

LAPORAN TUGAS KECIL 1
IF2211 STRATEGI ALGORITMA

Penyelesaian IQ Puzzler Pro Menggunakan Algoritma *Brute Force*



Disusun Oleh:
Adinda Putri
13523071
K-01

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2025

1. Pendahuluan

1.1. Algoritma Brute Force

Brute force merupakan algoritma yang bersifat straightforward dalam menyelesaikan suatu persoalan [1]. Brute force secara sistematis mengeksplorasi semua cara hingga solusi atas suatu permasalahan ditemukan. Dengan kata lain, algoritma ini tidak mempertimbangkan efisiensi. Oleh karena itu, algoritma ini bersifat sangat sederhana, langsung, dan jelas [1]. Beberapa contoh dari algoritma brute force adalah searching min-max, sequential search, perkalian dua buah matriks $n \times n$, bubble sort, dan lain-lain [1].

1.2. IQ Puzzler Pro

IQ Puzzler Pro merupakan permainan papan yang diproduksi oleh perusahaan Smart Games [1]. Tujuan dari permainan ini adalah pemain harus dapat mengisi seluruh papan dengan piece (blok puzzle) yang telah tersedia [1]. IQ Puzzler Pro terdiri dari dua komponen utama, yaitu:

1. Board (Papan) : Komponen utama yang menjadi tujuan permainan, dan menjadi tempat pengisian semua blok-blok yang telah disediakan [1].
2. Blok (Piece) : Blok adalah komponen yang digunakan pemain untuk mengisi papan kosong hingga terisi penuh [1]. Setiap blok memiliki bentuk yang unik dan semua blok harus digunakan untuk menyelesaikan puzzle [1].



Gambar 1 Papan IQ Puzzler Pro yang Penuh dengan Piece
(Sumber: <https://www.smartgamesusa.com>)

Untuk menemukan satu solusi dari IQ Puzzler Pro, atau menentukan bisa atau tidaknya board diisi penuh ketika tersedia piece dengan bentuk dan jumlah tertentu, dapat digunakan algoritma brute force, lalu algoritma tersebut dapat diimplementasikan dalam program berbahasa Java.

2. Algoritma Program

2.1 Algoritma IQ Puzzler Pro Solver

Algoritma brute force dalam IQ Puzzler Pro Solver diimplementasikan dengan menggunakan pendekatan backtracking untuk mencari solusi dari puzzle. Tujuan penggunaan algoritma ini adalah menempatkan semua *puzzle piece* pada *board* tanpa tumpang tindih dan sesuai dengan ukuran *board*. Algoritma dimulai dengan memproses *puzzle piece* pertama pada senarai berisi seluruh *puzzle piece* yang diinput oleh *user* dan mencoba setiap kemungkinan posisi pada *board*. Untuk setiap posisi (x, y), algoritma brute force memeriksa seluruh orientasi unik dari *puzzle piece* tersebut, termasuk rotasi dan *flip*. Setelah itu, digunakan metode *canPlacePiece* untuk memverifikasi apakah *puzzle piece* dapat ditempatkan pada posisi tertentu tanpa menabrak *puzzle piece* lain atau sel yang diblokir pada *board*.

Jika suatu orientasi *puzzle piece* dapat diletakkan pada posisi yang valid, metode *placePiece* akan digunakan untuk menempatkannya ke *board*. Setelah berhasil menempatkan *puzzle piece*, algoritma dilanjutkan secara rekursif ke *puzzle piece* berikutnya. Proses ini terus berlanjut hingga seluruh *puzzle piece* berhasil ditempatkan. Apabila semua *puzzle piece* telah diproses (ditandai dengan indeks yang mencapai jumlah total *puzzle piece*), metode *isBoardFull* akan memeriksa apakah *board* telah terisi penuh. Jika *board* telah penuh, fungsi akan mengembalikan nilai *true* sebagai tanda bahwa solusi telah ditemukan.

