

---

```
function HuffmanEncodingm()
clc;
clear;

% Step 1: Define symbols & probabilities
symbols = {'A','B','C','D','E'};
p = [0.30 0.25 0.20 0.15 0.10];

% Step 2: Create initial nodes
nodes = struct([]);

for i = 1:length(symbols)
    nodes(i).symbol = symbols{i};
    nodes(i).prob = p(i);
    nodes(i).left = [];
    nodes(i).right = [];
end

% Step 3: Build Huffman Tree
while length(nodes) > 1

    % Sort nodes by probability (ascending)
    [~, idx] = sort([nodes.prob]);
    nodes = nodes(idx);

    % Take two least probable nodes
    n1 = nodes(1);
    n2 = nodes(2);

    % Create parent node
    parent.symbol = '';
    parent.prob = n1.prob + n2.prob;
    parent.left = n1;
    parent.right = n2;

    % Remove merged nodes and add parent
    nodes = nodes(3:end);
    nodes(end+1) = parent;
end

huffmanTree = nodes;

% Step 4: Generate Huffman Codes
codebook = containers.Map();
generateCode(huffmanTree, '');

% Step 5: Display Huffman Codes
disp('Huffman Codes:'');
```

---

```

keys = codebook.keys;
for i = 1:length(keys)
    fprintf('Symbol %s -> Code %s\n', keys{i}, codebook(keys{i}));
end

% Step 6: Average Code Length

avgLen = 0;
for i = 1:length(symbols)
    avgLen = avgLen + p(i) * length(codebook(symbols{i}));
end

fprintf('\nAverage Code Length = %.3f bits/symbol\n', avgLen);

end % ---- END OF MAIN FUNCTION ----

% Local Function: generateCode

function generateCode(node, code)
% Recursive traversal to assign Huffman codes

% Access codebook from parent function
codebook = evalin('caller','codebook');

% If leaf node, assign code
if isempty(node.left) && isempty(node.right)
    codebook(node.symbol) = code;
    assignin('caller','codebook',codebook);
    return
end

% Traverse left (append 0)
if ~isempty(node.left)
    generateCode(node.left, [code '1']);
end

% Traverse right (append 1)
if ~isempty(node.right)
    generateCode(node.right, [code '0']);
end

end

Huffman Codes:
Symbol A -> Code 00
Symbol B -> Code 10
Symbol C -> Code 11
Symbol D -> Code 010
Symbol E -> Code 011

Average Code Length = 2.250 bits/symbol

```

---

