****

**Division of Electronics and Communication Engineering**

**III IA EVALUATION REPORT**

***for***

**BLENDED LEARNING PROJECT BASED LEARNING**

**Implementing the Huffman Method ALAP for ECG Signal in MATLAB**

***A report submitted by***

|  |  |
| --- | --- |
| ***Name of the Student*** | ***Balan M, Venkatesa Rakshan S*** |
| ***Register Number*** | ***URK22EC1036, URK22EC1002*** |
| ***Subject Name*** | ***Digital Communication*** |
| ***Subject Code*** | ***22EC2016*** |
| ***Date of Report submission*** | ***26/10/24*** |

**Total marks: \_\_\_\_\_/ 40 Marks**

**Signature of Faculty with date:**

****

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER** | **TITLE** | **PAGE NO.** |
|  | **INTRODUCTION** | **1** |
|  | **PROBLEM STATEMENT** | **3** |
|  | **MARKET SURVEY** | **4** |
|  | **MATHEMATICAL MODEL** | **5** |
|  | **HUFFMAN CODING ALAP ALGORITHM** | **7** |
|  | **CODE** | **9** |
|  | **RESULTS AND DISCUSSION** | **12** |
|  | **CONCLUSION** | **14** |
|  | **REFERENCES** | **15** |

In modern healthcare systems, the continuous monitoring of physiological signals like the electrocardiogram (ECG) plays a crucial role in diagnosing and managing heart conditions. ECG signals provide valuable insights into the electrical activity of the heart, helping clinicians detect abnormalities such as arrhythmias, ischemia, and other cardiovascular diseases. However, the constant acquisition of ECG data generates large volumes of information, leading to challenges in storage, processing, and transmission, especially in remote monitoring applications.

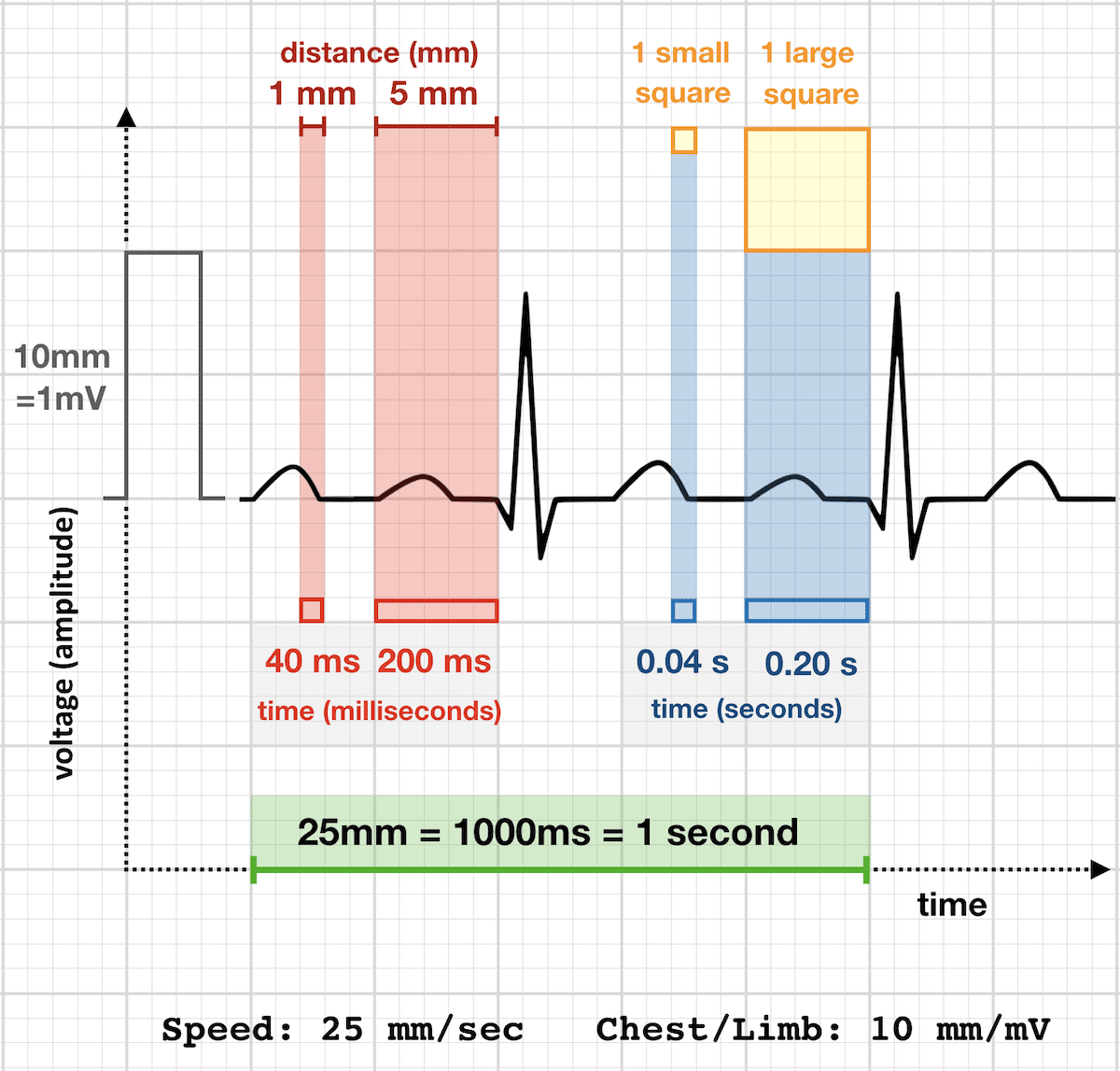
Data compression techniques, particularly lossless methods, have gained significance as a solution to these challenges. Lossless compression ensures that no information is lost during the compression and decompression processes, preserving the accuracy of the medical data, which is critical for clinical diagnoses. Among various lossless compression techniques, Huffman coding stands out due to its simplicity and effectiveness in minimizing data size based on the frequency of occurrence of symbols in the dataset.