Jika tidak ada orientasi atau posisi yang memungkinkan untuk menempatkan *puzzle piece* tertentu, algoritma akan melakukan backtracking dengan memanggil metode *removePiece* untuk menghapus *puzzle piece* yang sebelumnya diletakkan. Proses ini memungkinkan algoritma untuk mencari alternatif penempatan lainnya yang mungkin belum dicoba. Selama pencarian solusi, waktu eksekusi direkam dan variabel *iterations* akan di-*increment* untuk menghitung jumlah percobaan penempatan yang telah dilakukan, sehingga banyaknya kasus percobaan dapat ditinjau.

Pendekatan brute force memastikan semua konfigurasi yang mungkin diperiksa hingga ditemukan solusi atau semua kemungkinan habis diperiksa. Meskipun metode ini dapat menjamin pencarian solusi yang lengkap, waktu komputasi bisa menjadi sangat lama tergantung pada ukuran *board*, jumlah *puzzle piece*, dan kerumitan bentuk *puzzle piece*. Namun, penggunaan backtracking dalam algoritma ini membantu memangkas pencarian dengan membatalkan jalur yang tidak mungkin secepat mungkin, sehingga lebih efisien dibandingkan pengecekan semua kombinasi secara langsung tanpa optimasi.

2.2 Pseudocode Algoritma

Pseudocode dari algoritma IQ Puzzler Pro Solver dengan brute force adalah sebagai berikut.

```
function (board : Board, pieces : list of Piece, index : integer, iterations : array of
integer) -> boolean
    KAMUS LOKAL
    piece : Piece
    uniqueOrientations : set of Piece

    if (index = pieces.size()) then
        -> board.isBoardFull()
    piece <- pieces.get(index)
    uniqueOrientations <- piece.getUniqueOrientations()

    orientation traversal uniqueOrientations
        y traversal [0.. (board.getHeight() - orientation.getRow())
            i traversal [0.. (board_width - orientation.getCol())
                iterations[0] <- iterations[0] + 1

                if (board.canPlacePiece(orientation, x, y) then
                    board.placePiece(orientation, x,y)

                    if (solve(board, pieces, index+1, iterations)) then
                        -> true
                    board.removePiece(orientation, x, y)

    -> false
```

3. Source Code Program

Source code program IQ Puzzler Pro Solver ditulis dalam bahasa Java. Program ini terdiri dari dua folder, yaitu folder models dan utils, serta satu file sebagai program utama yaitu `Main.java`. Folder models berisi *source code* kelas-kelas yang digunakan dalam implementasi algoritma brute force. Sementara itu, folder utils berisi *source code* yang meng-handle kepentingan input dan output. Berikut merupakan *directory tree* dari program ini.

```
|— src
|   |— models
|   |   |— Board.java
|   |   |— Piece.java
|   |   |— Solver.java
|   |— utils
|   |   |— InputHandler.java
|   |   |— SolutionHandler.java
|   |— Main.java
```

2.1. Board.java

Berikut ini merupakan attribute dan method yang terdapat dalam class Board.

Attribute	Tipe Data	Keterangan
height	private iint	Attribut tinggi papan
width	private iint	Attribut lebar papan
board	private char[][]	Attribute papan
ansiColorMap	private static Map<Character, String>	Attribute untuk mapping character ke string
colorMap	private static final Map<Character, Color>	Attribute untuk mapping character ke color

Method	Returned Data Type	Keterangan
Board(int N, int M)	public	Konstruktor papan dengan lebar N dan tinggi M
setRow(int rowIndex, String line)	public void	Methode untuk membuat custom board
getColorMap()	public static Map<Character, Color>	Untuk mengakses color map
getWidth()	public int	Untuk mengakses lebar papan
getHeight()	public int	Untuk mengakses tinggi papan
getBoard()	public char[][]	Untuk mengakses papan
isBoardFull()	public boolean	Untuk mengecek penuh atau tidaknya papan
canPlacePiece(Piece piece, int x, int y)	public boolean	Untuk mengecek apakah dapat menempatkan piece pada papan
placePiece(Piece piece, int x,	public void	Untuk menempatkan piece

int y)		pada papan
removePiece(Piece piece, int x, int y)	public void	Untuk mengeluarkan piece dari papan
printBoard()	public void	Untuk menampilkan keadaan papan

