Name: Devanshi Jain
En-no: 22162101006
Batch: 51

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)

# 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('p10.html', result=result)

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

.html file:

```
<!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>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            background-color: #f0f0f0;
        }
        .container {
            background-color: white;
            padding: 30px;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
            width: 400px;
        }
        h2 {
            text-align: center;
        }
        label {
            display: block;
            margin-bottom: 10px;
            font-weight: bold;
        }
        input, select {
            width: 100%;
            padding: 10px;
            margin-bottom: 20px;
            border: 1px solid #ccc;
            border-radius: 5px;
        }
        button {
            width: 100%;
            padding: 10px;
            background-color: #4CAF50;
            color: white;
            border: none;
            border-radius: 5px;
            cursor: pointer;
        }
        button:hover {
            background-color: #45a049;
        }
        .result {
            margin-top: 20px;
            text-align: center;
            font-weight: bold;
            padding: 10px;
            background-color: #f1f1f1;
            border-radius: 5px;
        }
    </style>
</head>
<body>
    <div class="container">
        <h2>Huffman Coding</h2>
        <form method="POST">
```

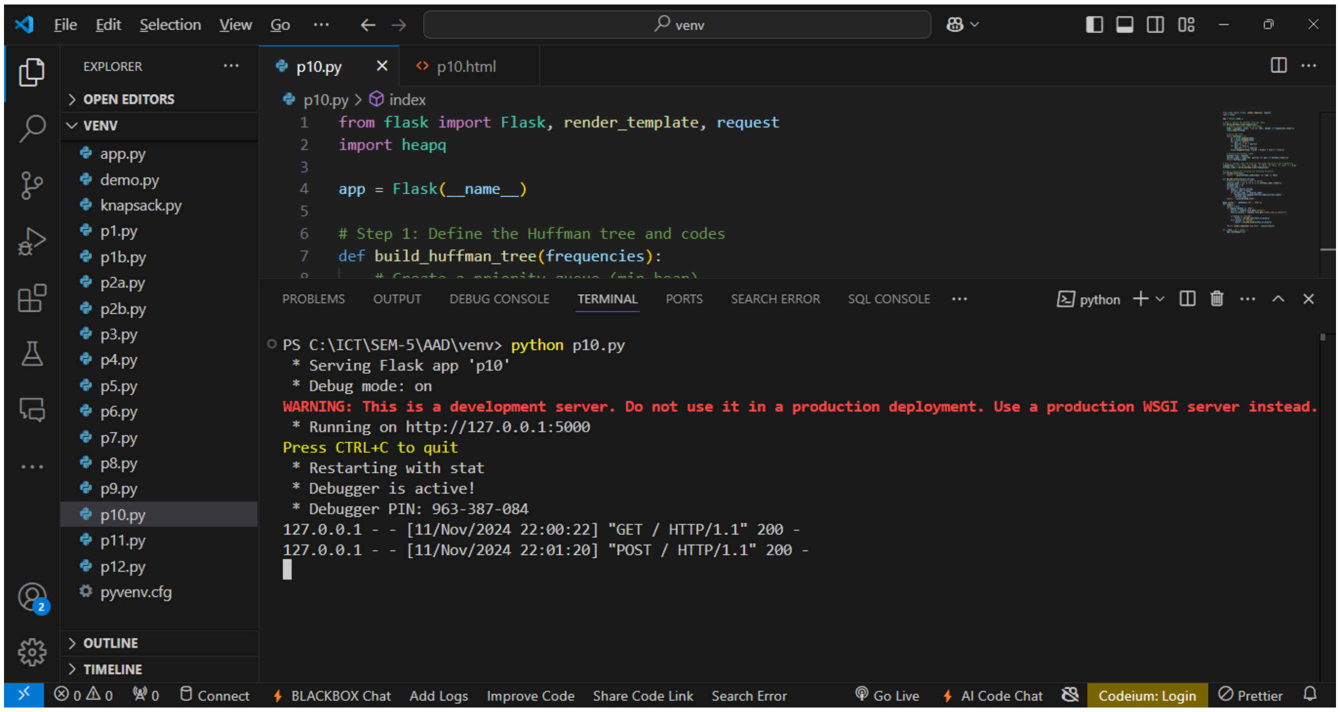
```
            <label for="input_text_or_binary">Enter Text or Binary:</label>
            <input type="text" id="input_text_or_binary"
name="input_text_or_binary" required>

            <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:



File Edit Selection View Go
venv
EXPLORER
OPEN EDITORS
VENV
app.py
demo.py
knapsack.py
p1.py
p1b.py
p2a.py
p2b.py
p3.py
p4.py
p5.py
p6.py
p7.py
p8.py
p9.py
p10.py
p11.py
p12.py
pyvenv.cfg
OUTLINE
TIMELINE
p10.py
p10.html
p10.py > index
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR SQL CONSOLE
python
Connect BLACKBOX Chat Add Logs Improve Code Share Code Link Search Error Go Live AI Code Chat Codeium: Login Prettier

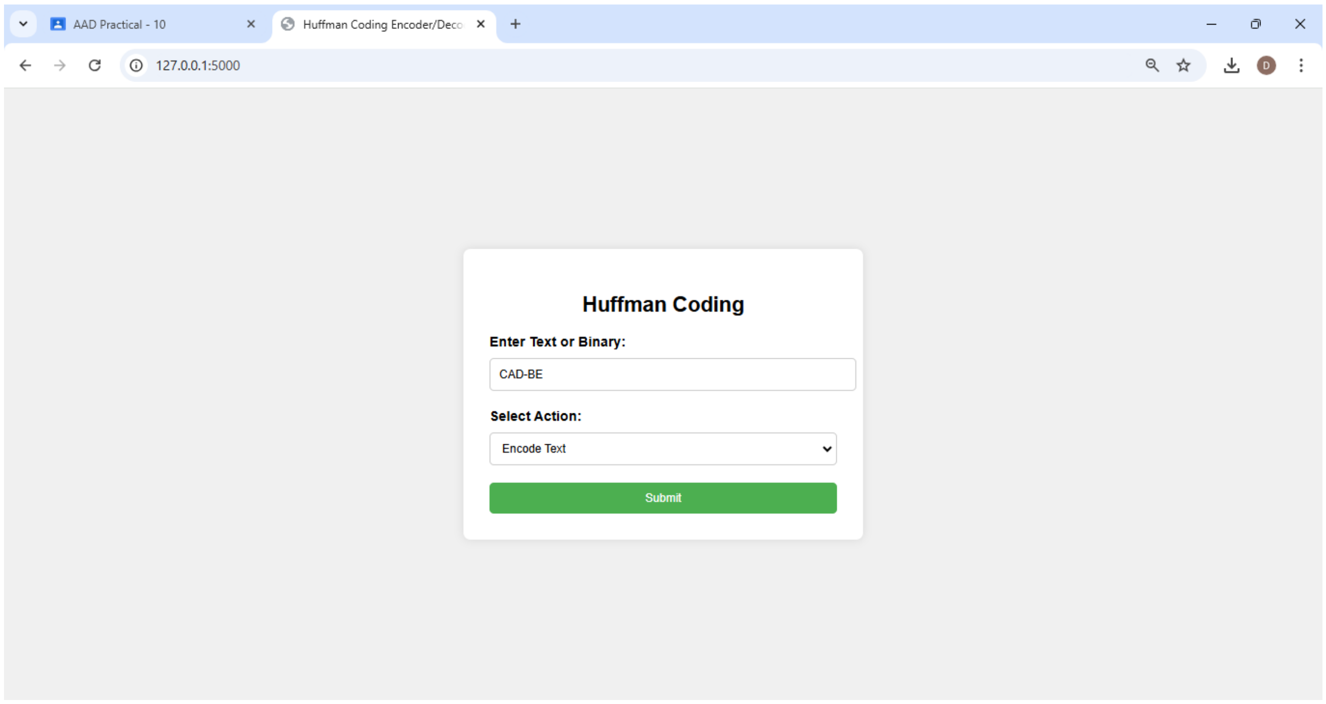
```
from flask import Flask, render_template, request
import heapq

