

**Laporan Tugas Kecil 2**  
**IF2211 Strategi Algoritma**  
**Kompresi Gambar dengan Metode Quadtree**



**Disusun Oleh**  
**13523073 - Alfian Hanif Fitria Yustanto**  
**13523095 - Rafif Farras**

**PROGRAM STUDI TEKNIK INFORMATIKA**  
**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**  
**INSTITUT TEKNOLOGI BANDUNG**  
**2024/2025**

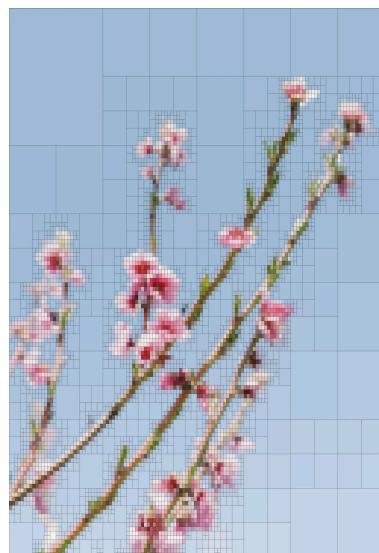
## DAFTAR ISI

<b>DAFTAR ISI.....</b>	<b>1</b>
<b>BAB 1</b>	
<b>DESKRIPSI MASALAH DAN ALGORITMA.....</b>	<b>2</b>
1.1. Quadtree.....	2
1.2. Algoritma Divide and Conquer.....	3
1.3. Penerapan Algoritma Divide and Conquer pada Kompresi Gambar dengan Metode Quadtree	3
<b>BAB 2</b>	
<b>IMPLEMENTASI PROGRAM.....</b>	<b>6</b>
2.1. File Compressor.java.....	6
2.2. File IO.java.....	6
2.3. File Image.java.....	7
2.4. File Pixel.java.....	9
2.5. File Quadtree.java.....	10
2.6. File Main.java.....	12
<b>BAB 3</b>	
<b>SOURCE CODE PROGRAM.....</b>	<b>13</b>
3.1. Repository Github.....	13
3.2. Source Code.....	13
1. Compressor.java.....	13
2. IO.java.....	13
3. Image.java.....	25
4. Pixel.java.....	32
5. Quadtree.java.....	33
6. main.java.....	37
<b>BAB 4</b>	
<b>UJI COBA PROGRAM.....</b>	<b>44</b>
<b>BAB 5</b>	
<b>ANALISIS.....</b>	<b>52</b>
<b>BAB 6</b>	
<b>IMPLEMENTASI BONUS.....</b>	<b>53</b>
6.1 SSIM.....	53
6.2 Target Kompresi.....	53
6.3 GIF.....	53
<b>LAMPIRAN.....</b>	<b>55</b>

# BAB 1

## DESKRIPSI MASALAH DAN ALGORITMA

### 1.1. Quadtree



Gambar 1. Quadtree dalam Kompresi Gambar

(Sumber: <https://medium.com/@tannerwyork/quadtrees-for-image-processing-302536c95c00>)

Quadtree adalah struktur data hierarkis yang digunakan untuk membagi ruang atau data menjadi bagian yang lebih kecil, yang sering digunakan dalam pengolahan gambar. Dalam konteks kompresi gambar, Quadtree membagi gambar menjadi blok-blok kecil berdasarkan keseragaman warna atau intensitas piksel. Prosesnya dimulai dengan membagi gambar menjadi empat bagian, lalu memeriksa apakah setiap bagian memiliki nilai yang seragam berdasarkan analisis sistem warna RGB, yaitu dengan membandingkan komposisi nilai merah (R), hijau (G), dan biru (B) pada piksel-piksel di dalamnya. Jika bagian tersebut tidak seragam, maka bagian tersebut akan terus dibagi hingga mencapai tingkat keseragaman tertentu atau ukuran minimum yang ditentukan.

Dalam implementasi teknis, sebuah Quadtree direpresentasikan sebagai simpul (node) dengan maksimal empat anak (children). Simpul daun (leaf) merepresentasikan area gambar yang seragam, sementara simpul internal menunjukkan area yang masih membutuhkan pembagian lebih lanjut. Setiap simpul menyimpan informasi seperti posisi (x, y), ukuran (width, height), dan nilai rata-rata warna atau intensitas piksel dalam area tersebut. Struktur ini memungkinkan pengkodean data gambar yang lebih efisien dengan menghilangkan redundansi pada area yang seragam. QuadTree sering digunakan dalam algoritma kompresi lossy karena mampu mengurangi ukuran file secara signifikan tanpa mengorbankan detail penting pada gambar.

## **1.2. Algoritma Divide and Conquer**

Algoritma Divide and Conquer merupakan salah satu algoritma dengan pendekatan rekursif yang memecah persoalan menjadi upapersoalan yang lebih kecil lalu menggabungkan solusinya kembali untuk mendapatkan solusi pada persoalan utama. Algoritma Divide and Conquer juga dapat didefinisikan dengan tiga tahapan yaitu divide (membagi persoalan), conquer (menyelesaikan upapersoalan), dan combine (menggabungkan solusi masing-masing upapersoalan).

## **1.3. Penerapan Algoritma Divide and Conquer pada Kompresi Gambar dengan Metode Quadtree**

Kompresi gambar menggunakan algoritma divide and conquer dilakukan dengan cara memecah gambar yang kompleks menjadi upagambar yang lebih sederhana dan menggabungkan solusi masing-masing upagambar tersebut untuk membentuk solusi akhir. Berikut adalah langkah-langkah detail dari algoritmanya:

### **1. Inisialisasi dan Pembentukan Quadtree Awal**

Proses kompresi gambar dengan Quadtree dimulai dengan inisialisasi struktur data Quadtree yang merepresentasikan seluruh area gambar sebagai satu blok besar. Blok ini menyimpan informasi seperti posisi baris dan kolom awal, dimensi panjang dan lebar blok, average pixel, serta referensi keempat sub-blok yang akan terbentuk bila blok tersebut perlu dibagi lebih lanjut. Pada tahap ini, blok dianggap sebagai daun (leaf) karena belum ada pembagian lebih jauh. Struktur awal ini berfungsi sebagai titik awal dari proses divide and conquer yang akan memecah gambar menjadi blok-blok lebih kecil berdasarkan homogenitas warna.

### **2. Perhitungan Rata-Rata Warna dalam Blok**

Setelah blok utama terbentuk, langkah berikutnya adalah melakukan perhitungan nilai rata-rata warna dalam blok tersebut. Setiap piksel dalam area blok diakses satu per satu untuk menjumlahkan nilai warna merah, hijau, dan biru secara terpisah. Total dari masing-masing warna ini kemudian dibagi dengan jumlah total piksel dalam blok untuk mendapatkan nilai rata-rata warna. Nilai ini sangat penting karena akan menjadi representasi warna dari blok tersebut ketika nanti proses kompresi selesai, terutama untuk blok-blok yang dianggap homogen dan tidak dibagi lagi.

### **3. Evaluasi Homogenitas Blok dengan Berbagai Metode Error**

Setelah rata-rata warna diketahui, blok dievaluasi apakah warnanya cukup homogen atau tidak. Evaluasi ini dilakukan dengan menghitung Metode Error menggunakan Variance, Mean Absolute Deviation (MAD), Max Pixel Difference, dan Entropy. Pengguna bisa memilih salah satu dari 4 metode ini untuk dijadikan patokan dalam pembagian blok. Jika Metode Error melebihi ambang batas (threshold) yang telah ditetapkan, maka blok tersebut dianggap masih memiliki variasi warna yang signifikan dan perlu diproses lebih lanjut.

### **4. Pembagian Blok menjadi Empat Sub-Blok (Divide)**

Jika hasil evaluasi Metode Error menunjukkan bahwa blok gambar tersebut tidak homogen atau masih lebih besar dari threshold dan masih memenuhi ukuran gambar diatas minimum block size, maka blok akan dibagi menjadi empat bagian yang lebih kecil. Setiap sub-blok ini merepresentasikan kuadran dari blok asal, yaitu Q1 (atas kiri), Q2 (atas kanan), Q3 (bawah kiri), dan Q4 (bawah kanan). Proses pembagian ini bertujuan untuk memisahkan area gambar yang memiliki perbedaan warna signifikan ke dalam blok-blok yang lebih spesifik, sehingga nantinya setiap blok kecil hanya akan mencakup area dengan warna yang lebih seragam.

#### 5. Rekursi pada Setiap Sub-Blok (Conquer)

Setelah blok dibagi menjadi empat bagian, proses kompresi dilanjutkan secara rekursif pada masing-masing sub-blok. Setiap sub-blok diproses dengan langkah yang sama: dihitung rata-rata warnanya, dievaluasi berdasarkan metode errornya, dan diputuskan apakah perlu dibagi lagi atau tidak. Proses ini dilakukan terus-menerus sampai seluruh blok yang terbentuk sudah memenuhi salah satu dari dua kondisi penghentian yaitu ukuran blok sudah mencapai batas minimum yang diperbolehkan, atau metode error blok sudah berada di bawah threshold.

#### 6. Penyusunan Struktur Pohon dan Hasil Akhir Kompresi

Ketika semua blok sudah memenuhi kriteria penghentian, hasil akhirnya adalah struktur pohon Quadtree yang menyimpan blok-blok homogen dari gambar tersebut. Setiap daun dari pohon Quadtree merepresentasikan satu blok hasil kompresi dengan nilai rata-rata warna yang telah dihitung sebelumnya. Struktur ini memungkinkan representasi gambar yang jauh lebih efisien, karena blok-blok homogen yang besar dapat disimpan sebagai satu unit tanpa harus menyimpan data setiap pikselnya. Selain itu, struktur pohon ini juga memudahkan proses rekonstruksi gambar kembali, atau untuk keperluan kompresi lebih lanjut.

