

IF2211 Strategi Algoritma
Laporan Tugas Kecil 2 – Kompresi Gambar Dengan Metode
Quadtree



Disusun oleh:

I Made Wiweka Putera – 13523160

PROGRAM STUDI TEKNIK INFORMATIKA SEKOLAH TEKNIK
ELEKTRO DAN INFORMATIKA INSTITUT TEKNOLOGI BANDUNG
JL. GANESA 10, BANDUNG 40132

2025

Daftar Isi

Daftar Isi.....	2
BAB I	
PENDAHULUAN.....	3
1.1 Deskripsi Masalah.....	3
1.2 Algoritma Divide and Conquer.....	3
BAB II	
PENERAPAN ALGORITMA DIVIDE AND CONQUER.....	4
2.1 Algoritma Kompresi Gambar.....	4
2.2 Pseudocode.....	4
BAB III	
SOURCE CODE.....	7
3.1 Arsitektur Program.....	7
3.2 Source Code.....	7
BAB IV	
EKSPERIMEN.....	8
4.1 Tangkapan Layar Eksperimen.....	8
4.2 Analisis.....	11
LAMPIRAN.....	13

BAB I

PENDAHULUAN

1.1 Deskripsi Masalah

Tugas ini bertujuan untuk mengembangkan program kompresi gambar berbasis struktur data Quadtree menggunakan metode Divide and Conquer. Program akan menerima input berupa gambar yang direpresentasikan sebagai matriks piksel RGB, serta parameter dari pengguna berupa metode perhitungan variansi, nilai ambang batas variansi (threshold), dan ukuran minimum blok.

Proses kompresi dilakukan dengan cara membagi gambar menjadi blok-blok kecil berdasarkan tingkat keseragaman warna. Untuk setiap blok, dihitung nilai variansi warnanya; jika nilai tersebut melebihi threshold dan ukuran blok masih bisa dibagi, maka blok akan dibagi menjadi empat sub-blok. Proses ini dilakukan secara rekursif hingga seluruh blok memenuhi kondisi penghentian, yaitu variansi di bawah threshold atau ukuran minimum blok tercapai.

Setiap blok yang tidak lagi dibagi akan dinormalisasi warnanya menjadi rata-rata nilai RGB blok tersebut. Gambar hasil kompresi kemudian direkonstruksi berdasarkan struktur Quadtree yang terbentuk dan disimpan dalam bentuk file hasil. Selain itu, program juga menghitung dan menampilkan persentase efisiensi kompresi yang dicapai.

1.2 Algoritma Divide and Conquer

Algoritma Divide and Conquer adalah pendekatan pemecahan masalah dengan cara membagi masalah besar menjadi beberapa submasalah yang lebih kecil, menyelesaikan setiap submasalah secara rekursif, lalu menggabungkan hasilnya untuk membentuk solusi akhir. Dalam konteks kompresi gambar dengan Quadtree, gambar dibagi menjadi empat bagian secara rekursif berdasarkan tingkat keseragaman warna. Proses pembagian terus dilakukan hingga blok dianggap seragam atau ukuran minimum blok tercapai, lalu hasil dari setiap sub-blok digabungkan untuk membentuk representasi gambar yang telah dikompresi.

BAB II

PENERAPAN ALGORITMA DIVIDE AND CONQUER

2.1 Algoritma Kompresi Gambar

1. Divide (Pembagian):

- Awalnya, seluruh gambar dianggap sebagai satu blok besar.
- Hitung seberapa "berbeda" atau "tidak homogen" blok tersebut (menggunakan Variance, MAD, Maximum Pixel Difference atau Entropy).
- Jika nilai perbedaan tersebut melebihi batas (threshold) yang sudah ditentukan dan ukuran blok masih cukup besar (lebih besar dari ukuran minimum), maka blok itu dibagi menjadi empat sub-blok.
- Langkah ini dilakukan secara rekursif: tiap kuadran yang dihasilkan akan diperlakukan sama untuk menentukan apakah perlu dibagi lagi atau tidak.

2. Conquer (Penyelesaian pada Blok Kecil):

- Setelah dilakukan pembagian, tiap sub-blok diproses kembali.
- Jika sub-blok tersebut sudah cukup homogen (nilai error di bawah threshold) atau mencapai ukuran minimum, maka proses pembagian berhenti pada blok itu.
- Pada titik ini, setiap blok yang tidak dibagi lagi akan disederhanakan dengan menyimpan nilai rata-rata warna (rata-rata dari kanal R, G, dan B) yang mewakili seluruh blok tersebut.

3. Combine (Penggabungan):

- Setelah seluruh gambar diproses dan setiap blok mendapatkan representasi warnanya, hasil akhirnya digabungkan untuk membentuk gambar terkompresi.
- Gambar baru ini tersusun dari blok-blok homogen yang masing-masing diwakili oleh nilai rata-rata warnanya.

