

LAPORAN TUGAS KECIL 1
IF2211 STRATEGI ALGORITMA

Penyelesaian IQ Puzzler Pro
dengan Algoritma *Brute Force*



Disusun oleh :
Carlo Angkisan - 13523091

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
JL. GANESHA 10, BANDUNG 40132
2025

DAFTAR ISI

DAFTAR ISI.....	2
BAB I.....	3
1.1 IQ Puzzler Pro.....	3
1.2 Algoritma Brute Force.....	3
1.3 Algoritma Penyelesaian IQ Puzzler Pro dengan Algoritma Brute Force.....	4
BAB II.....	6
3.1 Class Block.....	6
3.2 Class Board.....	7
3.3 Class PuzzleSolver.....	8
3.4 Class IO.....	9
3.5 Class CLI.....	10
3.6 Class GUI.....	10
BAB IV.....	12
4.1 Repositori Github.....	12
4.2 Struktur Program.....	12
4.3 Source Code Program.....	12
4.4 Antarmuka Program.....	30
BAB V.....	31
5.1 Kasus Uji 1.....	31
5.2 Kasus Uji 2.....	32
5.3 Kasus Uji 3.....	33
5.4 Kasus Uji 4.....	34
5.5 Kasus Uji 5.....	34
5.6 Kasus Uji 6.....	35
5.7 Kasus Uji 7.....	36
5.8 Kasus Uji 8.....	37
5.9 Kasus Uji 9.....	37
5.10 Kasus Uji 10.....	38
LAMPIRAN.....	39
Tabel Kelengkapan Spesifikasi.....	39

BAB I

DESKRIPSI MASALAH DAN ALGORITMA

1.1 IQ Puzzler Pro



Gambar 1 Permainan IQ Puzzler Pro
(Sumber: <https://www.smartgamesusa.com>)

IQ Puzzler Pro adalah permainan papan yang diproduksi oleh perusahaan Smart Games. Tujuan dari permainan ini adalah pemain harus dapat mengisi seluruh papan dengan piece (blok puzzle) yang telah tersedia. Komponen penting dari permainan IQ Puzzler Pro terdiri dari:

1. Board (Papan)

Board merupakan komponen utama yang menjadi tujuan permainan dimana pemain harus mampu mengisi seluruh area papan menggunakan blok-blok yang telah disediakan.

2. Blok/Piece

Blok adalah komponen yang digunakan pemain untuk mengisi papan kosong hingga terisi penuh. Setiap blok memiliki bentuk yang unik dan semua blok harus digunakan untuk menyelesaikan puzzle.

Permainan dimulai dengan papan yang kosong. Pemain dapat meletakkan blok puzzle sedemikian sehingga tidak ada blok yang bertumpang tindih (kecuali dalam kasus 3D). Setiap blok puzzle dapat dirotasikan maupun dicerminkan. Puzzle dinyatakan selesai jika dan hanya jika papan terisi penuh dan seluruh blok puzzle berhasil diletakkan.

1.2 Algoritma *Brute Force*

Algoritma *Brute Force* adalah metode penyelesaian masalah dengan mencoba semua kemungkinan kombinasi secara sistematis hingga menemukan solusi yang valid. Dalam konteks IQ Puzzler Pro, algoritma ini akan mengeksplorasi setiap kemungkinan penyusunan blok pada papan, termasuk berbagai rotasi dan pencerminan, hingga

menemukan konfigurasi yang memenuhi aturan permainan. Pendekatan ini memastikan bahwa solusi yang ditemukan benar-benar optimal karena seluruh kemungkinan diuji tanpa pengecualian.

Keunggulan utama *Brute Force* adalah kesederhanaannya dan jaminan menemukan solusi jika memang ada. Karena *Brute Force* tidak bergantung pada pola tertentu, algoritma ini dapat digunakan untuk berbagai jenis masalah tanpa memerlukan penyesuaian kompleks. Namun, kelemahannya adalah tidak efisien, terutama untuk masalah yang memiliki jumlah kemungkinan besar. Semakin banyak pilihan dan kombinasi yang harus diperiksa, semakin lama waktu komputasi yang dibutuhkan, sehingga metode ini sering kali tidak praktis untuk skala yang lebih besar.

Dalam laporan ini, **algoritma *Brute Force*** akan digunakan untuk menyelesaikan IQ Puzzler Pro, di mana seluruh kemungkinan konfigurasi penyusunan blok akan diperiksa satu per satu. Meskipun metode ini mungkin membutuhkan waktu yang cukup lama, pendekatan brute force menjamin bahwa solusi yang ditemukan benar-benar valid.

1.3 Algoritma Penyelesaian IQ Puzzler Pro dengan Algoritma Brute Force

Dalam menyelesaikan permainan IQ Puzzler Pro secara algoritmik, dapat menggunakan metode *Brute Force* yang dikombinasikan dengan teknik *backtracking*. Pendekatan ini memastikan bahwa setiap kemungkinan peletakan blok diuji secara menyeluruh. Adapun algoritma penyelesaian permainan ini sebagai berikut.

1. Representasikan papan *puzzle* sebagai sebuah *Board* matriks dengan ukuran baris dan kolom sesuai dengan masukan, dan kumpulan blok *puzzle* sebagai *List of Block* matriks dengan panjang dan lebar blok.
2. Program akan melakukan pencarian solusi dengan metode *backtracking* rekursif dimulai dari blok pertama pada *List of Block*.
3. Untuk setiap blok, program akan menghasilkan semua kemungkinan rotasi dan pencerminan dari blok tersebut.
4. Program mencoba menempatkan setiap variasi blok pada setiap posisi di papan, dimulai dari koordinat (0,0) hingga (rows-1, cols-1).
5. Setiap kali mencoba menempatkan blok, program memeriksa apakah penempatan valid, yaitu blok tidak tumpang tindih dengan blok lain atau keluar dari papan.
6. Jika penempatan valid, blok ditempatkan pada posisi tersebut dan program melanjutkan pencarian dengan blok berikutnya
7. Jika papan sudah penuh dan masih ada blok yang tersisa, maka solusi tersebut tidak valid dan program akan melakukan *backtrack* dengan menghapus blok yang baru saja ditempatkan.
8. Jika papan sudah penuh dan semua blok sudah digunakan, maka solusi telah ditemukan.

9. Jika tidak ada posisi valid yang ditemukan untuk suatu blok setelah mencoba semua variasi dan posisi, program akan kembali ke blok sebelumnya dan mencoba kemungkinan penempatan lainnya.
10. Program mencatat jumlah kasus yang diperiksa serta waktu eksekusi. Proses pencarian berakhir ketika solusi ditemukan atau seluruh kemungkinan telah dicoba tanpa menemukan solusi.