Pseudocode Algoritma Utama Proses Kompresi:

```
Procedure compress (input/output qt : Quadtree)
KAMUS LOKAL
variance : double // variansi warna dari potongan image
img : Image // variabel global referensi image masukkan
size : double
splitSize : double

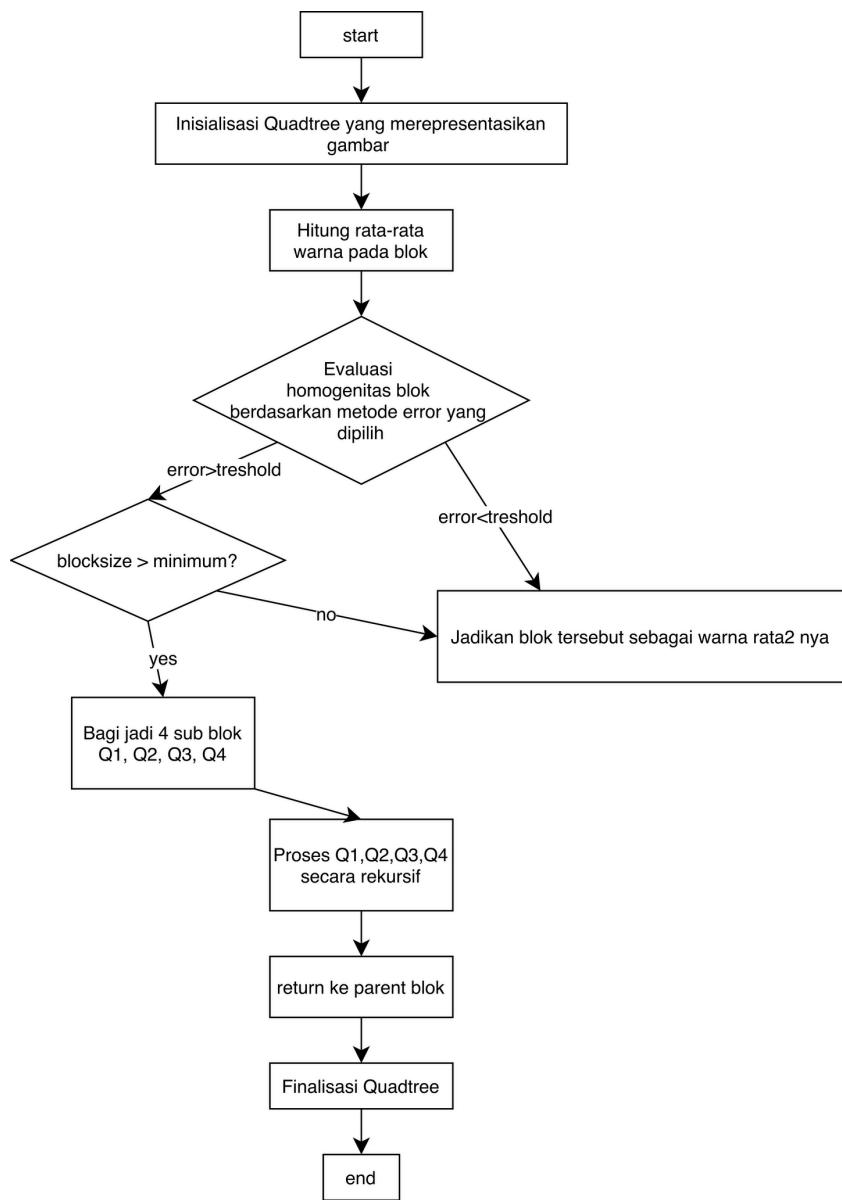
ALGORITMA
qt.calcAvgPixel() // menghitung rerata warna
variance <- img.getErrorMeasurement(qt.getStartRow,
                                    qt.getStartCol, qt.getRow, qt.getCol )
size <- qt.getCol() * qt.getRow()
splitSize <- (qt.getCol() / 2) * (qt.getRow() / 2)

// BASIS
if size <= img.minimumBlockSize
    or splitSize <= img.minimumBlockSize then
```

```

->
else if variance >= img.threshold then
->
// REKURENS
else
    qt.split() // membuat 4 cabang baru
    // melakukan rekurens untuk setiap cabang baru
    compress(qt.getQ1())
    compress(qt.getQ2())
    compress(qt.getQ3())
    compress(qt.getQ4())

```



Gambar 2. Flowchart Proses Kompresi Gambar Menggunakan Quadtree

## BAB 2

### IMPLEMENTASI PROGRAM

#### 2.1. File Compressor.java

Method	Tipe	Deskripsi
compress(Quadtree qt)	static void	Melakukan kompresi gambar menggunakan metode quadtree secara rekursif.

#### 2.2. File IO.java

Atribut	Tipe	Deskripsi
fileName	String	Nama file yang akan diproses
fileScanner	Scanner	Scanner untuk membaca isi file
inputScanner	Scanner (static)	Scanner untuk input pengguna
infoImage	Image (static)	Objek gambar hasil pembacaan

Method	Tipe	Deskripsi
IO(String fileName)	public (konstruktor)	Konstruktor untuk menginisialisasi nama file
getFileName()	public String	Mengembalikan nama file
openFile()	public void	Membuka dan membaca file
closeFile()	public void	Menutup file yang dibuka
readErrorMethod()	public static int	Membaca metode error yang dipilih pengguna
readThreshold(int max)	public static int	Membaca nilai threshold dari input pengguna
readMinBlock(Image image)	public static int	Membaca nilai minimum block size dari input pengguna
readOutputPath(String inputPath)	public static String	Membaca path output hasil kompresi dari input pengguna
readOutputGIFPath()	public static String	Membaca path output untuk animasi GIF

readFileName()	public static String	Membaca nama file dari input pengguna dan memvalidasinya
readImage(String fileName)	public static Image	Membaca file gambar dan mengembalikan objek Image
reconstructImage(Quadtree qt, BufferedImage img)	private static void	Meregenerasi gambar dari struktur quadtree
saveImage(String filepath, Quadtree qt)	public static void	Menyimpan image yang telah dilakukan kompresi
isSupportedFormat(String format)	public static boolean	Mengirimkan true jika format memiliki extension valid
getFileSize(String filepath)	public static long	Mengembalikan ukuran file
calculateCompressionPercentage(long inputsize, long outputsize)	public static double	Mengembalikan nilai perbandingan setelah kompresi
createProcessBufferedImage(Quadtree qt, int width, int height)	public static BufferedImage[]	Mengembalikan BufferedImage[] yang berisi frame-frame proses pembentukan gambar
drawFrame(Quadtree qt, BufferedImage[] frames, int currentDepth )	private static void	Fungsi rekursif untuk mengisi frame dengan warna pada proses kompresi
createGIF(BufferedImage[] frames, String outputPath, int delay)	public static void	Membentuk GIF dari BufferedImage[] dan menyimpan pada ouptutpath

### 2.3. File Image.java

Atribut	Tipe	Deskripsi
matrix	Pixel[][]	Matriks 2D yang merepresentasikan piksel gambar
rows	int	Jumlah baris gambar
cols	int	Jumlah kolom gambar
rowStart	int	Posisi awal baris gambar (untuk pemotongan/sub-image)

colStart	int	Posisi awal kolom gambar (untuk pemotongan/sub-image)
avgPixel	Pixel	Nilai rata-rata piksel dari gambar
method	int	Metode pengukuran error yang digunakan
threshold	double	Ambang batas error untuk pembagian gambar
minimumBlockSize	int	Ukuran blok minimum untuk kompresi
inputScanner	static Scanner	Scanner untuk membaca input dari pengguna

Method	Tipe	Deskripsi
Image(int _rows, int _cols)	public Constructor	Konstruktor untuk membuat gambar baru
Image(Image img, int _rows, int _cols, int rowStart, int colStart, int region)	public Constructor	Konstruktor untuk membuat sub-gambar dari gambar lain
setImageParam(int _method, double _threshold, int _minimumBlockSize)	public void	Mengatur parameter kompresi gambar
setPixel(int row, int col, Pixel p)	public void	Menetapkan piksel pada posisi tertentu
setAvgPixel()	public void	Menghitung dan menetapkan piksel rata-rata
getPixel(int row, int col)	public Pixel	Mengambil objek piksel pada posisi tertentu
getAvgPixel()	public Pixel	Mengambil piksel rata-rata dari gambar
getSize()	public int	Mengembalikan ukuran sisi gambar (diasumsikan persegi)
getRow()	public int	Mengambil jumlah baris dari gambar
getCol()	public int	Mengambil jumlah kolom dari gambar

getStartRow()	public int	Mengambil posisi awal baris gambar
getStartCol()	public int	Mengambil posisi awal kolom gambar
print(int limitRow)	public void	Menampilkan isi gambar sampai sejumlah baris tertentu
splitImage()	public Image[]	Membagi gambar menjadi empat bagian sub-gambar
getErrorMeasurement(int i, int j, int r, int c, Pixel avgP)	private double	Menghitung nilai error berdasarkan metode tertentu pada blok gambar
Variance(int i, int j, int r, int c, Pixel avgP)	private double	Menghitung varians blok piksel
MeanAbsoluteDeviation(int i, int j, int r, int c, Pixel avgP)	private double	Menghitung deviasi absolut rata-rata dari blok piksel
MaxPixelDifference(int i, int j, int r, int c, Pixel avgP)	private double	Menghitung selisih piksel terbesar pada blok terhadap rata-rata
Entropy(int i, int j, int r, int c, Pixel avgP)	private double	Menghitung entropi blok piksel
SSIM()	public double	Menghitung nilai SSIM (Structural Similarity Index) untuk seluruh gambar

#### 2.4. File Pixel.java

Atribut	Tipe	Deskripsi
red	int	Nilai komponen warna merah (Red)
green	int	Nilai komponen warna hijau (Green)
blue	int	Nilai komponen warna biru (Blue)

Method	Tipe	Deskripsi
Pixel(int _red, int _green, int _blue)	public Constructor	Konstruktor untuk membuat objek Pixel baru

Pixel(Pixel other)	public Constructor	Konstruktor salinan dari objek Pixel lain
getGreen()	public int	Mengambil nilai komponen hijau
setGreen(int green)	public void	Mengatur nilai komponen hijau
getRed()	public int	Mengambil nilai komponen merah
setRed(int red)	public void	Mengatur nilai komponen merah
getBlue()	public int	Mengambil nilai komponen biru
setBlue(int blue)	public void	Mengatur nilai komponen biru
print()	public void	Menampilkan nilai RGB dari objek Pixel

## 2.5. File Quadtree.java

Atribut	Tipe	Deskripsi
leaf	boolean	Menandakan apakah simpul merupakan daun
startRow	int	Baris awal dari area gambar yang direpresentasikan
startCol	int	Kolom awal dari area gambar yang direpresentasikan
row	int	Jumlah baris area gambar yang direpresentasikan
col	int	Jumlah kolom area gambar yang direpresentasikan
avgPixel	Pixel	Rata-rata nilai piksel dari area tersebut
Q1	Quadtree	Sub-kuadran kiri atas
Q2	Quadtree	Sub-kuadran kanan atas
Q3	Quadtree	Sub-kuadran kiri bawah
Q4	Quadtree	Sub-kuadran kanan bawah

Method	Tipe	Deskripsi
Quadtree(int row, int col, int startRow, int startCol)	public Constructor	Konstruktor untuk membuat objek Quadtree
getAvgPixel()	public Pixel	Mengembalikan nilai piksel rata-rata area
getRow()	public int	Mengembalikan jumlah baris area
getCol()	public int	Mengembalikan jumlah kolom area
getStartRow()	public int	Mengembalikan baris awal area
getStartCol()	public int	Mengembalikan kolom awal area
setLeaf(boolean b)	public void	Mengatur status simpul sebagai daun atau bukan
isLeaf()	public boolean	Mengecek apakah simpul merupakan daun
getQ1()	public Quadtree	Mengambil sub-kuadran kiri atas
getQ2()	public Quadtree	Mengambil sub-kuadran kanan atas
getQ3()	public Quadtree	Mengambil sub-kuadran kiri bawah
getQ4()	public Quadtree	Mengambil sub-kuadran kanan bawah
split()	public void	Membagi simpul menjadi 4 sub-kuadran jika merupakan daun
print(Quadtree qt, int n)	public void	Menampilkan struktur pohon quadtree secara hierarkis ke konsol
getDepth()	public int	Mengembalikan kedalaman maksimum dari pohon quadtree
countNodes()	public int	Menghitung total jumlah simpul pada pohon quadtree
calcAvgPixel()	public void	Menghitung rata-rata piksel

		dari area gambar yang direpresentasikan simpul
--	--	---

## 2.6. File Main.java

File ini akan memanggil fungsi-fungsi pada file lain dan akan menangani interaksi pengguna, mulai dari meminta input file, membaca konfigurasi file, menjalankan algoritma divide and conquer, hingga menampilkan hasil.

