

LAPORAN TUGAS KECIL 2 IF2211 STRATEGI ALGORITMA

Penerapan Divide and Conquer untuk Kompresi Gambar

dengan Metode Quadtree



disusun oleh:

Jessica Allen (13523059)

Naomi Risaka Sitorus (13523122)

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2025

DAFTAR ISI

DAFTAR ISI.....	2
BAB I DESKRIPSI MASALAH DAN ALGORITMA.....	3
1.1 Algoritma Divide and Conquer.....	3
1.2 Quadtree.....	4
1.3 Algoritma Kompresi Gambar dengan Pendekatan Divide and Conquer.....	5
BAB II IMPLEMENTASI ALGORITMA.....	8
2.1 Compression.....	8
2.2 Error Evaluation.....	9
2.3 GIF Sequence Writer (Bonus).....	10
2.4 GIF Exporter (Bonus).....	10
2.5 Program Utama.....	11
BAB III SOURCE CODE PROGRAM.....	12
3.1 Compression.java.....	12
3.2 ErrorEvaluation.java.....	14
3.3 GIFFormatter.java.....	15
3.4 GIFFooter.java.....	16
3.5 Main.java.....	18
BAB IV MASUKAN DAN KELUARAN PROGRAM.....	24
4.1 Test Case 1 (Variance).....	24
4.2 Test Case 2 (Mean Absolute Deviation dengan Gambar Solid Color).....	26
4.3 Test Case 3 (Max Pixel Difference dengan Gambar High Resolution 1736 KB).....	27
4.4 Test Case 4 (Entropy dan Gambar PNG).....	29
4.5 Test Case 5 (Variance dengan Gambar JPEG).....	31
4.6 Test Case 6 (Threshold Tinggi).....	32
4.7 Test Case 7 (Output Folder Tidak Ada/Tidak Sesuai).....	34
4.8 Test Case 8 (BONUS => SSIM).....	37
4.9 Test Case 9 (BONUS => Target Ratio dan Output Folder Tidak Ada).....	38
BAB V ANALISIS ALGORITMA.....	42
5.1 Deskripsi Umum Algoritma.....	42
5.2 Analisis Teoritis Kompleksitas dengan Master Theorem.....	42
5.3 Pengaruh Parameter dan Fitur Tambahan.....	43
DAFTAR PUSTAKA.....	45
LAMPIRAN.....	46

BAB I

DESKRIPSI MASALAH DAN ALGORITMA

1.1 Algoritma *Divide and Conquer*

Algoritma *divide and conquer* merupakan strategi penyelesaian masalah dengan cara memecah permasalahan menjadi bagian-bagian yang lebih kecil, menyelesaikan masing-masing upa-permasalahan tersebut, dan menggabungkannya menjadi solusi yang utuh. Strategi ini banyak digunakan dalam ilmu komputer karena mampu menyederhanakan masalah kompleks menjadi bagian-bagian yang lebih mudah dikelola. Umumnya, algoritma ini diimplementasikan menggunakan pendekatan rekursif agar proses pemecahan masalah dapat disajikan secara alami dan terstruktur. Algoritma ini terdiri dari tiga tahap utama, yaitu *divide*, *conquer*, and *combine*.

Divide adalah tahap memecah permasalahan menjadi beberapa upa-permasalahan lebih kecil yang lebih mudah diselesaikan. Permasalahan dalam algoritma ini umumnya dibagi menjadi dua bagian, dan idealnya setiap upa-permasalahan memiliki ukuran yang hampir sama. Tahap ini bertujuan untuk menyederhanakan masalah sehingga lebih mudah ditangani.

Tahap selanjutnya adalah *conquer*, yaitu menyelesaikan upa-permasalahan tersebut, baik secara langsung jika sudah berukuran kecil maupun secara rekursif jika masih berukuran besar. Jika upa-permasalahan yang dihasilkan masih cukup kecil atau sederhana (telah mencapai kondisi dasar atau *base case*), maka penyelesaiannya dapat dilakukan secara langsung tanpa perlu pemecahan lebih lanjut. Namun, jika ukurannya masih cukup besar, maka upa-permasalahan tersebut akan diselesaikan kembali dengan pendekatan *divide and conquer* secara rekursif.

Terakhir, solusi-solusi parsial dari upa-permasalahan digabungkan menjadi solusi utuh dari permasalahan awal pada tahap *combine*. Proses penggabungan ini tergantung pada jenis masalah yang dihadapi, misalnya, dalam algoritma *merge sort*, tahap ini melibatkan penggabungan dua array yang sudah terurut menjadi satu array terurut.

Algoritma *divide and conquer* memiliki berbagai kelebihan dalam penyelesaian masalah komputasi, salah satunya dari segi efisiensi waktu dalam menyelesaikan permasalahan dengan jumlah masukan besar dibandingkan algoritma *brute force*, karena permasalahan telah dipecah menjadi bagian-bagian yang lebih kecil. Penggunaan skema rekursif dalam membagi

permasalahan secara terstruktur juga dapat diaplikasikan secara luas dalam berbagai kasus. Selain itu, upa-permasalahan yang bersifat independen dalam metode ini dapat diselesaikan secara konkuren yang didukung oleh teknologi pemrosesan paralel pada komputer modern.

Akan tetapi, algoritma *divide and conquer* tidak selalu menghasilkan solusi optimal dalam setiap permasalahan. Hal ini disebabkan oleh sifat pemrosesan yang terstruktur dan deterministik, bukan acak atau adaptif terhadap konteks tertentu. Dalam kasus tertentu, salah satu dari tahap *divide* atau *combine* dapat menjadi kompleks demi menyederhanakan tahap lainnya. Selain itu, banyaknya upa-permasalahan yang harus diproses secara rekursif dapat menimbulkan beban memori yang tinggi, terutama jika tidak dioptimalkan dengan baik, misalnya dengan teknik *memoization* atau pembatasan kedalaman rekursi.

1.2 Quadtree

Quadtree merupakan suatu struktur data pohon yang digunakan untuk mempartisi ruang atau data dua dimensi secara rekursif hingga semua bagian memenuhi kriteria tertentu. Setiap simpul pada quadtree memiliki maksimal empat anak yang merepresentasikan empat kuadran dari suatu persegi. Kuadran-kuadran tersebut adalah kuadran kiri atas, kanan atas, kiri bawah, dan kanan bawah. Karena sifat rekursifnya, quadtree sangat cocok digunakan untuk menyimpan dan memproses informasi yang tersebar secara tidak merata dalam dua dimensi, seperti gambar digital, data geografis, atau peta.

Salah satu penerapan dari algoritma quadtree dapat ditemukan dalam bidang pengolahan citra digital, khususnya untuk kompresi gambar berbasis segmentasi spasial. Tujuan utama dari kompresi ini adalah untuk mengurangi ukuran file gambar tanpa mengorbankan terlalu banyak informasi visual yang penting. Dalam implementasinya pada kompresi gambar, quadtree membagi gambar ke dalam empat bagian secara rekursif hingga ukuran areanya cukup kecil, lalu mengevaluasi keseragaman warna setiap bagiannya.

Evaluasi keseragaman tersebut biasanya dilakukan dengan menggunakan sistem warna RGB (*Red*, *Green*, *Blue*). Dalam sistem ini, setiap piksel pada gambar terdiri dari tiga komponen warna, dan keseragaman suatu blok gambar ditentukan berdasarkan kesamaan distribusi nilai RGB antar piksel dalam blok tersebut. Berbagai metode pengukuran error dapat digunakan untuk menilai tingkat keseragaman ini, seperti variansi warna, *mean absolute deviation* (MAD), maksimum perbedaan antar piksel, atau bahkan metrik yang lebih kompleks seperti *entropy* dan *structural similarity index measure* (SSIM).

Jika suatu bagian dianggap homogen, maka bagian tersebut dianggap sebagai simpul daun dan tidak perlu dibagi lagi. Informasi posisi, ukuran, dan nilai rata-rata warna atau intensitas piksel pada area tersebut disimpan pada simpul internal pohon. Apabila keseragaman belum tercapai, area tersebut dibagi lagi menjadi empat bagian yang lebih kecil dan hal ini dilakukan secara terus-menerus secara rekursif hingga seluruh bagian gambar mencapai tingkat keseragaman yang telah ditentukan (*threshold*).

Pendekatan ini memungkinkan representasi gambar yang lebih efisien, dimana area yang homogen hanya direpresentasikan sekali, sementara area yang lebih kompleks mendapatkan lebih banyak pembagian untuk mempertahankan detail visualnya. Hasil akhirnya adalah sebuah representasi gambar dalam bentuk struktur pohon yang dapat digunakan untuk menyimpan gambar dalam ukuran file yang lebih kecil namun tetap mempertahankan kualitas visual yang cukup baik, terutama pada area yang penting secara visual. Selain itu, struktur quadtree juga mendukung proses seperti progressive rendering, di mana gambar dapat ditampilkan bertahap dari bentuk kasar ke bentuk yang lebih detail seiring waktu.

1.3 Algoritma Kompresi Gambar dengan Pendekatan *Divide and Conquer*

Kompresi gambar dapat dilakukan dengan pendekatan *divide and conquer* menggunakan metode *quadtree*, yang bekerja dengan cara membagi area gambar ke dalam bagian-bagian lebih kecil secara rekursif hingga seluruh bagian memenuhi kriteria keseragaman tertentu.

1. Program meminta gambar masukan untuk dikompresi dan parameter kompresi.

Pengguna diminta untuk memasukkan nama file gambar masukan berupa alamat absolutnya yang ingin dikompresi di CLI dengan syarat file gambar harus bertipe .jpg, .jpeg, atau .png. Program akan mencoba membaca gambar tersebut, dan jika berhasil, program kemudian akan lanjut meminta parameter kompresi seperti metode pengukuran error yang ingin digunakan (misalnya, variansi, MAD, atau lainnya), nilai ambang batas keseragaman (*threshold*) untuk menentukan homogenitas blok, ukuran blok minimum yang menandakan batas terkecil dari pembagian blok, serta target rasio kompresi (fitur bonus), dan nama file hasil kompresi. Nama file hasil kompresi juga harus berupa alamat absolut, yang merupakan alamat tempat gambar ingin disimpan. Untuk fitur target rasio kompresi, pengguna dapat memasukkan nilai nol (0) apabila tidak ingin mengaktifkannya. Semua masukan divalidasi oleh program untuk memastikan nilai-nilainya valid dan konsisten, dengan batas nilai threshold yang

bervariasi tergantung pada metode pengukuran yang dipilih, agar proses kompresi tetap akurat dan stabil.

2. Program menjalankan algoritma *divide and conquer* dengan metode quadtree untuk mengkompresi gambar.

Setelah pembacaan gambar masukan serta parameter kompresi berhasil, program menjalankan algoritma kompresi gambar dengan pendekatan *divide and conquer* menggunakan metode quadtree. Pada tahap ini, gambar input akan diproses dengan cara membagi setiap blok gambar ke dalam empat bagian secara rekursif, lalu dilanjutkan dengan melakukan evaluasi terhadap keseragaman warna piksel pada setiap blok menggunakan metode yang telah ditentukan pengguna. Metode-metode pengukuran error yang tersedia meliputi *variance*, *mean absolute deviation* (MAD), *maximum pixel difference*, dan *entropy*, yang masing-masing memiliki karakteristik dan sensitivitas berbeda terhadap variasi warna. Selain itu, tersedia juga metode tambahan berbasis persepsi visual, yaitu *structural similarity index* (SSIM), sebagai fitur bonus yang memberikan penilaian keseragaman berdasarkan kualitas visual yang biasanya dirasakan oleh manusia.

