Determine the signal-to-noise ratio (SNR) and the compression ratio of a JPEG coding of a 256x256 image.
Determine the signal-to-noise ratio (SNR) and the compression ratio of a JPEG coding of a 256x256 image.

**Overall Methodology**

The methodology begins by compressing a 256x256 image using various numbers of coefficients to find the compression ratio. Using a lower number of coefficients should result in a lower quality of image and thus a lower signal-to-noise ratio. The compression ratio is proportional to the number of coefficients used to represent the image. The signal to noise ratio is the energy of the original image over the energy of noise.

![Figure 1: Block diagram for the encoder and decoder of a block-transform coding system.](image)

The overall process of compressing and decompressing an image is shown on Figure 1. Input samples are 256x256 images. The quantizer will disassemble each tile of the image to basis images formed by the DCT transform. The inverse quantizer will do nothing because the quantizer only did rounder. Last, the inverse form will be applied to the image to decompress and recover the original image.
The Discrete Cosine Transform
The discrete cosine transform (DCT) is used in most modern image coding standards because of its variety of smooth combinations of blocks to choose from (Figure 2). The basis vectors of the DCT is defined by:

\[
 u_{k,n} = \alpha(k) \cos\left(\frac{(2n+1)k\pi}{2N}\right), \quad n = 0, 1, ..., N - 1 \quad \text{with} \quad \alpha(k) = \begin{cases} 
 \sqrt{\frac{1}{N}} & \text{if } k = 0 \\
 \sqrt{\frac{1}{N}} & \text{if } k = 1, 2, ..., N - 1
\end{cases}
\]

Figure 2: Basis images of an 8 x 8 DCT transform

The Forward and Backward Discrete Cosine Transform
forward transform: \( t = [U]^{-1} s = [V]s \)

\[
 t_k = \sum_{n=0}^{N-1} u_{k;n}^* s_n = \alpha(k) \sum_{n=0}^{N-1} s_n \cos\left(\frac{(2n+1)k\pi}{2N}\right)
\]

The forward discrete transform is applied in MATLAB code to compress the image using only the basis images form by the zigzag order of the DCT coefficients.

inverse transform: \( s = \sum_{k \in \mathcal{N}} t_k u_k = [U]t \)

\[
 s_n = \sum_{n=0}^{N-1} u_{k;n} t_k = \sum_{n=0}^{N-1} \alpha(k) t_k \cos\left(\frac{(2n+1)k\pi}{2N}\right)
\]

The inverse discrete transform is applied in MATLAB code to decompress the image.
The Zigzag Order of the DCT Coefficients
The optimal choice for the deciding which coefficient to use the not the zigzag pattern because
the luminance component for the basis blocks on the right is higher than the blocks on the
bottom (Figure 3). Some images might have different variances on the DCT coefficients because
of the orientation of the image. Therefore, the zigzag pattern (Figure 4) is chosen and manually
coded because it will fit the general images better in a random orientation.

![Quantization Matrix](image1)

**Figure 3:** Perceptual based quantization matrix

![Zigzag Ordering](image2)

**Figure 4:** Zigzag ordering of DCT coefficients

MATLAB Code

```matlab
function ECE434Project03
    clc; clear all; close all; % clear workspace
    % declare global variables so that other functions can reference them
    global base; global M;
    base = 8;
    image = im2double(imread(sprintf('GrayAlpine.jpg'), 'jpeg'));
    M = size(image, 1);
    figure
    imshow(image);
    title('original image');
    imageAfterForwardDCT = imageAfterForwardDCT(image);
    % limit the compression based on the selected coefficients
    selectedCoefficientMatrix = zeros(8,8);
    SNRArray = zeros(base * base, 1);
    % define the pattern to (de)compress each tile with
    zigZag = ...
        [1 1; 1 2; 2 1; % the diagonal is longest in the center of the block
```

3 1; 2 2; 1 3;
1 4; 2 3; 3 2; 4 1;
5 1; 4 2; 3 3; 2 4; 1 5;
1 6; 2 5; 3 4; 4 3; 5 2; 6 1;
7 1; 6 2; 5 3; 4 4; 3 5; 2 6; 1 7;
1 8; 2 7; 3 6; 4 5; 5 4; 6 3; 7 2; 8 1;
8 2; 7 3; 6 4; 5 5; 4 6; 3 7; 2 8;
3 8; 4 7; 5 6; 6 5; 7 4; 8 3;
8 4; 7 5; 6 6; 5 7; 4 8;
5 8; 6 7; 7 6; 8 5;
8 6; 7 7; 6 8;
7 8; 8 7; 8 8];