Algoritma tersebut dapat dituliskan dalam Pseudocode sebagai berikut.

```
FUNCTION solve(idx : integer) → boolean
{Fungsi rekursif untuk mencoba setiap kemungkinan penempatan blok}
  IF board is full THEN
    IF idx < blocks.length THEN
      → false
    → true

  IF idx = blocks.length THEN
    → false

  block ← blocks[idx]
  variations ← getAllRotationsAndFlips(block)
  i traversal [0..variations.length-1]
    currentBlock ← variations[i]
    x traversal [0..board.rows-1]
      y traversal [0..board.cols-1]
        IF canPlaceBlock(currentBlock, x, y) THEN
          totalCaseChecked ← totalCaseChecked + 1
          placeBlock(currentBlock, x, y)
          IF solve(idx + 1) THEN
            → true
          removeBlock(currentBlock, x, y)
        → false
    → false

PROCEDURE solvePuzzle()
{Prosedur utama untuk menyelesaikan puzzle}
  startTime ← getCurrentTime()
  isSolved ← solve(0)

  IF NOT isSolved THEN
    output("Solusi tidak ditemukan")
  ELSE
    output("Solusi ditemukan")
    printBoard(board)

  endTime ← getCurrentTime()
  duration ← endTime - startTime

  output("Waktu pencarian: " + duration + " ms")
  output("Banyak kasus yang ditinjau: " + totalCaseChecked)
```

BAB II

IMPLEMENTASI PROGRAM DENGAN BAHASA JAVA

Dalam implementasi program penyelesaian IQ Puzzler Pro menggunakan algoritma *Brute Force* dengan bahasa Java, pendekatan paradigma berorientasi objek (OOP) digunakan karena Java sendiri berbasis OOP. Dengan pendekatan ini, setiap komponen diorganisir dalam bentuk kelas (*class*) yang memiliki tanggung jawab spesifik, sehingga meningkatkan modularitas dan kemudahan dalam pengelolaan kode. Program ini tersedia dalam dua antarmuka, yaitu *Command Line Interface* (CLI) dan *Graphical User Interface* (GUI). Berikut adalah kelas-kelas yang digunakan dalam program ini.

3.1 Class Block

- Attribute

Nama	Tipe	Deskripsi
height	int	Menyimpan tinggi dalam dari blok.
width	int	Menyimpan lebar dari blok.
cells	char[][]	Matriks karakter yang merepresentasikan bentuk blok.

- Method

Nama	Tipe	Deskripsi
Block(char[][] cells)	Konstruktor	Menginisiasi objek <i>Block</i> dengan array matriks cells serta menetapkan nilai <i>height</i> dan <i>width</i> .
getHeight()	int	Mengembalikan tinggi dari blok.
getWidth()	int	Mengembalikan lebar dari blok.
getCells()	char[][]	Mengembalikan matriks <i>cells</i> dari blok.
rotateCW()	Block (private)	Mengembalikan objek <i>Block</i> baru yang merupakan hasil rotasi 90° searah jarum jam.
flipHorizontal()	Block (private)	Mengembalikan objek <i>Block</i> baru yang merupakan hasil pencerminan horizontal.
getAllRotationsAnd	List<Block>	Mengembalikan daftar semua rotasi dan

Flips()		pencerminan dari sebuah blok.
printCells()	void	Mencetak isi dari <i>cells</i> ke konsol dalam format matriks

3.2 Class Board

- Attribute

Nama	Tipe	Deskripsi
rows	int	Menyimpan jumlah baris pada board.
cols	int	Menyimpan jumlah kolom pada board.
board	char[][]	Matriks karakter yang merepresentasikan papan permainan (board).
COLORS	String[]	Array ANSI escape codes untuk warna tiap blok pada konsol.
RESET	String	ANSI escape code untuk mereset warna

- Method

Nama	Tipe	Deskripsi
Board(int rows, int cols, List<Block> blocks)	Konstruktor	Membuat papan dengan ukuran tertentu dan menginisialisasi dengan karakter '%' yang menandakan sel kosong.
getRows()	int	Mengembalikan jumlah baris dari papan.
getCols()	int	Mengembalikan jumlah kolom dari papan.
getBoard()	char[][]	Mengembalikan matriks Board kondisi papan saat ini.
getColors()	String()	Mengembalikan array warna yang digunakan untuk mencetak karakter.
setBoard(int x, int y, char c)	void	Menetapkan karakter c di posisi (x,y) pada papan.
printBoard()	void	Mencetak papan ke konsol dengan warna unik tiap blok.
isOutOfBounds(int row, int col)	boolean	Mengecek apakah posisi (row,col) berada di luar batas papan.

canPlaceBlock(Block block, int x, int y)	boolean	Mengecek apakah sebuah block dapat ditempatkan di posisi (x,y).
placeBlock(Block block, int x, int y)	void	Meletakkan block pada papan.
removeBlock(Block block, int x, int y)	void	Menghapus block dari posisi (x,y) pada papan.
isFull()	boolean	Mengecek apakah papan sudah penuh tanpa adanya sel kosong '%'.

3.3 Class PuzzleSolver

- Attribute

Nama	Tipe	Deskripsi
board	Board	Objek papan yang akan diisi oleh blok-blok.
blocks	List<Block>	Daftar blok yang akan digunakan untuk menyelesaikan puzzle.
totalCaseChecked	long	Jumlah total kasus yang diperiksa selama pencarian solusi.
duration	long	Waktu eksekusi dalam milidetik untuk menyelesaikan puzzle.
isSolved	boolean	Menunjukkan apakah puzzle telah diselesaikan atau tidak.

- Method

Nama	Tipe	Deskripsi
PuzzleSolver(Board board, List<Block> blocks)	Konstruktor	Membuat solver dengan papan dan daftar blok tertentu.
getTotalCaseChecked()	long	Mengembalikan jumlah kasus yang telah diperiksa.
getDuration()	long	Mengembalikan durasi eksekusi pencarian solusi.

getBoard()	Board	Mengembalikan objek papan permainan.
isSolved()	boolean	Mengembalikan status apakah puzzle sudah terpecahkan.
solve(int idx)	boolean	Algoritma rekursif yang mencoba menyusun blok pada papan.
solvePuzzle()	void	Memulai pencarian solusi, mengukur waktu eksekusi, dan mencetak hasil.

3.4 Class IO

- Attribute

Nama	Tipe	Deskripsi
board	Board	Objek papan yang akan diisi oleh blok-blok.
blocks	List<Block>	Daftar blok yang akan digunakan untuk menyelesaikan puzzle.
rows	int	Jumlah baris pada papan.
cols	int	Jumlah kolom pada papan.
totalBlock	int	Jumlah blok puzzle.
type	String	Tipe konfigurasi papan permainan.

- Method

Nama	Tipe	Deskripsi
getBlocks()	List<Block>	Mengembalikan array Block puzzle.
getType()	String	Mengembalikan tipe konfigurasi papan permainan.
getRows()	int	Mengembalikan jumlah baris pada papan.
getCols()	int	Mengembalikan jumlah kolom pada papan.
getTotalBlock()	int	Mengembalikan banyaknya block pada puzzle.
getBoard()	Board	Mengembalikan papan permainan.

readInputFile(String filePath)	void	Membaca masukan konfigurasi permainan.
processConfigCustom(BufferedReader br)	void	Membaca konfigurasi custom dari masukan.
processBlocks(BufferedReader br)	void	Membaca blok-blok puzzle dari masukan.
validateBlockCount()	void	Memvalidasi apakah jumlah blok sesuai dengan banyaknya blok pada masukan.
writeOutputFile(String filename, Board board, long duration, long totalCaseChecked)	void	Menulis keluaran ke dalam file .txt.
writeOutputImage(String filename, Board board)	void	Menyimpan keluaran ke dalam .png
convertAnsiToColor(String ansiColor)	Color	Mengonversi ANSI escape codes ke Color.

3.5 Class CLI

- Method

Nama	Tipe	Deskripsi
main(String[] args)	void	Method utama pada program CLI.

3.6 Class GUI

- Attribute

Nama	Tipe	Deskripsi
loadButton	JButton	Tombol untuk mengunggah file puzzle (.txt).
solveButton	JButton	Tombol untuk menyelesaikan puzzle.
savePngButton	JButton	Tombol untuk menyimpan solusi sebagai gambar (PNG).
saveTxtButton	JButton	Tombol untuk menyimpan solusi sebagai

		file teks (TXT).
boardPanel	JPanel	Panel untuk menampilkan papan puzzle.
statusLabel	JLabel	Label untuk menampilkan status proses penyelesaian puzzle.
io	IO	Objek untuk menangani input/output file.
solver	PuzzleSolver	Objek yang menangani logika penyelesaian puzzle.
board	Board	Objek yang merepresentasikan papan puzzle.
blocks	List<Block>	Daftar blok puzzle yang harus ditempatkan.
currentInputFileName	String	Nama input file puzzle yang sedang digunakan.