2.1 Algoritma Pembuatan GIF

1. Setelah quadtree dibangun dari gambar asli, kita ambil representasi gambar pada tiap level quadtree. Fungsi `captureFramesPerLevel` akan menghasilkan serangkaian frame/gambar (`levelFrames`) di mana tiap frame mewakili kondisi kompresi pada level tertentu.

2. Semua frame tersebut digabungkan secara berurutan dengan fungsi createGIF yang memanggil GifBegin(), GifWriteFrame() untuk setiap frame, dan GifEnd(). Hasil akhirnya adalah sebuah file GIF animasi yang menunjukkan proses kompresi dari level paling kasar (level root) hingga paling detail (level leaf).

2.2 Pseudocode

Berikut adalah pseudocode dari algoritma utama program ini.

BEGIN

1. Initialize FreeImage.

2. Input Data from User:

- Prompt and read inputImagePath.
 - *Validate*: Check that the file exists and that its extension is one of: .jpg, .jpeg, .png.
- Prompt and read the error method (errorMethodChoice) from the user.
- Prompt and read the threshold value with proper validation:
 - If error method is Variance, valid threshold is 0-65025.
 - If MAD or Max Pixel Difference, valid threshold is 0-255.
 - If Entropy, valid threshold is 0-8.
- Prompt and read the minBlockSize.
- Prompt and read the outputImagePath (again checking the extension is valid: .jpg, .jpeg, or .png).
- Prompt and read the outputGIFPath (extension must be .gif).

3. Record Start Time.

4. Load and Convert Image:

- Call loadImageAs24Bit(inputImagePath) to get:

- converted image (24-bit RGB image),
 - width and height.
- **IF** converted is NULL, then:
 - Deinitialize FreeImage and EXIT.
- Allocate an array imgData[width * height * 3] (for RGB data).
- **FOR** each y from 0 to height - 1:
 - Get the scanline: bits = FreeImage_GetScanLine(converted, y).
 - **FOR** each x from 0 to width - 1:
 - Compute srcIdx = x * 3.
 - Compute destIdx = (y * width + x) * 3.
 - Copy and swap channels:
 - imgData[destIdx + 0] ← bits[srcIdx + 2]
// Red
 - imgData[destIdx + 1] ← bits[srcIdx + 1]
// Green
 - imgData[destIdx + 2] ← bits[srcIdx + 0]
// Blue
 - **ENDFOR**
- **ENDFOR**
- Unload converted.

5. Get Original File Size:

- originalSize ← getFileSize(inputImagePath)

6. Build Quadtree:

- Create a QuadTree object qt.
- Call qt.build(imgData, width, height, threshold, minBlockSize, errorMethodChoice).

7. Capture Level-Based Frames:

- Call `levelFrames ← qt.captureFramesPerLevel(width, height)`
 - Reconstructs an image for every quadtree level, flips each frame vertically so that the GIF appears top-down.

8. Extend Final Frame Duration:

- **IF** `levelFrames` is not empty, then:
 - For 3 iterations, duplicate the last frame:
 - `levelFrames.push_back(last frame in levelFrames)`

9. Reconstruct Compressed Image:

- Allocate `compressedData[width * height * 3]`.
- Call `qt.reconstructImage(compressedData, width, height)`.

10. Save the Compressed Image:

- Determine output format:
 - `outFIF ← FreeImage_GetFIFFromFilename(outputImagePath)`
 - **IF** `outFIF` is UNKNOWN, set `outFIF ← JPEG`.
- Convert the raw compressed data to a FreeImage bitmap:
 - `outBmp ← FreeImage_ConvertFromRawBits(compressedData, width, height, width * 3, 24, maskR, maskG, maskB)`
- **IF** saving `outBmp` to `outputImagePath` fails, display an error.
- Unload `outBmp`.

11. Clean Up Data:

- Free allocated memory for `imgData` and `compressedData`.

12. Record End Time and Compute Execution Time:

- `endTime ← current time`

- $executionTime \leftarrow endTime - startTime$

13.Create Animated GIF:

- Call `createGIF(levelFrames, width, height, delayMs, outputGIFPath)`
 - **IF** GIF creation fails, display an error.
- $GIFexecutionTime \leftarrow \text{current time} - startTime$

14.Get Compressed Image Size:

- $compressedSize \leftarrow getFileSize(outputImagePath)$

15.Display Statistics:

- Output `executionTime`, `GIFexecutionTime`, `originalSize`, `compressedSize`, and calculated compression percentage.
- Output the quadtree metrics: tree depth and node count.
- Output the paths where the compressed image and GIF are saved.

16.Deinitialize FreeImage.