Source code file piece.java adalah sebagai berikut.

```
package models;

import java.awt.*;
import models.Board;
import java.util.Arrays;
import java.util.HashMap;
import java.util.Map;

public class Board {
    // attributes
    private int height, width;
    private char[][] board;
    private static Map<Character, String> ansiColorMap = new HashMap<>();
    private static final Map<Character, Color> colorMap = new HashMap<>();

    // constructor
    public Board(int N, int M) {
        this.height = N;
        this.width = M;
        this.board = new char[N][M];
        for (int i = 0; i < height; i++) {
            Arrays.fill(board[i], '.');
        }
    }

    // color mapping
    static {
        char[] letters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ".toCharArray();
        String[] pastelAnsiColors = {
            "\u001B[38;5;217m",
            "\u001B[38;5;151m",
            "\u001B[38;5;153m",
            "\u001B[38;5;222m",
            "\u001B[38;5;225m",
            "\u001B[38;5;144m",
        };
    }
}
```

```

        "\u001B[38;5;159m",
        "\u001B[38;5;186m",
        "\u001B[38;5;182m",
        "\u001B[38;5;176m",
        "\u001B[38;5;114m",
        "\u001B[38;5;174m",
        "\u001B[38;5;181m",
        "\u001B[38;5;225m",
        "\u001B[38;5;157m",
        "\u001B[38;5;180m",
        "\u001B[38;5;178m",
        "\u001B[38;5;147m",
        "\u001B[38;5;219m",
        "\u001B[38;5;186m",
        "\u001B[38;5;216m",
        "\u001B[38;5;156m",
        "\u001B[38;5;138m",
        "\u001B[38;5;117m",
        "\u001B[38;5;224m",
        "\u001B[38;5;226m"
    };

    Color[] colors = {
        new Color(255, 99, 132), // Red
        new Color(54, 162, 235), // Blue
        new Color(75, 192, 192), // Teal
        new Color(255, 206, 86), // Yellow
        new Color(153, 102, 255), // Purple
        new Color(255, 159, 64), // Orange
        new Color(201, 203, 207), // Light Gray
        new Color(255, 0, 0), // Bright Red
        new Color(0, 255, 0), // Bright Green
        new Color(0, 0, 255), // Bright Blue
        new Color(255, 255, 0), // Bright Yellow
        new Color(0, 255, 255), // Cyan
        new Color(255, 0, 255), // Magenta
        new Color(128, 0, 128), // Dark Purple
        new Color(255, 165, 0), // Dark Orange
        new Color(46, 139, 87), // Sea Green
        new Color(0, 206, 209), // Turquoise
        new Color(139, 69, 19), // Saddle Brown
        new Color(70, 130, 180), // Steel Blue
        new Color(219, 112, 147), // Pale Violet Red
        new Color(240, 230, 140), // Khaki
        new Color(210, 105, 30), // Chocolate
        new Color(154, 205, 50), // Yellow Green
        new Color(233, 150, 122), // Dark Salmon
        new Color(72, 61, 139), // Dark Slate Blue
    };

```

```

        new Color(244, 164, 96)    // Sandy Brown
    };

    for (int i = 0; i < letters.length; i++) {
        colorMap.put(letters[i], colors[i % colors.length]);
    }
    for (int i = 0; i < letters.length; i++) {
        ansiColorMap.put(letters[i], pastelAnsiColors[i % pastelAnsiColors.length]);
    }
}

// methods

// custom mode
public void setRow(int rowIndex, String line) {
    for (int j = 0; j < width; j++) {
        board[rowIndex][j] = (line.charAt(j) == 'X') ? '.' : '#';
    }
}

public static Map<Character, Color> getColorMap() {
    return colorMap;
}

public int getWidth() {
    return width;
}

public int getHeight() {
    return height;
}

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

// return true if the board is full, false if not
public boolean isBoardFull() {
    for (int i = 0; i < height; i++) {
        for (int j = 0; j < width; j++) {
            if (board[i][j] == '.') {
                return false;
            }
        }
    }
    return true;
}

