clc;

clear;

% === 1. Tạo hình ảnh nhân tạo ===

syntheticImage = uint8(reshape([1 1 1 1 0 0 0 0 1 1 0 0 0 1 1 1 0 0 1 1 1 1 1 0 0], 5, 5)); % Tạo hình ảnh 5x5

disp('Hình ảnh nhân tạo (Ma trận gốc):');

disp(syntheticImage);

% Kích thước ban đầu (tổng số pixel của hình ảnh)

original\_size = numel(syntheticImage);

% Hiển thị hình ảnh gốc kèm giá trị trực tiếp tại từng ô

figure;

imshow(syntheticImage, []);

title('Hình Ảnh Gốc với Giá Trị');

[row, col] = size(syntheticImage);

% Lặp qua từng ô và hiển thị giá trị trên hình ảnh

for r = 1:row

for c = 1:col

text(c, r, sprintf('%d', syntheticImage(r, c)), 'Color', 'red', 'FontSize', 12, ...

'HorizontalAlignment', 'center', 'VerticalAlignment', 'middle', 'FontWeight', 'bold');

end

end

% === 2. Mã hóa RLC ===

imageVector = syntheticImage(:); % Chuyển hình ảnh thành vector

disp('--- Nén bằng RLC ---');

encoded\_RLC = [];

count = 1;

for i = 2:length(imageVector)

if imageVector(i) == imageVector(i-1)

count = count + 1;

else

encoded\_RLC = [encoded\_RLC; imageVector(i-1), count];

count = 1;

end

end

encoded\_RLC = [encoded\_RLC; imageVector(end), count];

% Kích thước sau nén RLC

rlc\_size = numel(encoded\_RLC); % Tổng số phần tử sau nén

rlc\_compression\_ratio = rlc\_size / original\_size;

disp('Dữ liệu sau nén RLC:');

disp(encoded\_RLC);

% === 3. Nén bằng Huffman ===

disp('--- Nén bằng Huffman ---');

% Tạo bảng tần suất

symbols = unique(imageVector); % Các giá trị duy nhất

frequency = histc(imageVector, symbols); % Tần suất xuất hiện

probabilities = frequency / sum(frequency); % Xác suất tương ứng

% Tạo từ điển Huffman

huffmanDict = containers.Map('KeyType', 'char', 'ValueType', 'char'); % Khóa và giá trị đều là chuỗi

binaryCode = 0;

for i = 1:length(symbols)

key = num2str(symbols(i)); % Chuyển khóa sang chuỗi

huffmanDict(key) = dec2bin(binaryCode);

binaryCode = binaryCode + 1; % Tăng mã nhị phân

end

disp('Từ điển Huffman:');

keys = huffmanDict.keys;

for i = 1:length(keys)

fprintf('Giá trị: %s, Mã Huffman: %s\n', keys{i}, huffmanDict(keys{i}));

end

% Mã hóa Huffman

huffman\_encoded = '';

for i = 1:length(imageVector)

key = num2str(imageVector(i)); % Chuyển giá trị sang chuỗi

if isKey(huffmanDict, key) % Kiểm tra khóa tồn tại

huffman\_encoded = [huffman\_encoded huffmanDict(key)];

else

error('Giá trị %d không tồn tại trong từ điển Huffman!', imageVector(i));

end

end

% Kích thước sau nén Huffman (số bit)

huffman\_size = length(huffman\_encoded);

huffman\_compression\_ratio = huffman\_size / (original\_size \* 8); % Mỗi giá trị gốc dùng 8 bit

disp('Mã Huffman:');

disp(huffman\_encoded);

% === 4. Hiển thị kết quả ===

disp('--- Kết quả So sánh ---');

fprintf('Kích thước ban đầu: %d pixel\n', original\_size);

fprintf('RLC:\n - Kích thước sau nén: %d\n - Tỷ lệ nén: %.3f\n', rlc\_size, rlc\_compression\_ratio);

fprintf('Huffman:\n - Kích thước sau nén: %d bit\n - Tỷ lệ nén: %.3f\n', huffman\_size, huffman\_compression\_ratio);

% === 5. Vẽ biểu đồ So sánh ===

figure;

subplot(2, 2, 1);

imshow(syntheticImage, []);

title('Hình Ảnh Gốc với Giá Trị');

for r = 1:row

for c = 1:col

text(c, r, sprintf('%d', syntheticImage(r, c)), 'Color', 'red', 'FontSize', 12, ...

'HorizontalAlignment', 'center', 'VerticalAlignment', 'middle', 'FontWeight', 'bold');

end

end

subplot(2, 2, 2);

bar(1:size(encoded\_RLC, 1), encoded\_RLC(:, 2), 'FaceColor', 'r');

title('Biểu đồ Nén RLC');

xlabel('Nhóm');

ylabel('Số lần lặp');

subplot(2, 2, 3);

bar(symbols, probabilities, 'FaceColor', 'g');

title('Biểu đồ Nén Huffman');

xlabel('Giá trị Pixel');

ylabel('Xác suất Xuất hiện');

subplot(2, 2, 4);

categories = {'Ban đầu', 'RLC', 'Huffman'};

values = [original\_size, rlc\_size, huffman\_size / 8]; % Chuyển Huffman về số byte

bar(values, 'FaceColor', 'blue');

set(gca, 'XTickLabel', categories);

ylabel('Kích thước (Byte)');

title('So sánh Kích thước Dữ liệu');

grid on;