END

BAB III

SOURCE CODE

3.1 Arsitektur Program

Program dibuat dengan struktur proyek sebagai berikut:

```
Project Root
├── bin
│   └── FreeImage.dll
├── doc
│   └── Tucil2_13523160.pdf
├── src
│   ├── main.cpp
│   ├── modules
│   │   ├── errorMethods.hpp
│   │   ├── errorMethods.cpp
│   │   ├── gif_maker.hpp
│   │   ├── gif_maker.cpp
│   │   ├── gif.h
│   │   ├── imageUtils.hpp
│   │   ├── imageUtils.cpp
│   │   ├── quadTree.hpp
│   │   └── quadTree.cpp
│   └── obj
├── test
├── CMakeLists.txt
├── Makefile
└── README
```

3.2 Source Code

Berikut adalah [source code](#) program dalam bahasa pemrograman C++.

Program utama **main.cpp**

```
#include <iostream>
#include <string>
#include <vector>
#include <ctime>
#include <limits>
#include <sys/stat.h>
#include <FreeImage.h>
#include "modules/imageUtils.hpp"
```

```

#include "modules/quadTree.hpp"
#include "modules/errorMethods.hpp"
#include "modules/gif_maker.hpp"

using namespace std;

// Check if file exists.
bool fileExists(const string &filename)
{
    struct stat buffer;
    return (stat(filename.c_str(), &buffer) == 0);
}

// Convert a string to lower-case.
string toLower(const string &s)
{
    string result;
    for (char c : s)
        result.push_back(tolower(c));
    return result;
}

int main()
{
    // Initialize FreeImage.
    FreeImage_Initialise();

    cout << "=====\n";
    cout << "          INPUT          \n";
    cout << "=====\n";

    // 1. Input: Image path.
    string inputImagePath;
    while (true)
    {
        cout << "Masukkan path gambar yang akan dikompres: ";
        getline(cin, inputImagePath);
        if (!fileExists(inputImagePath))
        {
            cout << "File tidak ditemukan. Silakan masukkan path yang valid.\n";
            continue;
        }
        // Check if the file path contains an extension
        size_t pos = inputImagePath.find_last_of(".");
        if (pos == string::npos)
        {
            cout << "Path harus memiliki ekstensi .jpg, .jpeg, atau .png.\n";
            continue;
        }
    }
}

```

```

        string ext = toLower(inputImagePath.substr(pos));
        if (ext == ".jpg" || ext == ".jpeg" || ext == ".png")
            break;
        else
            cout << "Ekstensi file tidak didukung. Hanya .jpg, .jpeg,
atau .png yang diterima.\n";
    }

    // 2. Input: Error calculation method.
    int errorMethodChoice;
    cout << "Pilihan Metode Kalkulasi Error:\n";
    cout << "1: Variance\n";
    cout << "2: Mean Absolute Deviation (MAD)\n";
    cout << "3: Max Pixel Difference\n";
    cout << "4: Entropy\n";
    cout << "Masukkan pilihan: ";
    cin >> errorMethodChoice;
    if (errorMethodChoice < 1 || errorMethodChoice > 4)
    {
        cout << "Pilihan tidak valid.\n";
        return 1;
    }

    // 3. Input: Threshold.
    double threshold;
    bool validThreshold = false;
    while (!validThreshold)
    {
        if (errorMethodChoice == 1)
            cout << "Masukkan nilai threshold untuk Variance
(0-65025): ";
        else if (errorMethodChoice == 2 || errorMethodChoice == 3)
            cout << "Masukkan nilai threshold untuk MAD/Max Pixel
Difference (0-255): ";
        else if (errorMethodChoice == 4)
            cout << "Masukkan nilai threshold untuk Entropy (0-8): ";
        cin >> threshold;
        switch (errorMethodChoice)
        {
            case 1:
                validThreshold = (threshold >= 0 && threshold <= 65025);
                if (!validThreshold)
                    cout << "Masukan tidak valid. Untuk Variance,
masukkan nilai antara 0 dan 65025.\n";
                break;
            case 2:
            case 3:
                validThreshold = (threshold >= 0 && threshold <= 255);
                if (!validThreshold)
                    cout << "Masukan tidak valid. Untuk MAD/Max Pixel
Difference, masukkan nilai antara 0 dan 255.\n";

```

```

        break;
    case 4:
        validThreshold = (threshold >= 0 && threshold <= 8);
        if (!validThreshold)
            cout << "Masukan tidak valid. Untuk Entropy, masukkan
nilai antara 0 dan 8.\n";
        break;
    }
}

// 4. Input: Minimum block size.
int minBlockSize;
cout << "Masukkan minimum block size: ";
cin >> minBlockSize;

// 5. Input: Output image path.
cin.ignore(numeric_limits<streamsize>::max(), '\n');
string outputImagePath;
while (true)
{
    cout << "Masukkan path output image: ";
    getline(cin, outputImagePath);
    size_t pos = outputImagePath.find_last_of(".");
    if (pos == string::npos)
    {
        cout << "Path harus memiliki ekstensi .jpg, .jpeg, atau
.png.\n";
        continue;
    }
    string ext = toLower(outputImagePath.substr(pos));
    if (ext == ".jpg" || ext == ".jpeg" || ext == ".png")
        break;
    else
        cout << "Ekstensi file tidak didukung. Hanya .jpg, .jpeg,
atau .png yang diterima.\n";
}

// 6. Input: Output GIF path.
string outputGIFPath;
while (true)
{
    cout << "Masukkan path output GIF: ";
    getline(cin, outputGIFPath);
    size_t pos = outputGIFPath.find_last_of(".");
    if (pos == string::npos)
    {
        cout << "Path harus memiliki ekstensi .gif.\n";
        continue;
    }
    string ext = toLower(outputGIFPath.substr(pos));
    if (ext == ".gif")