## BAB 3

### SOURCE CODE PROGRAM

#### 3.1. Repository Github

Source Code lengkap dapat diakses melalui link github berikut:

[https://github.com/AlfianHanifFY/Tucil2\\_13523073\\_13523095.git](https://github.com/AlfianHanifFY/Tucil2_13523073_13523095.git)

#### 3.2. Source Code

##### 1. Compressor.java

```
package lib;

public class Compressor {

    public static void compress(Quadtree qt) {
        qt.calcAvgPixel();
        if (qt.getCol() * qt.getRow() <= IO.infoImage.minimumBlockSize
            || qt.getCol() / 2 * qt.getRow() / 2 <=
IO.infoImage.minimumBlockSize) {
            return;
        }

        double variance = IO.infoImage.getErrorMeasurement(qt.getStartRow(),
qt.getStartCol(), qt.getRow(),
qt.getCol(), qt.getAvgPixel());

        // System.out.println("Variance: " + variance);
        if (variance > IO.infoImage.treshold) {
            qt.split();
            compress(qt.getQ1());
            compress(qt.getQ2());
            compress(qt.getQ3());
            compress(qt.getQ4());
        }
    }
}
```

##### 2. IO.java

```
package lib;

import java.util.*;
import java.io.*;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
```

```

import javax.imageio.ImageIO;
import javax.imageio.ImageWriter;
import javax.imageio.stream.FileImageOutputStream;
import javax.imageio.stream.ImageOutputStream;

import java.awt.image.BufferedImage;
import java.awt.Color;
import com.madgag.gif.fmsware.AnimatedGifEncoder;

public class IO {

    public String fileName;
    public Scanner fileScanner;
    public static Scanner inputScanner = new Scanner(System.in);
    public static Image infoImage;

    // Konstruktor
    public IO(String fileName) {
        this.fileName = fileName;
    }

    public String getFileName() {
        return this.fileName;
    }

    // Open & Close File
    public void openFile() {
        try {
            File file = new File(getFileName());
            this.fileScanner = new Scanner(file);
        } catch (FileNotFoundException e) {
            System.out.println("\u001B[33m[WARNING]\u001B[0m" + " : File " +
getFileName()
                + " tidak ditemukan di directory...");}
            System.exit(0);
        }
    }

    public void closeFile() {
        if (fileScanner != null) {
            fileScanner.close();
        }
    }

    public static int readErorrMethod() {

```

```

        int method = -1;

        System.out.println("\n\u001B[34m[INFO]\u001B[0m : Pilih Error
Measurement !");
        System.out.println(" [1] Variance");
        System.out.println(" [2] Mean Absolute Deviation");
        System.out.println(" [3] Max Pixel Difference");
        System.out.println(" [4] Entropy");
        System.out.println(" [5] SSIM\n");

        while (true) {
            try {
                System.out.print("\u001B[38;5;214m[INPUT]\u001B[0m" + " : ");
                method = inputScanner.nextInt();
                if (method >= 1 && method <= 5) {
                    break;
                } else {
                    System.out.println("\u001B[33m[WARNING]\u001B[0m" + " :
Masukkan integer (1 -- 4) !");
                }
            }

            } catch (NumberFormatException e) {
                System.out.println("\u001B[33m[WARNING]\u001B[0m" + " :
Masukan wajib integer !");
            }
        }

        return method;
    }

    public static double readThreshold(int method) {
        double Threshold = -1;

        System.out.println("\n\u001B[34m[INFO]\u001B[0m : Masukkan Threshold
!");
        if (method == 1) {
            System.out.println("\u001B[34m[INFO]\u001B[0m : Threshold min
adalah : 0 dan max adalah : 255");

            System.out.println(
                "\u001B[34m[INFO]\u001B[0m : Rekomendasi Threshold dengan
pertimbangan kualitas gambar dan kompresi: 1 - 5");
        }

        else if (method == 2) {
    
```

```

        System.out.println("\u001B[34m[INFO]\u001B[0m : Threshold min
adalah : 0 dan max adalah : 255");

        System.out.println(
                "\u001B[34m[INFO]\u001B[0m : Rekomendasi Threshold dengan
pertimbangan kualitas gambar dan kompresi: 1 - 5");
    }

    else if (method == 3) {
        System.out.println("\u001B[34m[INFO]\u001B[0m : Threshold min
adalah : 0 dan max adalah : 255");

        System.out.println(
                "\u001B[34m[INFO]\u001B[0m : Rekomendasi Threshold dengan
pertimbangan kualitas gambar dan kompresi: 1 - 5");
    }

    else if (method == 4) {
        System.out.println("\u001B[34m[INFO]\u001B[0m : Threshold min
adalah : 0 dan max adalah : 8");

        System.out.println(
                "\u001B[34m[INFO]\u001B[0m : Rekomendasi Threshold dengan
pertimbangan kualitas gambar dan kompresi: 0,1 - 1");
    }

    else if (method == 5) {

        System.out.println("\u001B[34m[INFO]\u001B[0m : Threshold min
adalah : 0 dan max adalah : 1");

        System.out.println(
                "\u001B[34m[INFO]\u001B[0m : Rekomendasi Threshold dengan
pertimbangan kualitas gambar dan kompresi: 0,1-0,5");
    }

    System.out.println("");

    // System.out.println("\n\u001B[34m[INFO]\u001B[0m : Masukkan
Threshold !");

    while (true) {
        try {
            System.out.print("\u001B[38;5;214m[INPUT]\u001B[0m" + " : ");

```

```

        Threshold = inputScanner.nextDouble(); // Gunakan
nextDouble() untuk angka desimal

        if (Threshold < 0) {
            System.out.println("\u001B[33m[WARNING]\u001B[0m" + " :
Masukkan Threshold positif !");
            continue;
        }

        if (method == 1) {
            if (Threshold > 255) {
                System.out.println("\u001B[33m[WARNING]\u001B[0m" + " :
Masukkan Threshold <= 255 !");
                continue;
            }
        } else if (method == 2) {
            if (Threshold > 255) {
                System.out.println("\u001B[33m[WARNING]\u001B[0m" + " :
Masukkan Threshold <= 255 !");
                continue;
            }
        } else if (method == 3) {
            if (Threshold > 255) {
                System.out.println("\u001B[33m[WARNING]\u001B[0m" + " :
Masukkan Threshold <= 255 !");
                continue;
            }
        } else if (method == 4) {
            if (Threshold > 8) {
                System.out.println("\u001B[33m[WARNING]\u001B[0m" + " :
Masukkan Threshold <= 8 !");
                continue;
            }
        } else if (method == 5) {
            if (Threshold > 1) {
                System.out.println("\u001B[33m[WARNING]\u001B[0m" + " :
Masukkan Threshold <= 1 !");
                continue;
            }
        }

        break;
    } catch (InputMismatchException e) { // Tangkap
InputMismatchException
        System.out.println(

```

```

    "\u001B[33m[WARNING]\u001B[0m" + " : Masukkan angka
yang valid (gunakan koma untuk desimal)!");

        inputScanner.next(); // Bersihkan input yang salah
    }

}

return Threshold;
}

public static double readTargetCompression() {
    double targetCompression = -1;

    System.out.println("\n\u001B[34m[INFO]\u001B[0m : Masukkan Target
Compression (0 - 1) !");

    while (true) {
        try {
            System.out.print("\u001B[38;5;214m[INPUT]\u001B[0m" + " : ");
            targetCompression = inputScanner.nextDouble();
            if (targetCompression < 0 || targetCompression > 1) {
                System.out.println("\u001B[33m[WARNING]\u001B[0m" + " :
Masukkan Target Compression antara 0-1!");
                continue;
            }
            break;
        } catch (InputMismatchException e) {
            System.out.println(
                "\u001B[33m[WARNING]\u001B[0m" + " : Masukkan angka
yang valid (gunakan koma untuk desimal)!");

            inputScanner.next(); // Bersihkan input yang salah
        }
    }

    return targetCompression;
}

public static int readMinBlock() {
    int minBlock = -1;

    System.out.println("");

    System.out.print("\n\u001B[34m[INFO]\u001B[0m : Masukkan Minimum
Block !");
}

```

```

System.out.println("");

while (true) {
    try {
        System.out.print("\u001B[38;5;214m[INPUT]\u001B[0m" + " : ");
        minBlock = inputScanner.nextInt();
        /* nanti di atur validasi Threshold sesuai variance */
        if (minBlock > 0) {
            break;
        }
    } catch (NumberFormatException e) {
        System.out.println("\u001B[33m[WARNING]\u001B[0m" + " :
Masukkan wajib integer > 0 !");
    }
}

return minBlock;
}

public static String readOutputPath(String inputPath) {
    System.out.println("\n\u001B[34m[INFO]\u001B[0m : Masukkan alamat
(ABSOLUT) output hasil kompresi !");
    String fileName = "";
    inputScanner.nextLine();
    String inputFormat = inputPath.substring(inputPath.lastIndexOf('.') +
1).toLowerCase();
    while (true) {
        System.out.print("\u001B[38;5;214m[INPUT]\u001B[0m" + " : ");
        fileName = inputScanner.nextLine();
        // Ambil ekstensi file dari filePath
        String format = fileName.substring(fileName.lastIndexOf('.') +
1).toLowerCase();

        // Cek apakah format output sama seperti format input
        if (!format.equals(inputFormat)) {
            System.out.println(
                "\u001B[33m[WARNING]\u001B[0m : Format tidak sama
dengan format input! Gunakan format gambar yang sama (."
                + inputFormat + ")");
        } else {
            break;
        }
    }
    return fileName;
}

```

```

    }

    public static String readOutputGIFPath() {
        System.out.println("\n\u001B[34m[INFO]\u001B[0m : Masukkan alamat
(ABSOLUT) output GIF !");
        String fileName = "";
        while (true) {
            System.out.print("\u001B[38;5;214m[INPUT]\u001B[0m" + " : ");
            fileName = inputScanner.nextLine();
            // Ambil ekstensi file dari filePath
            String format = fileName.substring(fileName.lastIndexOf('.') +
1).toLowerCase();

            // Cek apakah format didukung oleh ImageIO
            if (!format.equals("gif")) {
                System.out.println(
                    "\u001B[33m[WARNING]\u001B[0m : Format tidak
didukung! Gunakan format GIF yang valid ! (.gif)");
            } else {
                break;
            }
        }
        return fileName;
    }

    // Read file name from input
    public static String readFileName() {
        String fileName = "";
        while (true) {
            try {
                System.out.print("\u001B[38;5;214m[INPUT]\u001B[0m" + " : ");
                fileName = inputScanner.nextLine();
                File file = new File(fileName);
                Scanner tempScanner = new Scanner(file);
                tempScanner.close();

                break;
            } catch (FileNotFoundException e) {
                System.out.println("\u001B[33m[WARNING]\u001B[0m" + " : File
" + fileName
                    + " tidak ditemukan di directory.");
            }
        }
        return fileName;
    }
}

```

```

}

public static void readImage(String fileName) {
    Image image = new Image(0, 0);
    try {
        // String fileName = IO.readFileName();
        File file = new File(fileName);
        BufferedImage img = ImageIO.read(file);
        int width = img.getWidth();
        int height = img.getHeight();
        image = new Image(height, width);

        // get rgb
        for (int i = 0; i < height; i++) {
            for (int j = 0; j < width; j++) {
                int rgb = img.getRGB(j, i);
                Color color = new Color(rgb);
                Pixel pixel = new Pixel(color.getRed(), color.getGreen(),
color.getBlue());
                image.setPixel(i, j, pixel);
            }
        }
    }

    } catch (Exception e) {
        System.out.println("\u001B[33m[WARNING]\u001B[0m" + " : Gagal
memproses image !");
    }
    infoImage = image;
}

private static void reconstructImage(Quadtree qt, BufferedImage image) {
    if (qt == null)
        return;

    if (qt.isLeaf()) {
        Pixel p = qt.getAvgPixel();
        int startRow = qt.getStartRow();
        int startCol = qt.getStartCol();
        int endRow = startRow + qt.getRow();
        int endCol = startCol + qt.getCol();

        for (int y = startRow; y < endRow; y++) {
            for (int x = startCol; x < endCol; x++) {
                // Periksa bahwa koordinat berada dalam batas gambar
            }
        }
    }
}