% go up to every coefficient index
for coefficientIndex = 1 : base * base
    % use the selected coefficient matrix based on the zig zag pattern
    selectedCoefficientMatrix(zigZag(coefficientIndex, 1),
        zigZag(coefficientIndex, 2)) = 1;
    selectedMatrix = repmat(selectedCoefficientMatrix, M / base, M / base);
    % compress the image
    imageAfterCompression = imageAfterForwardDCT .* selectedMatrix;
    % decompress the image
    imageAfterDecompression = imageAfterBackwardDCT(imageAfterCompression);
    if (frac(sym(log2(coefficientIndex))) == 0)
        figure
        imshow(imageAfterDecompression);
        title(sprintf('restored image: # of Coefficients = %d, compressionRatio = %2.3f, SNR = %2.3f dB', coefficientIndex, (base * base - coefficientIndex) / (base * base), 20*log10(calculateSNR(image, imageAfterDecompression))));
    end
end

function T = theForwardDiscreteCosineTransform(S)
    N = size(S, 1);
    u = zeros(N, N);
    n = 0 : N - 1;
    for k = 0 : N - 1
        u(k + 1, n + 1) = sqrt((1 + (k ~= 0)) / N) * cos((2 * n + 1) * k * pi / (2 * N));
    end
    T = u*S*(u');
end

function S = theBackwardDiscreteCosineTransform(T)
    N = size(T, 1);
    u = zeros(N, N);
    n = 0 : N - 1;
    for k = 0 : N - 1
        u(k + 1, n + 1) = sqrt((1 + (k ~= 0)) / N) * cos((2 * n + 1) * k * pi / (2 * N));
    end
    S = (u')*T*u;
end

function imageAfterForwardDCT = imageAfterForwardDCT(image)
global base;    global M;
imageAfterDCT = zeros(M, M);
for i = 0 : M / base - 1
    for j = 0 : M / base - 1
        imageAfterForwardDCT(i * base + [1:base], j * base + [1:base]) =
            theForwardDiscreteCosineTransform(image(i * base + [1:base], j * base + [1:base]));
    end
end

function imageAfterBackwardDCT = imageAfterBackwardDCT(image)
global base;    global M;
imageAfterDCT = zeros(M, M);
for i = 0 : M / base - 1
    for j = 0 : M / base - 1
        imageAfterBackwardDCT(i * base + [1:base], j * base + [1:base]) =
            theBackwardDiscreteCosineTransform(image(i * base + [1:base], j * base + [1:base]));
    end
end

function SNR = calculateSNR(originalImage, noisyImage)
    energyOfOriginalImage = sum(originalImage(:) .^ 2);
    energyOfNoise = sum((originalImage(:) - noisyImage(:)) .^ 2);
    SNR = energyOfOriginalImage / energyOfNoise;

Results (Crowd Image)

Figure 5: Original Image
Figure 6: Restored Image using 1 coefficient
Figure 7: Restored Image using 2 coefficients

Figure 8: Restored Image using 4 coefficients

Figure 9: Restored Image using 8 coefficients

Figure 10: Restored Image using 16 coefficients
### Figure 11: Restored Image using 32 coefficients

### Figure 12: Restored Image using 64 coefficients

<table>
<thead>
<tr>
<th>Number of Coefficients</th>
<th>Compression Ratio</th>
<th>SNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.984</td>
<td>19.091</td>
</tr>
<tr>
<td>2</td>
<td>0.969</td>
<td>21.597</td>
</tr>
<tr>
<td>4</td>
<td>0.938</td>
<td>25.222</td>
</tr>
<tr>
<td>8</td>
<td>0.875</td>
<td>31.097</td>
</tr>
<tr>
<td>16</td>
<td>0.750</td>
<td>37.506</td>
</tr>
<tr>
<td>32</td>
<td>0.500</td>
<td>47.706</td>
</tr>
<tr>
<td>64</td>
<td>0.000</td>
<td>606.944</td>
</tr>
</tbody>
</table>

**Results (Gray Alpine Image)**

### Figure 13: Original Image

### Figure 14: Restored Image using 1 coefficient
Figure 15: Restored Image using 2 coefficients

Figure 16: Restored Image using 4 coefficients

Figure 17: Restored Image using 8 coefficients

Figure 18: Restored Image using 16 coefficients
<table>
<thead>
<tr>
<th>Number of Coefficients</th>
<th>Compression Ratio</th>
<th>SNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.984</td>
<td>29.672</td>
</tr>
<tr>
<td>2</td>
<td>0.969</td>
<td>30.759</td>
</tr>
<tr>
<td>4</td>
<td>0.938</td>
<td>34.202</td>
</tr>
<tr>
<td>8</td>
<td>0.875</td>
<td>36.814</td>
</tr>
<tr>
<td>16</td>
<td>0.750</td>
<td>41.783</td>
</tr>
<tr>
<td>32</td>
<td>0.500</td>
<td>49.346</td>
</tr>
<tr>
<td>64</td>
<td>0.000</td>
<td>606.884</td>
</tr>
</tbody>
</table>

**Conclusion**

The number of coefficients used and the image quality determines the compression ratio and the signal to noise ratio of an image. Images are capable of having very high compression ratios by lowering the signal to noise ratio, which decreases the quality of the image (Figure 6, 14). On the other hand, images could also be compressed and decompressed with a high signal to noise ratio at the cost of compression and memory storage (Figure 12, 20). For most purposes, compressing images using between 16 and 32 coefficients is a good balance between the compression ratio and the signal to noise ratio.