```

```

        break;
    else
        cout << "Ekstensi file tidak didukung. Hanya .gif yang
diterima.\n";
    }

    // Start time
    clock_t startTime = clock();

    // Load image using FreeImage.
    int width = 0, height = 0;
    FIBITMAP *converted = loadImageAs24Bit(inputImagePath, width,
height);
    if (!converted)
    {
        FreeImage_DeInitialise();
        return 1;
    }

    // Copy image data into a 1D array and convert BGR -> RGB.
    unsigned char *imgData = new unsigned char[width * height * 3];
    for (int y = 0; y < height; y++)
    {
        BYTE *bits = FreeImage_GetScanLine(converted, y);
        for (int x = 0; x < width; x++)
        {
            int srcIdx = x * 3;
            int destIdx = (y * width + x) * 3;
            imgData[destIdx + 0] = bits[srcIdx + 2]; // R
            imgData[destIdx + 1] = bits[srcIdx + 1]; // G
            imgData[destIdx + 2] = bits[srcIdx + 0]; // B
        }
    }
    FreeImage_Unload(converted);

    // Get original file size.
    long originalSize = getFileSize(inputImagePath);

    // Build quadtree and record frames.
    // Frame: vector<uint8_t> = an image of every QuadTree level
    QuadTree qt;
    qt.build(imgData, width, height, threshold, minBlockSize,
errorMethodChoice);
    vector<vector<uint8_t>> levelFrames =
qt.captureFramesPerLevel(width, height);

    // Push the last frame 3 more times so that the final image
appears longer
    if (!levelFrames.empty())
    {
        for (int i = 0; i < 3; i++)

```

```

        {
            levelFrames.push_back(levelFrames.back());
        }
    }

    // Reconstruct compressed image.
    unsigned char *compressedData = new unsigned char[width * height
* 3];
    qt.reconstructImage(compressedData, width, height);

    // Determine output format.
    FREE_IMAGE_FORMAT outFIF =
FreeImage_GetFIFFromFilename(outputImagePath.c_str());
    if (outFIF == FIF_UNKNOWN)
    {
        cerr << "Unknown input image format; defaulting output to
JPEG.\n";
        outFIF = FIF_JPEG;
    }

    // Save the compressed image.
    FIBITMAP *outBmp = FreeImage_ConvertFromRawBits(compressedData,
width, height, width * 3, 24, 0xFF0000, 0x00FF00, 0x0000FF);
    if (!FreeImage_Save(outFIF, outBmp, outputImagePath.c_str(), 0))
        cerr << "Error saving compressed image to " <<
outputImagePath << "\n";
    FreeImage_Unload(outBmp);

    delete[] compressedData;
    delete[] imgData;

    // End time (compression process)
    clock_t endTime = clock();
    double executionTime = double(endTime - startTime) /
CLOCKS_PER_SEC;

    // Create animated GIF from recorded frames.
    if (!createGIF(levelFrames, width, height, 50, outputGIFPath))
        cerr << "GIF creation failed." << "\n";

    // End time (GIF creation process)
    endTime = clock();
    double GIFexecutionTime = double(endTime - startTime) /
CLOCKS_PER_SEC;

    long compressedSize = getFileSize(outputImagePath);
    double compressionPercentage = 0.0;
    if (originalSize > 0)
        compressionPercentage = (1.0 - double(compressedSize) /
originalSize) * 100.0;

```

```

        cout << "\n=====\\n";
        cout << "          OUTPUT          \\n";
        cout << "=====\\n";
        cout << "Waktu Eksekusi Kompresi (s): " << executionTime << "\\n";
        cout << "Waktu Eksekusi hingga Pembuatan GIF (s): " <<
GIFexecutionTime << "\\n";
        cout << "Ukuran Gambar Sebelum (bytes): " << originalSize <<
"\\n";
        cout << "Ukuran Gambar Setelah (bytes): " << compressedSize <<
"\\n";
        cout << "Persentase Kompresi: " << compressionPercentage <<
"%\\n";
        cout << "Kedalaman Pohon: " << qt.getTreeDepth() << "\\n";
        cout << "Banyak Simpul Pohon: " << qt.getNodeCount() << "\\n";
        cout << "Gambar Hasil Kompresi Disimpan pada: " <<
outputImagePath << "\\n";
        cout << "GIF Disimpan pada: " << outputGIFPath << "\\n";

        FreeImage_DeInitialise();
        return 0;
    }

```

Fungsi program utama **main.cpp**:

1. Inisialisasi dan Validasi Input:

Program menginisialisasi FreeImage dan meminta input dari pengguna berupa path gambar yang akan dikompres, metode kalkulasi error, nilai threshold (dengan validasi sesuai metode), nilai minimum block size, serta path output untuk gambar terkompres dan file GIF. Validasi dilakukan untuk memastikan file ada dan ekstensi sesuai (input image: .jpg, .jpeg, atau .png; output GIF: .gif).