- Method

Nama	Tipe	Deskripsi
public GUI()	Konstruktor	Konstruktor untuk membuat antarmuka pengguna grafis (GUI).
drawBoard()	void	Menampilkan papan puzzle dalam boardPanel.
main(String[] args)	void	Method utama pada program GUI.

BAB IV

SOURCE CODE DAN STRUKTUR PROGRAM

4.1 Repositori Github

https://github.com/carllix/Tucil1_13523091

4.2 Struktur Program

Program memiliki struktur folder sebagai berikut.

- **src** : berisi source code program dalam bentuk file **.java**.
- **bin** : hasil kompilasi dalam bentuk file **.class** yang siap dijalankan.
- **test** : terdiri dari folder **input/** yang berisi file puzzle uji coba, serta **output/** yang menyimpan hasil penyelesaian dalam format teks dan gambar.
- **doc** : berisi laporan tugas kecil dan dokumentasi program.

4.3 Source Code Program

1. Block.java

```
package src;

import java.util.ArrayList;
import java.util.List;

public class Block {
    private int height, width;
    private char[][] cells;

    public Block(char[][] cells) {
        this.cells = cells;
        this.height = cells.length;
        this.width = cells[0].length;
    }

    public int getHeight() {
        return this.height;
    }

    public int getWidth() {
        return this.width;
    }

    public char[][] getCells() {
        return this.cells;
    }

    private Block rotateCW() {
        char[][] newCells = new char[width][height];
        for (int i = 0; i < width; i++) {
            for (int j = 0; j < height; j++) {
```

```

        newCells[i][j] = cells[height - j - 1][i];
    }
}
return new Block(newCells);
}

private Block flipHorizontal() {
    char[][] newCells = new char[height][width];
    for (int i = 0; i < height; i++) {
        for (int j = 0; j < width; j++) {
            newCells[i][j] = cells[i][width - j - 1];
        }
    }
    return new Block(newCells);
}

public List<Block> getAllRotationsAndFlips() {
    List<Block> rotationAndFlips = new ArrayList<>();
    Block current = this;

    for (int i = 0; i < 4; i++) {
        rotationAndFlips.add(current);
        rotationAndFlips.add(current.flipHorizontal());
        current = current.rotateCW();
    }

    return rotationAndFlips;
}

public void printCells() {
    for (int i = 0; i < height; i++) {
        for (int j = 0; j < width; j++) {
            System.out.print(cells[i][j] + " ");
        }
        System.out.println();
    }
}
}

```

2. Board.java

```

package src;

import java.util.List;

public class Board {
    private int rows, cols;
    private char[][] board;

    private static final String[] COLORS = {
        "\u001B[31m", "\u001B[32m", "\u001B[33m", "\u001B[34m",
        "\u001B[35m", "\u001B[36m",
        "\u001B[91m", "\u001B[92m", "\u001B[93m", "\u001B[94m",
        "\u001B[95m", "\u001B[96m",
    };
}

```

```

        "\u001B[41m", "\u001B[42m", "\u001B[43m", "\u001B[44m",
"\u001B[45m", "\u001B[46m",
        "\u001B[101m", "\u001B[102m", "\u001B[103m", "\u001B[104m",
"\u001B[105m", "\u001B[106m",
        "\u001B[48;5;208m", "\u001B[48;5;129m"
    };
    private static final String RESET = "\u001B[0m";

    public Board(int rows, int cols, List<Block> blocks) {
        this.rows = rows;
        this.cols = cols;
        this.board = new char[rows][cols];
        initializeEmptyBoard();
    }

    private void initializeEmptyBoard() {
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                board[i][j] = '%';
            }
        }
    }

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

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

    public char[][] getBoard() {
        return this.board;
    }

    public String[] getColors() {
        return COLORS;
    }

    public void setBoard (int x, int y, char c) {
        this.board[x][y] = c;
    }

    public void printBoard() {
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                char cell = board[i][j];

                if (cell >= 'A' && cell <= 'Z') {
                    int colorIdx = cell - 'A';
                    System.out.print(COLORS[colorIdx] + cell + RESET + " ");
                } else if (cell == '.') {
                    System.out.print("\u001B[30m" + cell + RESET + " ");
                } else {
                    System.out.print(cell + " ");
                }
            }
        }
    }

```

```

    }
    }
    System.out.println();
}

private boolean isOutOfBounds(int row, int col) {
    return row >= rows || col >= cols;
}

public boolean canPlaceBlock(Block block, int x, int y) {
    for (int i = 0; i < block.getHeight(); i++) {
        for (int j = 0; j < block.getWidth(); j++) {
            char[][] cells = block.getCells();
            if (cells[i][j] != '%') {
                if (isOutOfBounds(x + i, y + j) || board[x + i][y + j]
!= '%') {
                    return false;
                }
            }
        }
    }
    return true;
}

public void placeBlock(Block block, int x, int y) {
    for (int i = 0; i < block.getHeight(); i++) {
        for (int j = 0; j < block.getWidth(); j++) {
            char[][] cells = block.getCells();
            if (cells[i][j] != '%') {
                board[x + i][y + j] = cells[i][j];
            }
        }
    }
}

public void removeBlock(Block block, int x, int y) {
    for (int i = 0; i < block.getHeight(); i++) {
        for (int j = 0; j < block.getWidth(); j++) {
            char[][] cells = block.getCells();
            if (cells[i][j] != '%') {
                board[x + i][y + j] = '%';
            }
        }
    }
}

public boolean isFull() {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            if (board[i][j] == '%') {
                return false;
            }
        }
    }
}

```

```
        return true;
    }
}
```

3. PuzzleSolver.java

```
package src;

import java.util.List;

public class PuzzleSolver {
    private Board board;
    private List<Block> blocks;
    private long totalCaseChecked = 0;
    private long duration = 0;
    private boolean isSolved = false;

    public PuzzleSolver(Board board, List<Block> blocks) {
        this.board = board;
        this.blocks = blocks;
    }

    public long getTotalCaseChecked() {
        return this.totalCaseChecked;
    }

    public long getDuration() {
        return this.duration;
    }

    public Board getBoard() {
        return this.board;
    }

    public boolean isSolved() {
        return this.isSolved;
    }

    private boolean solve(int idx) {

        if (board.isFull()) {
            if (idx < blocks.size()) {
                return false;
            }
            return true;
        }

        if (idx == blocks.size()) {
            return false;
        }

        Block block = blocks.get(idx);
        List<Block> allRotationsAndFlips = block.getAllRotationsAndFlips();
        for (int i = 0; i < allRotationsAndFlips.size(); i++) {
            Block cells = allRotationsAndFlips.get(i);
```



```

        for (int x = 0; x < board.getRows(); x++) {
            for (int y = 0; y < board.getCols(); y++) {
                if (board.canPlaceBlock(cells, x, y)) {
                    totalCaseChecked++;
                    board.placeBlock(cells, x, y);

                    if (solve(idx + 1))
                        return true;

                    board.removeBlock(cells, x, y);
                }
            }
        }
    }
    return false;
}

public void solvePuzzle() {
    long startTime = System.currentTimeMillis();

    boolean isSolved = solve(0);
    this.isSolved = isSolved;

    if (!isSolved) {
        System.out.println("Solusi tidak ditemukan.");
    } else {
        System.out.println("Solusi ditemukan.");
        board.printBoard();
    }

    long endTime = System.currentTimeMillis();
    this.duration = endTime - startTime;

    System.out.println("\nWaktu pencarian: " + (this.duration) + " ms");
    System.out.println("\nBanyak kasus yang ditinjau: " +
this.totalCaseChecked);
}
}