Apabila blok nilai keseragamannya sudah di bawah atau sama dengan *threshold* atau ukuran blok sudah mencapai ukuran blok minimum, maka blok tersebut akan dianggap homogen. Pada titik ini, warna blok akan dinormalisasi, yang artinya semua piksel dalam blok akan diubah menjadi warna rata-rata RGB dari blok tersebut. Namun, jika blok belum homogen dan ukurannya masih melebihi batas minimum dari yang diperlukan, maka blok akan dibagi lagi menjadi empat sub-blok, dan proses ini terus dilanjutkan secara rekursif. Hasil dari proses ini adalah sebuah struktur pohon *quadtree*, di mana simpul daun menyimpan informasi posisi, ukuran, dan warna blok akhir. Setelah seluruh blok dianalisis dan dibagi sesuai kebutuhan, simpul-simpul daun digabungkan kembali untuk menyusun gambar hasil kompresi secara utuh. Gambar hasil ini kemudian disimpan di alamat keluaran sesuai input pengguna.

Apabila pengguna menggunakan fitur target rasio kompresi yang merupakan spesifikasi bonus pada tugas ini, program akan menyesuaikan nilai *threshold* secara otomatis agar mendapatkan rasio kompresi yang diinginkan dengan toleransi kesalahan tertentu. Penyesuaian tersebut dilakukan dengan metode pencarian biner (*binary search*) yang menciptakan gambar hasil kompresi sementara secara iteratif hingga memperoleh gambar yang rasio kompresinya mendekati target. Metode

pencarian biner tersebut juga bersifat *divide and conquer* karena menyesuaikan *threshold* kompresi gambar ke dua kemungkinan arah yang terpisah, yaitu menaikkan atau menurunkan *threshold* dari nilai yang digunakan pada percobaan saat ini.

3. Program menampilkan informasi kompresi gambar serta gambar hasil kompresi pada alamat yang sudah ditentukan.

Jika gambar berhasil dikompresi, program akan menampilkan berbagai informasi kompresi gambar di CLI (*command line interface*). Informasi ini meliputi waktu eksekusi program, ukuran file gambar awal, ukuran file gambar hasil kompresi, persentase pengurangan ukuran (rasio kompresi), kedalaman pohon *quadtree*, dan jumlah simpul (*nodes*) pada pohon, yang merepresentasikan kompleksitas segmentasi gambar. Selain itu, pengguna juga dapat melihat gambar hasil kompresi yang telah disimpan pada alamat absolut keluaran yang telah ditentukan. Jika proses kompresi gagal karena kesalahan dalam file atau parameter, program akan menampilkan pesan “Gambar gagal dikompresi” beserta rincian pesan kesalahan (*error message*) yang membantu pengguna mengidentifikasi penyebab kegagalan.

BAB II

IMPLEMENTASI ALGORITMA

2.1 Compression

Berisi *method* untuk melakukan kompresi gambar dengan algoritma *quadtree* yang mengimplementasikan pendekatan *divide and conquer*. Selain itu, kelas ini mendukung penyesuaian *threshold* secara otomatis untuk mencapai target rasio kompresi gambar yang merupakan fitur BONUS. Terdapat dalam file Compression.java yang memiliki kelas Compression.

<i>Method</i>	Deskripsi
public static BufferedImage compressImage (BufferedImage img, int method, double threshold, int minSize)	Mengembalikan gambar hasil kompresi gambar menggunakan metode quadtree
public static BufferedImage compressWithTargetRatio (BufferedImage original, int method, double ogThreshold, double tolerance, int minBlockSize, long ogSize, double targetRatio, String outputFormat)	Mengembalikan gambar hasil kompresi gambar dengan metode quadtree penyesuaian threshold secara otomatis agar mendekati rasio kompresi target
private static void quadtreeCompress (BufferedImage original, BufferedImage output, int x, int y, int sizeX, int sizeY, int depth, int method, double threshold, int minSize)	Mengkompresi gambar dengan metode quadtree menjadi 4 bagian
public static boolean isHomogenous (BufferedImage img, int x, int y, int w, int h, int method, double threshold)	Mengembalikan <i>true</i> jika blok sudah dianggap homogen berdasarkan metode dan threshold yang digunakan, dan <i>false</i> jika belum
public static void fillBlock (BufferedImage img, int x, int y, int w, int h, int color)	Mengisi blok dengan satu warna yang merupakan warna hasil rata-rata

2.2 Error Evaluation

Berisi *method* untuk mengukur error pada blok gambar dengan metode sesuai pilihan. Terdapat lima metode pengukuran error yang tersedia, yaitu *variance*, *mean absolute deviation* (MAD), *maximum pixel difference*, *entropy*, dan *structural similarity index measure* (SSIM) yang merupakan fitur BONUS. Terdapat dalam file ErrorEvaluation.java yang memiliki kelas ErrorEvaluation.

<i>Method</i>	Deskripsi
public static int getAvgColor (BufferedImage img, int x, int y, int w, int h)	Mengembalikan nilai RGB warna rata-rata dari blok gambar
public static double calculateVariance (BufferedImage img, int x, int y, int w, int h)	Mengembalikan nilai variansi warna dalam blok gambar
public static double calculateMAD (BufferedImage img, int x, int y, int w, int h)	Mengembalikan nilai <i>mean absolute deviation</i> (MAD) dari blok gambar terhadap warna rata-rata
public static double calculateMaxPixelDifference (BufferedImage img, int x, int y, int w, int h)	Mengembalikan selisih nilai piksel maksimum terhadap warna rata-rata dari blok gambar
public static double calculateEntropy (BufferedImage img, int x, int y, int w, int h)	Mengembalikan nilai entropi blok berdasarkan nilai <i>grayscale</i> -nya
public static double calculateSSIM (BufferedImage original, int x, int y, int w, int h)	Mengembalikan nilai <i>structural similarity index</i> terhadap warna rata-rata dari blok gambar
private static double calculateSSIMChannel (BufferedImage img, int refValue, int x, int y, int w, int h, char channel) {	Mengembalikan nilai SSIM pada salah satu <i>channel</i> warna RGB, merupakan fungsi bantu dari <i>calculateSSIM()</i>

2.3 GIF Sequence Writer (Bonus)

Merupakan utility class yang dipakai untuk membuat GIF animasi dari sekumpulan BufferedImage. Terdapat dalam file `GIFSequenceWriter.java` yang memiliki kelas `GIFSequenceWriter`.

<i>Constructor</i>	Deskripsi
<pre>public GIFSequenceWriter(ImageOutputStream outputStream, int imageType, int timeBetweenFramesMS, boolean loopContinuously)</pre>	Menyiapkan <i>GIF sequence writer</i> untuk menulis GIF

<i>Method</i>	Deskripsi
<pre>public void writeToSequence(RenderedImage img)</pre>	Menulis frame gambar ke dalam <i>GIF sequence</i>
<pre>public void close()</pre>	Mengakhiri penulisan <i>GIF sequence</i>
<pre>private static ImageWriter getWriter()</pre>	Mendapatkan <i>ImageWriter</i> untuk GIF
<pre>private static IIOMetadataNode getNode(IIOMetadataNode rootNode, String nodeName)</pre>	Mendapatkan atau menambahkan <i>node</i> metadata berdasarkan namanya

2.4 GIF Exporter (Bonus)

Berisi *method* untuk membuat GIF animasi yang menampilkan proses bertahap kompresi quadtree yang merupakan fitur BONUS. Terdapat dalam file `GIFExporter.java` yang memiliki kelas `GIFExporter`.

<i>Method</i>	Deskripsi
<pre>public static void exportGIFPerDepth(BufferedImage original, int method, double threshold, int minBlockSize, String gifPath, int maxDepth, int width, int height)</pre>	Mengekspor gambar ke format GIF dengan kompresi quadtree

<pre>private static void quadtreeCompressGIF(BufferedImage original, BufferedImage output, int x, int y, int sizeX, int sizeY, int depth, int method, double threshold, int minSize, int maxAllowedDepth)</pre>	<p>Menciptakan GIF dari proses kompresi gambar dengan metode quadtree</p>
--	---

2.5 Program Utama

Berisi *method* untuk menjalankan program “Kompresi Gambar dengan Metode Quadtree” secara keseluruhan dan memproses file masukan serta keluaran, baik dalam bentuk gambar maupun GIF (BONUS). Terdapat dalam file Main.java yang memiliki class Main.

Method	Deskripsi
public static void main (String[] args)	Menjalankan program secara keseluruhan dengan memanggil method untuk membaca gambar masukan, mengkompresi gambar, dan menyimpan hasil kompresi, baik dalam bentuk gambar maupun GIF (bonus)
public static void printHeader ()	Mencetak <i>header ASCII art</i>

BAB III

SOURCE CODE PROGRAM

3.1 Compression.java

```
1  import java.awt.image.BufferedImage;
2  import java.io.File;
3  import javax.imageio.ImageIO;
4
5  public class Compression {
6      public static int nodeAmt = 0;
7      public static int maxDepth = 0;
8
9      // Fungsi utama untuk mengkompresi gambar menggunakan metode quadtree
10     public static BufferedImage compressImage(BufferedImage img, int method, double threshold, int minSize) {
11         int width = img.getWidth();
12         int height = img.getHeight();
13         BufferedImage result = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
14         quadtreeCompress(img, result, x:0, y:0, width, height, depth:0, method, threshold, minSize);
15         return result;
16     }
17
18     // Kompresi dengan penyesuaian threshold secara dinamis agar mendekati rasio kompresi target
19     public static BufferedImage compressWithTargetRatio(
20         BufferedImage original, int method, double ogThreshold, double tolerance, int minBlockSize, long ogSize,
21         double targetRatio, String outputFormat)
22     {
23         double low = method == 5 ? 0.0 : 0.0;
24         double high = method == 5 ? 1.0 : Math.max(ogThreshold * 5, b:500);
25         double bestThreshold = ogThreshold;
26         BufferedImage bestImage = null;
27
28         double bestRatioDiff = Double.MAX_VALUE;
29         int maxIter = 30; // Batas iterasi untuk mencegah loop tak berujung
30
31         for (int i = 0; i < maxIter; i++) {
32             double mid = (low + high) / 2;
33             BufferedImage tempImage = compressImage(original, method, mid, minBlockSize);
34
35             try {
36                 // Simpan hasil kompres sementara ke file temp
37                 File tempFile = File.createTempFile(prefix:"temp_compressed", "." + outputFormat);
38                 ImageIO.write(tempImage, outputFormat, tempFile);
39                 long tempSize = tempFile.length();
40                 tempFile.delete();
41
42                 double achievedRatio = 1.0 - (double) tempSize / ogSize;
43                 double ratioDiff = Math.abs(achievedRatio - targetRatio);
44
45                 // Simpan hasil terbaik sejauh ini
46                 if (ratioDiff < bestRatioDiff) {
47                     bestRatioDiff = ratioDiff;
48                     bestThreshold = mid;
49                     bestImage = tempImage;
50                 }
51
52                 // Sesuaikan batas bawah dan atas binary search
53                 if (achievedRatio < targetRatio) {
54                     if (method == 5) high = mid;
55                     else low = mid;
56                 } else {
57                     if (method == 5) low = mid;
58                     else high = mid;
59                 }
60             }
61         }
62     }
63 }
```

```

61     } catch (Exception e) {
62         e.printStackTrace();
63     }
64 }
65
66 // Cetak threshold hasil penyesuaian
67 System.out.println();
68 System.out.printf(format:"Threshold disesuaikan ke: %.4f\n", bestThreshold);
69 return bestImage;
70 }
71
72 // Fungsi rekursif utama untuk kompresi quadtree
73 private static void quadtreeCompress(
74     BufferedImage original, BufferedImage output, int x, int y, int sizeX, int sizeY, int depth,
75     int method, double threshold, int minSize
76 ) {
77     nodeAmt++;
78     maxDepth = Math.max(maxDepth, depth);
79
80     // Jika blok cukup kecil atau homogen, isikan dengan warna rata-rata
81     if (sizeX <= minSize || sizeY <= minSize || isHomogenous(original, x, y, sizeX, sizeY, method, threshold)) {
82         int avgColor = ErrorEvaluation.getAvgColor(original, x, y, sizeX, sizeY);
83         fillBlock(output, x, y, sizeX, sizeY, avgColor);
84     } else {
85         // Bagi blok jadi 4 kuadran dan kompres masing-masing
86         int halfX = sizeX / 2;
87         int halfY = sizeY / 2;
88         int remX = sizeX - halfX;
89         int remY = sizeY - halfY;