```

```

                if (x >= 0 && x < image.getWidth() && y >= 0 && y <
image.getHeight()) {
                    int r = p.getRed();
                    int g = p.getGreen();
                    int b = p.getBlue();
                    image.setRGB(x, y, new Color(r, g, b).getRGB());
                }
            }
        }

        // System.out.println(
        // " | Filling block at: (" + startCol + "," + startRow + ")
size: "
        // + qt.getInfoImage().getCol()
        // + "x" + qt.getInfoImage().getRow());

    } else {
        if (qt.getQ1() != null)
            reconstructImage(qt.getQ1(), image);
        if (qt.getQ2() != null)
            reconstructImage(qt.getQ2(), image);
        if (qt.getQ3() != null)
            reconstructImage(qt.getQ3(), image);
        if (qt.getQ4() != null)
            reconstructImage(qt.getQ4(), image);
    }
}

public static void saveImage(String filePath, Quadtree qt) {

    try {
        int width = qt.getCol();
        int height = qt.getRow();
        BufferedImage image = new BufferedImage(width, height,
BufferedImage.TYPE_INT_RGB);

        reconstructImage(qt, image);

        // Ambil ekstensi file dari filePath
        String format = filePath.substring(filePath.lastIndexOf('.') +
1).toLowerCase();

        // Cek apakah format didukung oleh ImageIO
        if (!isSupportedFormat(format)) {

```

```

        System.out.println(
            "\u001B[33m[WARNING] \u001B[0m : Format tidak
didukung! Gunakan format gambar yang valid.");
    }

    File outputFile = new File(filePath);
    ImageIO.write(image, format, outputFile);
    // System.out.println("\u001B[32m[SUKSES] \u001B[0m : Gambar
berhasil disimpan
        // sebagai " + filePath);

} catch (Exception e) {
    System.out.println("\u001B[31m[ERROR] \u001B[0m : Gagal menyimpan
gambar!");
    e.printStackTrace();
}
}

/***
 * Mengecek apakah format gambar didukung oleh ImageIO
 */
public static boolean isSupportedFormat(String format) {
    Iterator<ImageWriter> writers =
ImageIO.getImageWritersBySuffix(format);
    return writers.hasNext();
}

public static long getFileSize(String filePath) {
    Path path = Paths.get(filePath);
    long fileSize = -1; // Jika file tidak ditemukan atau error, return
-1

    try {
        fileSize = Files.size(path);
    } catch (IOException e) {
        System.err.println("\u001B[31m[ERROR] \u001B[0m : Gagal membaca
ukuran file " + filePath);
        e.printStackTrace();
    }

    return fileSize;
}

public static double calculateCompressionPercentage(long inputSize, long
outputSize) {
    if (inputSize == 0) {

```

```

        return 0; // Hindari pembagian dengan nol
    }

    return 100.0 * (1 - ((double) outputSize / inputSize));
}

public static BufferedImage[] createProcessBufferedImages(Quadtree qt,
int width, int height) {
    int depth = qt.getDepth();
    BufferedImage[] frames = new BufferedImage[depth + 1];

    for (int i = 0; i <= depth; i++) {
        frames[i] = new BufferedImage(width, height,
        BufferedImage.TYPE_INT_RGB);
    }

    drawFrame(qt, frames, 0);

    return frames;
}

private static void drawFrame(Quadtree qt, BufferedImage[] frames, int
currentDepth) {
    if (qt == null)
        return;

    Pixel p = qt.getAvgPixel();
    int startRow = qt.getStartRow();
    int startCol = qt.getStartCol();
    int endRow = startRow + qt.getRow();
    int endCol = startCol + qt.getCol();

    for (int y = startRow; y < endRow; y++) {
        for (int x = startCol; x < endCol; x++) {
            if (x >= 0 && x < frames[0].getWidth() && y >= 0 && y <
frames[0].getHeight()) {
                int color = new Color(p.getRed(), p.getGreen(),
p.getBlue()).getRGB();
                for (int i = currentDepth; i < frames.length; i++) {
                    frames[i].setRGB(x, y, color);
                }
            }
        }
    }

    if (qt.getQ1() != null)

```

```

        drawFrame(qt.getQ1(), frames, currentDepth + 1);
        if (qt.getQ2() != null)
            drawFrame(qt.getQ2(), frames, currentDepth + 1);
        if (qt.getQ3() != null)
            drawFrame(qt.getQ3(), frames, currentDepth + 1);
        if (qt.getQ4() != null)
            drawFrame(qt.getQ4(), frames, currentDepth + 1);
    }

    public static void createGIF(BufferedImage[] frames, String
outputFilePath, int delay) {
    AnimatedGifEncoder e = new AnimatedGifEncoder();
    e.start(outputFilePath);
    e.setDelay(delay);
    e.setRepeat(0); // 0 = loop forever

    for (BufferedImage frame : frames) {
        e.addFrame(frame);
    }

    e.finish();
}

}

```

### 3. Image.java

```

package lib;

import java.util.Scanner;

public class Image {
    private Pixel[][] matrix;
    private int rows;
    private int cols;
    private int rowStart;
    private int colStart;
    private Pixel avgPixel;

    // param
    int method;
    double treshold;
    int minimumBlockSize;

    static Scanner inputScanner = new Scanner(System.in);
}

```

```

// === ini buat nge tes co === //
public static void main(String[] args) {
    // Scanner scanner = new Scanner(System.in);
}

// Konstruktor
public Image(int _rows, int _cols) {
    this.rows = _rows;
    this.cols = _cols;
    this.rowStart = 0;
    this.colStart = 0;
    this.matrix = new Pixel[_rows][_cols];
    this.method = 1;
    this.treshold = 10;
    this.minimumBlockSize = 8;
}

// cctor
public Image(Image img, int _rows, int _cols, int rowStart, int colStart,
int region) {
    this.rows = _rows;
    this.cols = _cols;
    this.method = img.method;
    this.treshold = img.treshold;
    this.minimumBlockSize = img.minimumBlockSize;
    this.matrix = new Pixel[_rows][_cols];

    // rowStart dan colStart di sini sudah merupakan koordinat absolut
untuk
    // sub-image
    this.rowStart = rowStart;
    this.colStart = colStart;

    // Salin piksel dari image induk dengan konversi indeks lokal
    for (int i = 0; i < _rows; i++) {
        for (int j = 0; j < _cols; j++) {
            int localRow = (this.rowStart + i) - img.getStartRow();
            int localCol = (this.colStart + j) - img.getStartCol();
            Pixel originalPixel = img.getPixel(localRow, localCol);
            setPixel(i, j, originalPixel);
        }
    }
}
}

```

```

    public void setImageParam(int _method, double _threshold, int
_minimumBlockSize) {
    this.method = _method;
    this.threshold = _threshold;
    this.minimumBlockSize = _minimumBlockSize;
}

public void setPixel(int row, int col, Pixel p) {
    this.matrix[row][col] = p;
}

public void setAvgPixel() {
    int red = 0;
    int green = 0;
    int blue = 0;
    int totalPixels = getRow() * getCol();

    for (int i = 0; i < getRow(); i++) {
        for (int j = 0; j < getCol(); j++) {
            Pixel p = getPixel(i, j);
            red += p.getRed();
            green += p.getGreen();
            blue += p.getBlue();
        }
    }

    this.avgPixel = new Pixel(red / totalPixels, green / totalPixels,
blue / totalPixels);
}

public Pixel getPixel(int row, int col) {
    return this.matrix[row][col];
}

public Pixel getAvgPixel() {
    return this.avgPixel;
}

public int getSize() {
    return this.rows * this.cols;
}

public int getRow() {
    return this.rows;
}

```

```

public int getCol() {
    return this.cols;
}

public int getStartRow() {
    return this.rowStart;
}

public int getStartCol() {
    return this.colStart;
}

public void print(int limitRow) {
    int R;
    if (limitRow == 0) {
        R = getRow();
    } else {
        R = limitRow;
    }
    for (int i = 0; i < R; i++) {
        for (int j = 0; j < getCol(); j++) {
            Pixel p = getPixel(i, j);
            p.print();
            System.out.println();
        }
    }
}

public Image[] splitImage() {
    // membagi image menjadi 4 bagian

    /*
     * ILUSTRASI SPLIT
     *
     * collection[0] | collection[1]
     * -----
     * collection[2] | collection[3]
     *
     */
    Image[] collection = new Image[4];
    int midRow = rows / 2;
    int midCol = cols / 2;
}

```

```

        collection[0] = new Image(this, midRow, midCol, this.rowStart,
this.colStart, 0); // Top-left
        collection[1] = new Image(this, midRow, cols - midCol, this.rowStart,
this.colStart + midCol, 1); // Top-right
        collection[2] = new Image(this, rows - midRow, midCol, this.rowStart
+ midRow, this.colStart, 2); // Bottom-left
        collection[3] = new Image(this, rows - midRow, cols - midCol,
this.rowStart + midRow, this.colStart + midCol,
            3); // Bottom-right

    return collection;
}

/* Error Measurement Method */

public double getErrorMeasurement(int i, int j, int r, int c, Pixel avgP)
{
    if (method == 1) {
        return Variance(i, j, r, c, avgP);
    }

    if (method == 2) {
        return MeanAbsoluteDeviation(i, j, r, c, avgP);
    }

    if (method == 3) {
        return MaxPixelDifference(i, j, r, c, avgP);
    }

    if (method == 4) {
        return Entropy(i, j, r, c, avgP);
    }

    if (method == 5) {
        return SSIM();
    }

    // kalo ga ada method yang sesuai
    return 0;
}

private double Variance(int i, int j, int r, int c, Pixel avgP) {
    double redVariance = 0;
    double greenVariance = 0;
    double blueVariance = 0;
}

```

```

        for (int x = i; x < r + i; x++) {
            for (int y = j; y < c + j; y++) {
                Pixel p = getPixel(x, y);
                redVariance += Math.pow(p.getRed() - avgP.getRed(), 2);
                greenVariance += Math.pow(p.getGreen() - avgP.getGreen(), 2);
                blueVariance += Math.pow(p.getBlue() - avgP.getBlue(), 2);
            }
        }
        redVariance /= getSize();
        greenVariance /= getSize();
        blueVariance /= getSize();

        return (redVariance + greenVariance + blueVariance) / 3;
    }

    public double MeanAbsoluteDeviation(int i, int j, int r, int c, Pixel avgP) {
        double redMAD = 0;
        double greenMAD = 0;
        double blueMAD = 0;

        for (int x = i; x < r + i; x++) {
            for (int y = j; y < c + j; y++) {
                Pixel p = getPixel(x, y);
                redMAD += Math.abs(p.getRed() - avgP.getRed());
                greenMAD += Math.abs(p.getGreen() - avgP.getGreen());
                blueMAD += Math.abs(p.getBlue() - avgP.getBlue());
            }
        }
        redMAD /= getSize();
        greenMAD /= getSize();
        blueMAD /= getSize();

        return (redMAD + greenMAD + blueMAD) / 3;
    }

    public double MaxPixelDifference(int i, int j, int r, int c, Pixel avgP)
    {
        double redDiff = 0;
        double greenDiff = 0;
        double blueDiff = 0;
        double maxRed = 0;
        double maxGreen = 0;