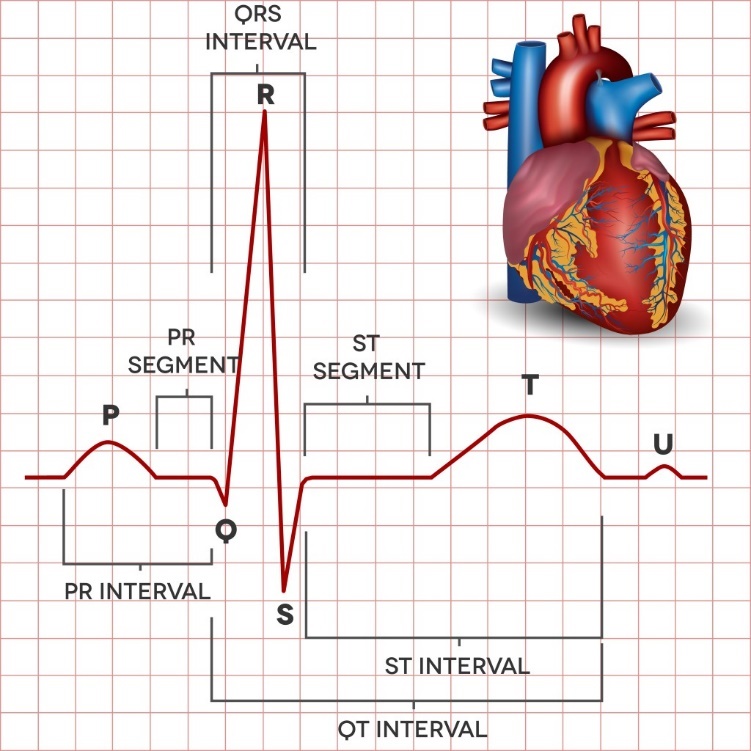
This report focuses on implementing Huffman coding, a well-established algorithm for data compression, to compress ECG signals. By leveraging the "As Late As Possible" (ALAP) scheduling strategy, the goal is to maximize the efficiency of the compression process. The ALAP approach is particularly useful in optimizing the computation scheduling in such a way that the compression algorithm is applied at the final stages of processing, minimizing computational overhead while maintaining high-quality signal reconstruction.