```

```

91         quadtreeCompress(original, output, x, y, halfX, halfY, depth + 1, method, threshold, minSize);
92         quadtreeCompress(original, output, x + halfX, y, remX, halfY, depth + 1, method, threshold, minSize);
93         quadtreeCompress(original, output, x, y + halfY, halfX, remY, depth + 1, method, threshold, minSize);
94         quadtreeCompress(original, output, x + halfX, y + halfY, remX, remY, depth + 1, method, threshold, minSize);
95     }
96 }
97
98 // Mengecek apakah blok homogen berdasarkan metode dan threshold tertentu
99 public static boolean isHomogenous(BufferedImage img, int x, int y, int w, int h, int method, double threshold) {
100     switch (method) {
101         case 1: return ErrorEvaluation.calculateVariance(img, x, y, w, h) < threshold;
102         case 2: return ErrorEvaluation.calculateMAD(img, x, y, w, h) < threshold;
103         case 3: return ErrorEvaluation.calculateMaxPixelDifference(img, x, y, w, h) < threshold;
104         case 4: return ErrorEvaluation.calculateEntropy(img, x, y, w, h) < threshold;
105         case 5: return ErrorEvaluation.calculateSSIM(img, x, y, w, h) > threshold; // SSIM lebih tinggi = lebih mirip
106         default: return false;
107     }
108 }
109
110 // Mengisi blok dengan satu warna (hasil rata-rata)
111 public static void fillBlock(BufferedImage img, int x, int y, int w, int h, int color) {
112     for (int i = x; i < x + w; i++) {
113         for (int j = y; j < y + h; j++) {
114             if (i >= 0 && i < img.getWidth() && j >= 0 && j < img.getHeight()) {
115                 img.setRGB(i, j, color);
116             }
117         }
118     }
119 }
120 }
121

```

3.2 ErrorEvaluation.java

```
1 import java.awt.Color;
2 import java.awt.image.BufferedImage;
3
4 public class ErrorEvaluation {
5
6     // Menghitung warna rata-rata dari blok (x, y, w, h) dan mengembalikan nilai RGB-nya
7     public static int getAvgColor(BufferedImage img, int x, int y, int w, int h) {
8         int r = 0, g = 0, b = 0, count = 0;
9         for (int i = x; i < x + w; i++) {
10             for (int j = y; j < y + h; j++) {
11                 Color c = new Color(img.getRGB(i, j));
12                 r += c.getRed();
13                 g += c.getGreen();
14                 b += c.getBlue();
15                 count++;
16             }
17         }
18         return new Color(r / count, g / count, b / count).getRGB();
19     }
20
21     // Menghitung variansi warna dalam blok (x, y, w, h)
22     public static double calculateVariance(BufferedImage img, int x, int y, int w, int h) {
23         long sumR = 0, sumG = 0, sumB = 0;
24         int count = 0;
25
26         // Hitung jumlah total setiap kanal warna
27         for (int i = x; i < x + w; i++) {
28             for (int j = y; j < y + h; j++) {
29                 Color c = new Color(img.getRGB(i, j));
30                 sumR += c.getRed();
31                 sumG += c.getGreen();
32                 sumB += c.getBlue();
33                 count++;
34             }
35         }
36
37         double avgR = sumR / (double) count;
38         double avgG = sumG / (double) count;
39         double avgB = sumB / (double) count;
40
41         // Hitung variansi total
42         double variance = 0.0;
43         for (int i = x; i < x + w; i++) {
44             for (int j = y; j < y + h; j++) {
45                 Color c = new Color(img.getRGB(i, j));
46                 variance += Math.pow(c.getRed() - avgR, 2);
47                 variance += Math.pow(c.getGreen() - avgG, 2);
48                 variance += Math.pow(c.getBlue() - avgB, 2);
49             }
50         }
51         return variance / count;
52     }
}
```

3.3 GIFSequenceWriter.java

```
1 import java.awt.image.RenderedImage;
2 import java.io.IOException;
3 import java.util.Iterator;
4 import javax.imageio.ImageIO;
5 import javax.imageio.ImageTypeSpecifier;
6 import javax.imageio.ImageWriteParam;
7 import javax.imageio.ImageWriter;
8 import javax.imageio.metadata.IIOMetadata;
9 import javax.imageio.metadata.IIOMetadataNode;
10 import javax.imageio.stream.ImageOutputStream;
11
12 public class GIFSequenceWriter {
13     protected ImageWriter gifWriter;
14     protected ImageWriteParam imageWriteParam;
15     protected IIOMetadata imageMetaData;
16
17     // Menyiapkan GIF sequence writer untuk menulis GIF
18     public GIFSequenceWriter(ImageOutputStream outputStream, int imageType, int timeBetweenFramesMS, boolean loopContinuously) throws IOException {
19         gifWriter = getWriter();
20         imageWriteParam = gifWriter.getDefaltultWriteParam();
21         ImageTypeSpecifier imageTypeSpecifier = ImageTypeSpecifier.createFromBufferedImageType(imageType);
22
23         imageMetaData = gifWriter.getDefaltultImageMetadata(imageTypeSpecifier, imageWriteParam);
24
25         String metaFormatName = imageMetaData.getNativeMetadataFormatName();
26         IIOMetadataNode root = (IIOMetadataNode) imageMetaData.getAsTree(metaFormatName);
27
28         // Atur node metadata untuk GIF
29         IIOMetadataNode graphicsControlExtensionNode = getNode(root, nodeName:"GraphicControlExtension");
30         graphicsControlExtensionNode.setAttribute(name:"disposalMethod", value:"none");
31         graphicsControlExtensionNode.setAttribute(name:"userInputFlag", value:"FALSE");
32         graphicsControlExtensionNode.setAttribute(name:"transparentColorFlag", value:"FALSE");
33         graphicsControlExtensionNode.setAttribute(name:"delayTime", Integer.toString(timeBetweenFramesMS / 10));
34         graphicsControlExtensionNode.setAttribute(name:"transparentColorIndex", value:"0");
35
36         IIOMetadataNode appExtensionsNode = getNode(root, nodeName:"ApplicationExtensions");
37         IIOMetadataNode appExtension = new IIOMetadataNode(nodeName:"ApplicationExtension");
38
39         appExtension.setAttribute(name:"applicationID", value:"NETSCAPE");
40         appExtension.setAttribute(name:"authenticationCode", value:"2.0");
41
42         int loop = loopContinuously ? 0 : 1;
43         appExtension.setUserObject(new byte[]{0x1, (byte) (loop & 0xFF), (byte) ((loop >> 8) & 0xFF)});
44         appExtensionsNode.appendChild(appExtension);
45         root.appendChild(appExtensionsNode);
46
47         imageMetaData.setFromTree(metaFormatName, root);
48         gifWriter.setOutput(outputStream);
49         gifWriter.prepareWriteSequence(streamMetadata:null);
50     }
51
52     // Menulis frame gambar ke dalam GIF sequence
53     public void writeToSequence(RenderedImage img) throws IOException {
54         gifWriter.writeToSequence(new javax.imageio.IIOMage(img, thumbnails:null, imageMetaData), imageWriteParam);
55     }
56
57 }
```

```

57     // Mengakhiri penulisan GIF sequence
58     public void close() throws IOException {
59         gifWriter.endWriteSequence();
60     }
61
62     // Mendapatkan ImageWriter untuk GIF
63     private static ImageWriter getWriter() throws IOException {
64         Iterator<ImageWriter> writers = ImageIO.getImageWritersBySuffix(fileSuffix:"gif");
65         if (!writers.hasNext()) {
66             throw new IOException(message:"No GIF Image Writers Exist");
67         }
68         return writers.next();
69     }
70
71     // Mendapatkan atau menambahkan node metadata berdasarkan namanya
72     private static IIOMetadataNode getNode(IIOMetadataNode rootNode, String nodeName) {
73         for (int i = 0; i < rootNode.getLength(); i++) {
74             if (rootNode.item(i).getNodeName().equalsIgnoreCase(nodeName)) {
75                 return (IIOMetadataNode) rootNode.item(i);
76             }
77         }
78         IIOMetadataNode node = new IIOMetadataNode(nodeName);
79         rootNode.appendChild(node);
80         return node;
81     }
82 }
```

3.4 GIFExporter.java

```

1  import java.awt.Color;
2  import java.awt.Graphics2D;
3  import java.awt.image.BufferedImage;
4  import java.io.File;
5  import java.io.IOException;
6  import java.util.ArrayList;
7  import javax.imageio.stream.FileImageOutputStream;
8
9  public class GIFExporter {
10     // Mengeksport gambar ke format GIF dengan kompresi quadtree
11     public static void exportGIFPerDepth(
12         BufferedImage original,
13         int method,
14         double threshold,
15         int minBlockSize,
16         String gifPath,
17         int maxDepth,
18         int width,
19         int height
20     ) {
21         try {
22             ArrayList<BufferedImage> gifFrames = new ArrayList<>();
23             int backgroundColor = ErrorEvaluation.getAvgColor(original, x:0, y:0, width, height);
24
25             for (int targetDepth = 0; targetDepth <= maxDepth; targetDepth++) {
26                 BufferedImage frame = new BufferedImage(original.getWidth(), original.getHeight(), original.getType());
27
28                 Graphics2D g2d = frame.createGraphics();
29                 g2d.setColor(new Color(backgroundColor));
30                 g2d.fillRect(x:0, y:0, width, height);
31                 g2d.dispose();
32             }
33         } catch (IOException e) {
34             e.printStackTrace();
35         }
36     }
37 }
```

```

33     quadtreeCompressGIF(original, frame, x:0, y:0, width, height, depth:0, method, threshold, minBlockSize, targetDepth);
34     gifFrames.add(frame);
35   }
36
37   FileImageOutputStream outputGIF = new FileImageOutputStream(new File(gifPath));
38   GIFSequenceWriter writer = new GIFSequenceWriter(outputGIF, BufferedImage.TYPE_INT_RGB, timeBetweenFramesMS:500, loopContinuously:true);
39
40   for (BufferedImage frame : gifFrames) {
41     writer.writeToSequence(frame);
42   }
43
44   writer.close();
45   outputGIF.close();
46
47 } catch (IOException e) {
48   System.out.println("Gagal menyimpan GIF: " + e.getMessage());
49 }
50 }
51
// Fungsi rekursif utama untuk kompresi quadtree versi GIF
52 private static void quadtreeCompressGIF(
53   BufferedImage original,
54   BufferedImage output,
55   int x, int y,
56   int sizeX, int sizeY,
57   int depth,
58   int method,
59   double threshold,
60   int minSize,
61   int maxAllowedDepth
62 ) {
63
64   if (
65     sizeX <= minSize || sizeY <= minSize || depth >= maxAllowedDepth ||
66     Compression.isHomogenous(original, x, y, sizeX, sizeY, method, threshold)
67   ) {
68     int avgColorInt = ErrorEvaluation.getAvgColor(original, x, y, sizeX, sizeY);
69     Compression.fillBlock(output, x, y, sizeX, sizeY, avgColorInt);
70   } else {
71     int halfX = sizeX / 2;
72     int halfY = sizeY / 2;
73     int remX = sizeX - halfX;
74     int remY = sizeY - halfY;
75
76     quadtreeCompressGIF(original, output, x, y, halfX, halfY, depth + 1, method, threshold, minSize, maxAllowedDepth);
77     quadtreeCompressGIF(original, output, x + halfX, y, remX, halfY, depth + 1, method, threshold, minSize, maxAllowedDepth);
78     quadtreeCompressGIF(original, output, x, y + halfY, halfX, remY, depth + 1, method, threshold, minSize, maxAllowedDepth);
79     quadtreeCompressGIF(original, output, x + halfX, y + halfY, remX, remY, depth + 1, method, threshold, minSize, maxAllowedDepth);
80   }
81 }
82 }
83