```



```

// check if the piece can fit the board
public boolean canPlacePiece(Piece piece, int x, int y) {
    char[][] shape = piece.getPiece();
    for (int i = 0; i < piece.getRow(); i++) {
        for (int j = 0; j < piece.getCol(); j++) {
            if (shape[i][j] != '.') {
                if (y + i >= height || x + j >= width || board[y + i][x + j] != '.') {
                    return false;
                }
            }
        }
    }
    return true;
}

// place the piece into the board if can
public void placePiece(Piece piece, int x, int y) {
    char[][] shape = piece.getPiece();
    for (int i = 0; i < piece.getRow(); i++) {
        for (int j = 0; j < piece.getCol(); j++) {
            if (shape[i][j] != '.' && board[y + i][x + j] == '.') {
                board[y + i][x + j] = shape[i][j];
            }
        }
    }
}

public void removePiece(Piece piece, int x, int y) {
    char[][] shape = piece.getPiece();
    for (int i = 0; i < piece.getRow(); i++) {
        for (int j = 0; j < piece.getCol(); j++) {
            if (shape[i][j] != '.' && board[y + i][x + j] != '#') {
                board[y + i][x + j] = '.';
            }
        }
    }
}

// board printing
public void printBoard() {
    System.out.print("\u001B[0m");
    for (int i = 0; i < this.height; i++) {
        for (int j = 0; j < this.width; j++) {
            char c = board[i][j];
            System.out.print(ansiColorMap.getOrDefault(c, "") + c + " ");
        }
    }
}

```

```

        System.out.println();
    }
    System.out.print("\u001B[0m\n");
}
}

```

2.2. Piece.java

Berikut ini merupakan attribute dan method yang terdapat dalam class Piece.

Attribute	Tipe Data	Keterangan
row	private int	Attribute untuk menyimpan tinggi piece
col	private int	Attribute untuk menyimpan lebar terpanjang piece
piece	private char[][]	Attribute untuk menyimpan bentuk piece dalam Array 2d

Method	Returned Data Type	Keterangan
Piece(char[][] shape)	public	Konstruktor Piece dengan shape tertentu
getRow()	public int	Aksesori baris piece
getCol()	public int	Aksesori kolom piece
getPiece()	public char[][]	Aksesori piece
printPiece()	public void	Untuk menampilkan piece
rotate()	public void	Merotasikan piece 90 derajat searah jarum jam
flip()	public void	Untuk membalikkan piece secara horizontal
copyShape()	public char[][]	Untuk menyalin shape suatu piece
setPiece(char[][] newShape)	public void	Untuk set piece hasil salin (copyShape())

getUniqueOrientations()	public Set<Piece>	Aksesor untuk semua orientasi unik (4 rotasi dan 2 flip)
serializeShape()	private String	

Source code file Piece.java adalah sebagai berikut.

```
package models;

import java.util.Arrays;
import java.util.HashSet;
import java.util.Set;

public class Piece {
    // attributes
    private int row, col;
    private char[][] piece;

    // constructor
    public Piece(char[][] shape) {
        this.row = shape.length;
        this.col = shape[0].length;
        this.piece = new char[row][col];
        for (int i = 0; i < row; i++) {
            System.arraycopy(shape[i], 0, this.piece[i], 0, col);
        }
    }

    // methods

    public int getRow() {
        return row;
    }

    public int getCol() {
        return col;
    }

    public char[][] getPiece() {
        return piece;
    }

    // print the piece
    public void printPiece() {
```

```

        for (int i = 0; i < row; i++) {
            for (int j = 0; j < col; j++) {
                System.out.print(piece[i][j] + " ");
            }
            System.out.println();
        }
    }

    // rotate the piece 90 degrees clockwise
    public void rotate() {
        int newRow = col;
        int newCol = row;
        char[][] rotated = new char[newRow][newCol];

        for (int r = 0; r < row; r++) {
            for (int c = 0; c < col; c++) {
                rotated[c][newCol - 1 - r] = piece[r][c];
            }
        }

        this.piece = rotated;
        this.row = newRow;
        this.col = newCol;
    }

    // flip the piece horizontally
    public void flip() {
        for (int r = 0; r < row; r++) {
            for (int c = 0; c < col / 2; c++) {
                char temp = piece[r][c];
                piece[r][c] = piece[r][col - 1 - c];
                piece[r][col - 1 - c] = temp;
            }
        }
    }

    // copy piece
    public char[][] copyShape() {
        char[][] copy = new char[row][col];
        for (int i = 0; i < row; i++) {
            System.arraycopy(piece[i], 0, copy[i], 0, col);
        }
        return copy;
    }

    // set piece copied
    public void setPiece(char[][] newShape) {