```

4. IO.java

```

package src;

import java.io.*;
import java.util.*;
import java.awt.image.BufferedImage;
import java.awt.Color;
import java.awt.Graphics2D;
import javax.imageio.ImageIO;

public class IO {
    private List<Block> blocks = new ArrayList<>();
    private String type;
    private int rows, cols, totalBlock;
    private Board board;
}

```

```

    public List<Block> getBlocks() {
        return this.blocks;
    }

    public String getType() {
        return this.type;
    }

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

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

    public int getTotalBlock() {
        return this.totalBlock;
    }

    public Board getBoard() {
        return this.board;
    }

    private String getNextToken(StringTokenizer tokenizer, String fieldName)
    throws IOException {
        if (!tokenizer.hasMoreTokens()) {
            throw new IOException("Data tidak lengkap: " + fieldName + "
            tidak ditemukan.");
        }
        return tokenizer.nextToken();
    }

    public void readInputFile(String filePath) throws IOException {
        this.blocks.clear();

        BufferedReader br = new BufferedReader(new FileReader(filePath));

        try (br) {
            String firstLine = br.readLine();

            if (firstLine == null || firstLine.trim().isEmpty()) {
                throw new IOException("File kosong atau format tidak
                valid.");
            }

            StringTokenizer tokenizer = new StringTokenizer(firstLine);
            try {
                this.rows = Integer.parseInt(getNextToken(tokenizer,
                "rows"));
                this.cols = Integer.parseInt(getNextToken(tokenizer,
                "cols"));
                this.totalBlock = Integer.parseInt(getNextToken(tokenizer,
                "totalBlock"));
            } catch (NumberFormatException e) {

```

```

        throw new IOException("Format angka tidak valid dalam
file.");
    }

    String type = br.readLine();
    if (type == null || type.trim().isEmpty()) {
        throw new IOException("Tipe tidak boleh kosong.");
    }
    type = type.trim().toUpperCase();

    if (!type.equals("DEFAULT") && !type.equals("CUSTOM")) {
        throw new IOException("Tipe tidak valid. Harus 'DEFAULT'
atau 'CUSTOM'.");
    }
    this.type = type;

    this.board = new Board(this.rows, this.cols, this.blocks);
    if (type.equals("CUSTOM")) {
        processConfigCustom(br);
    }

    processBlocks(br);
}

private boolean isValidConfigCustomLine(String line) {
    for (int i = 0; i < line.length(); i++) {
        char c = line.charAt(i);
        if (c != '.' && c != 'X') {
            return false;
        }
    }
    return true;
}

private char[][] configCustomStringToMatrix(List<String> lines) throws
IOException {
    int width = lines.get(0).length();
    for (int i = 1; i < lines.size(); i++) {
        if (lines.get(i).length() != width) {
            throw new IOException("Lebar konfigurasi tidak sama.");
        }
    }

    char[][] matrix = new char[lines.size()][width];
    for (int i = 0; i < lines.size(); i++) {
        for (int j = 0; j < width; j++) {
            matrix[i][j] = lines.get(i).charAt(j);
        }
    }
    return matrix;
}

private void processConfigCustom(BufferedReader br) throws IOException {
    List<String> configLines = new ArrayList<>();

```

```

        String currentConfigLine;
        for (int i = 0; i < rows; i++) {
            currentConfigLine = br.readLine();
            if (!isValidConfigCustomLine(currentConfigLine)) {
                throw new IOException("Format konfigurasi tidak valid.");
            } else {
                configLines.add(currentConfigLine);
            }
        }

        char[][] matrixConfigCustom =
configCustomStringToMatrix(configLines);

        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                if (matrixConfigCustom[i][j] == '.') {
                    board.setBoard(i, j, '.');
                }
            }
        }
    }

    private void processBlocks(BufferedReader br) throws IOException {
        List<String> currentShapeLines = new ArrayList<>();
        char currentBlockId = '\\0';

        String currentLine;
        while ((currentLine = br.readLine()) != null) {
            char foundBlockId = findId(currentLine);

            if (foundBlockId != currentBlockId &&
!currentShapeLines.isEmpty()) {
                blocks.add(new Block(toCharMatrix(currentShapeLines)));
                currentShapeLines.clear();
            }

            currentShapeLines.add(currentLine);
            currentBlockId = foundBlockId;
        }

        if (!currentShapeLines.isEmpty()) {
            blocks.add(new Block(toCharMatrix(currentShapeLines)));
        }

        validateBlockCount();
    }

    private void validateBlockCount() throws IOException {
        if (blocks.size() != totalBlock) {
            throw new IOException("Jumlah block tidak sesuai, silahkan cek
kembali file input!");
        }
    }
}

```

```

private char findId(String line) {
    for (int i = 0; i < line.length(); i++) {
        if (Character.isLetter(line.charAt(i))) {
            return line.charAt(i);
        }
    }
    return '%';
}

private char[][] toCharMatrix(List<String> lines) {
    int maxWidth = 0;
    for (String line : lines) {
        if (line.length() > maxWidth) {
            maxWidth = line.length();
        }
    }

    char[][] matrix = new char[lines.size()][maxWidth];
    for (int i = 0; i < lines.size(); i++) {
        for (int j = 0; j < maxWidth; j++) {
            if (j < lines.get(i).length()) {
                matrix[i][j] = (lines.get(i).charAt(j) == ' ') ? '%' :
lines.get(i).charAt(j);
            } else {
                matrix[i][j] = '%';
            }
        }
    }

    return matrix;
}

public void writeOutputFile(String filename, Board board, long duration,
long totalCaseChecked) {
    String outputDir = "test/output/file/";
    File dir = new File(outputDir);
    if (!dir.exists()) {
        dir.mkdirs();
    }

    File outputFile = new File(outputDir + filename);

    try (BufferedWriter writer = new BufferedWriter(new
FileWriter(outputFile))) {
        char[][] boardState = board.getBoard();

        for (char[] row : boardState) {
            writer.write(new String(row));
            writer.newLine();
        }

        writer.newLine();
        writer.write("Waktu pencarian: " + duration + " ms");
        writer.newLine();
        writer.write("Banyak kasus yang ditinjau: " + totalCaseChecked);
    }
}

```

```

        writer.flush();

        System.out.println("Output berhasil disimpan di: " + outputDir +
filename);
    } catch (IOException e) {
        System.out.println("Terjadi kesalahan saat menulis file: " +
e.getMessage());
    }
}

public void writeOutputImage(String filename, Board board) {
    int cellSize = 50;
    int rows = board.getRows();
    int cols = board.getCols();
    char[][] boardState = board.getBoard();

    int width = cols * cellSize;
    int height = rows * cellSize;

    BufferedImage image = new BufferedImage(width, height,
BufferedImage.TYPE_INT_RGB);
    Graphics2D g2d = image.createGraphics();

    String[] colors = board.getColors();

    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            char cell = boardState[i][j];
            Color cellColor = Color.WHITE;

            if (cell >= 'A' && cell <= 'Z') {
                int colorIdx = cell - 'A';
                String ansiColor = colors[colorIdx];
                cellColor = convertAnsiToColor(ansiColor);
            } else if (cell == '.') {
                cellColor = Color.BLACK;
            }

            g2d.setColor(cellColor);
            g2d.fillRect(j * cellSize, i * cellSize, cellSize,
cellSize);

            g2d.setColor(Color.BLACK);
            g2d.drawRect(j * cellSize, i * cellSize, cellSize,
cellSize);
        }
    }

    g2d.dispose();

    String outputDir = "test/output/image/";
    File dir = new File(outputDir);
    if (!dir.exists()) {
        dir.mkdirs();
    }
}