```

3.5 Main.java

```
1 import java.awt.image.BufferedImage;
2 import java.io.File;
3 import java.util.Scanner;
4 import javax.imageio.ImageIO;
5
6 // Main class untuk menjalankan program kompresi gambar berbasis Quadtree
7 public class Main {
8     Run | Debug | Run main | Debug main
9     public static void main(String[] args) {
10         Scanner scanner = new Scanner(System.in);
11
12         // Header awal
13         System.out.println("=====");
14         printHeader();
15         System.out.println("=====");
16
17         File inputFile = null;
18         String inputPath = "";
19
20         // Input nama file gambar dan validasi
21         while (true) {
22             System.out.println();
23             System.out.print("Masukkan nama file gambar (diakhiri dengan .jpg, .jpeg, atau .png): ");
24             inputPath = scanner.nextLine().trim();
25
26             // Validasi ekstensi file
27             if (!((inputPath.toLowerCase().endsWith(suffix:".jpg") || inputPath.toLowerCase().endsWith(suffix:".jpeg") || inputPath.toLowerCase().endsWith(suffix:".png")))) {
28                 System.out.println("Format file tidak didukung. Harus diakhiri dengan .jpg, .jpeg, atau .png.");
29                 continue;
30             }
31
32             inputFile = new File(inputPath);
33             if (!inputFile.exists()) {
34                 System.out.println("File input tidak ditemukan: " + inputPath);
35                 continue;
36             }
37             break;
38
39             // Pemilihan metode pengukuran error
40             System.out.println();
41             System.out.println("=====");
42             System.out.println("METODE PENGUKURAN ERROR");
43             System.out.println("=====");
44             System.out.println("1. Variance");
45             System.out.println("2. Mean Absolute Deviation (MAD)");
46             System.out.println("3. Max Pixel Difference");
47             System.out.println("4. Entropy");
48             System.out.println("5. Structural Similarity Index (SSIM)");
49             System.out.println("=====");
50
51             int method = 0;
52             while (true) {
53                 System.out.print("Masukkan metode yang ingin digunakan: ");
54                 String methodInput = scanner.nextLine().trim();
55
56                 if (methodInput.isEmpty()) {
57                     System.out.println("Input tidak boleh kosong.");
58                     continue;
59                 }
60
61                 try {
62                     method = Integer.parseInt(methodInput);
63                     if (method >= 1 && method <= 5) break;
64                     else System.out.println("Masukkan angka antara 1 sampai 5.");
65                 } catch (NumberFormatException e) {
66                     System.out.println("Input harus berupa angka.");
67                 }
68             }
69
70             // Proses kompresi
71             System.out.println("=====");
72             System.out.println("KOMPRESI GAMBAR");
73             System.out.println("=====");
74             compressImage(inputPath, outputPath, quality);
75             System.out.println("=====");
76             System.out.println("=====");
77         }
78
79         // Footer akhir
80         System.out.println("=====");
81         System.out.println("TERIMA KASIH");
82         System.out.println("=====");
83     }
84 }
```

```

70     // Penjelasan tambahan untuk metode SSIM
71     if (method == 5) {
72         System.out.println();
73         System.out.println(x:"=====");
74         System.out.println(x:"Anda memilih metode SSIM.");
75         System.out.println(x:"Threshold SSIM berkisar antara 0 sampai 1.");
76         System.out.println(x:"Semakin tinggi threshold, maka blok akan digabung jika sangat mirip.");
77         System.out.println(x:"Semakin rendah threshold, maka blok digabung walau kurang mirip.");
78         System.out.println(x:"=====");
79         System.out.println();
80     }
81
82     // Input nilai threshold dengan validasi berdasarkan metode
83     double threshold = 0;
84     while (true) {
85         System.out.print(s:"Masukkan nilai ambang batas (threshold): ");
86         String thresholdInput = scanner.nextLine().trim();
87
88         if (thresholdInput.isEmpty()) {
89             System.out.println(x:"Input tidak boleh kosong.");
90             System.out.println();
91             continue;
92         }
93
94         try {
95             threshold = Double.parseDouble(thresholdInput);
96         } catch (NumberFormatException e) {
97             System.out.println(x:"Input harus berupa angka.");
98             System.out.println();
99             continue;
100        }
101
102        boolean isValid = true;
103        switch (method) {
104            case 1:
105                if (threshold < 0) {
106                    System.out.println(x:"Threshold metode ini tidak boleh bernilai negatif.");
107                    System.out.println();
108                    isValid = false;
109                }
110                break;
111            case 2:
112            case 3:
113                if (threshold < 0 || threshold > 255) {
114                    System.out.println(x:"Threshold metode ini harus bernilai antara 0 sampai 255.");
115                    System.out.println();
116                    isValid = false;
117                }
118                break;
119            case 4:
120                if (threshold < 0 || threshold > 8) {
121                    System.out.println(x:"Threshold metode ini harus bernilai antara 0 sampai 8.");
122                    System.out.println();
123                    isValid = false;
124                }
125                break;
126            case 5:
127                if (threshold < 0 || threshold > 1) {
128                    System.out.println(x:"Threshold metode ini harus bernilai antara 0 sampai 1.");
129                    System.out.println();
130                    isValid = false;
131                }
132                break;

```

```

130             isValid = false;
131         }
132     }
133 }
134 if (isValid) break;
135 }

// Input ukuran blok minimum
138 int minBlockSize = 0;
139 while (true) {
140     System.out.print(s:"Masukkan ukuran blok minimum: ");
141     String minBlockInput = scanner.nextLine().trim();

143 if (minBlockInput.isEmpty()) {
144     System.out.println(x:"Input tidak boleh kosong.");
145     System.out.println();
146     continue;
147 }
148 try {
149     minBlockSize = Integer.parseInt(minBlockInput);
150     if (minBlockSize > 0) break;
151     else {
152         System.out.println(x:"Ukuran blok minimum harus lebih besar dari 0.");
153         System.out.println();
154     }
155 } catch (NumberFormatException e) {
156     System.out.println(x:"Input harus berupa angka.");
157     System.out.println();
158 }
159 }

162 // Input target rasio kompresi (opsional)
163 double targetRatio = 0;
164 double tolerance = 0;
165
166 while (true) {
167     System.out.print(s:"Masukkan target rasio kompresi (0 jika tidak ingin menggunakan fitur ini): ");
168     String ratioInput = scanner.nextLine().trim();
169
170 if (ratioInput.isEmpty()) {
171     System.out.println(x:"Input tidak boleh kosong.\n");
172     continue;
173 }
174 try {
175     targetRatio = Double.parseDouble(ratioInput);
176     if (targetRatio < 0 || targetRatio > 1) {
177         System.out.println(x:"Target rasio kompresi harus berada antara 0 dan 1.\n");
178     } else if (targetRatio == 0) {
179         break;
180     } else {
181         tolerance = 0.003;
182         break;
183     }
184 } catch (NumberFormatException e) {
185     System.out.println(x:"Input harus berupa angka.");
186     System.out.println();
187 }
188 }

```

```

191 // Input nama file output dan validasi format
192 String outputPath;
193 while (true) {
194     System.out.print(s:"Masukkan nama file hasil kompresi (diakhiri dengan .jpg, .jpeg, atau .png): ");
195     outputPath = scanner.nextLine().trim();
196
197     if (outputPath.isEmpty()) {
198         System.out.println(x:"Input tidak boleh kosong.\n");
199         continue;
200     }
201
202     if (!(outputPath.toLowerCase().endsWith(suffix:".jpg") || outputPath.toLowerCase().endsWith(suffix:".jpeg") || outputPath.toLowerCase().endsWith(suffix:".png"))) {
203         System.out.println(x:"Format file tidak didukung. Harus berakhiran dengan .jpg, .jpeg, atau .png.\n");
204         continue;
205     }
206
207     File outputFile = new File(outputPath);
208     File parentDir = outputFile.getParentFile();
209
210     if (parentDir != null && !parentDir.exists()) {
211         boolean retryOutput = false;
212         while (true) {
213             System.out.print(s:"Folder tujuan tidak ada. Apakah Anda ingin membuatnya? (ya/tidak): ");
214             String confirm = scanner.nextLine().trim().toLowerCase();
215
216             if (confirm.isEmpty()) {
217                 System.out.println(x:"Input tidak boleh kosong.\n");
218                 continue;
219             }
220
221             if (confirm.equals(anObject:"ya") || confirm.equals(anObject:"y")) {
222                 try {
223                     if (!parentDir.mkdirs()) {
224                         System.out.println(x:"Gagal membuat folder tujuan.");
225                         System.exit(status:1);
226                     }
227                     break;
228                 } catch (SecurityException e) {
229
230                     } catch (SecurityException e) {
231                         System.out.println(x:"Tidak memiliki izin untuk membuat folder.");
232                         System.exit(status:1);
233                     }
234                 } else if (confirm.equals(anObject:"tidak") || confirm.equals(anObject:"t")) {
235                     System.out.println(x:"Silakan masukkan ulang nama file output.\n");
236                     retryOutput = true;
237                     break;
238                 } else {
239                     System.out.println(x:"Pilihan tidak valid. Silakan masukkan 'ya' atau 'tidak'.\n");
240                 }
241
242                 if (retryOutput) continue;
243             }
244
245             break;
246
247             // Input pilihan untuk menyimpan GIF
248             boolean exportGIF = false;
249             String gifPath = "";
250             while (true) {
251                 System.out.print(s:"Apakah Anda ingin menyimpan GIF hasil kompresi? (ya/tidak): ");
252                 String gifOption = scanner.nextLine().trim().toLowerCase();
253
254                 if (gifOption.isEmpty()) {
255                     System.out.println(x:"Input tidak boleh kosong.");
256                     System.out.println();
257                     continue;
258                 }

```

```

260
261     if (gifOption.equals(anObject:"ya") || gifOption.equals(anObject:"y")) {
262         exportGIF = true;
263         gifPath = outputPath.replaceAll(regex:"\\.[^.]+$", replacement:".gif");
264         break;
265     } else if (gifOption.equals(anObject:"tidak") || gifOption.equals(anObject:"t")) {
266         break;
267     } else {
268         System.out.println(x:"Pilihan tidak valid. Silakan masukkan 'ya' atau 'tidak'.");
269         System.out.println();
270     }
271
272     try {
273         // Membaca gambar input
274         BufferedImage ogImage = ImageIO.read(inputFile);
275         int ogSize = (int) inputFile.length();
276
277         // Melakukan kompresi
278         long startTime = System.nanoTime();
279         BufferedImage compressedImage;

281         if (targetRatio == 0) {
282             compressedImage = Compression.compressImage(ogImage, method, threshold, minBlockSize);
283         } else {
284             String format = outputPath.substring(outputPath.lastIndexOf(ch:'.') + 1).toLowerCase();
285             compressedImage = Compression.compressWithTargetRatio(
286                 ogImage,
287                 method,
288                 threshold, tolerance,
289                 minBlockSize,
290                 ogSize,
291                 targetRatio,
292                 format
293             );
294         }
295
296         long endTime = System.nanoTime();
297
298         // Menyimpan gambar hasil kompresi
299         String outputFormat = outputPath.substring(outputPath.lastIndexOf(ch:'.') + 1).toLowerCase();
300         ImageIO.write(compressedImage, outputFormat, new File(outputPath));
301         int compressedSize = (int) new File(outputPath).length();
302
303         // Menghitung rasio kompresi dan waktu eksekusi
304         double compressionRatio = (1 - (double) compressedSize / ogSize) * 100;
305         double executionTime = (endTime - startTime) / 1e6;