The implementation of Huffman coding for ECG signal compression in MATLAB is explored in this project. MATLAB, a powerful tool for signal processing, allows for efficient manipulation and compression of large datasets, making it an ideal platform for developing and testing the Huffman compression algorithm. The aim of this project is not only to reduce the size of the ECG data but also to ensure that the compressed signal can be fully recovered without any loss, making it suitable for medical applications where signal integrity is paramount.

In the subsequent sections of this report, we will delve into the theoretical background of Huffman coding, the steps involved in implementing the ALAP strategy, and the detailed methodology of applying these techniques to ECG signals. The results of the compression will be evaluated in terms of compression ratio, computational efficiency, and the fidelity of the reconstructed ECG signal, demonstrating the practicality of this approach for real-world applications in healthcare.

**ECG Signal:**





The continuous collection of ECG signals produces large volumes of data, making storage and transmission challenging, especially in real-time health monitoring systems. There is a need for an efficient lossless compression method to reduce data size without losing critical information. This project aims to implement Huffman coding using the "As Late as Possible" (ALAP) strategy in MATLAB to compress ECG signals while ensuring full data recovery and evaluating its performance in terms of compression ratio, computational efficiency, and signal integrity.

**1. Growing Demand for ECG Monitoring Systems**

The global market for electrocardiogram (ECG) monitoring is expanding rapidly, driven by the increasing prevalence of cardiovascular diseases, advancements in wearable health monitoring devices, and the shift towards telemedicine. According to recent reports, the ECG monitoring market is expected to grow significantly in the coming years due to the rising demand for non-invasive and real-time heart monitoring solutions. Wearable devices and remote monitoring systems generate massive amounts of ECG data, creating the need for efficient data compression to handle storage and transmission challenges.

**2. Need for Efficient Compression in Real-Time Monitoring**

One of the primary challenges in ECG monitoring systems is the handling of large, continuous data streams in real time. Medical devices and monitoring systems often collect high-resolution ECG data, which needs to be processed, transmitted, and stored efficiently. Traditional compression methods, such as standard lossy algorithms, are not suitable for medical signals because they can result in data loss, potentially affecting diagnostic accuracy. This has led to the rising importance of lossless compression techniques like Huffman coding, which ensure data integrity while reducing file size.

**3. Adoption of Lossless Compression Methods**

Huffman coding is a well-established lossless compression technique that has been widely adopted in various industries for reducing data size while ensuring that no information is lost during compression. In the healthcare domain, especially for ECG data, maintaining the full accuracy of the signal is essential. The Huffman method offers an effective solution to the problem of ECG data compression, ensuring that the original signal can be fully reconstructed without any loss of critical medical information. With advancements in computing, algorithms like Huffman coding can be combined with scheduling strategies such as the "As Late As Possible" (ALAP) method to optimize computational efficiency.

**4. MATLAB as a Platform for ECG Signal Processing**

MATLAB is a widely used tool in both academic research and the healthcare industry for signal processing tasks, including ECG data analysis and compression. Its extensive libraries and built-in functions for signal manipulation make it an ideal platform for implementing complex algorithms like Huffman coding. The ability to simulate and analyze ECG signals in MATLAB has become crucial for testing new compression techniques, including those aimed at reducing data volume without losing accuracy.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Symbol**  **Si** | **Probability P(Si)** | **Stage 1** | **Stage 2** | **Stage 3**  0 | **Codeword** | **Length L(Si)** |
| S0 | 0.4 | 0.4 | 0.4 | 0.6 | 1 | 1 |
| S1 | 0.2 | 0.2  0 | 0.4  0 | 0.4  1 | 11 | 2 |
| S2 | 0.2  0 | 0.2 | 0.2  1 |  | 100 | 3 |
| S3 | 0.1 | 0.2  1 |  |  | 1110 | 4 |
| S4 | 0.1  1 |  |  |  | 1111 | 4 |

**Average Codeword Length:**

Formula:

Substituting values:

= (0.4×1) +(0.2×2) +(0.2×3) +(0.1×4) +(0.1×4)

= 0.4+0.4+0.6+0.4+0.4

**= 2.2 bits / symbol**

**Entropy:**

Formula:

Substituting values:

**H = 2.1219 bits / Symbol**

**Coding Efficiency:**

Substituting values:

**Variance:**

Substituting values:

**1. Introduction to Huffman Coding**

Huffman coding is a lossless data compression technique that efficiently reduces data size by using variable-length codes for symbols based on their frequencies. In medical applications like ECG signal processing, it helps minimize storage and transmission needs while preserving critical information.