```

```

        this.piece = newShape;
        this.row = newShape.length;
        this.col = newShape[0].length;
    }

    // 4 rotation and 2 flip
    public Set<Piece> getUniqueOrientations() {
        Set<String> seen = new HashSet<>();
        Set<Piece> orientations = new HashSet<>();
        char[][] original = copyShape();

        for (int flipCount = 0; flipCount < 2; flipCount++) {
            for (int rotationCount = 0; rotationCount < 4; rotationCount++) {
                String serialized = serializeShape();
                if (seen.add(serialized)) {
                    orientations.add(new Piece(copyShape()));
                }
                rotate();
            }
            flip();
            setPiece(original);
        }

        return orientations;
    }

    private String serializeShape() {
        StringBuilder sb = new StringBuilder();
        for (char[] row : piece) {
            for (char c : row) {
                sb.append(c);
            }
            sb.append(';');
        }
        return sb.toString();
    }
}

```

2.3 Solver.java

Berikut ini merupakan attribute dan method yang terdapat dalam class Solver.

Method	Returned Data Type	Keterangan
--------	--------------------	------------

solve(Board board, List<Piece> pieces, int index, int[] iterations)	public static boolean	Method untuk menemukan solusi IQ Puzzler Pro dengan menggunakan method-method dalam kelas pada Board.java dan Piece.java
---	-----------------------	---

Source code file Solver.java adalah sebagai berikut.

```
package models;

import java.util.List;
import java.util.Set;

public class Solver {

    public static boolean solve(Board board, List<Piece> pieces, int index, int[] iterations) {
        if (index == pieces.size()) {
            return board.isBoardFull();
        }

        Piece piece = pieces.get(index);
        Set<Piece> uniqueOrientations = piece.getUniqueOrientations();

        for (Piece orientation : uniqueOrientations) {
            for (int y = 0; y <= board.getHeight() - orientation.getRow(); y++) {
                for (int x = 0; x <= board.getWidth() - orientation.getCol(); x++) {
                    iterations[0]++;

                    if (board.canPlacePiece(orientation, x, y)) {
                        board.placePiece(orientation, x, y);

                        if (solve(board, pieces, index + 1, iterations)) {
                            return true;
                        }

                        board.removePiece(orientation, x, y);
                    }
                }
            }
        }

        return false;
    }
}
```

2.4 InputHandler.java

Berikut ini merupakan attribute dan method yang terdapat dalam class InputHandler.

Attribute	Tipe Data	Keterangan
board	private Board	Attribute board
pieces	private List<Piece>	Attribute list of Piece
mode	private String	Attribute penentu mode (DEFAULT, CUSTOM, PYRAMID)

Method	Returned Data type	Keterangan
InputHandler(String filePath)	public	Konstruktor Inputhandler dengan filepath tertentu
readInputFromFile(String filePath)	private void	Untuk membaca input dari file tertentu
readCustomBoard(BufferedReader reader, int rows, int cols)	private Board	Untuk membaca input pada papan custom
readPieces	private List<Piece>	Untuk membaca piece tertentu
convertToPiece(List<String> pieceLines, char letter)	private Piece	Untuk mengonversi string ke piece
readNonEmptyLine(BufferedReader reader)	private String	Untuk membaca line yang tidak kosong
getBoard()	public Board	Untuk mengakses papan
getPieces()	public List<Piece>	Untuk mengakses list of piece
getMode()	public String	Untuk mengakses mode yang diinput oleh user