```


BAB IV

MASUKAN DAN KELUARAN PROGRAM

4.1 Test Case 1 (Variance)

Masukan (CLI)
PS D:\Semester 4\Strategi Algoritma\tucil 2\tucil2_13523059_13523122\src> javac -d ..\bin Main.java PS D:\Semester 4\Strategi Algoritma\tucil 2\tucil2_13523059_13523122\src> java -cp ..\bin Main =====  ===== Masukkan nama file gambar (diakhiri dengan .jpg, .jpeg, atau .png): D:\Semester 4\Strategi Algoritma\tucil 2\tucil2_13523059_13523122\test\input\flowers.jpg ===== METODE PENGUKURAN ERROR ===== 1. Variance 2. Mean Absolute Deviation (MAD) 3. Max Pixel Difference 4. Entropy 5. Structural Similarity Index (SSIM) ===== Masukkan metode yang ingin digunakan: 6 Masukkan angka antara 1 sampai 5. Masukkan metode yang ingin digunakan: 1 Masukkan nilai ambang batas (threshold): 5 Masukkan ukuran blok minimum: 6 Masukkan target rasio kompresi (0 jika tidak ingin menggunakan fitur ini): 9 Target rasio kompresi harus berada antara 0 dan 1. Masukkan target rasio kompresi (0 jika tidak ingin menggunakan fitur ini): 0.5 Masukkan nama file hasil kompresi (diakhiri dengan .jpg, .jpeg, atau .png): D:\Semester 4\Strategi Algoritma\tucil 2\tucil2_13523059_13523122\test\output\hasilvar.jpg Apakah Anda ingin menyimpan GIF hasil kompresi? (ya/tidak): ya
Masukan (File Gambar)

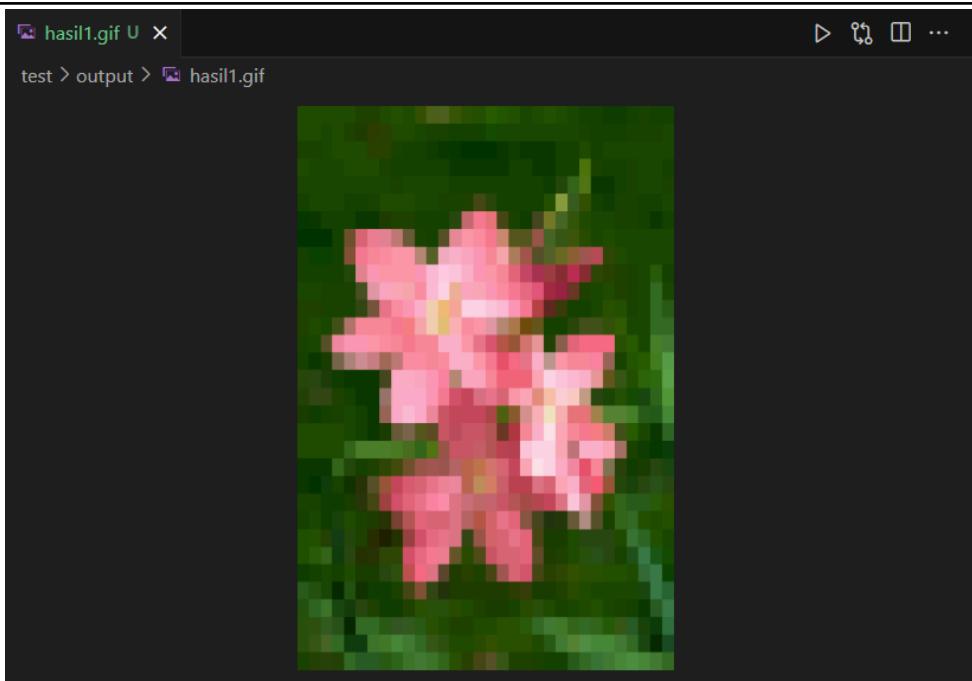

Keluaran (CLI)

```
Threshold disesuaikan ke: 499.0234
=====
HASIL KOMPRESI GAMBAR
=====
Kompresi gambar berhasil.
Waktu eksekusi: 3933.4452 ms
Ukuran gambar asli: 53710 bytes
Ukuran gambar hasil kompresi: 45236 bytes
Persentase kompresi: 15.78%
Kedalaman pohon: 7
Jumlah simpul pada pohon: 171654
=====
Gambar hasil kompresi disimpan di: D:\Semester 4\Strategi Algoritma\Tucil 2\Tucil2_13523059_13523122\test\output\hasilvar.jpg
GIF hasil kompresi disimpan di: D:\Semester 4\Strategi Algoritma\Tucil 2\Tucil2_13523059_13523122\test\output\hasilvar.gif
=====
```

Keluaran (File Gambar)



Keluaran (File GIF)

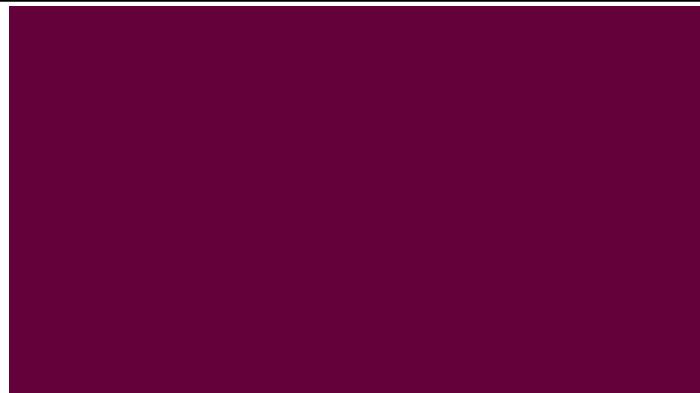


4.2 Test Case 2 (Mean Absolute Deviation dengan Gambar Solid Color)

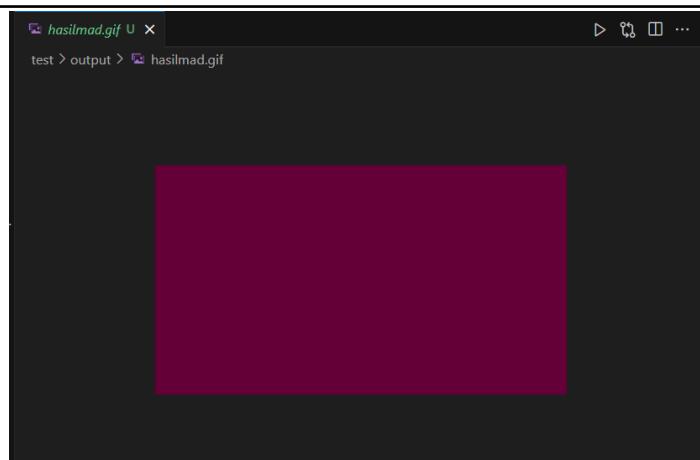
Masukan (CLI)
PS D:\Semester 4\Strategi Algoritma\Tucil 2\Tucil2_13523059_13523122\src> javac -d ..\bin Main.java PS D:\Semester 4\Strategi Algoritma\Tucil 2\Tucil2_13523059_13523122\src> java -cp ..\bin Main =====  ===== Masukkan nama file gambar (diakhiri dengan .jpg, .jpeg, atau .png): D:\Semester 4\Strategi Algoritma\Tucil 2\Tucil2_13523059_13523122\test\input\solid.jpg ===== METODE PENGUKURAN ERROR ===== 1. Variance 2. Mean Absolute Deviation (MAD) 3. Max Pixel Difference 4. Entropy 5. Structural Similarity Index (SSIM) ===== Masukkan metode yang ingin digunakan: 2 ===== Masukkan nilai ambang batas (threshold): 5 Masukkan ukuran blok minimum: 8 Masukkan target rasio kompresi (0 jika tidak ingin menggunakan fitur ini): 0 Masukkan nama file hasil kompresi (diakhiri dengan .jpg, .jpeg, atau .png): hasilmad.jpg Apakah Anda ingin menyimpan GIF hasil kompresi? (ya/tidak): y =====
Masukan (File Gambar)

Keluaran (CLI)
===== HASIL KOMPRESI GAMBAR ===== Kompreksi gambar berhasil. Waktu eksekusi: 43.9277 ms Ukuran gambar asli: 2117 bytes Ukuran gambar hasil kompresi: 2029 bytes Percentase kompresi: 4.16% Keadalan pohon: 0 Jumlah simpul pada pohon: 1 ===== Gambar hasil kompresi disimpan di: hasilmad.jpg GIF hasil kompresi disimpan di: hasilmad.gif =====

Keluaran (File Gambar)



Keluaran (File GIF)



4.3 Test Case 3 (Max Pixel Difference dengan Gambar High Resolution 1736 KB)

Masukan (CLI)

Masukan (File Gambar)



Keluaran (CLI)

HASIL KOMPRESI GAMBAR

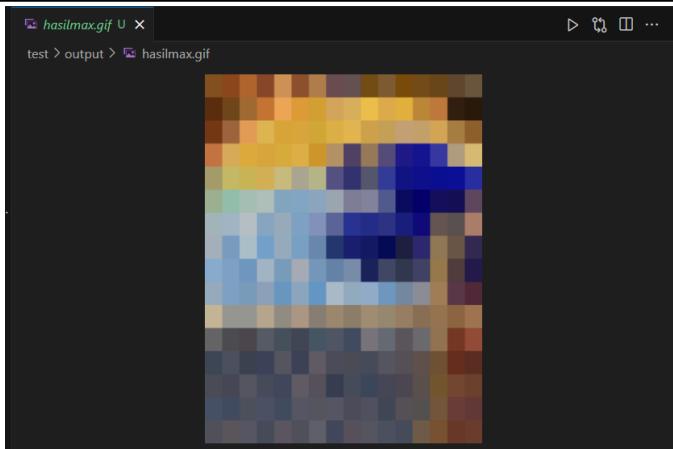
```
Kompresi gambar berhasil.  
waktu eksekusi: 3476.8258 ms  
Ukuran gambar asli: 1777037 bytes  
Ukuran gambar hasil kompresi: 1327320 bytes  
Persentase kompresi: 25.31%  
Keadalaman pohon: 8  
Jumlah simpul pada pohon: 87381
```

```
Gambar hasil kompresi disimpan di: D:\Semester 4\Strategi Algoritma\Tucil 2\tucil2_13523059_13523122\test\output\hasilmax.jpg  
GIF hasil kompresi disimpan di: D:\Semester 4\Strategi Algoritma\Tucil 2\tucil2_13523059_13523122\test\output\hasilmax.gif
```

Keluaran (File Gambar)



Keluaran (File GIF)



4.4 Test Case 4 (Entropy dan Gambar PNG)

Masukan (CLI)

Masukan (File Gambar)