**2. Loading and Preprocessing the ECG Signal**

The process begins by loading the ECG signal data into MATLAB, followed by generating a time axis based on the sampling frequency. This step enables visualization of the ECG waveform, providing a reference for later analysis.

**3. Quantization of the ECG Signal**

Quantization reduces the number of unique voltage levels in the ECG signal by applying a quantization factor. This introduces controlled lossy compression, simplifying the data while retaining essential diagnostic fidelity.

**4. Generating Symbol Frequencies and Building the Huffman Tree**

The algorithm identifies unique quantized symbols and calculates their frequencies, forming a probability distribution. This distribution is used to construct the Huffman tree, where nodes are merged based on their probabilities to optimize symbol encoding.

**5. Encoding the Data Using Huffman Codes**

With the Huffman tree in place, binary codes are generated for each symbol. More frequent symbols receive shorter codes, leading to a compact binary representation of the ECG signal, which is crucial for effective data compression.

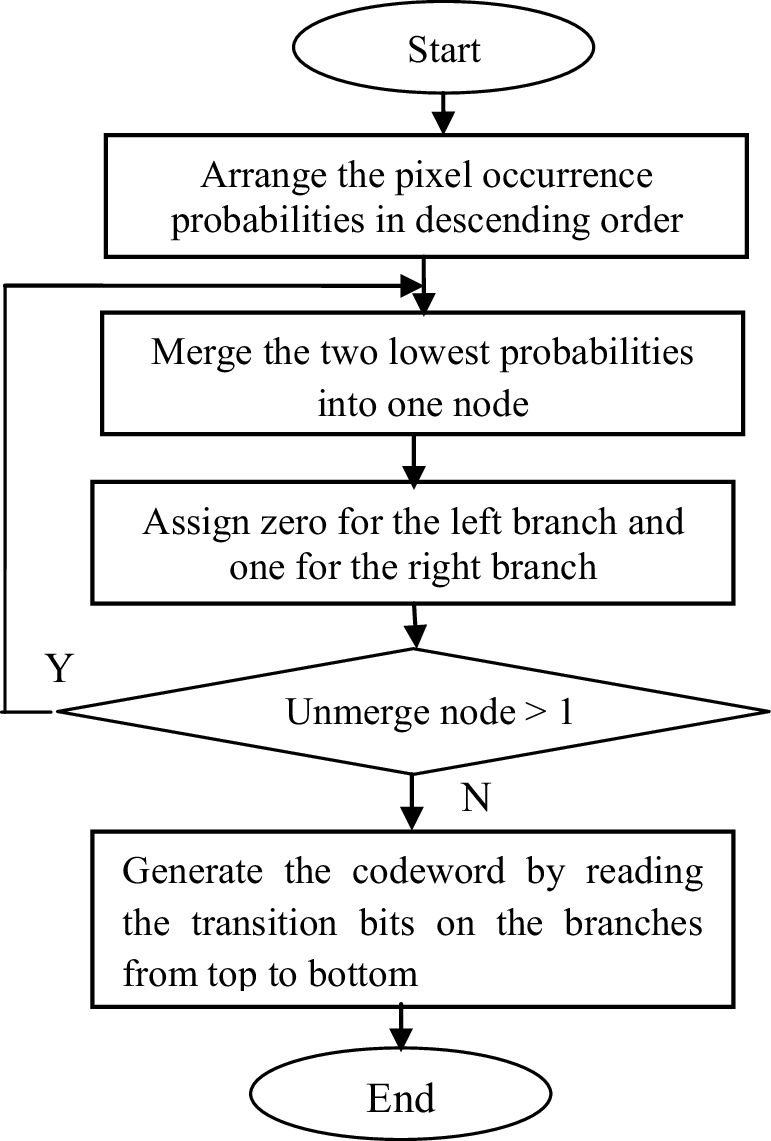
**6. Decoding the Compressed Data and Verification**

A decoding process reconstructs the original quantized data from the compressed representation using Huffman codes. Verification ensures that the decoded data matches the original, confirming the integrity and reliability of the compression.