```

```

        double maxBlue = 0;
        double minRed = 255;
        double minGreen = 255;
        double minBlue = 255;

        // Find max and min red, green, and blue
        for (int x = i; x < r + i; x++) {
            for (int y = j; y < c + j; y++) {
                Pixel p = getPixel(x, y);
                if (p.getRed() > maxRed) {
                    maxRed = p.getRed();
                } else if (p.getRed() < minRed) {
                    minRed = p.getRed();
                }
                if (p.getGreen() > maxGreen) {
                    maxGreen = p.getGreen();
                } else if (p.getGreen() < minGreen) {
                    minGreen = p.getGreen();
                }
                if (p.getBlue() > maxBlue) {
                    maxBlue = p.getBlue();
                } else if (p.getBlue() < minBlue) {
                    minBlue = p.getBlue();
                }
            }
        }

        redDiff = maxRed - minRed;
        greenDiff = maxGreen - minGreen;
        blueDiff = maxBlue - minBlue;

        return (redDiff + greenDiff + blueDiff) / 3;
    }

    public double Entropy(int i, int j, int r, int c, Pixel avgP) {
        double redEntropy = 0;
        double greenEntropy = 0;
        double blueEntropy = 0;
        int redCount[] = new int[256];
        int greenCount[] = new int[256];
        int blueCount[] = new int[256];

        // masukkan ke list agar bisa dihitung peluangnya
        for (int x = i; x < r + i; x++) {
            for (int y = j; y < c + j; y++) {

```

```

        Pixel p = getPixel(x, y);
        redCount[p.getRed()]++;
        greenCount[p.getGreen()]++;
        blueCount[p.getBlue()]++;
    }
}

for (int k = 0; k < 256; k++) {
    if (redCount[k] != 0) {
        redEntropy += (double) redCount[k] / getSize() *
Math.log((double) redCount[k] / getSize());
    }
    if (greenCount[k] != 0) {
        greenEntropy += (double) greenCount[k] / getSize() *
Math.log((double) greenCount[k] / getSize());
    }
    if (blueCount[k] != 0) {
        blueEntropy += (double) blueCount[k] / getSize() *
Math.log((double) blueCount[k] / getSize());
    }
}

redEntropy *= -1;
greenEntropy *= -1;
blueEntropy *= -1;

return (redEntropy + greenEntropy + blueEntropy) / 3;
}

public double SSIM() {
    return 0;
}
}

```

#### 4. Pixel.java

```

package lib;

public class Pixel {
    private int red;
    private int green;
    private int blue;

    public Pixel(int _red, int _green, int _blue) {
        this.red = _red;
        this.green = _green;
    }
}

```

```

        this.blue = _blue;
    }

    public Pixel(Pixel other) {
        this.red = other.red;
        this.green = other.green;
        this.blue = other.blue;
    }

    public int getGreen() {
        return this.green;
    }

    public void setGreen(int green) {
        this.green = green;
    }

    public int getRed() {
        return this.red;
    }

    public void setRed(int red) {
        this.red = red;
    }

    public int getBlue() {
        return this.blue;
    }

    public void setBlue(int blue) {
        this.blue = blue;
    }

    public void print() {
        System.out.print("R : " + getRed());
        System.out.print(" | G : " + getGreen());
        System.out.print(" | B : " + getBlue());
    }
}

```

## 5. Quadtree.java

```
package lib;
```

```

public class Quadtree {
    private boolean leaf;
    private int startRow;
    private int startCol;
    private int row;
    private int col;
    private Pixel avgPixel;
    private Quadtree Q1, Q2, Q3, Q4;

    public Quadtree(int row, int col, int startRow, int startCol) {
        this.row = row;
        this.col = col;
        this.startCol = startCol;
        this.startRow = startRow;
        this.avgPixel = null;
        this.leaf = true;
        this.Q1 = null;
        this.Q2 = null;
        this.Q3 = null;
        this.Q4 = null;
    }

    public Pixel getAvgPixel() {
        return this.avgPixel;
    }

    public int getRow() {
        return this.row;
    }

    public int getCol() {
        return this.col;
    }

    public int getStartRow() {
        return this.startRow;
    }

    public int getStartCol() {
        return this.startCol;
    }

    public void setLeaf(boolean b) {
        this.leaf = b;
    }
}

```

```

public boolean isLeaf() {
    return this.leaf;
}

public Quadtree getQ1() {
    if (!isLeaf()) {
        return this.Q1;
    }
    return null;
}

public Quadtree getQ2() {
    if (!isLeaf()) {
        return this.Q2;
    }
    return null;
}

public Quadtree getQ3() {
    if (!isLeaf()) {
        return this.Q3;
    }
    return null;
}

public Quadtree getQ4() {
    if (!isLeaf()) {
        return this.Q4;
    }
    return null;
}

public void split() {
    if (isLeaf()) {
        setLeaf(false);
        int midRow = row / 2;
        int midCol = col / 2;
        this.Q1 = new Quadtree(midRow, midCol, this.startRow,
this.startCol);
        this.Q2 = new Quadtree(midRow, this.col - midCol, this.startRow,
this.startCol + midCol);
        this.Q3 = new Quadtree(this.row - midRow, midCol, this.startRow +
midRow, this.startCol);
    }
}

```

```

        this.Q4 = new Quadtree(row - midRow, col - midCol, this.startRow
+ midRow, this.startCol + midCol);
    }
}

public void print(Quadtree qt, int n) {
    if (qt == null)
        return;

    System.out.println(" ".repeat(n) + (qt.isLeaf() ? "A" : "Q"));

    if (!qt.isLeaf()) {
        print(qt.getQ1(), n + 2);
        print(qt.getQ2(), n + 2);
        print(qt.getQ3(), n + 2);
        print(qt.getQ4(), n + 2);
    }
}

public int getDepth() {
    if (isLeaf())
        return 0;
    } else {
        int d1 = Q1.getDepth();
        int d2 = Q2.getDepth();
        int d3 = Q3.getDepth();
        int d4 = Q4.getDepth();
        return 1 + Math.max(Math.max(d1, d2), Math.max(d3, d4));
    }
}

public int countNodes() {
    if (isLeaf())
        return 1;
    } else {
        return 1 + Q1.countNodes() + Q2.countNodes() + Q3.countNodes() +
Q4.countNodes();
    }
}

public void calcAvgPixel() {
    int red = 0;
    int green = 0;
    int blue = 0;
    int totalPixels = getRow() * getCol();
}

```

```

        for (int i = startRow; i < getRow() + startRow; i++) {
            for (int j = startCol; j < getCol() + startCol; j++) {
                Pixel p = IO.infoImage.getPixel(i, j);
                red += p.getRed();
                green += p.getGreen();
                blue += p.getBlue();
            }
        }
        this.avgPixel = new Pixel(red / totalPixels, green / totalPixels,
blue / totalPixels);
    }
}

```

## 6. main.java

```

import java.awt.image.BufferedImage;
import java.util.*;
import lib.*;

public class main {
    static Scanner inputScanner = new Scanner(System.in);

    public static void main(String[] args) {
        System.out.println("\u001B[34m[INFO]\u001B[0m : Wilujeng
Sumping...\n");

        boolean status = true;

        while (status) {
            System.out.println("\u001B[34m[INFO]\u001B[0m : Masukkan alamat
ABSOLUT image !\n");

            String inputFile = IO.readFileName();

            int method = IO.readErorrMethod();
            double Threshold = IO.readThreshold(method);
            int minBlock = IO.readMinBlock();
            double targetKompresi = IO.readTargetCompression();

            String outputFile = IO.readOutputPath(inputFile);
            String outputGIFFile = IO.readOutputGIFPath();

            Quadtree quadtree;

            boolean compressionSuccess = false;
            boolean gifSuccess = false;
        }
    }
}

```

```

        long startTime = System.currentTimeMillis();
        if (targetKompresi == 0) {
            System.out.println("\n\u001B[34m[INFO]\u001B[0m : Memproses
gambar dengan Quadtree...\n");
            IO.readImage(inputFile);
            IO.infoImage.setImageParam(method, Threshold, minBlock);
            quadtree = new Quadtree(IO.infoImage.getRow(),
IO.infoImage.getCol(), 0, 0);
            try {
                Compressor.compress(quadtree);

                IO.saveImage(outputFile, quadtree);
                compressionSuccess = true;
            } catch (Exception e) {
                System.out.println(
                    "\u001B[31m[ERROR]\u001B[0m : Terjadi kesalahan
saat kompresi gambar - " + e.getMessage());
            }
        } else {
            System.out.println(
                "\n\u001B[34m[INFO]\u001B[0m : Memproses gambar
dengan Quadtree untuk mencapai target kompresi ... \n");
            IO.readImage(inputFile);
            IO.infoImage.setImageParam(method, Threshold, minBlock);
            quadtree = new Quadtree(IO.infoImage.getRow(),
IO.infoImage.getCol(), 0, 0);
            double lowerBound = 0;
            double upperBound = 255;
            if (method == 1) {
                upperBound = 65025;
            } else if (method == 2) {
                upperBound = 255;
            } else if (method == 3) {
                upperBound = 255;
            } else if (method == 4) {
                upperBound = 8;
            } else if (method == 5) {
                upperBound = 1;
            }
            double tolerance = 0.01;
            double bestThreshold = Threshold;
            double currentCompressionRate = 0;
            boolean found = false;

```

```

        double maxCompressionRate;
        IO.infoImage.setImageParam(method, upperBound, minBlock);
        Compressor.compress(quadtree);
        IO.saveImage(outputFile, quadtree);
        maxCompressionRate =
        IO.calculateCompressionPercentage(IO.getFileSize(inputFile),
                                         IO.getFileSize(outputFile));
        double minCompressionRate;
        IO.infoImage.setImageParam(method, lowerBound, minBlock);
        Compressor.compress(quadtree);
        IO.saveImage(outputFile, quadtree);
        minCompressionRate =
        IO.calculateCompressionPercentage(IO.getFileSize(inputFile),
                                         IO.getFileSize(outputFile));

        while (upperBound - lowerBound > tolerance) {
            double midThreshold = (lowerBound + upperBound) / 2;
            // System.out.println("[INFO] : Mencoba threshold " +
midThreshold);

            IO.infoImage.setImageParam(method, midThreshold,
minBlock);
            Quadtree tempquadtree = new
Quadtree(IO.infoImage.getRow(), IO.infoImage.getCol(), 0, 0);

            try {
                Compressor.compress(tempquadtree);

                // Simpan hasil kompres sementara ke file sementara
                String tempOutputFile = "temp_output_" + midThreshold
+ ".jpg";
                IO.saveImage(tempOutputFile, tempquadtree);

                long inputSize = IO.getFileSize(inputFile);
                long outputSize = IO.getFileSize(tempOutputFile);
                currentCompressionRate =
                IO.calculateCompressionPercentage(inputSize, outputSize);
                // System.out.println("[INFO] : lowerBound : " +
lowerBound);
                // System.out.println("[INFO] : upperBound : " +
upperBound);
                // System.out.println("[INFO] : Persentase kompresi :
" +
                // currentCompressionRate/100 + "%");

```

```

        // Hapus file sementara
        new java.io.File(tempOutputFile).delete();

        if (Math.abs(currentCompressionRate / 100 -
targetKompresi) <= tolerance) {
            bestThreshold = midThreshold;
            found = true;
            compressionSuccess = true;
            break;
        }

        if (currentCompressionRate / 100 > targetKompresi) {
            upperBound = midThreshold;
        } else {
            lowerBound = midThreshold;
        }

    } catch (OutOfMemoryError e) {
        System.out.println("\u001B[31m[ERROR]\u001B[0m :
Memori tidak cukup saat mencoba threshold "
+ midThreshold);
        upperBound = midThreshold; // Coba threshold lebih
rendah
    } catch (Exception e) {
        System.out.println("\u001B[31m[ERROR]\u001B[0m :
Kesalahan saat mencoba threshold "
+ midThreshold + " - " + e.getMessage());
        upperBound = midThreshold; // Coba threshold lebih
rendah
    }
}

if (found) {
    System.out.println("\u001B[34m[INFO]\u001B[0m : Threshold
terbaik ditemukan: " + bestThreshold);
    IO.infoImage.setImageParam(method, bestThreshold,
minBlock);
    Quadtree finalQuadtree = new
Quadtree(IO.infoImage.getRow(), IO.infoImage.getCol(), 0, 0);
    quadtree = finalQuadtree;
    Compressor.compress(finalQuadtree);
    IO.saveImage(outputFile, finalQuadtree);

} else {
    System.out.println(