```

```

        File outputFile = new File(outputDir + filename);

        try {
            ImageIO.write(image, "png", outputFile);
            System.out.println("Gambar berhasil disimpan di: " +
outputFile);
        } catch (IOException e) {
            System.out.println("Gagal menyimpan gambar: " + e.getMessage());
        }
    }

    public static Color convertAnsiToColor(String ansiColor) {
        switch (ansiColor) {
            case "\u001B[31m":
                return new Color(255, 0, 0);
            case "\u001B[32m":
                return new Color(0, 255, 0);
            case "\u001B[33m":
                return new Color(255, 255, 0);
            case "\u001B[34m":
                return new Color(0, 0, 255);
            case "\u001B[35m":
                return new Color(128, 0, 128);
            case "\u001B[36m":
                return new Color(0, 255, 255);
            case "\u001B[91m":
                return new Color(255, 102, 102);
            case "\u001B[92m":
                return new Color(102, 255, 102);
            case "\u001B[93m":
                return new Color(255, 255, 153);
            case "\u001B[94m":
                return new Color(102, 102, 255);
            case "\u001B[95m":
                return new Color(255, 102, 255);
            case "\u001B[96m":
                return new Color(102, 255, 255);
            case "\u001B[41m":
                return new Color(255, 0, 0);
            case "\u001B[42m":
                return new Color(0, 255, 0);
            case "\u001B[43m":
                return new Color(255, 255, 0);
            case "\u001B[44m":
                return new Color(0, 0, 255);
            case "\u001B[45m":
                return new Color(128, 0, 128);
            case "\u001B[46m":
                return new Color(0, 255, 255);
            case "\u001B[101m":
                return new Color(255, 102, 102);
            case "\u001B[102m":
                return new Color(102, 255, 102);
            case "\u001B[103m":

```

```

        return new Color(255, 255, 153);
    case "\u001B[104m":
        return new Color(102, 102, 255);
    case "\u001B[105m":
        return new Color(255, 102, 255);
    case "\u001B[106m":
        return new Color(102, 255, 255);
    case "\u001B[48;5;208m":
        return new Color(255, 140, 0);
    case "\u001B[48;5;129m":
        return new Color(139, 0, 139);
    default:
        return Color.WHITE;
    }
}
}

```

5. CLI.java

```

package src;

import java.io.IOException;
import java.util.List;
import java.util.Scanner;

public class CLI {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        IO io = new IO();

        System.out.println("=====");
        System.out.println("    Selamat Datang di IQ Puzzler Pro Solver");
        System.out.println("=====");

        System.out.print("\nMasukkan nama file input (contoh: 1.txt): ");
        String fileName = scanner.nextLine().trim();
        String filePath = "test/input/" + fileName;
        System.out.println();

        try {
            io.readInputFile(filePath);

            List<Block> blocks = io.getBlocks();
            Board board = io.getBoard();
            PuzzleSolver solver = new PuzzleSolver(board, blocks);
            solver.solvePuzzle();

            long duration = solver.getDuration();
            long totalCaseChecked = solver.getTotalCaseChecked();

            String choose = "";
            while (!choose.equalsIgnoreCase("Ya") &&
!choose.equalsIgnoreCase("Tidak")) {
                System.out.print("\nApakah Anda ingin menyimpan solusi?
(Ya/Tidak): ");
                choose = scanner.nextLine().trim();
            }
        }
    }
}

```



```

        if (!choose.equalsIgnoreCase("Ya") &&
!choose.equalsIgnoreCase("Tidak")) {
            System.out.println("\nInput tidak valid! Harap masukkan
'Ya' atau 'Tidak'.");
        }
    }

    if (choose.equalsIgnoreCase("Ya")) {
        int option = -1;
        while (option < 1 || option > 3) {
            System.out.println("\nPilih format penyimpanan:");
            System.out.println("1. File (TXT)");
            System.out.println("2. Image (PNG)");
            System.out.println("3. File (TXT) + Image (PNG)");
            System.out.print("\nPilihan (1-3): ");

            if (scanner.hasNextInt()) {
                option = scanner.nextInt();
                scanner.nextLine();
            } else {
                System.out.println("\nInput tidak valid! Harap
masukkan angka 1-3.");
                scanner.nextLine();
                continue;
            }

            switch (option) {
                case 1:
                    io.writeOutputFile(fileName, solver.getBoard(),
duration, totalCaseChecked);
                    break;
                case 2:
                    io.writeOutputImage(fileName.replace(".txt",
".png"), solver.getBoard());
                    break;
                case 3:
                    io.writeOutputFile(fileName, solver.getBoard(),
duration, totalCaseChecked);
                    io.writeOutputImage(fileName.replace(".txt",
".png"), solver.getBoard());
                    break;
                default:
                    System.out.println("\nPilihan tidak valid!
Silakan coba lagi.");
            }
        }
    } else {
        System.out.println("\nSolusi tidak disimpan.");
    }

} catch (IOException e) {
    System.out.println("\nTerjadi kesalahan saat membaca file: " +
e.getMessage());
}

```

```

        scanner.close();
        System.out.println("\nTerima kasih telah menggunakan IQ Puzzler Pro Solver!");
    }
}

```

6. GUI.java

```

package src;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.*;
import java.util.List;

public class GUI extends JFrame {
    private JButton loadButton, solveButton, savePngButton, saveTxtButton;
    private JPanel boardPanel;
    private JLabel statusLabel;
    private JLabel fileNameLabel;
    private IO io;
    private PuzzleSolver solver;
    private Board board;
    private List<Block> blocks;
    private String currentInputFileName;

    public GUI() {
        setTitle("IQ Puzzler Pro Solver");
        setSize(720, 720);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());

        JPanel topPanel = new JPanel(new BorderLayout());
        JPanel buttonPanel = new JPanel();
        loadButton = new JButton("Unggah File Puzzle (.txt)");
        solveButton = new JButton("Selesaikan");
        savePngButton = new JButton("Simpan sebagai PNG");
        saveTxtButton = new JButton("Simpan sebagai TXT");

        buttonPanel.add(loadButton);
        buttonPanel.add(solveButton);
        buttonPanel.add(savePngButton);
        buttonPanel.add(saveTxtButton);

        fileNameLabel = new JLabel("File: -");
        fileNameLabel.setBorder(BorderFactory.createEmptyBorder(5, 10, 5, 10));

        topPanel.add(buttonPanel, BorderLayout.CENTER);
        topPanel.add(fileNameLabel, BorderLayout.SOUTH);

        boardPanel = new JPanel();
        boardPanel.setLayout(new GridLayout(6, 6));
    }
}