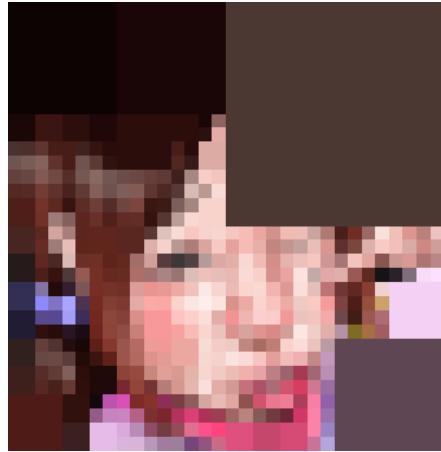
Keluaran (CLI)

```
=====
HASIL KOMPRESI GAMBAR
=====

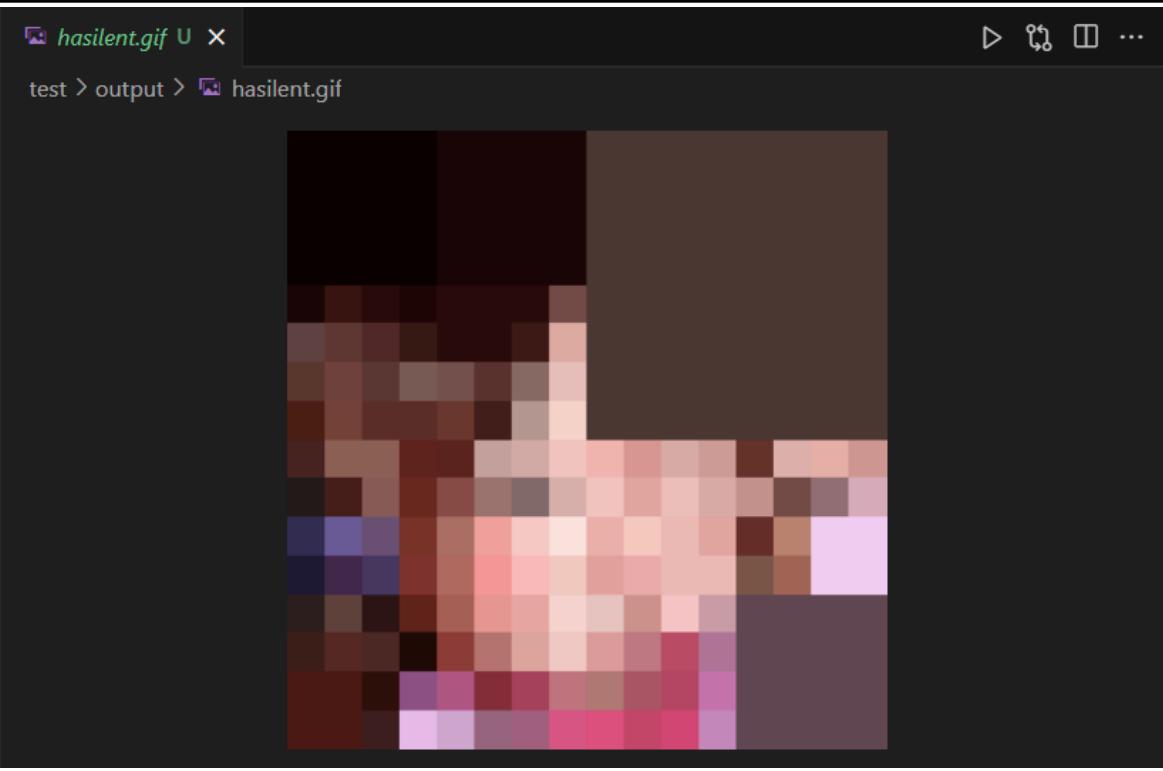
Kompresi gambar berhasil.
Waktu eksekusi: 52.1583 ms
Ukuran gambar asli: 196250 bytes
Ukuran gambar hasil kompresi: 5088 bytes
Percentase kompresi: 97.41%
Keadalaman pohon: 5
Jumlah simpul pada pohon: 613
=====

Gambar hasil kompresi disimpan di: D:\Semester 4\Strategi Algoritma\Tucil 2\Tucil2_13523059_13523122\test\output\hasilent.png
GIF hasil kompresi disimpan di: D:\Semester 4\Strategi Algoritma\Tucil 2\Tucil2_13523059_13523122\test\output\hasilent.gif
=====
```

Keluaran (File Gambar)



Keluaran (File GIF)



4.5 Test Case 5 (Variance dengan Gambar JPEG)

Masukan (CLI)

Masukkan nama file gambar (diakhiri dengan .jpg, .jpeg, atau .png): D:\Semester 4\Strategi Algoritma\Tucil 2\Tucil2_13523059_13523122\test\input\bebek.jpg
File input tidak ditemukan: D:\Semester 4\Strategi Algoritma\Tucil 2\Tucil2_13523059_13523122\test\input\bebek.jpg

Masukkan nama file gambar (diakhiri dengan .jpg, .jpeg, atau .png): d:\Semester 4\Strategi Algoritma\tucil 2\tucil2_13523059_13523122\test\input\bebek.jpeg

METODE PENGUKURAN ERROR

- 1. Variance
 - 2. Mean Absolute Deviation (MAD)
 - 3. Max Pixel Difference
 - 4. Entropy
 - 5. Structural Similarity Index (ssim)

Masukkan metode yang ingin digunakan: 1

Masukkan nilai ambang batas (threshold): 7

Masukkan ukuran blok minimum: 15

Masukkan

Masukkan nama file hasil kompresi (diakhiri dengan .jpg, .jpeg, atau .png): D:\Semester 4\Strategi Algoritma\Tucil 2\Tucil2_13523059_13523122\test\output\hasilval

r1.jpg

Apakah Anda ingin menyimpan GIF hasil kompresi? (ya/tidak): h
Pilihlah tidak valid jika benarannya bukan atau tidak!

Pilihan tidak valid. Silakan masukkan 'ya' atau 'tidak'.

Anak-anak Anda ingin menyimpen CIE hasil kompetisi? (ya/tidak)

Apakah Anda ingin menyimpan GIF hasil kompresi? (ya/tidak)

[Home](#) | [About Us](#) | [Services](#) | [Contact Us](#)

Masukan (File Gambar)



Keluaran (CLI)

HASIL KOMPRESI GAMBAR

Kompresi gambar berhasil.

Waktu eksekusi: 48.7431 ms

Ukuran gambar asli: 10777 bytes

Ukuran gambar asli: 10777 bytes

Ukuran gambar hasil kompres

Persentase kompresi

Kedalaman pohon: 4

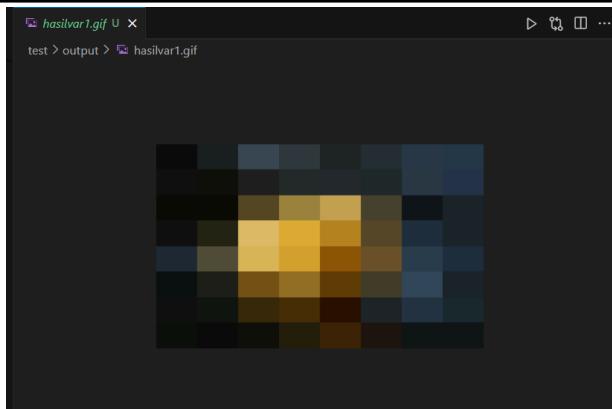
Jumlah simpul pada pohon: 341

Gambar hasil kompresi disimpan di: D:\Semester 4\Strategi Algoritma\Tucil_2\Tucil_2_13523059_13523122\test\output\hasilvar1.jpg

Keluaran (File Gambar)



Keluaran (File GIF)



4.6 Test Case 6 (Threshold Tinggi)

Masukan (CLI)

```
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS
=====
[ / \ ] [ / \ ] [ o } d ) / [ / \ ] [ / \ ] [ o } o ) / [ / \ ]
[ / \ ] [ / \ ] [ o } d ) / [ / \ ] [ / \ ] [ o } o ) / [ / \ ]
[ / \ ] [ / \ ] [ o } d ) / [ / \ ] [ / \ ] [ o } o ) / [ / \ ]
=====

Masukkan nama file gambar (diakhiri dengan .jpg, .jpeg, atau .png): D:\Semester 4\Strategi Algoritma\Tucil 2\Tucil2_13523059_13523122\test\input\bunny.jpg
=====

METODE PENGUKURAN ERROR
=====
1. Variance
2. Mean Absolute Deviation (MAD)
3. Max Pixel Difference
4. Entropy
5. Structural Similarity Index (SSIM)

Masukkan metode yang ingin digunakan: 2
=====

Masukkan nilai ambang batas (threshold): 67890
Threshold metode ini harus bernilai antara 0 sampai 255.

Masukkan nilai ambang batas (threshold): 100
Masukkan ukuran blok minimum: 0
Ukuran blok minimum harus lebih besar dari 0.

Masukkan ukuran blok minimum: 15
Masukkan target rasio kompresi (0 jika tidak ingin menggunakan fitur ini): 0
Masukkan nama file hasil kompresi (diakhiri dengan .jpg, .jpeg, atau .png): D:\Semester 4\Strategi Algoritma\Tucil 2\Tucil2_13523059_13523122\test\output\hasilmad1.jpg
Apakah Anda ingin menyimpan GIF hasil kompresi? (ya/tidak): ya
```

Masukan (File Gambar)



Keluaran (CLI)

```
=====
HASIL KOMPRESI GAMBAR
=====

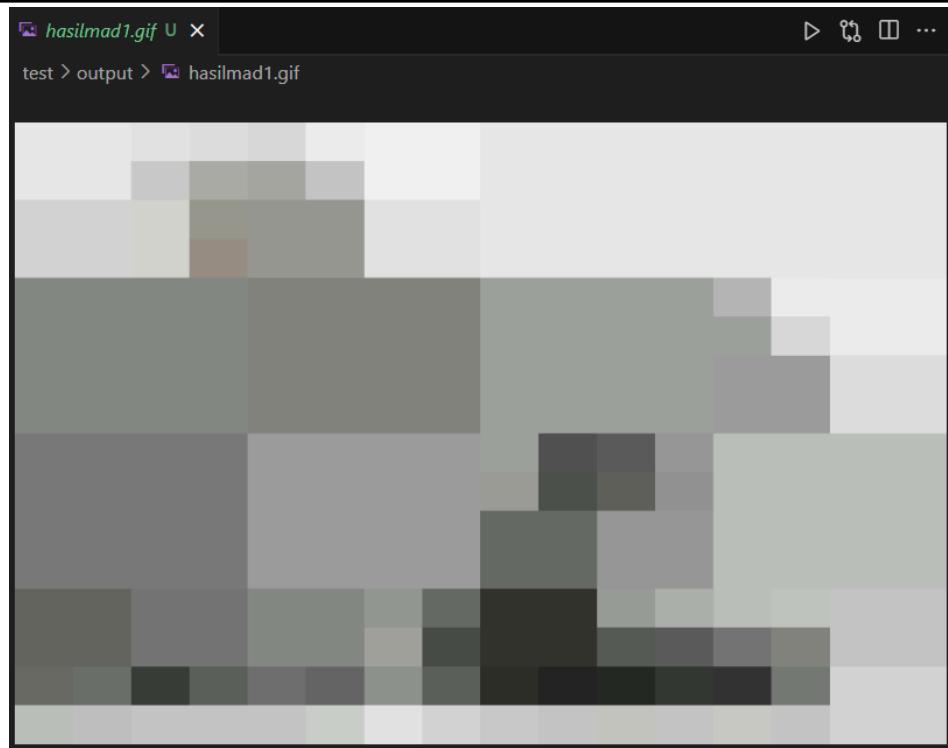
Kompresi gambar berhasil.
Waktu eksekusi: 686.806 ms
Ukuran gambar asli: 1359498 bytes
Ukuran gambar hasil kompresi: 70891 bytes
Persentase kompresi: 94.79%
Keadalaman pohon: 7
Jumlah simpul pada pohon: 209
=====

Gambar hasil kompresi disimpan di: D:\Semester 4\Strategi Algoritma\Tucil 2\Tucil2_13523059_13523122\test\output\hasilmad1.jpeg
GIF hasil kompresi disimpan di: D:\Semester 4\Strategi Algoritma\Tucil 2\Tucil2_13523059_13523122\test\output\hasilmad1.gif
=====
```

Keluaran (File Gambar)



Keluaran (File GIF)



4.7 Test Case 7 (Output Folder Tidak Ada/Tidak Sesuai)

Masukan (CLI)

```
=====
| / \ | / \ | / \ | / \ | / \ | / \ | / \ | / \ | / \ | | |
| \ / | o | \ / | o | \ / | o | \ / | o | \ / | o | \ / |
| \ / | \ / | \ / | \ / | \ / | \ / | \ / | \ / | \ / |
| \ / | \ / | \ / | \ / | \ / | \ / | \ / | \ / | \ / |
=====

Masukkan nama file gambar (diakhiri dengan .jpg, .jpeg, atau .png): D:\Semester 4\Strategi Algoritma\Tucil 2\Tucil2_13523059_13523122\test\input\panda.jpg

=====
METODE PENGUKURAN ERROR
=====
1. Variance
2. Mean Absolute Deviation (MAD)
3. Max Pixel Difference
4. Entropy
5. Structural Similarity Index (SSIM)

=====
Masukkan metode yang ingin digunakan: 2
Masukkan nilai ambang batas (threshold): 0
Masukkan ukuran blok minimum: 15
Masukkan target rasio kompresi (0 jika tidak ingin menggunakan fitur ini): 0
Masukkan nama file hasil kompresi (diakhiri dengan .jpg, .jpeg, atau .png): D:\Semester 4\Strategi Algoritma\Tucil 2\Tucil2_13523059_13523122\test\output\abc\hasil.jpg
Format file tidak didukung. Harus berakhir dengan .jpg, .jpeg, atau .png.

Masukkan nama file hasil kompresi (diakhiri dengan .jpg, .jpeg, atau .png): D:\Semester 4\Strategi Algoritma\Tucil 2\Tucil2_13523059_13523122\test\output\abc\hasil.gif
Folder tujuan tidak ada. Apakah Anda ingin membuatnya? (ya/tidak): ya
Apakah Anda ingin menyimpan GIF hasil kompresi? (ya/tidak): ya
```