2. Pemrosesan Gambar:

- Gambar dimuat menggunakan FreeImage dan dikonversi ke 24-bit RGB.
- Data gambar disalin ke array satu dimensi sambil mengonversi urutan saluran dari BGR (yang diperoleh dari FreeImage) ke RGB.
- Ukuran file asli dihitung.

3. Pembangunan Quadtree dan Rekonstruksi:

- Program membangun struktur Quadtree berdasarkan data gambar dan parameter (threshold, minimum block size, dan metode error).
- Setelah Quadtree terbentuk, program merekonstruksi gambar terkompres menggunakan Quadtree tersebut.
- Program mengambil snapshot pada tiap tingkat (level) Quadtree menggunakan fungsi `captureFramesPerLevel`, kemudian menduplikasi frame terakhir beberapa kali agar tampilan akhir pada animasi GIF lebih lama.

4. Penyimpanan Output:

- Gambar terkompres disimpan ke file output dengan FreeImage (dengan konversi dari data mentah ke bitmap, kemudian disimpan sesuai format yang valid).

- File GIF animasi dibuat dari deretan frame (setiap frame mewakili tingkat pembentukan Quadtree) dengan durasi tertentu per frame.
5. Statistik dan Akhiri Program:
- Program mengukur waktu eksekusi untuk proses kompresi dan pembuatan GIF, menghitung ukuran file terkompres, serta persentase kompresi. Selanjutnya, ditampilkan statistik kompresi, kedalaman Quadtree, dan jumlah simpul. Terakhir, FreeImage dinonaktifkan dan program selesai.



Secara keseluruhan, main.cpp bertanggung jawab untuk menerima input, memproses gambar dengan kompresi berbasis Quadtree, menghasilkan output gambar terkompres serta animasi GIF yang menggambarkan proses pembentukan Quadtree, dan menampilkan hasil serta statistik kinerja kompresi.

BAB IV



EKSPERIMEN

4.1 Tangkapan Layar Eksperimen

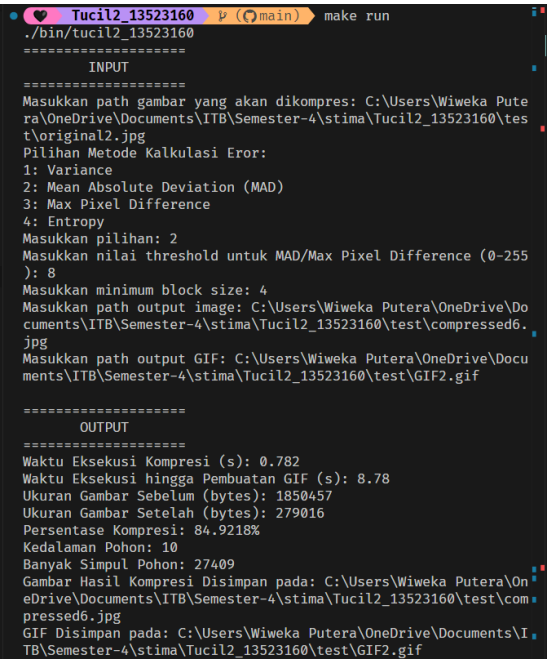

Berikut adalah tangkapan layar input dan output eksperimen. Gambar input dan output tersedia pada folder *test repository*.