```

```

        "\u001B[31m[ERROR]\u001B[0m : Tidak dapat
menemukan threshold yang memenuhi sesuai dengan metode error untuk mencapai
target kompresi dengan min block size "
                + minBlock);
System.out.println("\u001B[31m[ERROR]\u001B[0m :
Compression rate terbesar yang dicapai: "
                + maxCompressionRate / 100);
System.out.println("\u001B[31m[ERROR]\u001B[0m :
Compression rate terkecil yang dicapai: "
                + minCompressionRate / 100);
}

}

// Buat GIF
try {
    BufferedImage[] frames =
IO.createProcessBufferedImages(quadtree, IO.infoImage.getCol(),
                                IO.infoImage.getRow());
    IO.createGIF(frames, outputGIFFFile, 500);
    // System.out.println("\u001B[32m[SUKSES]\u001B[0m : GIF
berhasil dibuat di " +
                // outputGIFFFile);
    gifSuccess = true;
} catch (OutOfMemoryError e) {
    System.out.println("\u001B[31m[ERROR]\u001B[0m : Memori tidak
cukup saat membuat GIF.");
} catch (Exception e) {
    System.out
        .println("\u001B[31m[ERROR]\u001B[0m : Terjadi
kesalahan saat membuat GIF - " + e.getMessage());
}

// OUTPUT
try {
    long endTime = System.currentTimeMillis();
    long executionTime = endTime - startTime;
    System.out.println("\n\n\u001B[32m-----[OUTPUT
KOMPRESI]-----\u001B[0m");
    System.out.println("\u001B[32m[SUKSES]\u001B[0m : Waktu
Eksekusi : " + executionTime + " milliseconds");

    long fileInputSizeInBytes = IO.getFileSize(inputFile);
    System.out.println(
        "\u001B[32m[SUKSES]\u001B[0m : Ukuran File Input : "
        + fileInputSizeInBytes + " bytes");
}

```

```

        long fileOutputSizeInBytes = IO.getFileSize(outputFile);
        System.out.println(
                "\u001B[32m[SUKSES]\u001B[0m : Ukuran File Output : "
+ fileOutputSizeInBytes + " bytes");

        double compressionRate =
IO.calculateCompressionPercentage(fileInputSizeInBytes,
fileOutputSizeInBytes);
        System.out.println("\u001B[32m[SUKSES]\u001B[0m : Persentase
Kompresi : " + compressionRate + "%");

        int depth = quadtree.getDepth();
        System.out.println("\u001B[32m[SUKSES]\u001B[0m : Kedalaman
Pohon : " + depth);

        int nodeCount = quadtree.countNodes();
        System.out.println("\u001B[32m[SUKSES]\u001B[0m : Banyak
Simpul : " + nodeCount);

        System.out
                .println("\u001B[32m[SUKSES]\u001B[0m : Gambar
berhasil disimpan sebagai " + outputFile + "\n");

        System.out.println("\u001B[32m[SUKSES]\u001B[0m : GIF
berhasil dibuat di " + outputGIFFile);
        // Kesimpulan
        System.out.println("\n\u001B[34m[INFO]\u001B[0m : Status
akhir proses:");
        System.out.println(" - Kompresi Gambar : "
                + (compressionSuccess ? "\u001B[32mBerhasil\u001B[0m"
: "\u001B[31mGagal\u001B[0m"));
        System.out.println(" - Pembuatan GIF : "
                + (gifSuccess ? "\u001B[32mBerhasil\u001B[0m" :
"\u001B[31mGagal\u001B[0m"));

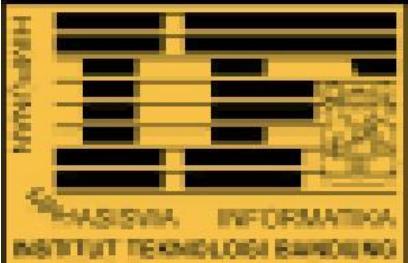
        // Tanya lanjut
        System.out.println("\n\u001B[34m[INFO]\u001B[0m : Apakah Anda
ingin memproses gambar lain?");
        System.out.println(" [1] Ya");
        System.out.println(" [2] Tidak (Keluar)");
        System.out.print("\u001B[38;5;214m[INPUT]\u001B[0m : ");
        int pilihan = inputScanner.nextInt();
        inputScanner.nextLine();
        status = (pilihan == 1);
    }
}

```

```
        System.out.println();
    } catch (Exception e) {
        System.out.println(
            "\u001B[31m[ERROR]\u001B[0m : Terjadi kesalahan saat
menampilkan hasil - " + e.getMessage());
    }
    System.out.println("\u001B[34m[INFO]\u001B[0m : Program selesai.
Hatur nuhun!");
}
}
```

## BAB 4

### UJI COBA PROGRAM

Test Case #1 Metode 1	
Input	
Output	 <a href="#">GIF</a>
CLI	<pre>[INFO] : Masukkan alamat ABSOLUT image ! [INPUT] : /Users/rafiffarras/SEMESTER_4/Strategi_Algoritma/Tucil2/Tucil2_13523073_13523095/test/input/hmif.jpg [INFO] : Pilih Error Measurement ! [1] Variance [2] Mean Absolute Deviation [3] Max Pixel Difference [4] Entropy [5] SSIM  [INPUT] : 1 1  [INFO] : Masukkan Threshold ! [INFO] : Threshold min adalah : 0 dan max adalah : 255 [INFO] : Rekomendasi Threshold dengan pertimbangan kualitas gambar dan kompresi: 1 - 5  [INPUT] : -1 [WARNING] : Masukkan Threshold positif ! [INPUT] : 256 [WARNING] : Masukkan Threshold &lt;= 255 [INPUT] : 1  [INFO] : Ukuran Gambar Anda : 200 x 300 atau 60000 pixel [INFO] : Masukkan Minimum Block ! [INFO] : Rekomendasi Minimum Block : 25 - 36 (5x5 - 6x6)  [INPUT] : 10 [INFO] : Masukkan Target Compression (0 - 1) ! [INPUT] : 0  [INFO] : Masukkan alamat (ABSOLUT) output hasil kompresi ! [INPUT] : /Users/rafiffarras/SEMESTER_4/Strategi_Algoritma/Tucil2/Tucil2_13523073_13523095/test/output/hmif-variance-t1-m10.jpg  [INFO] : Masukkan alamat (ABSOLUT) output GIF ! [INPUT] : /Users/rafiffarras/SEMESTER_4/Strategi_Algoritma/Tucil2/Tucil2_13523073_13523095/test/output/hmif-variance-t1-m10.gif  [INFO] : Memproses gambar dengan Quadtree... -----[OUTPUT KOMPRESI]----- [SUKSES] : Waktu Eksekusi : 80 milliseconds [SUKSES] : Ukuran File Input : 30326 bytes [SUKSES] : Ukuran File Output : 16160 bytes [SUKSES] : Persentase Kompresi : 46.712392006858806% [SUKSES] : Kedalaman Pohon : 6 [SUKSES] : Banyak Simpul : 4109 [SUKSES] : Gambar berhasil disimpan sebagai /Users/rafiffarras/SEMESTER_4/Strategi_Algoritma/Tucil2/Tucil2_13523073_13523095/test/output/hmif-variance-t1-m10.jpg  [SUKSES] : GIF berhasil dibuat di /Users/rafiffarras/SEMESTER_4/Strategi_Algoritma/Tucil2/Tucil2_13523073_13523095/test/output/hmif-variance-t1-m10.gif  [INFO] : Status akhir proses: - Kompresi Gambar : Berhasil - Pembuatan GIF : Berhasil  [INFO] : Apakah Anda ingin memproses gambar lain? [1] Ya [2] Tidak (Keluar)</pre>

Test Case #2 Metode 3	
Input	
Output	 <a href="#">GIF</a>
CLI	<pre>[INFO] : Wilujeng Sumping... [INFO] : Masukkan alamat ABSOLUT image ! [INPUT] : /Users/raffiffarras/SEMESTER_4/Strategi_Algoritma/Tucil2/Tucil2_13523073_13523095/test/output/pemandangan [WARNING] : File /Users/raffiffarras/SEMESTER_4/Strategi_Algoritma/Tucil2/Tucil2_13523073_13523095/test/output/pemandangan tidak ditemukan di directory. [INPUT] : /Users/raffiffarras/SEMESTER_4/Strategi_Algoritma/Tucil2/Tucil2_13523073_13523095/test/input/pemandangan.jpg  [INFO] : Pilih Error Measurement ! [1] Variance [2] Mean Absolute Deviation [3] Max Pixel Difference [4] Entropy [5] SSIM [INPUT] : 3  [INFO] : Masukkan Threshold ! [INFO] : Threshold min adalah : 0 dan max adalah : 255 [INFO] : Rekomendasi Threshold dengan pertimbangan kualitas gambar dan kompresi: 1 - 5 [INPUT] : 5 [INFO] : Ukuran Gambar Anda : 650 x 1024 atau 665600 pixel [INFO] : Masukkan Minimum Block ! [INFO] : Rekomendasi Minimum Block : 64 - 100 (8x8 - 10x10) [INPUT] : 10 [INFO] : Masukkan Target Compression (0 - 1) ! [INPUT] : 0 [INFO] : Masukkan alamat (ABSOLUT) output hasil kompresi ! [INPUT] : /Users/raffiffarras/SEMESTER_4/Strategi_Algoritma/Tucil2/Tucil2_13523073_13523095/test/output/pemandangan-mdp-t5-m10.jpg [INFO] : Masukkan alamat (ABSOLUT) output GIF ! [INPUT] : /Users/raffiffarras/SEMESTER_4/Strategi_Algoritma/Tucil2/Tucil2_13523073_13523095/test/output/pemandangan-mdp-t5-m10.gif [INFO] : Memproses gambar dengan Quadtree...  [OUTPUT KOMPRESI] [SUKSES] : Waktu Eksekusi : 2274 milliseconds [SUKSES] : Ukuran File Input : 1123210 bytes [SUKSES] : Ukuran File Output : 68671 bytes [SUKSES] : Persentase Kompresi : 93.886183495072% [SUKSES] : Kedalaman Pohon : 8 [SUKSES] : Banyak Simpul : 26633 [SUKSES] : Gambar berhasil disimpan sebagai /Users/raffiffarras/SEMESTER_4/Strategi_Algoritma/Tucil2/Tucil2_13523073_13523095/test/output/pemandangan-mdp-t5-m10.jpg [SUKSES] : GIF berhasil dibuat di /Users/raffiffarras/SEMESTER_4/Strategi_Algoritma/Tucil2/Tucil2_13523073_13523095/test/output/pemandangan-mdp-t5-m10.gif  [INFO] : Status akhir proses: - Kompresi Gambar : Berhasil - Pembuatan GIF : Berhasil</pre>