**7. Conclusion and Visualization of Results**

The final step involves plotting the decoded ECG signal for comparison with the original, allowing for an assessment of the compression effectiveness. This implementation of the Huffman coding algorithm enhances the management and transmission of vital medical data, contributing to improved telemedicine applications.

**Flowchart:**



load('data.mat');

sampling\_frequency = 360;

time\_axis = (0:length(data)-1) / sampling\_frequency;

% Plot the original ECG signal

figure;

subplot(2,1,1);

plot(time\_axis, data);

xlabel('Time (seconds)');

ylabel('Voltage (mV)');

title('Original ECG Signal');

grid on;

% Quantization of the ECG signal (lossy compression)

quantization\_factor = 10; % Change this value for more or less precision

quantized\_data = round(data / quantization\_factor) \* quantization\_factor;

% Unique symbols and their probabilities for quantized data

symbols = unique(quantized\_data);

symbols = reshape(symbols, [], 1); % Ensure symbols is a column vector

bin\_edges = [symbols; max(symbols) + 1];

% Calculate probabilities based on the occurrence of each symbol

probabilities = histcounts(quantized\_data, bin\_edges) / numel(quantized\_data);

% Create nodes for each symbol

nodes = cell(1, numel(symbols));

for i = 1:numel(symbols)

nodes{i} = struct('symbol', symbols(i), 'probability', probabilities(i));

end

% Build the Huffman Tree

while numel(nodes) > 1

% Sort the nodes by probability

[~, sort\_idx] = sort(cellfun(@(x) x.probability, nodes));

% Extract two nodes with the smallest probabilities

node1 = nodes{sort\_idx(1)};

node2 = nodes{sort\_idx(2)};

% Create a new combined node

new\_node = struct('symbol', [], 'probability', node1.probability + node2.probability, 'left', node1, 'right', node2);

% Remove the two nodes and add the new node

nodes = [nodes(sort\_idx(3:end)), new\_node];

end

% Generate Huffman codes using containers.Map to avoid issues with field names

function codes = huffman\_codes(node)

codes = containers.Map('KeyType', 'double', 'ValueType', 'char');

traverse\_tree(node, '', codes);

end

function traverse\_tree(node, code, codes)

% Traverse the Huffman tree recursively to assign binary codes

if isempty(node.symbol)

% If the node is internal, traverse left and right

traverse\_tree(node.left, [code '0'], codes);

traverse\_tree(node.right, [code '1'], codes);

else

% If the node is a leaf, assign the code to the symbol

codes(node.symbol) = code;

end

end

% Generate Huffman codes

codes = huffman\_codes(nodes{1});

% Compress the data using the generated Huffman codes

compressed\_data = arrayfun(@(x) codes(x), quantized\_data, 'UniformOutput', false);

% Sort symbols and probabilities from most probable to least probable

[sorted\_probabilities, sorted\_indices] = sort(probabilities, 'ascend');

sorted\_symbols = symbols(sorted\_indices);

% Display the Huffman codes for each symbol, sorted by probability

for i = 1:numel(sorted\_symbols)

fprintf('Symbol: %d, Probability: %.5f, Code: %s\n', sorted\_symbols(i), sorted\_probabilities(i), codes(sorted\_symbols(i)));

end

% Entropy calculation

entropy = -sum(probabilities(probabilities > 0) .\* log2(probabilities(probabilities > 0)));

% Average code length calculation

average\_code\_length = 0;

for i = 1:numel(symbols)

code\_length = length(codes(symbols(i)));

average\_code\_length = average\_code\_length + probabilities(i) \* code\_length;

end

% Redundancy calculation

redundancy = average\_code\_length - entropy;

% Display results

fprintf('Entropy: %.5f bits\n', entropy);

fprintf('Average Code Length: %.5f bits\n', average\_code\_length);

fprintf('Redundancy: %.5f bits\n', redundancy);

% Save the compressed data as a cell array of binary strings

save('compressed\_data.mat', 'compressed\_data');

% Decoding the compressed data

decode\_map = containers.Map(values(codes), keys(codes)); % Invert the Huffman code map for decoding

% Convert compressed data into a single binary string

encoded\_str = strjoin(compressed\_data, '');

% Initialize variables for decoding

decoded\_data = zeros(1, length(quantized\_data)); % Preallocate array for decoded data

current\_code = '';

decoded\_idx = 1;

