Name: Maurya Patel R
En-no: 22162101014
Batch: 51
Branch: CBA

Institute of Computer Technology
B. Tech Computer Science and Engineering

Sub: Algorithm Analysis and Design

Practical 10

Huffman coding assigns variable length code words to fixed length input characters based on their frequencies. More frequent characters are assigned shorter code words and less frequent characters are assigned longer code words. All edges along the path to a character contain a code digit. If they are on the left side of the tree, they will be a 0 (zero). If on the right, they'll be a 1 (one). Only the leaves will contain a letter and its frequency count. All other nodes will contain a null instead of a character, and the count of the frequency of all of it and its descendant characters.

Construct the Huffman tree for the following data and obtain its Huffman code.

Characters	A	B	C	D	E	-
Frequency/ Probability	0.5	0.35	0.5	0.1	0.4	0.2

(i)   Encode text CAD-BE using the above code.

 Input: CAD-BE

Output: 10011100110111100

(ii)   Decode the text 1100110110 using the above information.

 Input: 0011011100011100

Output: E-DAD

CODE:

.pyfile:

```python
from flask import Flask, render_template, request
import heapq

app = Flask(__name__)

# Step 1: Define the Huffman tree and codes
def build_huffman_tree(frequencies):
    # Create a priority queue (min-heap)
    heap = [[weight, [char, ""]] for char, weight in frequencies.items()]
    heapq.heapify(heap)

    # Build the tree
    while len(heap) > 1:
        lo = heapq.heappop(heap)
        hi = heapq.heappop(heap)
        for pair in lo[1:]:
            pair[1] = '0' + pair[1]
        for pair in hi[1:]:
            pair[1] = '1' + pair[1]
        heapq.heappush(heap, [lo[0] + hi[0]] + lo[1:] + hi[1:])

    # Extract the huffman codes
    huffman_tree = heap[0]
    huffman_codes = {pair[0]: pair[1] for pair in huffman_tree[1:]}
    return huffman_codes

# Step 2: Huffman code and tree for the given characters and frequencies
frequencies = {'A': 0.5, 'B': 0.35, 'C': 0.5, 'D': 0.1, 'E': 0.4, '-': 0.2}
huffman_codes = build_huffman_tree(frequencies)
```

```python
# Step 3: Define the encoding and decoding functions
def encode_text(text):
    return ''.join(huffman_codes[char] for char in text)


def decode_binary(binary_string):
    # Reverse the huffman codes to decode
    reverse_codes = {v: k for k, v in huffman_codes.items()}
    decoded_text = []
    current_code = ''
    for digit in binary_string:
        current_code += digit
        if current_code in reverse_codes:
            decoded_text.append(reverse_codes[current_code])
            current_code = ''
    return ''.join(decoded_text)


@app.route('/', methods=['GET', 'POST'])
def index():
    result = None
    if request.method == 'POST':
        action = request.form.get('action')
        text_or_binary = request.form.get('input_text_or_binary')

        if action == 'encode':
            result = encode_text(text_or_binary)
        elif action == 'decode':
            result = decode_binary(text_or_binary)

    return render_template('index.html', result=result)


if __name__ == '__main__':
    app.run(debug=True)
```

.html file:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Huffman Coding Encoder/Decoder</title>
    <style>

        * {
```

```css
        margin: 0;
        padding: 0;
        box-sizing: border-box;
    }

    body {
        font-family: 'Arial', sans-serif;
        height: 100vh;
        display: flex;
        justify-content: center;
        align-items: center;
        background-color: #f7f7f7;
        margin: 0;
    }

    .container {
        background-color: #ffffff;
        padding: 40px;
        border-radius: 12px;
        box-shadow: 0 0 15px rgba(0, 0, 0, 0.1);
        width: 80%;
        max-width: 650px;
        text-align: center;
        display: flex;
        flex-direction: column;
        align-items: center;
    }


    h2 {
        font-size: 2.5em;
        color: #333;
        margin-bottom: 30px;
    }


    form {
        width: 100%;
        margin-bottom: 30px;
    }

    label {
        font-size: 1.2em;
        text-align: left;
        color: #555;
```

```css
        margin-bottom: 10px;
        width: 100%;
    }

    input, select {
        width: 100%;
        padding: 12px;
        font-size: 1.2em;
        margin-bottom: 20px;
        border: 1px solid #ddd;
        border-radius: 8px;
        background-color: #f9f9f9;
    }

    button {
        width: 100%;
        padding: 14px;
        font-size: 1.2em;
        background-color: #4CAF50;
        color: #fff;
        border: none;
        border-radius: 8px;
        cursor: pointer;
        transition: background-color 0.3s;
    }

    button:hover {
        background-color: #45a049;
    }

    /* Result Styling */
    .result {
        font-size: 2.5em;
        font-weight: bold;
        color: #333;
        padding: 20px;
        background-color: #f0f0f0;
        border-radius: 8px;
        word-wrap: break-word;
        text-align: center;
        max-width: 100%;
        margin-top: 20px;
    }