No	Input/Output CLI	Output Image	Metode
1	<pre> • ❤️ Tucil2_13523160 P (main) make run ./bin/tucil2_13523160 ===== INPUT ===== Masukkan path gambar yang akan dikompres: C:\Users\Wiwe ka Putera\OneDrive\Documents\ITB\Semester-4\stima\Tucil 2_13523160\test\ori1.jpg Pilihan Metode Kalkulasi Error: 1: Variance 2: Mean Absolute Deviation (MAD) 3: Max Pixel Difference 4: Entropy Masukkan pilihan: 1 Masukkan nilai threshold untuk Variance (0-16384): 64 Masukkan minimum block size: 8 Masukkan path output image: C:\Users\Wiweka Putera\OneD rive\Documents\ITB\Semester-4\stima\Tucil2_13523160\tes t\compressed1.jpg ===== OUTPUT ===== Waktu Eksekusi (s): 0.48 Ukuran Gambar Sebelum (bytes): 1873941 Ukuran Gambar Setelah (bytes): 368706 Persentase Kompresi: 80.3246% Kedalaman Pohon: 9 Banyak Simpul Pohon: 27205 Gambar Hasil Kompresi Disimpan pada: C:\Users\Wiweka Pu tera\OneDrive\Documents\ITB\Semester-4\stima\Tucil2_135 23160\test\compressed1.jpg </pre>		Variance Threshold: 64 Min block size: 8
2	<pre> • ❤️ Tucil2_13523160 P (main) make run ./bin/tucil2_13523160 ===== INPUT ===== Masukkan path gambar yang akan dikompres: C:\Users\Wiwe ka Putera\OneDrive\Documents\ITB\Semester-4\stima\Tucil 2_13523160\test\ori1.jpg Pilihan Metode Kalkulasi Error: 1: Variance 2: Mean Absolute Deviation (MAD) 3: Max Pixel Difference 4: Entropy Masukkan pilihan: 2 Masukkan nilai threshold untuk MAD/Max Pixel Difference (0-255): 4 Masukkan minimum block size: 2 Masukkan path output image: C:\Users\Wiweka Putera\OneD rive\Documents\ITB\Semester-4\stima\Tucil2_13523160\tes t\compressed2.jpg ===== OUTPUT ===== Waktu Eksekusi (s): 0.545 Ukuran Gambar Sebelum (bytes): 1873941 Ukuran Gambar Setelah (bytes): 419564 Persentase Kompresi: 77.6106% Kedalaman Pohon: 11 Banyak Simpul Pohon: 213793 Gambar Hasil Kompresi Disimpan pada: C:\Users\Wiweka Pu tera\OneDrive\Documents\ITB\Semester-4\stima\Tucil2_135 23160\test\compressed2.jpg </pre>		MAD Threshold: 4 Min block size: 2

3	<pre> • Tucil2_13523160 P (main) make run ./bin/tucil2_13523160 ===== INPUT ===== Masukkan path gambar yang akan dikompres: C:\Users\Wiwe ka Putera\OneDrive\Documents\ITB\Semester-4\stima\Tucil 2_13523160\test\ori1.jpg Pilihan Metode Kalkulasi Error: 1: Variance 2: Mean Absolute Deviation (MAD) 3: Max Pixel Difference 4: Entropy Masukkan pilihan: 3 Masukkan nilai threshold untuk MAD/Max Pixel Difference (0-255): 4 Masukkan minimum block size: 2 Masukkan path output image: C:\Users\Wiweka Putera\OneD rive\Documents\ITB\Semester-4\stima\Tucil2_13523160\tes t\compressed3.jpg ===== OUTPUT ===== Waktu Eksekusi (s): 0.516 Ukuran Gambar Sebelum (bytes): 1873941 Ukuran Gambar Setelah (bytes): 438331 Persentase Kompresi: 76.6091% Kedalaman Pohon: 11 Banyak Simpul Pohon: 556689 Gambar Hasil Kompresi Disimpan pada: C:\Users\Wiweka Pu tera\OneDrive\Documents\ITB\Semester-4\stima\Tucil2_135 23160\test\compressed3.jpg </pre>		<p>Max Pixel Difference</p> <p>Threshold: 4</p> <p>Min block size: 2</p>
4	<pre> • Tucil2_13523160 P (main) make run ./bin/tucil2_13523160 ===== INPUT ===== Masukkan path gambar yang akan dikompres: C:\Users\Wiwe ka Putera\OneDrive\Documents\ITB\Semester-4\stima\Tucil 2_13523160\test\ori1.jpg Pilihan Metode Kalkulasi Error: 1: Variance 2: Mean Absolute Deviation (MAD) 3: Max Pixel Difference 4: Entropy Masukkan pilihan: 4 Masukkan nilai threshold untuk Entropy (0-8): 4 Masukkan minimum block size: 2 Masukkan path output image: C:\Users\Wiweka Putera\OneD rive\Documents\ITB\Semester-4\stima\Tucil2_13523160\tes t\compressed4.jpg ===== OUTPUT ===== Waktu Eksekusi (s): 0.605 Ukuran Gambar Sebelum (bytes): 1873941 Ukuran Gambar Setelah (bytes): 393445 Persentase Kompresi: 79.0044% Kedalaman Pohon: 11 Banyak Simpul Pohon: 106569 Gambar Hasil Kompresi Disimpan pada: C:\Users\Wiweka Pu tera\OneDrive\Documents\ITB\Semester-4\stima\Tucil2_135 23160\test\compressed4.jpg </pre>		<p>Entropy</p> <p>Threshold: 4</p> <p>Min block size: 2</p>

