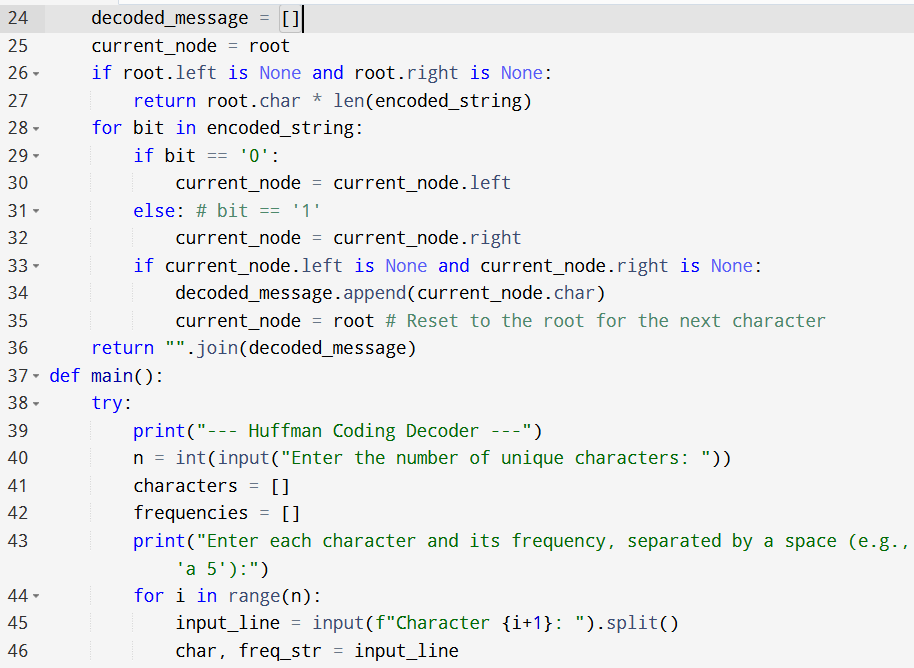
**Aim:** To reconstruct the original message from a Huffman-encoded binary string using the character frequencies.

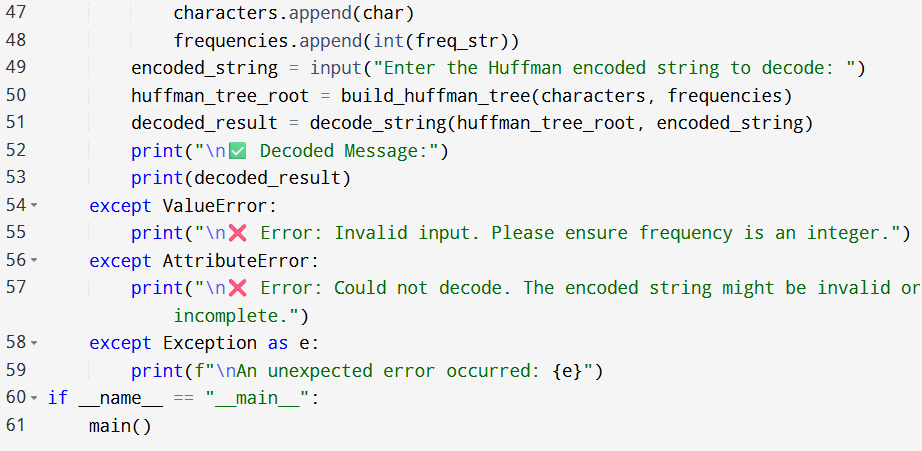
**Algorithm:**

1. We are given the set of characters with their frequencies  
   f = {c₁: f₁, c₂: f₂, …, cₙ: fₙ},  
   and a Huffman-encoded binary string.
2. First, rebuild the Huffman Tree using the same procedure as in encoding:
3. Insert all characters into a min-heap with their frequencies.
4. Iteratively merge the two lowest-frequency nodes until one root remains.
5. Now, start at the root of the Huffman Tree.
6. Scan the encoded binary string bit by bit:
7. If the bit is 0, move to the left child.
8. If the bit is 1, move to the right child.
9. When a leaf node is reached, output the character stored at that node.
10. Return to the root of the tree and continue decoding from the next bit.
11. Repeat until all bits in the encoded string are consumed.
12. The concatenation of all decoded characters gives the original message.

**Program:**

****

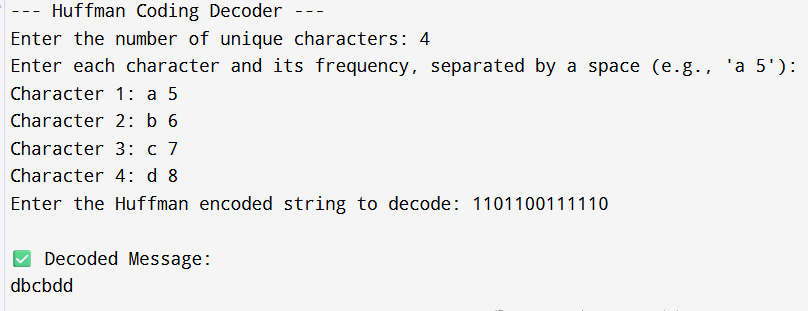




**Input:**

* Enter the number of unique characters: 4
* Enter each character and its frequency, separated by a space (e.g., 'a 5'):

**Output:**

****

**Result:** Thus, the program is executed successfully and output is verified.

**Performance analysis:**

* Time Complexity: O(nlogn+m)
* Space Complexity: O(n+k).