    /* Responsive Adjustments */
```

```html
        @media (max-width: 600px) {
            .container {
                padding: 20px;
            }

            h2 {
                font-size: 2em;
                margin-bottom: 20px;
            }

            input, select, button {
                font-size: 1.1em;
            }

            .result {
                font-size: 2em;
                padding: 15px;
            }
        }
    </style>
</head>
<body>
    <div class="container">
        <h2>Huffman Coding Encoder/Decoder</h2>
        <form method="POST">
            <label for="input_text_or_binary">Enter Text or Binary:</label>
            <input type="text" id="input_text_or_binary"
name="input_text_or_binary" required>

            <label for="action">Select Action:</label>
            <select id="action" name="action" required>
                <option value="encode">Encode Text</option>
                <option value="decode">Decode Binary</option>
            </select>

            <button type="submit">Submit</button>
        </form>

        {% if result is not none %}
        <div class="result">
            Result: {{ result }}
        </div>
        {% endif %}
    </div>
</body>
```
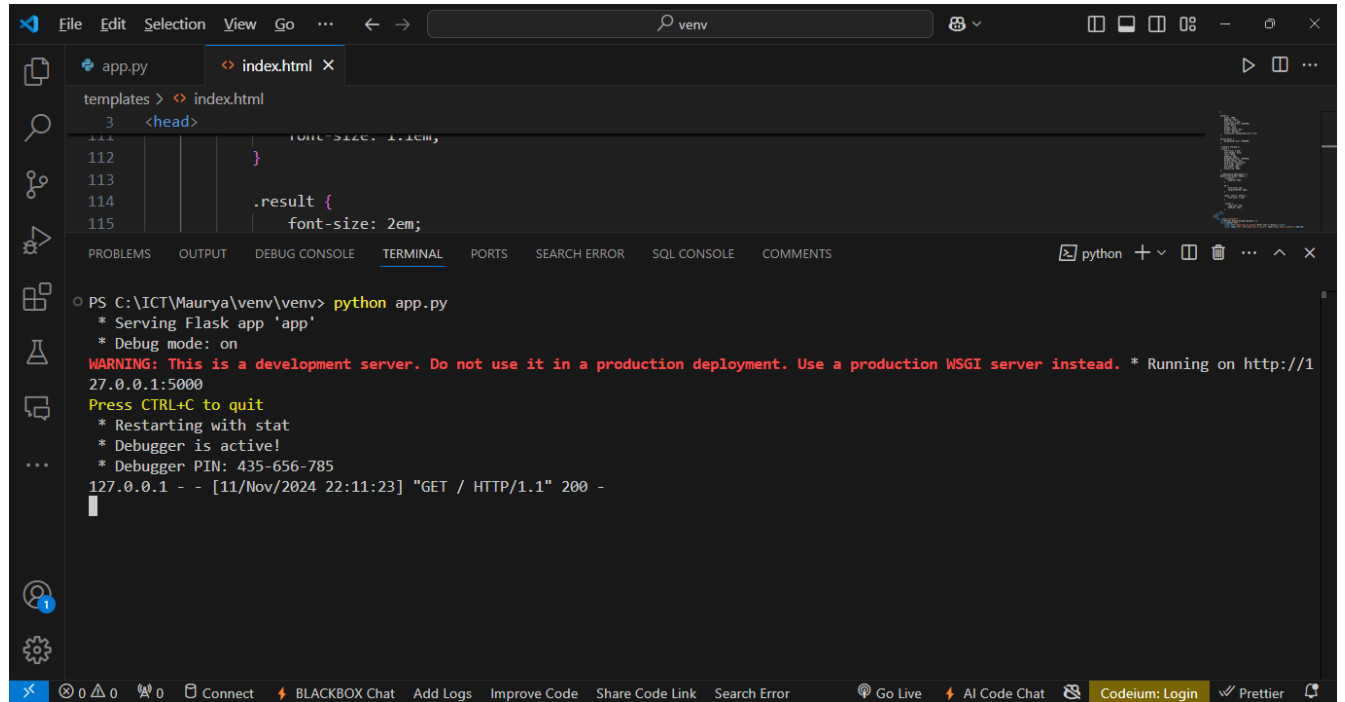
```
</html>
```

SCREENSHOT:

OUTPUT:

Huffman Coding Encoder/Decoder

Enter Text or Binary:

Select Action:

Encode Text

Submit

Result: 10011100110111100

Huffman Coding Encoder/Decoder

Enter Text or Binary:

0011011100011100

Select Action:

Decode Binary

Submit

Result: 10011100110111100

127.0.0.1:5000

Huffman Coding Encoder/Decoder

Enter Text or Binary:

Select Action:

Encode Text

Submit

Result: E-DAD