Input	
Output	 <a href="#"><u>GIF</u></a>
CLI	<pre>[INFO] : Masukkan alamat ABSOLUT image ! [INPUT] : /Users/alfianhaniffy/INFORMATIKA/STIMA/Tucil2/test/gal.jpg [INFO] : Pilih Error Measurement ! [1] Variance [2] Mean Absolute Deviation [3] Max Pixel Difference [4] Entropy [5] SSIM [INPUT] : 2  [INFO] : Masukkan Threshold ! [INFO] : Threshold min adalah : 0 dan max adalah : 255 [INFO] : Rekomendasi Threshold dengan pertimbangan kualitas gambar dan kompresi: 1 - 5 [INPUT] : 0  [INFO] : Masukkan Minimum Block ! [INPUT] : 2 [INFO] : Masukkan Target Compression (0 - 1) ! [INPUT] : 0 [INFO] : Masukkan alamat (ABSOLUT) output hasil kompresi ! [INPUT] : /Users/alfianhaniffy/INFORMATIKA/STIMA/Tucil2/test/2.jpg [INFO] : Masukkan alamat (ABSOLUT) output GIF ! [INPUT] : /Users/alfianhaniffy/INFORMATIKA/STIMA/Tucil2/test/2.gif [INFO] : Memproses gambar dengan Quadtree...</pre>

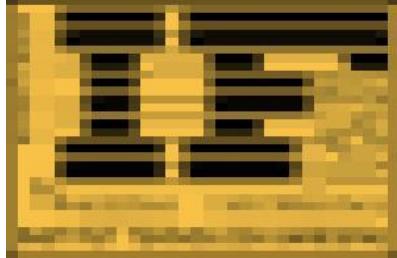
	<pre>[OUTPUT KOMPRESI] [SUKSES] : Waktu Eksekusi : 3117 milliseconds [SUKSES] : Ukuran File Input : 101310 bytes [SUKSES] : Ukuran File Output : 82920 bytes [SUKSES] : Persentase Kompresi : 18.15228610008884% [SUKSES] : Kedalaman Pohon : 9 [SUKSES] : Banyak Simpul : 113265 [SUKSES] : Gambar berhasil disimpan sebagai /Users/alfianhaniffy/INFORMATIKA/STIMA/Tucil2/test/2.jpg  [SUKSES] : GIF berhasil dibuat di /Users/alfianhaniffy/INFORMATIKA/STIMA/Tucil2/test/2.gif  [INFO] : Status akhir proses: - Kompresi Gambar : Berhasil - Pembuatan GIF : Berhasil</pre>
<b>Test Case #4 Metode 4</b>	
Input	
Output	 <p><a href="#">GIF</a></p>
CLI	<pre>[INFO] : Masukkan alamat ABSOLUT image ! [INPUT] : /Users/raffifarras/SEMESTER_4/Strategi_Algoritma/Tucil2/Tucil2_13523073_13523095/test/input/matahari.jpg  [INFO] : Pilih Error Measurement ! [1] Variance [2] Mean Absolute Deviation [3] Max Pixel Difference [4] Entropy [5] SSIM  [INPUT] : 4  [INFO] : Masukkan Threshold ! [INFO] : Threshold min adalah : 0 dan max adalah : 8 [INFO] : Rekomendasi Threshold dengan pertimbangan kualitas gambar dan kompresi: 0,1 - 1  [INPUT] : 0,01  [INFO] : Masukkan Minimum Block ! [INPUT] : 2  [INFO] : Masukkan Target Compression (0 - 1) ! [INPUT] : 0  [INFO] : Masukkan alamat (ABSOLUT) output hasil kompresi ! [INPUT] : /Users/raffifarras/SEMESTER_4/Strategi_Algoritma/Tucil2/Tucil2_13523073_13523095/test/output/ok.jpg  [INFO] : Masukkan alamat (ABSOLUT) output GIF ! [INPUT] : /Users/raffifarras/SEMESTER_4/Strategi_Algoritma/Tucil2/Tucil2_13523073_13523095/test/output/ok.gif  [INFO] : Memproses gambar dengan Quadtree...  [OUTPUT KOMPRESI] [SUKSES] : Waktu Eksekusi : 362 milliseconds [SUKSES] : Ukuran File Input : 341548 bytes [SUKSES] : Ukuran File Output : 20325 bytes [SUKSES] : Persentase Kompresi : 94.04915268132152% [SUKSES] : Kedalaman Pohon : 6 [SUKSES] : Banyak Simpul : 2173 [SUKSES] : Gambar berhasil disimpan sebagai /Users/raffifarras/SEMESTER_4/Strategi_Algoritma/Tucil2/Tucil2_13523073_13523095/test/output/ok.jpg  [SUKSES] : GIF berhasil dibuat di /Users/raffifarras/SEMESTER_4/Strategi_Algoritma/Tucil2/Tucil2_13523073_13523095/test/output/ok.gif  [INFO] : Status akhir proses: - Kompresi Gambar : Berhasil - Pembuatan GIF : Berhasil  [INFO] : Apakah Anda ingin memproses gambar lain? [1] Ya [2] Tidak (Keluar)</pre>

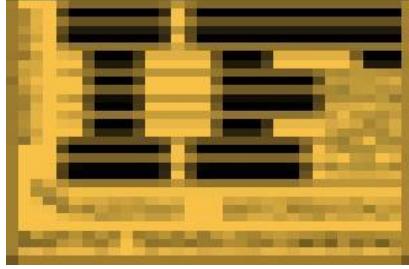
### Test Case #5 Metode 5

Input	
Output	 <u><a href="#">GIF</a></u>
CLI	<pre>[INFO] : Masukkan alamat ABSOLUT image ! [INFO] : /Users/rafiffarras/SEMESTER_4/Strategi_Algoritma/Tucil2/Tucil2_13523073_13523095/test/input/pemandangan.jpg  [INFO] : Pilih Error Measurement ! [1] Variance [2] Mean Absolute Deviation [3] Max Pixel Difference [4] Entropy [5] SSIM  [INPUT] : 5  [INFO] : Masukkan Threshold ! [INFO] : Threshold min adalah : 0 dan max adalah : 1 [INFO] : Rekomendasi Threshold dengan pertimbangan kualitas gambar dan kompresi: 0,1-0,5  [INPUT] : 0,04  [INFO] : Masukkan Minimum Block ! [INPUT] : 3  [INFO] : Masukkan Target Compression (0 - 1) ! [INPUT] : 0  [INFO] : Masukkan alamat (ABSOLUT) output hasil kompresi ! [INPUT] : /Users/rafiffarras/SEMESTER_4/Strategi_Algoritma/Tucil2/Tucil2_13523073_13523095/test/output/ssim.jpg  [INFO] : Masukkan alamat (ABSOLUT) output GIF ! [INPUT] : /Users/rafiffarras/SEMESTER_4/Strategi_Algoritma/Tucil2/Tucil2_13523073_13523095/test/output/ssim.gif  [INFO] : Memproses gambar dengan Quadtree...  [OUTPUT KOMPRESI] [OKSES] : Waktu Eksekusi : 1405 milliseconds [OKSES] : Ukuran File Input : 1123210 bytes [OKSES] : Ukuran File Output : 93092 bytes [OKSES] : Persentase Kompresi : 91.71196837634993% [OKSES] : Kedalaman Pohon : 8 [OKSES] : Banyak Simpul : 87381 [OKSES] : Gambar berhasil disimpan sebagai /Users/rafiffarras/SEMESTER_4/Strategi_Algoritma/Tucil2/Tucil2_13523073_13523095/test/output/ssim.jpg  [OKSES] : GIF berhasil dibuat di /Users/rafiffarras/SEMESTER_4/Strategi_Algoritma/Tucil2/Tucil2_13523073_13523095/test/output/ssim.gif  [INFO] : Status akhir proses: - Kompresi Gambar : Berhasil - Pembuatan GIF : Berhasil  [INFO] : Apakah Anda ingin memproses gambar lain? [1] Ya [2] Tidak (Keluar)</pre>

### Test Case #6 Target Kompresi

Input	
Output	 <a href="#">GIF</a>
CLI	<pre>[INFO] : Masukkan alamat ABSOLUT image ! [INPUT] : /Users/raffifarras/SEMESTER_4/Strategi_Algoritma/Tucil2/Tucil2_13523073_13523095/test/input/gunung.jpg  [INFO] : Pilih Error Measurement ! [1] Variance [2] Mean Absolute Deviation [3] Max Pixel Difference [4] Entropy [5] SSIM  [INPUT] : 1  [INFO] : Masukkan Threshold ! [INFO] : Threshold min adalah : 0 dan max adalah : 255 [INFO] : Rekomendasi Threshold dengan pertimbangan kualitas gambar dan kompresi: 1 - 5 [INPUT] : 1  [INFO] : Masukkan Minimum Block ! [INPUT] : 2  [INFO] : Masukkan Target Compression (0 - 1) ! [INPUT] : 0,5  [INFO] : Masukkan alamat (ABSOLUT) output hasil kompresi ! [INPUT] : /Users/raffifarras/SEMESTER_4/Strategi_Algoritma/Tucil2/Tucil2_13523073_13523095/test/output/gunungg.jpg  [INFO] : Masukkan alamat (ABSOLUT) output GIF ! [INPUT] : /Users/raffifarras/SEMESTER_4/Strategi_Algoritma/Tucil2/Tucil2_13523073_13523095/test/output/gunungg.gif  [INFO] : Memproses gambar dengan Quadtree untuk mencapai target kompresi ... [INFO] : Threshold terbaik ditemukan: 0.49610137939453125  [OUTPUT KOMPRESI] [SUkses] : Waktu Eksekusi : 665 milliseconds [SUkses] : Ukuran File Input : 38955 bytes [SUkses] : Ukuran File Output : 19122 bytes [SUkses] : Persentase Kompresi : 50.91259145167501% [INFO] : Komprimsi berhasil [INFO] : Banyak Simbol : 1385 [INFO] : Gambar berhasil disimpan sebagai /Users/raffifarras/SEMESTER_4/Strategi_Algoritma/Tucil2/Tucil2_13523073_13523095/test/output/gunungg.jpg [INFO] : GIF berhasil dibuat di /Users/raffifarras/SEMESTER_4/Strategi_Algoritma/Tucil2/Tucil2_13523073_13523095/test/output/gunungg.gif  [INFO] : Status akhir proses: - Komprimsi Gambar : Berhasil - Pembuatan GIF : Berhasil  [INFO] : Apakah Anda ingin memproses gambar lain? [1] Ya [2] Tidak (Keluar)</pre>
<b>Test Case #7 Pengaruh Threshold (Semua Parameter sama seperti TC#1, kecuali threshold)</b>	
Input	