% Decode the binary string

for i = 1:length(encoded\_str)

current\_code = [current\_code, encoded\_str(i)]; % Append the current bit

if isKey(decode\_map, current\_code)

decoded\_symbol = decode\_map(current\_code); % Get the corresponding symbol

decoded\_data(decoded\_idx) = decoded\_symbol; % Store in the decoded array

decoded\_idx = decoded\_idx + 1;

current\_code = ''; % Reset the current code to start decoding the next symbol

end

end

% Verify that the decoded data matches the quantized data

if isequal(quantized\_data, decoded\_data)

disp('Decoding successful. The decoded data matches the quantized data.');

else

disp('Decoding failed. The decoded data does not match the quantized data.');

end

% Plot the decoded ECG signal

subplot(2,1,2);

plot(time\_axis, decoded\_data); % Plot the decoded data

xlabel('Time (seconds)');

ylabel('Voltage (mV)');

title('Decoded ECG Signal (Lossy)');

grid on;

% Create a bar graph of sorted symbols and their probabilities

figure;

bar(sorted\_symbols, sorted\_probabilities);

xlabel('Symbols');

ylabel('Probability');

title('Probability Distribution of Symbols');

grid on;

% Calculate original and compressed sizes

original\_size = numel(data) \* 8; % Assuming original data uses 8-bit symbols

compressed\_size = sum(cellfun(@length, compressed\_data)); % Total length of Huffman encoded data

% Data for the bar graph

sizes = [compressed\_size, original\_size - compressed\_size];

% Plot the bar graph

figure;

bar(sizes, 'FaceColor', 'flat');

set(gca, 'XTickLabel', {'Compressed', 'Uncompressed'});

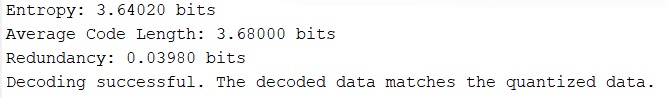
ylabel('Size (Bits)');

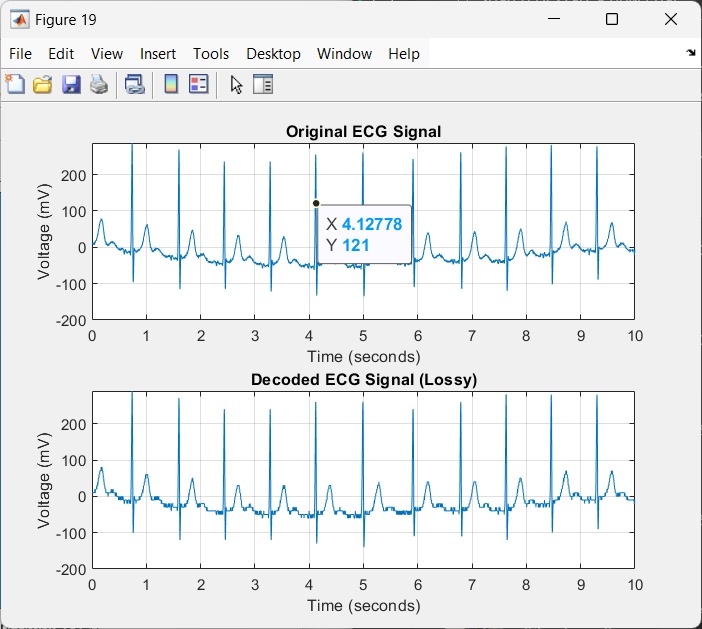
title('Compression Ratio');

grid on;