```

```

        statusLabel = new JLabel(" ");

        add(topPanel, BorderLayout.NORTH);
        add(boardPanel, BorderLayout.CENTER);
        add(statusLabel, BorderLayout.SOUTH);

        io = new IO();

        loadButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                JFileChooser fileChooser = new JFileChooser();
                if (fileChooser.showOpenDialog(null) ==
JFileChooser.APPROVE_OPTION) {
                    File file = fileChooser.getSelectedFile();

                    currentInputFileName =
file.getName().replaceFirst("[.][^.]+" + "$", "");
                    fileNameLabel.setText("File: " + file.getName());

                    try {
                        board = null;
                        blocks = null;
                        solver = null;
                        boardPanel.removeAll();
                        boardPanel.revalidate();
                        boardPanel.repaint();

                        io.readInputFile(file.getAbsolutePath());
                        blocks = io.getBlocks();
                        board = io.getBoard();

                        if (board == null || blocks == null ||
blocks.isEmpty()) {
                            throw new IOException("File tidak valid atau
tidak berisi data yang cukup.");
                        }

                        int rows = board.getRows();
                        int cols = board.getCols();
                        boardPanel.setLayout(new GridLayout(rows, cols));
                        drawBoard();

                        JOptionPane.showMessageDialog(null, "File berhasil
dimuat.", "Success",
JOptionPane.INFORMATION_MESSAGE);
                    } catch (IOException ex) {
                        fileNameLabel.setText("File: -");
                        currentInputFileName = null;
                        JOptionPane.showMessageDialog(null, "Terjadi
kesalahan saat memuat file: " + ex.getMessage(),
"Error", JOptionPane.ERROR_MESSAGE);
                    }
                }
            }
        });

```

```

    }
    });

    solveButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            if (board != null && blocks != null) {
                solver = new PuzzleSolver(board, blocks);
                solver.solvePuzzle();
                boolean isSolved = solver.isSolved();

                if (!isSolved) {
                    statusLabel.setText("Waktu pencarian: " +
solver.getDuration() + " ms | Banyak kasus yang ditinjau: " +
solver.getTotalCaseChecked());
                    JOptionPane.showMessageDialog(null, "Solusi tidak
ditemukan.", "Solution",
                        JOptionPane.INFORMATION_MESSAGE);
                    return;
                }

                drawBoard();
                statusLabel.setText("Waktu pencarian: " +
solver.getDuration() + " ms | Banyak kasus yang ditinjau: " +
solver.getTotalCaseChecked());

                JOptionPane.showMessageDialog(null,
                    "Waktu pencarian: " + solver.getDuration() + "
ms\n" + "Banyak kasus yang ditinjau: "
                        + solver.getTotalCaseChecked(),
                    "Solution",
                    JOptionPane.INFORMATION_MESSAGE);
            } else {
                JOptionPane.showMessageDialog(null, "Tidak ada puzzle
yang dimuat untuk diselesaikan.", "Error",
                    JOptionPane.ERROR_MESSAGE);
            }
        }
    });

    savePngButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            if (solver != null && currentInputFileName != null) {
                io.writeOutputImage(currentInputFileName + ".png",
solver.getBoard());
                JOptionPane.showMessageDialog(null,
                    "Gambar berhasil disimpan di: " +
"test/output/image/" + currentInputFileName + ".png",
                    "Success",
                    JOptionPane.INFORMATION_MESSAGE);
            } else {
                JOptionPane.showMessageDialog(null, "Tidak ada solusi
yang dapat disimpan.", "Error",
                    JOptionPane.ERROR_MESSAGE);
            }
        }
    });

```

```

    }
}

});

saveTxtButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        if (solver != null && currentInputFileName != null) {
            io.writeOutputFile(currentInputFileName + ".txt",
solver.getBoard(), solver.getDuration(),
            solver.getTotalCaseChecked());
            JOptionPane.showMessageDialog(null,
                "Ouput berhasil disimpan di: " +
"test/output/file/" + currentInputFileName + ".txt",
                "Success",
                JOptionPane.INFORMATION_MESSAGE);
        } else {
            JOptionPane.showMessageDialog(null, "Tidak ada solusi
yang dapat disimpan.", "Error",
                JOptionPane.ERROR_MESSAGE);
        }
    }
});
}

private void drawBoard() {
    boardPanel.removeAll();

    if (board == null) {
        JOptionPane.showMessageDialog(null, "Board atau belum
diinisialisasi.", "Error",
            JOptionPane.ERROR_MESSAGE);
        return;
    }

    int rows = board.getRows();
    int cols = board.getCols();
    boardPanel.setLayout(new GridLayout(rows, cols));

    char[][] solution = board.getBoard();
    String[] colors = board.getColors();

    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            char cell = solution[i][j];
            if (cell == '%') {
                cell = ' ';
            }
            JLabel label = new JLabel(String.valueOf(cell),
SwingConstants.CENTER);
            label.setOpaque(true);

            if (cell >= 'A' && cell <= 'Z') {
                int colorIdx = cell - 'A';
                String ansiColor = colors[colorIdx];

```

```

        label.setBackground(IO.convertAnsiToColor(ansiColor));
    } else if (cell == '.') {
        label.setBackground(Color.BLACK);
    } else {
        label.setBackground(Color.WHITE);
    }
}

label.setBorder(BorderFactory.createLineBorder(Color.BLACK));
label.setFont(label.getFont().deriveFont(20f));
boardPanel.add(label);
    }
}

boardPanel.revalidate();
boardPanel.repaint();
}

public static void main(String[] args) {
    SwingUtilities.invokeLater(() -> {
        GUI gui = new GUI();
        gui.setVisible(true);
    });
}
}

```

4.4 Antarmuka Program

1. CLI

```

=====
Selamat Datang di IQ Puzzler Pro Solver
=====

Masukkan nama file input (contoh: 1.txt): 1.txt

Solusi ditemukan.
A G G G C
A A B C C
E E B B F
E E D F F
E D D F F

Waktu pencarian: 126 ms

Banyak kasus yang ditinjau: 27023

Apakah Anda ingin menyimpan solusi? (Ya/Tidak): Ya

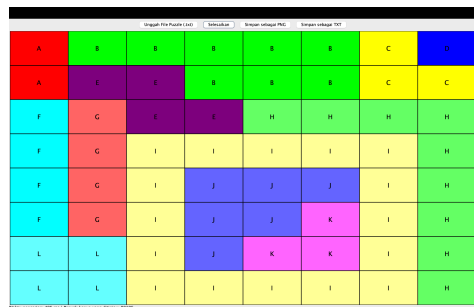
Pilih format penyimpanan:
1. File (TXT)
2. Image (PNG)
3. File (TXT) + Image (PNG)

Pilihan (1-3): 1
Output berhasil disimpan di: test/output/file/1.txt

Terima kasih telah menggunakan IQ Puzzler Pro Solver!
=====

```

2. GUI



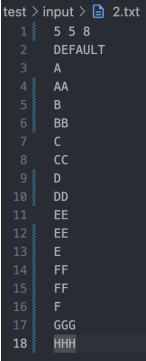
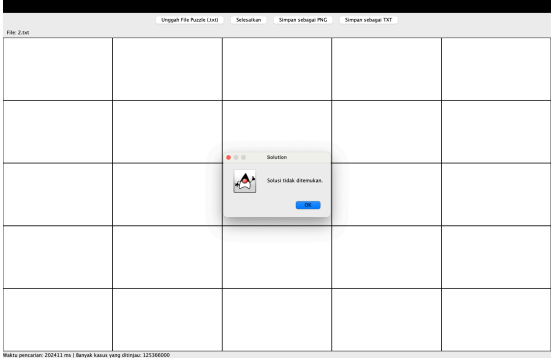
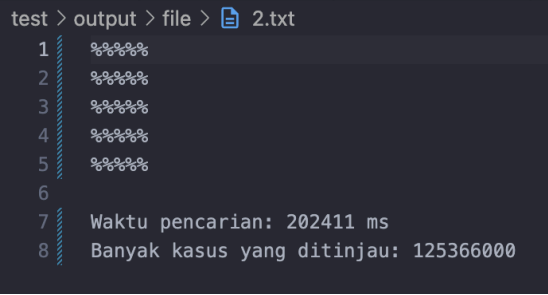
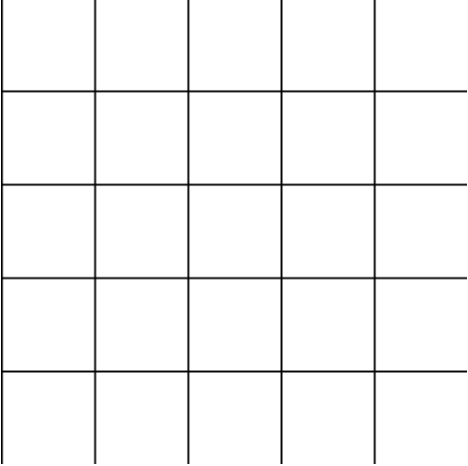
BAB V