Source code file InputHandler.java adalah sebagai berikut.

```
package utils;

import java.io.*;
import java.util.*;
import models.*;

public class InputHandler {
    private Board board;
    private List<Piece> pieces;
    private String mode;

    public InputHandler(String filePath) throws IOException {
        readInputFromFile(filePath);
    }

    private void readInputFromFile(String filePath) throws IOException {
        try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {

            // read board dimensions and number of pieces
            String firstLine = readNonEmptyLine(reader);
            if (firstLine == null) throw new IllegalArgumentException("File kosong.");

            String[] parts = firstLine.trim().split(" ");
            if (parts.length != 3) throw new IllegalArgumentException("Format baris pertama harus 3 angka.");

            int N = Integer.parseInt(parts[0]);
            int M = Integer.parseInt(parts[1]);
            int P = Integer.parseInt(parts[2]);

            // read mode
            String modeLine = readNonEmptyLine(reader);
            if (modeLine == null) throw new IllegalArgumentException("File tidak memiliki mode permainan.");
            mode = modeLine.trim();

            if (mode.equals("DEFAULT")) {
                board = new Board(N, M); // empty board
            } else if (mode.equals("CUSTOM")) {
                board = readCustomBoard(reader, N, M); // custom filled board
            } else {
                throw new IllegalArgumentException("Mode permainan tidak valid: " + mode);
            }
        }
    }
}
```



```

        // read puzzle pieces
        pieces = readPieces(reader, P);

        // validate piece count
        if (pieces.size() != P) {
            throw new IllegalArgumentException("Jumlah puzzle piece tidak sesuai. Diharapkan " + P + ", ditemukan " + pieces.size());
        }

    } catch (IOException e) {
        throw new IllegalArgumentException("Gagal membaca file: " + e.getMessage());
    }
}

private Board readCustomBoard(BufferedReader reader, int rows, int cols) throws IOException {
    Board customBoard = new Board(rows, cols);
    for (int i = 0; i < rows; i++) {
        String line = readNonEmptyLine(reader);
        if (line.length() != cols) {
            throw new IllegalArgumentException("Panjang baris ke-" + (i + 1) + " tidak sesuai dengan lebar papan.");
        }
        customBoard.setRow(i, line);
    }
    return customBoard;
}

private List<Piece> readPieces(BufferedReader reader, int totalPieces) throws IOException {
    List<Piece> pieceList = new ArrayList<>();
    List<String> pieceLines = new ArrayList<>();
    char currentLetter = '\0';
    String line;

    while ((line = reader.readLine()) != null) {
        line = line.replace(' ', '.').trim();
        if (line.isEmpty()) continue;

        char firstChar = line.replace(".", "").isEmpty() ? '.' :
line.charAt(line.indexOf(line.replace(".", "").charAt(0)));

        if (firstChar != currentLetter && !pieceLines.isEmpty()) {
            pieceList.add(convertToPiece(pieceLines, currentLetter));
            pieceLines.clear();
        }

        currentLetter = firstChar;
    }
}

```

```

        pieceLines.add(line);
    }

    // save the last piece
    if (!pieceLines.isEmpty()) {
        pieceList.add(convertToPiece(pieceLines, currentLetter));
    }

    return pieceList;
}

private Piece convertToPiece(List<String> pieceLines, char letter) {
    int rows = pieceLines.size();
    int cols = pieceLines.stream().mapToInt(String::length).max().orElse(0);
    char[][] shape = new char[rows][cols];

    for (int i = 0; i < rows; i++) {
        String line = pieceLines.get(i);
        for (int j = 0; j < cols; j++) {
            shape[i][j] = (j < line.length() && line.charAt(j) == letter) ? letter : '.';
        }
    }
    return new Piece(shape);
}

private String readNonEmptyLine(BufferedReader reader) throws IOException {
    String line;
    while ((line = reader.readLine()) != null) {
        line = line.trim();
        if (!line.isEmpty()) return line;
    }
    return null;
}

public Board getBoard() {
    return board;
}

public List<Piece> getPieces() {
    return pieces;
}

public String getMode() {
    return mode;
}
}

```

2.5 SolutionHandler.java

Berikut ini merupakan attribute dan method yang terdapat dalam class SolutionHandler.