Output	 <a href="#">GIF</a>
CLI	<pre>[INFO] : Wilujeng Sumping... [INFO] : Masukkan alamat ABSOLUT image ! [INPUT] : /Users/rafiffarras/SEMESTER_4/Strategi_Algoritma/Tucil2/Tucil2_13523073_13523095/test/input/hmif.jpg [INFO] : Pilih Error Measurement ! [1] Variance [2] Mean Absolute Deviation [3] Max Pixel Difference [4] Entropy [5] SSIM  [INPUT] : 1  [INFO] : Masukkan Threshold ! [INFO] : Threshold min adalah : 0 dan max adalah : 255 [INFO] : Rekomendasi Threshold dengan pertimbangan kualitas gambar dan kompresi: 1 - 5  [INPUT] : 10  [INFO] : Masukkan Minimum Block ! [INPUT] : 10  [INFO] : Masukkan Target Compression (0 - 1) ! [INPUT] : 0  [INFO] : Masukkan alamat (ABSOLUT) output hasil kompresi ! [INPUT] : /Users/rafiffarras/SEMESTER_4/Strategi_Algoritma/Tucil2/Tucil2_13523073_13523095/test/output/tres.jpg  [INFO] : Masukkan alamat (ABSOLUT) output GIF ! [INPUT] : /Users/rafiffarras/SEMESTER_4/Strategi_Algoritma/Tucil2/Tucil2_13523073_13523095/test/output/tres.gif  [INFO] : Memproses gambar dengan Quadtree...  [OUTPUT KOMPRESI] [SUKSES] : Waktu Eksekusi : 1099 milliseconds [SUKSES] : Ukuran File Input : 30326 bytes [SUKSES] : Ukuran File Output : 12845 bytes [SUKSES] : Persentase Kompresi : 57.64360614654092% [SUKSES] : Kedalaman Pohon : 5 [SUKSES] : Banyak Simpul : 1117 [SUKSES] : Gambar berhasil disimpan sebagai /Users/rafiffarras/SEMESTER_4/Strategi_Algoritma/Tucil2/Tucil2_13523073_13523095/test/output/tres.jpg  [SUKSES] : GIF berhasil dibuat di /Users/rafiffarras/SEMESTER_4/Strategi_Algoritma/Tucil2/Tucil2_13523073_13523095/test/output/tres.gif  [INFO] : Status akhir proses: - Kompresi Gambar : Berhasil - Pembuatan GIF : Berhasil  [INFO] : Apakah Anda ingin memproses gambar lain? [1] Ya [2] Tidak (Keluar)</pre>
<b>Test Case #8 Pengaruh Threshold (Semua Parameter sama seperti TC#1, kecuali minimum size block)</b>	
Input	

Output	 <a href="#">GIF</a>
CLI	<pre>[INFO] : Wilujeng Sumping... [INFO] : Masukkan alamat ABSOLUT image ! [INPUT] : /Users/raiffarras/SEMESTER_4/Strategi_Algoritma/Tucil2/Tucil2_13523073_13523095/test/input/hmif.jpg [INFO] : Pilih Error Measurement ! [1] Variance [2] Mean Absolute Deviation [3] Max Pixel Difference [4] Entropy [5] SSIM [INPUT] : 1  [INFO] : Masukkan Threshold ! [INFO] : Threshold min adalah : 0 dan max adalah : 255 [INFO] : Rekomendasi Threshold dengan pertimbangan kualitas gambar dan kompresi: 1 - 5 [INPUT] : 1  [INFO] : Masukkan Minimum Block ! [INPUT] : 40  [INFO] : Masukkan Target Compression (0 - 1) ! [INPUT] : 0  [INFO] : Masukkan alamat (ABSOLUT) output hasil kompresi ! [INPUT] : /Users/raiffarras/SEMESTER_4/Strategi_Algoritma/Tucil2/Tucil2_13523073_13523095/test/output/minsi.jpg [INFO] : Masukkan alamat (ABSOLUT) output GIF ! [INPUT] : /Users/raiffarras/SEMESTER_4/Strategi_Algoritma/Tucil2/Tucil2_13523073_13523095/test/output/minsi.gif [INFO] : Memproses gambar dengan Quadtree...</pre> <pre>[OUTPUT KOMPRESI] [SUKSES] : Waktu Eksekusi : 1123 milliseconds [SUKSES] : Ukuran File Input : 30326 bytes [SUKSES] : Ukuran File Output : 13240 bytes [SUKSES] : Persentase Kompresi : 56.341093451164014% [SUKSES] : Kedalaman Pohon : 5 [SUKSES] : Banyak Simpul : 1309 [SUKSES] : Gambar berhasil disimpan sebagai /Users/raiffarras/SEMESTER_4/Strategi_Algoritma/Tucil2/Tucil2_13523073_13523095/test/output/minsi.jpg [SUKSES] : GIF berhasil dibuat di /Users/raiffarras/SEMESTER_4/Strategi_Algoritma/Tucil2/Tucil2_13523073_13523095/test/output/minsi.gif  [INFO] : Status akhir proses: - Kompresi Gambar : Berhasil - Pembuatan GIF : Berhasil  [INFO] : Apakah Anda ingin memproses gambar lain? [1] Ya [2] Tidak (Keluar)</pre>

## BAB 5

### ANALISIS

Hasil kompresi dengan quadtree menunjukkan karakteristik yang sangat bergantung pada konten gambar yang diproses. Pada gambar dengan area homogen luas, algoritma mampu mencapai rasio kompresi tinggi tanpa degradasi kualitas yang signifikan, karena area tersebut dapat direpresentasikan dengan blok-blok besar. Sebaliknya, pada gambar dengan detail kompleks, rasio kompresi cenderung lebih rendah untuk mempertahankan kualitas visual yang baik. Dari test case yang membandingkan gambar pemandangan alam yang sama dengan variasi threshold dan minsize, terlihat bahwa penurunan threshold meningkatkan kedalaman pohon secara signifikan dan jumlah simpul, menghasilkan gambar yang mempertahankan lebih banyak detail namun dengan ukuran file yang lebih besar. Perubahan ukuran blok minimum pada gambar yang sama memperlihatkan peningkatan pada representasi detail halus terutama di area kompleks, namun dengan peningkatan jumlah simpul pohon yang berdampak signifikan pada ukuran hasil kompresi.

Pemilihan metode perhitungan error, nilai threshold, dan ukuran blok minimum secara signifikan memengaruhi hasil kompresi quadtree. Nilai threshold menentukan tingkat toleransi terhadap variasi dalam blok gambar, threshold rendah menghasilkan bagian lebih banyak dan mempertahankan detail lebih baik dengan tingkat kompresi lebih rendah, sementara threshold tinggi menghasilkan lebih sedikit bagian dengan kompresi lebih tinggi namun potensi kehilangan detail. Ukuran blok minimum membatasi kedalaman rekursi dan memengaruhi tingkat detail yang bisa dipertahankan, nilai kecil memungkinkan representasi detail halus tapi mengurangi rasio kompresi, sementara nilai besar menghasilkan kompresi lebih tinggi namun dapat menghilangkan detail penting.

Kompresi quadtree memiliki kelebihan dalam kemampuan adaptifnya terhadap konten gambar, implementasi yang relatif sederhana karena hanya melalui threshold maupun ukuran blok minimum. Kelebihan lainnya adalah kemampuan menghasilkan visualisasi proses kompresi. Namun, teknik ini juga memiliki keterbatasan seperti efek blok yang terlihat pada tingkat kompresi tinggi, ketidakefisienan pada gambar dengan tekstur kompleks merata, dan performansi yang tidak selalu optimal dibandingkan algoritma modern lainnya. Dari hasil perbandingan test case dengan variasi parameter pada gambar yang sama, terlihat bahwa penyesuaian threshold dan ukuran blok minimum memberikan kontrol signifikan untuk menyeimbangkan kualitas visual dan tingkat kompresi. Pengembangan yang dapat dilakukan meliputi implementasi paralelisasi untuk mempercepat perhitungan error, dan adaptasi threshold dinamis berdasarkan karakteristik lokal gambar.

## BAB 6

### IMPLEMENTASI BONUS

#### 6.1 SSIM

Implementasi SSIM pada kode ini menggunakan pendekatan "pseudo SSIM" dengan menghitung rata-rata dan variansi warna untuk setiap blok gambar pada tiga kanal warna (R, G, B). Mula-mula, sistem menghitung rata-rata nilai warna setiap kanal, lalu dilanjutkan dengan perhitungan variansi sebagai ukuran seberapa besar penyimpangan warna di blok tersebut. Formula SSIM yang digunakan memanfaatkan hasil rata-rata dan variansi untuk menilai keseragaman warna, dengan konstanta kecil (6.5) untuk mencegah pembagian nol. Nilai SSIM tiap kanal dihitung secara terpisah, kemudian dirata-ratakan untuk mendapatkan skor akhir SSIM blok tersebut. Semakin kecil variansi warna, semakin mendekati nilai 1.0, yang berarti kualitas blok sangat baik dan mendekati aslinya.

#### 6.2 Target Kompresi

Pada implementasi ini, target kompresi dicapai dengan metode binary search untuk mencari nilai threshold terbaik yang menghasilkan rasio kompresi mendekati target yang diinginkan. Prosesnya dimulai dengan mendefinisikan batas bawah dan atas threshold berdasarkan metode kompresi yang dipilih, lalu secara iteratif sistem mencoba threshold tengah, mengompresi gambar, dan menghitung persentase kompresinya. Jika hasilnya belum mencapai target dalam toleransi yang ditentukan, maka batas pencarian akan disesuaikan: jika kompresi terlalu tinggi, upper bound diturunkan, jika terlalu rendah, lower bound dinaikkan. Proses ini berlanjut hingga threshold optimal ditemukan atau selisih batas pencarian sudah lebih kecil dari toleransi, sehingga gambar dapat terkompresi sesuai target dengan tetap menjaga keseimbangan kualitas.

#### 6.3 GIF

Implementasi dilakukan dengan menampilkan proses kompresi menggunakan struktur quadtree dalam bentuk gambar bergerak (GIF). GIF ini dibentuk dengan menyusun sejumlah *frame* yang merepresentasikan hasil rekonstruksi gambar pada setiap kedalaman (*depth*) dari quadtree. Setiap *frame* berupa objek *BufferedImage* yang menunjukkan tahapan kompresi pada level tertentu. Seluruh *BufferedImage* tersebut kemudian digabungkan menjadi sebuah animasi menggunakan **AnimatedGifEncoder** —*encoder* eksternal yang dapat diunduh melalui pranala berikut: [<https://github.com/rtyley/animated-gif-lib-for-java>].

Proses implementasi terdiri atas tiga tahap utama:

- Pembuatan frame. fungsi `createProcessBufferedImages()` membentuk array *BufferedImage* sebanyak kedalaman quadtree ditambah satu, dengan tiap elemen merepresentasikan kondisi gambar pada level tertentu.

- Pengisian warna berdasarkan struktur quadtree. metode drawFrame() digunakan untuk mengisi warna piksel berdasarkan node quadtree. Warna dari setiap node diisi untuk *frame* pada kedalaman saat ini hingga frame akhir, sehingga informasi warna pada kedalaman sebelumnya dimiliki kedalaman selanjutnya. Proses ini dilakukan secara rekursif pada keempat kuadran node.
- Penyusunan file GIF. Setelah seluruh frame terbentuk, fungsi createGIF() digunakan untuk menyusun animasi GIF dari array BufferedImage tersebut, dengan menentukan waktu jeda antar frame dan pengulangan.

## LAMPIRAN

Tautan Repository: [https://github.com/AlfianHanifFY/Tucil2\\_13523073\\_13523095.git](https://github.com/AlfianHanifFY/Tucil2_13523073_13523095.git)

NO	Poin	Ya	Tidak
1	Program berhasil dikompilasi tanpa kesalahan	<input checked="" type="checkbox"/>	
2	Program berhasil dijalankan	<input checked="" type="checkbox"/>	
3	Program berhasil melakukan kompresi gambar sesuai parameter yang ditentukan	<input checked="" type="checkbox"/>	
4	Mengimplementasi seluruh metode perhitungan error wajib	<input checked="" type="checkbox"/>	
5	[Bonus] Implementasi persentase kompresi sebagai parameter tambahan	<input checked="" type="checkbox"/>	
6	[Bonus] Implementasi Structural Similarity Index (SSIM) sebagai metode pengukuran error	<input checked="" type="checkbox"/>	
7	[Bonus] Output berupa GIF Visualisasi Proses pembentukan Quadtree dalam Kompresi Gambar	<input checked="" type="checkbox"/>	
8	Program dan laporan dibuat (kelompok) sendiri	<input checked="" type="checkbox"/>	

- Pranala *library* yang digunakan untuk Bonus visualisasi GIF :  
<https://github.com/rtyley/animated-gif-lib-for-java>