EKSPERIMEN

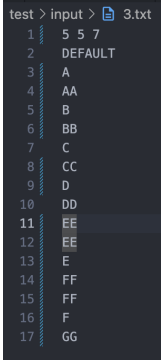
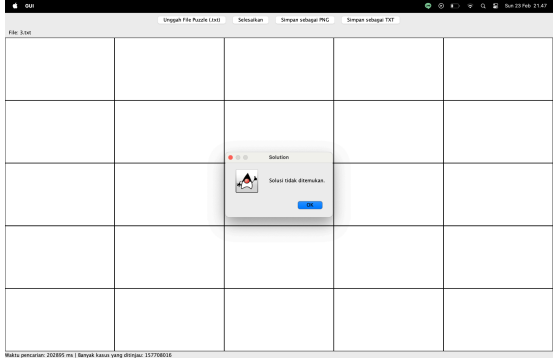
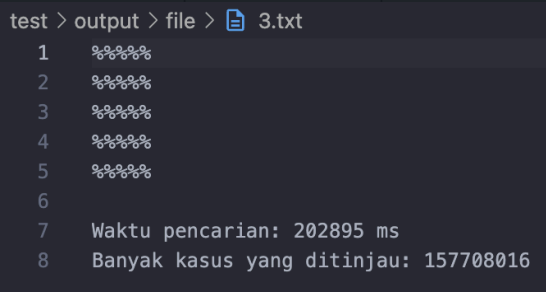
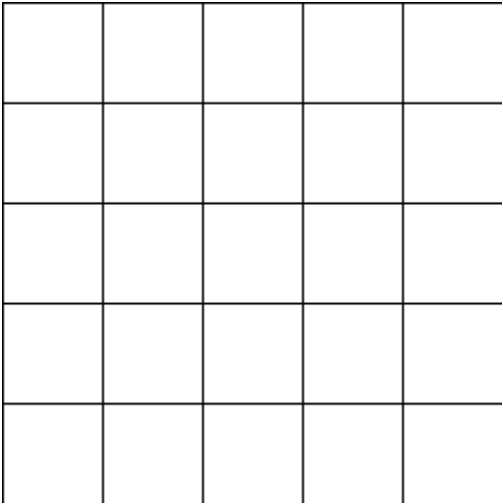
5.1 Kasus Uji 1

Input File	Output GUI
<pre>test > input > 1.txt 1 5 5 7 2 DEFAULT 3 A 4 AA 5 B 6 BB 7 C 8 CC 9 D 10 DD 11 EE 12 EE 13 E 14 FF 15 FF 16 F 17 GGG</pre>	
Output Text (.txt)	Output Image (.png)
<pre>test > output > file > 1.txt 1 AGGGC 2 AABCC 3 EEBBF 4 EEDFF 5 EDDFF 6 7 Waktu pencarian: 127 ms 8 Banyak kasus yang ditinjau: 27023</pre>	
<p>Keterangan:</p> <p>Pengujian dilakukan menggunakan kasus uji normal dengan tipe konfigurasi DEFAULT yang terdapat pada spesifikasi.</p>	

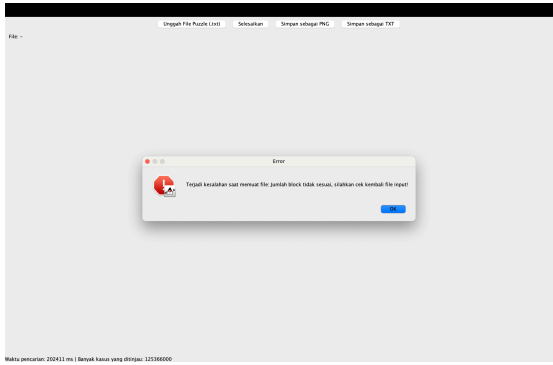
5.2 Kasus Uji 2

Input File	Output GUI
	
Output Text (.txt)	Output Image (.png)
	
Keterangan:	
<p>Pengujian dilakukan menggunakan kasus uji yang seharusnya dapat diselesaikan, tetapi masih terdapat blok atau potongan berlebih, sehingga tidak memungkinkan terbentuknya solusi yang valid. Pada GUI, akan muncul pop-up dengan pesan "<i>Solusi Tidak Ditemukan Solution</i>". Jika hasilnya disimpan ke dalam file, papan akan ditampilkan dalam kondisi kosong yang disimbolkan dengan karakter '%'. Begitu pula pada output gambar, akan dihasilkan papan kosong.</p>	

5.3 Kasus Uji 3

Input File	Output GUI
 <pre>test > input > 3.txt 1 5 5 7 2 DEFAULT 3 A 4 AA 5 B 6 BB 7 C 8 CC 9 D 10 DD 11 EE 12 EE 13 E 14 FF 15 FF 16 F 17 GG</pre>	
Output Text (.txt)	Output Image (.png)
 <pre>test > output > file > 3.txt 1 %%% 2 %%% 3 %%% 4 %%% 5 %%% 6 7 Waktu pencarian: 202895 ms 8 Banyak kasus yang ditinjau: 157708016</pre>	
Keterangan:	
Pengujian dilakukan dengan menggunakan kasus uji yang tidak dapat diselesaikan karena masih terdapat sel kosong pada papan.	

5.4 Kasus Uji 4

Input File	Output GUI
<pre>test > input > 4.txt 1 5 5 8 2 DEFAULT 3 A 4 AA 5 B 6 BB 7 C 8 CC 9 D 10 DD 11 EE 12 EE 13 E 14 FF 15 FF 16 F 17 GGG</pre>	
<p>Keterangan:</p> <p>Pengujian dilakukan dengan kasus uji yang memiliki jumlah blok yang tidak sesuai dengan banyaknya blok yang terdapat pada file input. Dapat dilihat pada file masukan seharusnya memiliki jumlah blok sebanyak 8, tetapi hanya terdapat 7 blok.</p>	

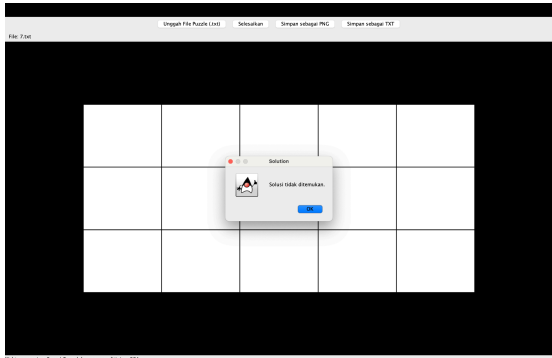
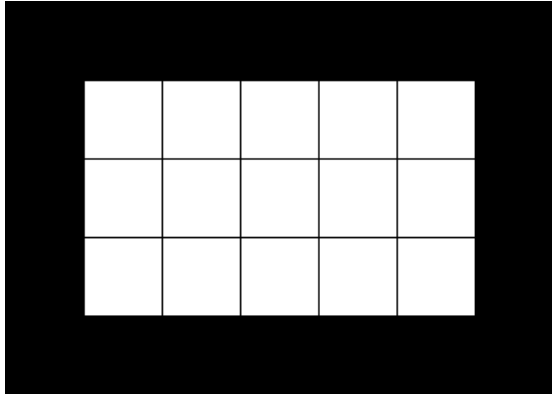
5.5 Kasus Uji 5