5	<pre> • ♥ Tucil2_13523160 P (main) make run ./bin/tucil2_13523160 ===== INPUT ===== Masukkan path gambar yang akan dikompres: C:\Users\Wiweka P utera\OneDrive\Documents\ITB\Semester-4\stima\Tucil2_135231 60\test\ori2.jpg Pilihan Metode Kalkulasi Error: 1: Variance 2: Mean Absolute Deviation (MAD) 3: Max Pixel Difference 4: Entropy Masukkan pilihan: 1 Masukkan nilai threshold untuk Variance (0-16384): 16 Masukkan minimum block size: 2 Masukkan path output image: C:\Users\Wiweka Putera\OneDrive \Documents\ITB\Semester-4\stima\Tucil2_13523160\test\compre ssed5.jpg ===== OUTPUT ===== Waktu Eksekusi (s): 1.096 Ukuran Gambar Sebelum (bytes): 1850457 Ukuran Gambar Setelah (bytes): 386866 Persentase Kompresi: 79.0935% Kedalaman Pohon: 11 Banyak Simpul Pohon: 234437 Gambar Hasil Kompresi Disimpan pada: C:\Users\Wiweka Putera \OneDrive\Documents\ITB\Semester-4\stima\Tucil2_13523160\te st\compressed5.jpg </pre>		Variance Threshold: 16 Min block size: 2
6	<pre> • ♥ Tucil2_13523160 P (main) make run ./bin/tucil2_13523160 ===== INPUT ===== Masukkan path gambar yang akan dikompres: C:\Users\Wiweka P utera\OneDrive\Documents\ITB\Semester-4\stima\Tucil2_135231 60\test\ori2.jpg Pilihan Metode Kalkulasi Error: 1: Variance 2: Mean Absolute Deviation (MAD) 3: Max Pixel Difference 4: Entropy Masukkan pilihan: 2 Masukkan nilai threshold untuk MAD/Max Pixel Difference (0- 255): 8 Masukkan minimum block size: 4 Masukkan path output image: C:\Users\Wiweka Putera\OneDrive \Documents\ITB\Semester-4\stima\Tucil2_13523160\test\compre ssed6.jpg ===== OUTPUT ===== Waktu Eksekusi (s): 0.872 Ukuran Gambar Sebelum (bytes): 1850457 Ukuran Gambar Setelah (bytes): 279016 Persentase Kompresi: 84.9218% Kedalaman Pohon: 10 Banyak Simpul Pohon: 27409 Gambar Hasil Kompresi Disimpan pada: C:\Users\Wiweka Putera \OneDrive\Documents\ITB\Semester-4\stima\Tucil2_13523160\te st\compressed6.jpg </pre>		MAD Threshold: 8 Min block size: 4

7	<pre> • Tucil2_13523160 (main) make run ./bin/tucil2_13523160 ===== INPUT ===== Masukkan path gambar yang akan dikompres: C:\Users\Wiweka P utera\OneDrive\Documents\ITB\Semester-4\stima\Tucil2_135231 60\test\ori2.jpg Pilihan Metode Kalkulasi Error: 1: Variance 2: Mean Absolute Deviation (MAD) 3: Max Pixel Difference 4: Entropy Masukkan pilihan: 3 Masukkan nilai threshold untuk MAD/Max Pixel Difference (0- 255): 8 Masukkan minimum block size: 4 Masukkan path output image: C:\Users\Wiweka Putera\OneDrive \Documents\ITB\Semester-4\stima\Tucil2_13523160\test\compre ssed7.jpg ===== OUTPUT ===== Waktu Eksekusi (s): 0.768 Ukuran Gambar Sebelum (bytes): 1850457 Ukuran Gambar Setelah (bytes): 400941 Persentase Kompresi: 78.3329% Kedalaman Pohon: 10 Banyak Simpul Pohon: 164089 Gambar Hasil Kompresi Disimpan pada: C:\Users\Wiweka Putera \OneDrive\Documents\ITB\Semester-4\stima\Tucil2_13523160\te st\compressed7.jpg </pre>		<p>Max Pixel Difference</p> <p>Threshold: 8</p> <p>Min block size: 4</p>
8	<pre> • Tucil2_13523160 (main) make run ./bin/tucil2_13523160 ===== INPUT ===== Masukkan path gambar yang akan dikompres: C:\Users\Wiweka Pute ra\OneDrive\Documents\ITB\Semester-4\stima\Tucil2_13523160\tes t\original1.jpg Pilihan Metode Kalkulasi Error: 1: Variance 2: Mean Absolute Deviation (MAD) 3: Max Pixel Difference 4: Entropy Masukkan pilihan: 1 Masukkan nilai threshold untuk Variance (0-16384): 64 Masukkan minimum block size: 8 Masukkan path output image: C:\Users\Wiweka Putera\OneDrive\Do cuments\ITB\Semester-4\stima\Tucil2_13523160\test\compressed1. jpg Masukkan path output GIF: C:\Users\Wiweka Putera\OneDrive\Docu ments\ITB\Semester-4\stima\Tucil2_13523160\test\GIF1.gif ===== OUTPUT ===== Waktu Eksekusi Kompresi (s): 0.82 Waktu Eksekusi hingga Pembuatan GIF (s): 10.894 Ukuran Gambar Sebelum (bytes): 1873941 Ukuran Gambar Setelah (bytes): 368706 Persentase Kompresi: 80.3246% Kedalaman Pohon: 9 Banyak Simpul Pohon: 27205 Gambar Hasil Kompresi Disimpan pada: C:\Users\Wiweka Putera\On eDrive\Documents\ITB\Semester-4\stima\Tucil2_13523160\test\com pressed1.jpg GIF Disimpan pada: C:\Users\Wiweka Putera\OneDrive\Documents\I TB\Semester-4\stima\Tucil2_13523160\test\GIF1.gif </pre>		<p>(GIF) Variance</p> <p>Threshold: 64</p> <p>Min block size: 8</p>