app = Flask(__name__)

# Step 1: Define the Huffman tree and codes
def build_huffman_tree(frequencies):
```

```
PS C:\ICT\SEM-5\AAD\venv> python p10.py
 * Serving Flask app 'p10'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 963-387-084
127.0.0.1 - - [11/Nov/2024 22:00:22] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [11/Nov/2024 22:01:20] "POST / HTTP/1.1" 200 -
```

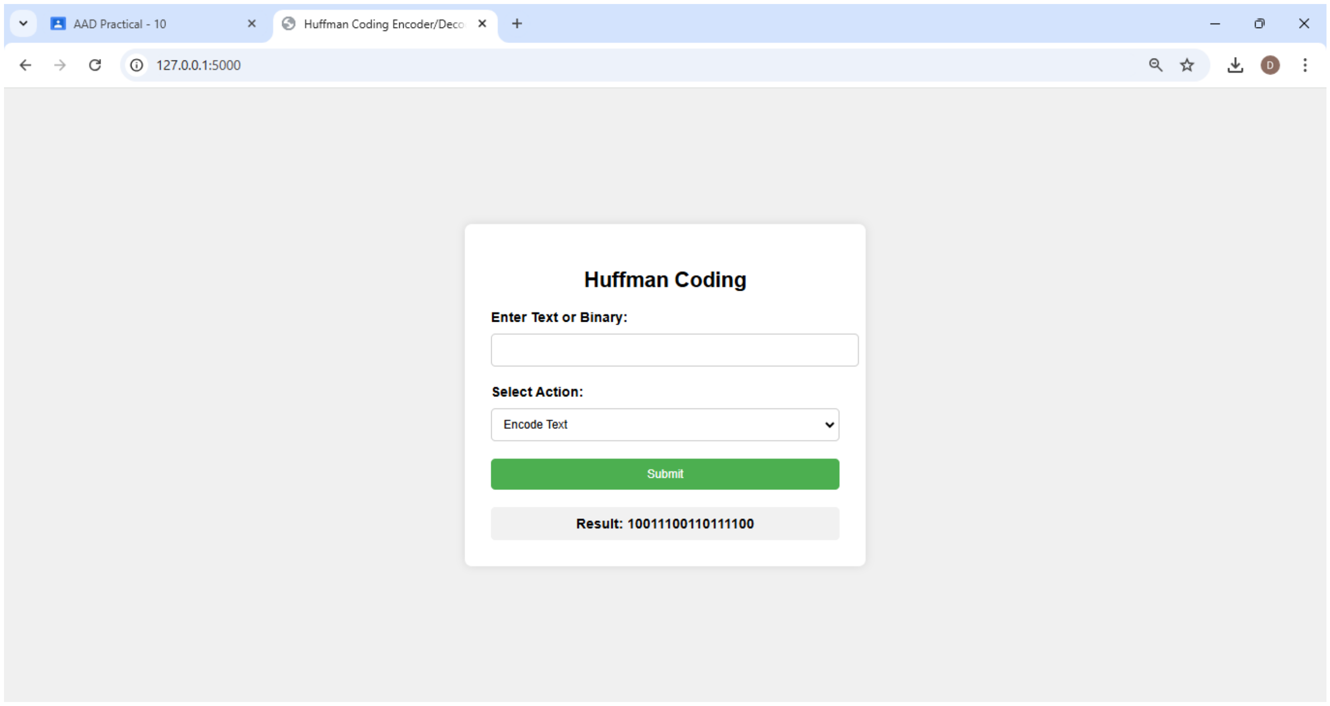
OUTPUT:



Huffman Coding

Enter Text or Binary: CAD-BE

Select Action: Encode Text

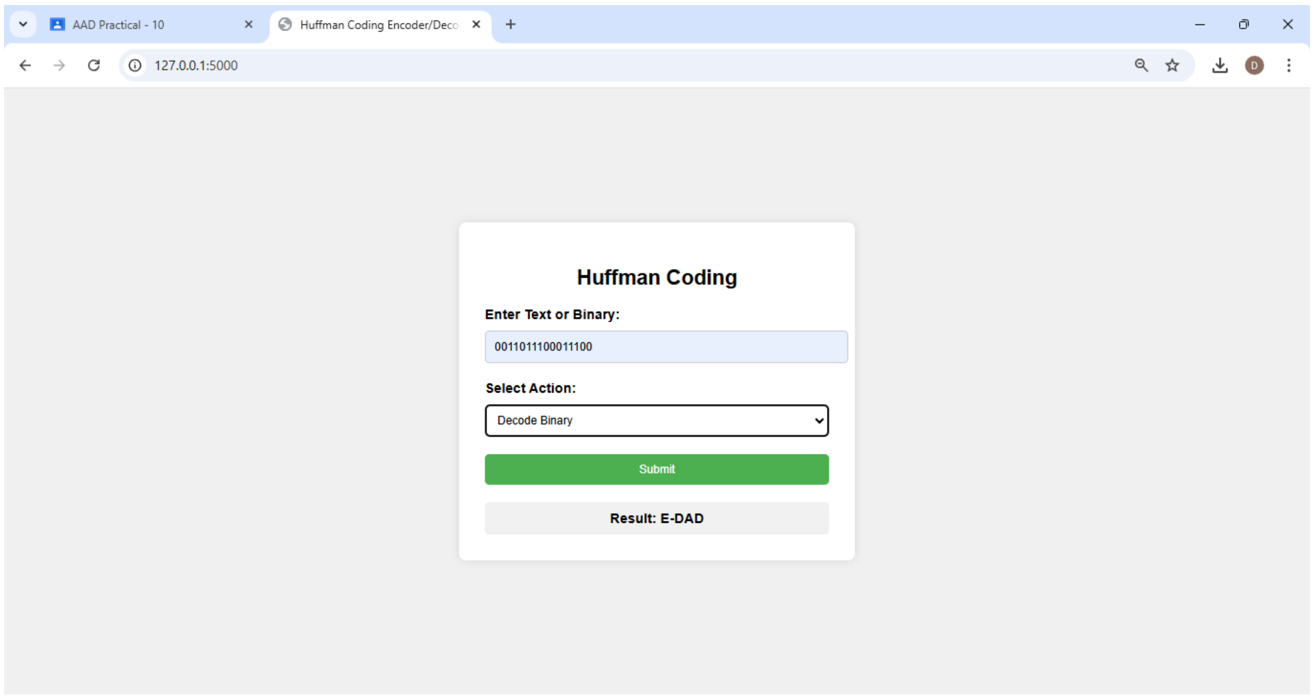


Huffman Coding

Enter Text or Binary:

Select Action: Encode Text

Result: 10011100110111100



Huffman Coding

Enter Text or Binary: 0011011100011100

Select Action: Decode Binary

Submit

Result: E-DAD