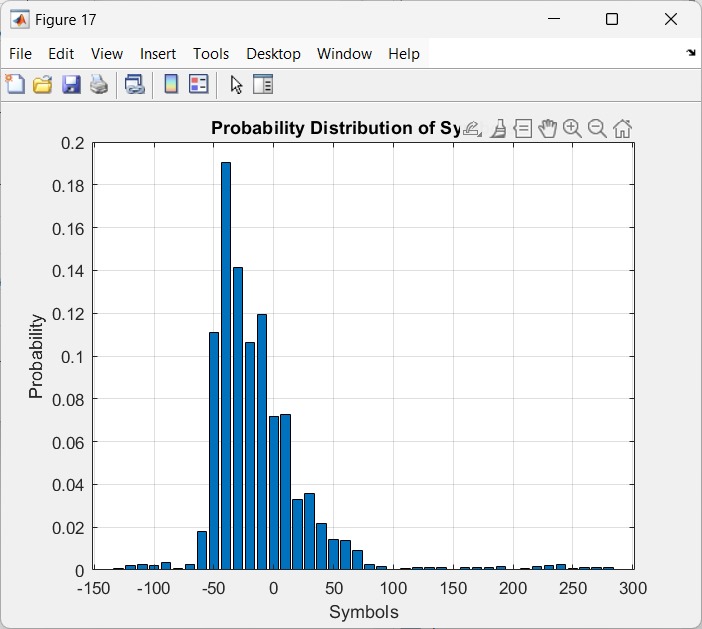
% Add values on top of bars

text(1:length(sizes), sizes, num2str(sizes'), 'vert', 'bottom', 'horiz', 'center');





**INPUT AND DECODED GRAPH**

****

**PROBABILITY OF SYMBOLS**

**Ijettjournal**

**About the data of the ECG Signal used**

Ac Para Tablet is a combination medication that merges two active ingredients: aceclofenac and paracetamol. This formulation is primarily utilized for the effective management of pain and inflammation associated with various medical conditions, particularly those affecting the musculoskeletal system. The dual action of these components makes Ac Para Tablet a versatile option for patients suffering from acute or chronic pain.

Each Ac Para Tablet contains 100 mg of aceclofenac and 325 mg of paracetamol. This combination leverages the anti-inflammatory properties of aceclofenac along with the analgesic effects of paracetamol, providing comprehensive relief from discomfort. Ac Para Tablet is indicated for the relief of pain and swelling related to arthritis, including both osteoarthritis and rheumatoid arthritis, spondylitis, low back pain, gynaecological pain such as menstrual cramps and post-operative discomfort, toothache, headaches including tension-type headaches and migraines, and pain and swelling in the ear, nose, and throat regions.

The therapeutic efficacy of Ac Para Tablet is attributed to its active ingredients. Aceclofenac is a non-steroidal anti-inflammatory drug (NSAID) that works by inhibiting cyclooxygenase (COX) enzymes, leading to a reduction in the synthesis of prostaglandins—chemicals that mediate inflammation and pain. Paracetamol acts primarily in the central nervous system to increase the pain threshold and reduce fever. Together, these ingredients provide a synergistic effect that enhances pain relief while minimizing inflammation.

Ac Para Tablet should be taken according to a healthcare professional's instructions. It is typically recommended to take one tablet every 8 to 12 hours as needed for pain relief. The tablet can be consumed with or without food; however, taking it with food may help reduce gastrointestinal discomfort. It is crucial to swallow the tablet whole with water without chewing or crushing it to ensure proper absorption.

While Ac Para Tablet is generally well-tolerated, some patients may experience side effects such as nausea, vomiting, stomach pain or discomfort, indigestion or heartburn, constipation or diarrhea, and dizziness or headache. Most side effects are mild and transient; however, if any severe reactions occur—such as allergic reactions (rash, itching, swelling), gastrointestinal bleeding (black or bloody stools), or signs of liver damage (yellowing of the skin or eyes)—medical attention should be sought immediately.

.

Ac Para Tablet is an effective combination medication that harnesses the analgesic and anti-inflammatory properties of aceclofenac and paracetamol to provide relief from various types of pain and inflammation. Its versatility makes it suitable for managing conditions such as arthritis, low back pain, and headaches. While generally well-tolerated, it is essential for patients to use Ac Para Tablet under medical supervision to mitigate potential side effects and contraindications. Adhering to prescribed dosages and communicating any adverse reactions to healthcare providers will ensure optimal therapeutic outcomes and enhance patient safety. Overall, Ac Para Tablet serves as a valuable option for individuals seeking effective pain management.

<https://ijettjournal.org/Volume-72/Issue-2/IJETT-V72I2P121.pdf>

<https://www.geeksforgeeks.org/huffman-coding-greedy-algo-3/>

<https://www.nature.com/articles/s41598-024-68022-5>

<https://pmc.ncbi.nlm.nih.gov/articles/PMC7147367/>