9	 <pre> Tucil2_13523160 (main) make run ./bin/tucil2_13523160 ===== INPUT ===== Masukkan path gambar yang akan dikompres: C:\Users\Wiweka Putera\OneDrive\Documents\ITB\Semester-4\stima\Tucil2_13523160\test\original2.jpg Pilihan Metode Kalkulasi Error: 1: Variance 2: Mean Absolute Deviation (MAD) 3: Max Pixel Difference 4: Entropy Masukkan pilihan: 2 Masukkan nilai threshold untuk MAD/Max Pixel Difference (0-255): 8 Masukkan minimum block size: 4 Masukkan path output image: C:\Users\Wiweka Putera\OneDrive\Documents\ITB\Semester-4\stima\Tucil2_13523160\test\compressed6.jpg Masukkan path output GIF: C:\Users\Wiweka Putera\OneDrive\Documents\ITB\Semester-4\stima\Tucil2_13523160\test\GIF2.gif ===== OUTPUT ===== Waktu Eksekusi Kompresi (s): 0.782 Waktu Eksekusi hingga Pembuatan GIF (s): 8.78 Ukuran Gambar Sebelum (bytes): 1850457 Ukuran Gambar Setelah (bytes): 279016 Persentase Kompresi: 84.9218% Kedalaman Pohon: 10 Banyak Simpul Pohon: 27409 Gambar Hasil Kompresi Disimpan pada: C:\Users\Wiweka Putera\OneDrive\Documents\ITB\Semester-4\stima\Tucil2_13523160\test\compressed6.jpg GIF Disimpan pada: C:\Users\Wiweka Putera\OneDrive\Documents\ITB\Semester-4\stima\Tucil2_13523160\test\GIF2.gif </pre>		<p>(GIF) MAD</p> <p>Threshold: 8 Min block size: 4</p>
---	--	--	---

4.2 Analisis

Dalam sembilan percobaan yang telah dilakukan, algoritma divide and conquer menggunakan pendekatan Quadtree terbukti efektif dalam mengurangi ukuran file gambar. Pada setiap percobaan, dengan variasi nilai threshold dan minimum blok, gambar berhasil dikompresi secara signifikan. Penggunaan metode perhitungan error (baik Variance, MAD, Max Pixel Difference, maupun Entropy) memungkinkan bagian-bagian gambar yang homogen diwakili oleh satu nilai rata-rata warna, sehingga data yang disimpan lebih ringkas. Hasil eksperimen menunjukkan bahwa meskipun detail di beberapa area tetap dipertahankan, keseluruhan ukuran file dapat menurun, yang menunjukkan bahwa algoritma ini efektif untuk kompresi gambar dalam konteks aplikasi yang dikembangkan. Analisis Kompleksitas Algoritma:

- Worst-case Complexity:

Jika gambar memiliki variasi warna yang sangat tinggi secara menyeluruh, algoritma mungkin akan terus membagi blok hingga mencapai ukuran minimum (misalnya, piksel individual). Dalam kasus ini, jumlah node yang dihasilkan bisa mendekati $O(N)$ di mana N adalah jumlah total piksel pada gambar. Proses rekursif untuk setiap blok terjadi secara bersamaan, sehingga total waktu eksekusi dapat mendekati linear terhadap jumlah piksel, meskipun dengan overhead rekursif.

- Average-case Complexity:

Untuk gambar yang memiliki area homogen, banyak blok tidak perlu dibagi lebih lanjut karena error sudah di bawah threshold. Hal ini mengakibatkan struktur quadtree menjadi lebih sederhana (dengan jumlah node yang jauh lebih sedikit daripada kasus worst-case) dan pemrosesan menjadi lebih cepat. Secara umum, kompleksitas algoritma dalam kasus rata-rata menjadi lebih rendah dibandingkan worst-case, dengan pembagian yang sebagian besar hanya dilakukan pada area dengan detail tinggi saja.

Secara keseluruhan, algoritma ini memiliki performa yang baik dalam praktik, terutama untuk gambar dengan banyak area homogen, sehingga meskipun dalam kasus paling buruk kompleksitasnya mendekati $O(N)$, dalam kebanyakan kondisi nyata, waktu eksekusi jauh lebih rendah berkat pembagian yang selektif.

LAMPIRAN

Pranala ke repository yang berisi kode program:

https://github.com/wiwekaputera/Tucil2_13523160

No	Poin	Ya	Tidak
1.	Program berhasil dikompilasi tanpa keasalahan	✓	
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	✓	