Input File	Output GUI
<pre>test > input > 5.txt 1 5 5 7 2 LINGKARAN 3 A 4 AA 5 B 6 BB 7 C 8 CC 9 D 10 DD 11 EE 12 EE 13 E 14 FF 15 FF 16 F 17 GGG</pre>	
<p>Keterangan:</p> <p>Pengujian dilakukan dengan kasus uji yang memiliki tipe tidak valid (selain DEFAULT dan CUSTOM)</p>	

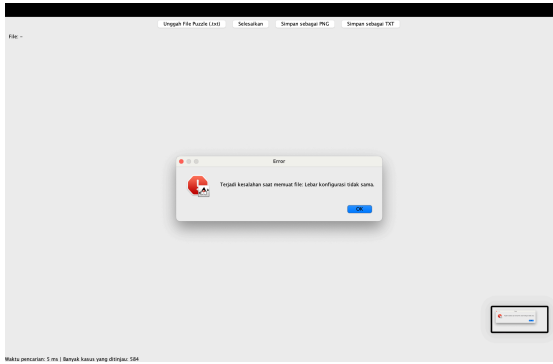
5.6 Kasus Uj 6

Input File	Output GUI
<pre>test > input > 6.txt 1 5 7 5 2 CUSTOM 3 ...X... 4 .XXXXX. 5 XXXXXXX 6 .XXXXX. 7 ...X... 8 A 9 AAA 10 BB 11 BBB 12 CCCC 13 C 14 D 15 EEE 16 E</pre>	
Output Text (.txt)	Output Image (.png)
<pre>test > output > file > 6.txt 1 ...A... 2 .BBAAA. 3 BBBCCCC 4 .EEECD. 5 ...E... 6 7 Waktu pencarian: 8 ms 8 Banyak kasus yang ditinjau: 217</pre>	
Keterangan:	
Pengujian dilakukan menggunakan kasus uji normal dengan tipe konfigurasi CUSTOM yang terdapat pada spesifikasi.	

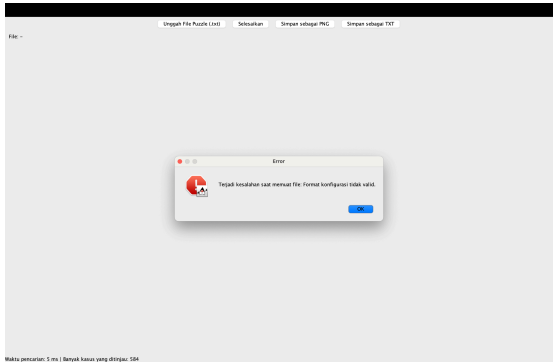
5.7 Kasus Uji 7

Input File	Output GUI
<pre> test > input > 7.txt 1 5 7 5 2 CUSTOM 3 4 .XXXXX. 5 .XXXXX. 6 .XXXXX. 7 8 A 9 AAA 10 BB 11 BBB 12 CCCC 13 C 14 D 15 EEE 16 E </pre>	
Output Text (.txt)	Output Image (.png)
<pre> test > output > file > 7.txt 1 2 .%-%-%-%-%. 3 .%-%-%-%-%. 4 .%-%-%-%-%. 5 6 7 Waktu pencarian: 5 ms 8 Banyak kasus yang ditinjau: 584 </pre>	
Keterangan:	
<p>Pengujian dilakukan menggunakan kasus uji yang tidak memiliki solusi dengan tipe konfigurasi CUSTOM. Pada GUI, akan muncul pop-up dengan pesan "<i>Solusi Tidak Ditemukan Solution</i>". Jika hasilnya disimpan ke dalam file, papan akan ditampilkan dalam kondisi kosong yang disimbolkan dengan karakter ‘%’ dengan ‘.’ merupakan hasil konfigurasi <i>custom</i> yang tidak boleh diisi. Begitu pula pada gambar keluaran, akan dihasilkan papan kosong.</p>	


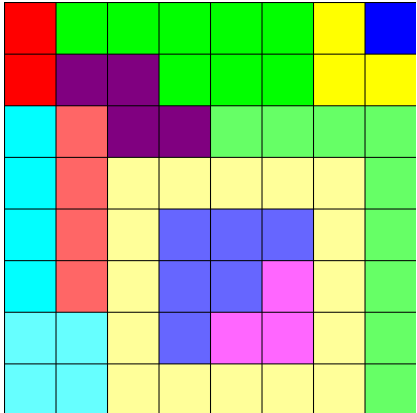
5.8 Kasus Uj 8

Input File	Output GUI
<pre>test > input > 8.txt 1 5 7 5 2 CUSTOM 3 ...X... 4 .XXX 5 XXXXXX 6 .XXXXX. 7 ...X... 8 A 9 AAA 10 BB 11 BBB 12 CCCC 13 C 14 D 15 EEE 16 E</pre>	
<p>Keterangan:</p> <p>Pengujian dilakukan menggunakan kasus uji dengan konfigurasi papan CUSTOM yang rusak (tidak memiliki lebar yang sama/tidak valid)</p>	

5.9 Kasus Uji 9

Input File	Output GUI
<pre>test > input > 9.txt 1 5 7 5 2 CUSTOM 3 ...X... 4 .XXXXX. 5 XXXXXX 6 A 7 AAA 8 BB 9 BBB 10 CCCC 11 C 12 D 13 EEE 14 E</pre>	
<p>Keterangan:</p> <p>Pengujian dilakukan menggunakan kasus uji dengan konfigurasi papan CUSTOM yang kurang (tidak sesuai ukuran papan).</p>	

5.10 Kasus Uji 10

Input File	Output GUI
<pre>test > input > 10.txt 1 8 8 12 2 DEFAULT 3 A 4 A 5 BBBB 6 BBB 7 C 8 CC 9 D 10 EE 11 EE 12 FFFF 13 GGGG 14 HHHH 15 H 16 H 17 H 18 H 19 H 20 IIIII 21 I I 22 I I 23 I I 24 IIIII 25 JJJ 26 JJ 27 J 28 KK 29 K 30 LL 31 LL</pre>	
Output Text (.txt)	Output Image (.png)
<pre>test > output > file > 10.txt 1 ABBBBBCD 2 AEEBBCC 3 FGEEHHH 4 FGIIIIIH 5 FGIJJJIH 6 FGIJJKIH 7 LLIJKKIH 8 LLIIIIIIH 9 10 Waktu pencarian: 403 ms 11 Banyak kasus yang ditinjau: 90195</pre>	
Keterangan:	
Pengujian dilakukan menggunakan kasus uji normal dengan tipe konfigurasi DEFAULT .	

LAMPIRAN

Tabel Kelengkapan Spesifikasi

No	Poin	Ya	Tidak
1	Program berhasil dikompilasi tanpa kesalahan	✓	
2	Program berhasil dijalankan	✓	
3	Solusi yang diberikan program benar dan mematuhi aturan permainan	✓	
4	Program dapat membaca masukan berkas .txt serta menyimpan solusi dalam berkas .txt	✓	
5	Program memiliki <i>Graphical User Interface</i> (GUI)	✓	
6	Program dapat menyimpan solusi dalam bentuk file gambar	✓	
7	Program dapat menyelesaikan kasus konfigurasi <i>custom</i>	✓	
8	Program dapat menyelesaikan kasus konfigurasi Piramida (3D)		✓
9	Program dibuat oleh saya sendiri	✓	