Masukan (File Gambar)

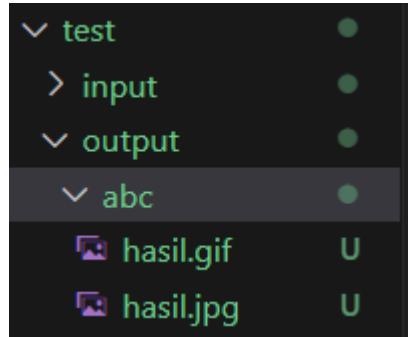


Keluaran (CLI)

```
=====
HASIL KOMPRESI GAMBAR
=====

Kompresi gambar berhasil.
Waktu eksekusi: 339.5218 ms
Ukuran gambar asli: 135500 bytes
Ukuran gambar hasil kompresi: 75221 bytes
Persentase kompresi: 44.49%
Keadalan pohon: 6
Jumlah simpul pada pohon: 5349

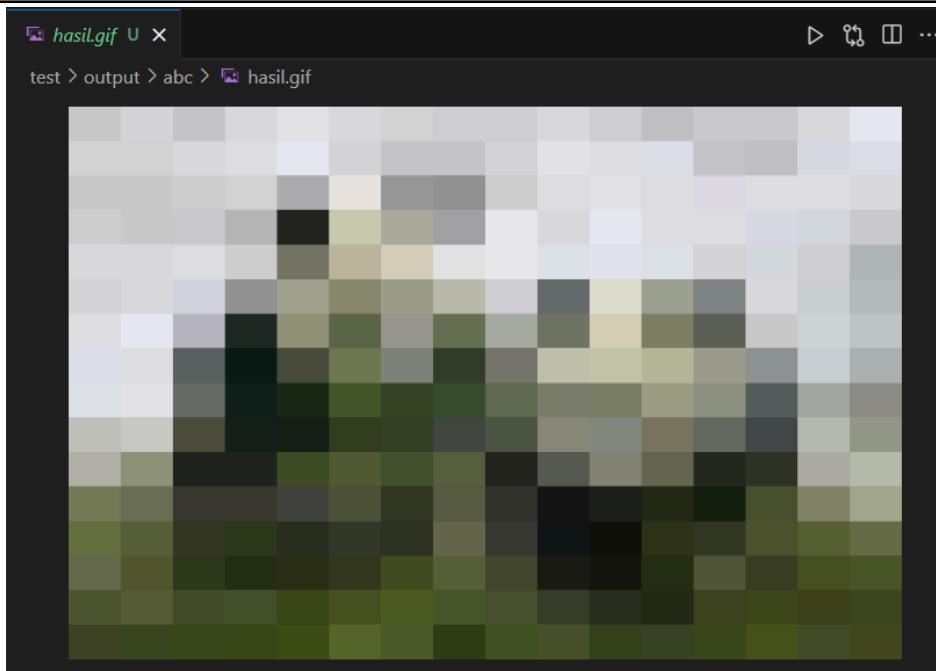
=====
Gambar hasil kompresi disimpan di: D:\Semester 4\Strategi Algoritma\Tucil 2\Tucil2_13523059_13523122\test\output\abc\hasil.jpg
GIF hasil kompresi disimpan di: D:\Semester 4\Strategi Algoritma\Tucil 2\Tucil2_13523059_13523122\test\output\abc\hasil.gif
=====
```



Keluaran (File Gambar)



Keluaran (File GIF)



4.8 Test Case 8 (BONUS => SSIM)

Masukan (CLI)



```
Masukkan nama file gambar (diakhiri dengan .jpg, .jpeg, atau .png): D:\Semester 4\Strategi Algoritma\Tucil 2\Tucil2_13523059_13523122\test\input\koala.png  
File input tidak ditemukan: D:\Semester 4\Strategi Algoritma\Tucil 2\Tucil2_13523059_13523122\test\input\koala.png
```

```
Masukkan nama file gambar (diakhiri dengan .jpg, .jpeg, atau .png): D:\Semester 4\Strategi Algoritma\Tucil 2\Tucil2_13523059_13523122\test\input\koala.gifjk  
Format file tidak didukung. Harus diakhiri dengan .jpg, .jpeg, atau .png.
```

```
Masukkan nama file gambar (diakhiri dengan .jpg, .jpeg, atau .png): D:\Semester 4\Strategi Algoritma\Tucil 2\Tucil2_13523059_13523122\test\input\koala.jpg
```

=====
METODE PENGUKURAN ERROR
=====

1. Variance
2. Mean Absolute Deviation (MAD)
3. Max Pixel Difference
4. Entropy
5. Structural Similarity Index (SSIM)

```
=====  
Masukkan metode yang ingin digunakan: 5
```

```
=====  
Anda memilih metode SSIM.  
Threshold SSIM berkisar antara 0 sampai 1.  
Semakin tinggi threshold, maka blok akan digabung jika sangat mirip.  
Semakin rendah threshold, maka blok digabung walaupun kurang mirip.
```

```
=====  
Masukkan nilai ambang batas (threshold): 9  
Threshold metode ini harus bernilai antara 0 sampai 1.
```

```
Masukkan nilai ambang batas (threshold): 0.5
```

```
Masukkan ukuran blok minimum: 15
```

```
Masukkan target rasio kompresi (0 jika tidak ingin menggunakan fitur ini): 0
```

```
Masukkan nama file hasil kompresi (diakhiri dengan .jpg, .jpeg, atau .png): D:\Semester 4\Strategi Algoritma\Tucil 2\Tucil2_13523059_13523122\test\output\hasilSSIM.jpg
```

```
Apakah Anda ingin menyimpan GIF hasil kompresi? (ya/tidak): ya
```

Masukan (File Gambar)



Keluaran (CLI)

```
=====
HASIL KOMPRESI GAMBAR
=====

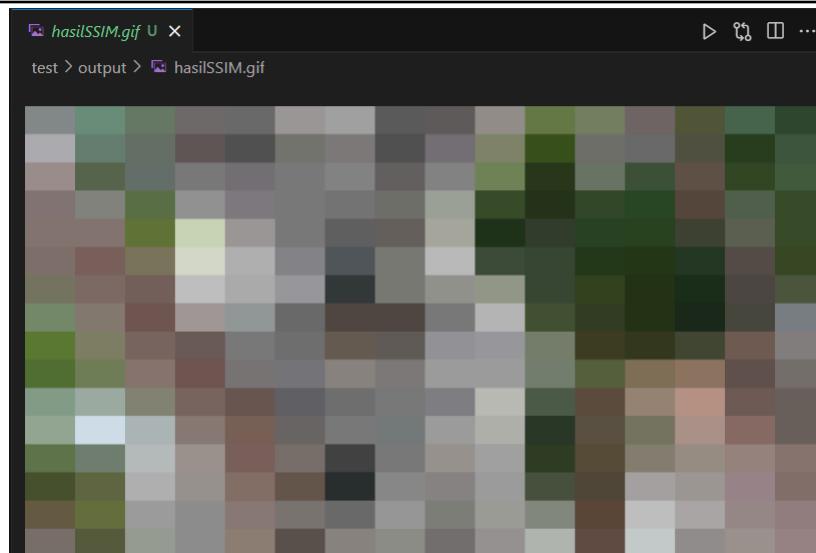
Kompresi gambar berhasil.
Waktu eksekusi: 1797.7092 ms
Ukuran gambar asli: 532192 bytes
Ukuran gambar hasil kompresi: 162602 bytes
Percentase kompresi: 69.45%
Keadalan pohon: 7
Jumlah simpul pada pohon: 16457
=====

Gambar hasil kompresi disimpan di: D:\Semester 4\Strategi Algoritma\Tucil 2\Tucil2_13523059_13523122\test\output\hasilSSIM.jpg
GIF hasil kompresi disimpan di: D:\Semester 4\Strategi Algoritma\Tucil 2\Tucil2_13523059_13523122\test\output\hasilSSIM.gif
=====
```

Keluaran (File Gambar)



Keluaran (File GIF)



4.9 Test Case 9 (BONUS => Target Ratio dan Output Folder Tidak Ada)

Masukan (CLI)



Masukkan nama file gambar (diakhiri dengan .jpg, .jpeg, atau .png): D:\Semester 4\Strategi Algoritma\Tucil 2\Tucil2_13523059_13523122\test\input\redpanda.jpg

=====
METODE PENGUKURAN ERROR
=====

1. Variance
2. Mean Absolute Deviation (MAD)
3. Max Pixel Difference
4. Entropy
5. Structural Similarity Index (SSIM)

Masukkan metode yang ingin digunakan: 3

Masukkan nilai ambang batas (threshold): 998

Threshold metode ini harus bernilai antara 0 sampai 255.

Masukkan nilai ambang batas (threshold): 5

Masukkan ukuran blok minimum: 0

Ukuran blok minimum harus lebih besar dari 0.

Masukkan ukuran blok minimum: 15

Masukkan target rasio kompresi (0 jika tidak ingin menggunakan fitur ini): 890

Target rasio kompresi harus berada antara 0 dan 1.

Masukkan target rasio kompresi (0 jika tidak ingin menggunakan fitur ini): 0.5

Masukkan nama file hasil kompresi (diakhiri dengan .jpg, .jpeg, atau .png): D:\Semester 4\Strategi Algoritma\Tucil 2\Tucil2_13523059_13523122\test\abcd\hasil.yu
iuo

Format file tidak didukung. Harus berakhir dengan .jpg, .jpeg, atau .png.

Masukkan nama file hasil kompresi (diakhiri dengan .jpg, .jpeg, atau .png): D:\Semester 4\Strategi Algoritma\Tucil 2\Tucil2_13523059_13523122\test\abcd\hasil.jp

g

Folder tujuan tidak ada. Apakah Anda ingin membuatnya? (ya/tidak): h

Pilihan tidak valid. Silakan masukkan 'ya' atau 'tidak'.

Folder tujuan tidak ada. Apakah Anda ingin membuatnya? (ya/tidak): tidak

Silakan masukkan ulang nama file output.

Masukkan nama file hasil kompresi (diakhiri dengan .jpg, .jpeg, atau .png): D:\Semester 4\Strategi Algoritma\Tucil 2\Tucil2_13523059_13523122\test\output\hasilma

x1.jpg

Apakah Anda ingin menyimpan GIF hasil kompresi? (ya/tidak): ya

Masukan (File Gambar)



Keluaran (CLI)

```
Threshold disesuaikan ke: 117.1875
```

```
=====
```

```
HASIL KOMPRESI GAMBAR
```

```
=====
```

```
Kompresi gambar berhasil.
```

```
Waktu eksekusi: 14300.3189 ms
```

```
Ukuran gambar asli: 228387 bytes
```

```
Ukuran gambar hasil kompresi: 114166 bytes
```

```
Persentase kompresi: 50.01%
```

```
Keadalaman pohon: 7
```

```
Jumlah simpul pada pohon: 210126
```

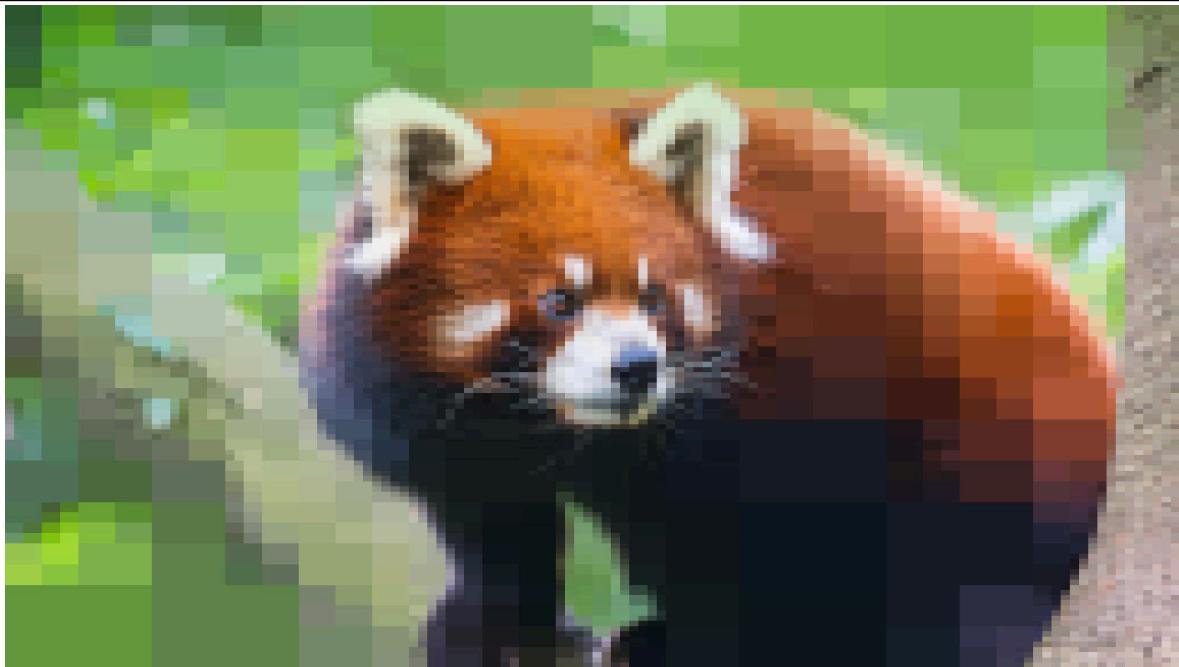
```
=====
```

```
Gambar hasil kompresi disimpan di: D:\Semester 4\Strategi Algoritma\Tucil 2\Tucil2_13523059_13523122\test\output\hasilmax1.jpg
```

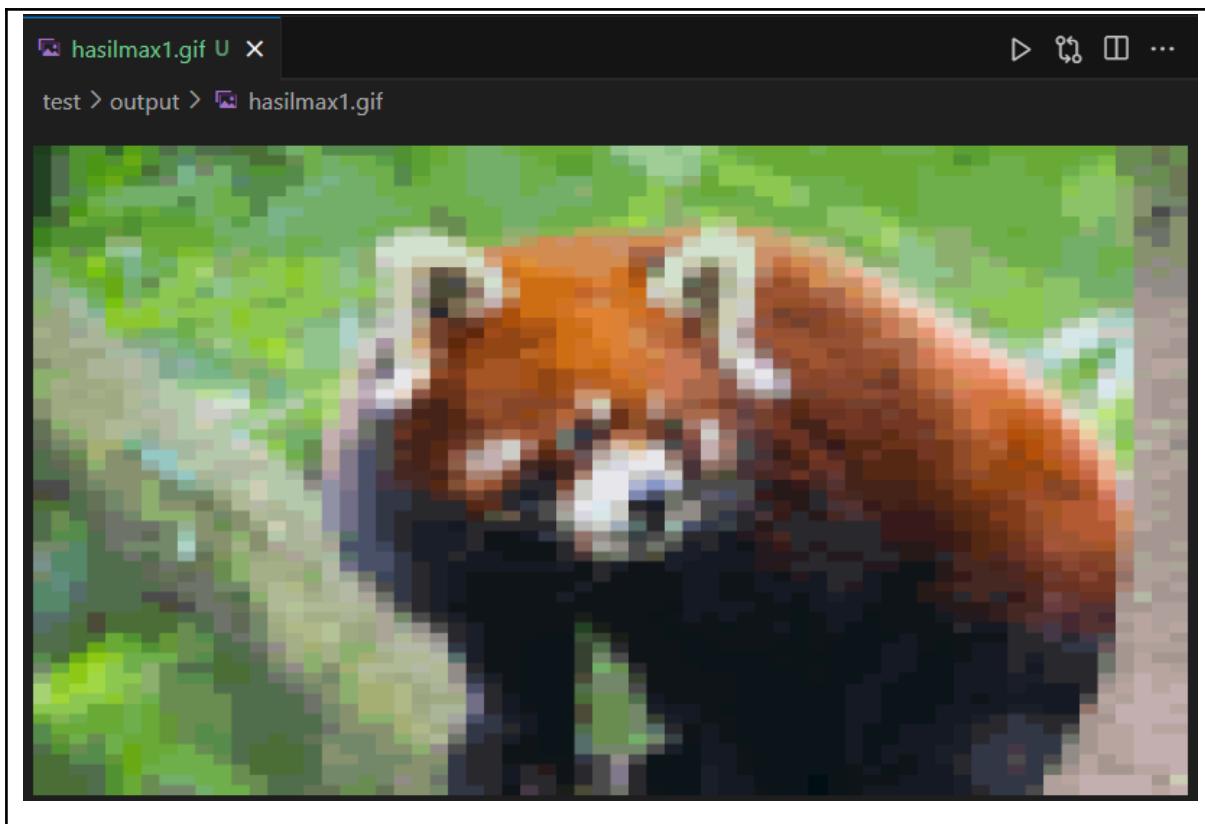
```
GIF hasil kompresi disimpan di: D:\Semester 4\Strategi Algoritma\Tucil 2\Tucil2_13523059_13523122\test\output\hasilmax1.gif
```

```
=====
```

Keluaran (File Gambar)



Keluaran (File GIF)



BAB V

ANALISIS ALGORITMA

5.1 Deskripsi Umum Algoritma

Program ini menggunakan metode kompresi berbasis Quadtree (*divide and conquer*) untuk mereduksi kompleksitas visual suatu gambar. Setiap blok gambar akan dicek, apakah blok tersebut cukup mirip (homogen). Jika tidak, blok gambar akan dibagi menjadi 4 bagian yang lebih kecil secara rekursif. Proses ini akan dilakukan sampai kedalaman tertentu atau hingga blok dapat dinyatakan cukup mirip.

Algoritma utama pada program ini adalah algoritma pada fungsi **compressToDepth()** yang merupakan algoritma *divide and conquer*. Pada setiap level rekursi, blok akan dibagi menjadi 4 sub-blok jika tidak homogen. Proses berlanjut hingga kedalaman maksimum atau blok cukup homogen. Misalkan gambar berukuran $n \times n$, dan kedalaman maksimal adalah k .

Pada setiap level terdapat 4^d sub-blok di level ke- d karena setiap blok dibagi menjadi 4. Setiap blok kemudian akan dicek homogenitasnya, dan jika tidak terhitung homogen, dihitung kemudian warna rata-ratanya dan di-*fill*.

5.2 Analisis Teoritis Kompleksitas dengan Master Theorem

Untuk menganalisis kompleksitas algoritma, digunakan **Master Theorem** yang bentuk umumnya adalah sebagai berikut.

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$$

Rumus ini digunakan untuk algoritma *divide and conquer* yang memecah masalah ukuran n menjadi a sub masalah. Setiap submasalah berukuran $\frac{n}{b}$. $f(n)$ adalah pekerjaan di luar rekursi (biasanya pembagian, pengecekan, penggabungan, dsb). Kemudian, untuk *Quadtree*, komponennya dapat didefinisikan sebagai berikut.

- $a = 4$ (gambar dibagi jadi 4 bagian)
- $b = 2$ (setiap sisi dibagi dua)

- $f(n) = O(n^2)$, karena evaluasi error (mean/variance/dll) butuh mengakses seluruh blok (area)

Jadi,

$$T(n) = 4T\left(\frac{n}{2}\right) + O(n^2) \Rightarrow T(n) = O(n^2 \log(n))$$

Dalam implementasi nyata, setiap level *Quadtree* hanya memproses area yang tidak tumpang tindih, dan setiap piksel gambar hanya diproses satu kali secara keseluruhan. Oleh karena itu, dalam kasus praktis (*best/average case*), total pekerjaan bersifat linear terhadap jumlah piksel, sehingga kompleksitas efektifnya $T(n)$ dapat dianggap sebagai $O(n^2)$.

5.3 Pengaruh Parameter dan Fitur Tambahan

Metode-metode evaluasi error yang digunakan pada program seperti *variance*, *mean absolute deviation* (MAD), max pixel difference, entropy, dan SSIM, perlu melakukan iterasi terhadap semua piksel dalam satu blok. Jadi, kompleksitas per blok adalah $O(k^2)$ untuk blok berukuran $k \times k$. Karena semua gambar terdiri dari $O(n^2)$ piksel, semua evaluasi totalnya masih akan tetap dalam batas $O(n^2)$.

Kemudian, fitur BONUS *target compression ratio* dengan penyesuaian nilai *threshold* menggunakan *binary search* yang memiliki kompleksitas $O(n^2 \log(T))$. Hal ini karena dilakukan sebanyak $O(\log(T))$ iterasi (dengan T adalah rentang *threshold*). Setiap iterasi memanggil proses kompresi penuh. Lalu, proses penyimpanan hasil gambar atau GIF dilakukan secara langsung menggunakan API I/O dan encoding. Karena setiap frame berukuran $n \times n$, dan jumlah *frame* adalah D (biasanya setara dengan *maxDepth* + 1), maka kompleksitasnya adalah $O(n^2 \cdot D) \approx O(n^2)$.

Jika digabungkan, dapat dilihat daftar kompleksitas yang dimiliki oleh program adalah sebagai berikut.

Fungsi	Kompleksitas Waktu
Quadtree Compression	$O(n^2)$ atau $O(n^2 \log n)$

Error Evaluation per Block	$O(n^2)$
Target Ratio Adjustment (Binary Search)	$O(n^2 \log T)$
Penyimpanan Output (gambar/GIF)	$O(n^2)$

Jadi, kompleksitas program secara keseluruhan (dalam kasus target ratio) adalah $O(n^2 \log(T))$. Namun, jika pengguna memilih untuk tidak menggunakan fitur target ratio, maka kompleksitas algoritmanya menjadi $O(n^2)$. Dalam kasus *worst case scenario*, jika target ratio digunakan oleh pengguna dan rekursi penuh (*full recursion*) terjadi, kompleksitas algoritma program dapat meningkat menjadi $O(n^2 \log(n) \log(T))$. Hal ini akan terjadi jika semua blok tidak homogen.

DAFTAR PUSTAKA

- Glushach, R. 2021. “What is a Quadtree and How it Works”. [Online]. Tersedia: <https://romanglushach.medium.com/what-is-a-quadtrees-and-how-it-works-6286791fb46>. [7 April 2025].
- Munir, R. 2025. “Tugas Kecil 2: Penerapan Divide and Conquer untuk Kompresi Gambar dengan Metode Quadtree”. [Online]. Tersedia: <https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/Tucil2-Stima-2025.pdf>. [28 Maret 2025].
- Munir, R. 2025. “Algoritma Divide and Conquer (2025) Bagian 1”. [Online]. Tersedia: [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/07-Algoritma-Divide-and-Conquer-\(2025\)-Bagian1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/07-Algoritma-Divide-and-Conquer-(2025)-Bagian1.pdf). [7 April 2025].
- Munir, R. 2025. “Algoritma Divide and Conquer (2025) Bagian 2”. [Online]. Tersedia: [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/08-Algoritma-Divide-and-Conquer-\(2025\)-Bagian2.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/08-Algoritma-Divide-and-Conquer-(2025)-Bagian2.pdf). [10 April 2025].
- ScholarHat. 2023. “Divide and Conquer Algorithm Tutorial”. [Online]. Tersedia: <https://www.scholarhat.com/tutorial/datastructures/divide-and-conquer-algorithm>. [7 April 2025].

LAMPIRAN

Lampiran 1. Tautan *Repository*

https://github.com/naomirisaka/Tucil2_13523059_13523122

Lampiran 2. Tabel Checklist Program

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