Attribute	Tipe Data	Keterangan
CELL_SIZE	private static final int	Attribute untuk menentukan ukuran sel pada output gambar dalam satuan piksel
PADDING	private static final int	Attribute untuk menentukan jarak (padding) antara board dengan batas gambar

Method	Returned Data Type	Keterangan
displaySolution(Board board, long time, int iterations)	public static void	Untuk menampilkan solusi board pada terminal beserta waktu pencarian dan jumlah kasus yang ditinjau
saveSolution(Board board, long time, int iterations, String inputFilename)	public static void	Untuk menyimpan solusi berdasarkan konfirmasi pengguna, apabila disetujui method ini akan memanggil method saveTextSolution dan saveImageSolution
saveTextSolution(Board board, long time, int iterations, String filename)	private static void	Untuk menyimpan solusi dalam file teks
saveImageSolution(Board board, String filename)	private static void	Untuk menyimpan solusi dalam file PNG

Source code file SolutionHandler.java adalah sebagai berikut.

```
package utils;

import models.Board;
```

```

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

public class SolutionHandler {

    private static final int CELL_SIZE = 50;
    private static final int PADDING = 10;

    public static void displaySolution(Board board, long time, int iterations) {
        board.printBoard();
        System.out.println("Waktu pencarian: " + time + " ms");
        System.out.println("Banyak kasus yang ditinjau: " + iterations);
    }

    public static void saveSolution(Board board, long time, int iterations, String inputFilename)
    {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Apakah Anda ingin menyimpan solusi? (ya/tidak): ");
        String response = scanner.nextLine().trim().toLowerCase();

        if (!response.equals("ya")) {
            System.out.println("Solusi tidak disimpan.");
            return;
        }

        String baseFilename = inputFilename.replace(".txt", "_solution");
        saveTextSolution(board, time, iterations, "test/" + baseFilename + ".txt");
        saveImageSolution(board, "test/" + baseFilename + ".png");
    }

    private static void saveTextSolution(Board board, long time, int iterations, String filename)
    {
        try (BufferedWriter writer = new BufferedWriter(new FileWriter(filename))) {
            char[][] boardArray = board.getBoard();

            for (char[] row : boardArray) {
                for (char cell : row) {
                    writer.write(cell);
                }
                writer.newLine();
            }

            writer.newLine();
            writer.write("Waktu pencarian: " + time + " ms\n");
        }
    }
}

```

```

        writer.write("Banyak kasus yang ditinjau: " + iterations + "\n");

        System.out.println("Solusi berhasil disimpan di: " + filename);
    } catch (IOException e) {
        System.err.println("Gagal menyimpan solusi ke file: " + e.getMessage());
    }
}

private static void saveImageSolution(Board board, String filename) {
    int width = board.getWidth();
    int height = board.getHeight();
    int imageWidth = width * CELL_SIZE + PADDING * 2;
    int imageHeight = height * CELL_SIZE + PADDING * 2;

    BufferedImage image = new BufferedImage(imageWidth, imageHeight,
BufferedImage.TYPE_INT_RGB);
    Graphics2D g = image.createGraphics();

    g.setColor(Color.WHITE);
    g.fillRect(0, 0, imageWidth, imageHeight);

    char[][] boardArray = board.getBoard();
    for (int i = 0; i < height; i++) {
        for (int j = 0; j < width; j++) {
            char cell = boardArray[i][j];
            if (cell != '.' && cell != '#') {
                g.setColor(Board.getColorMap().getOrDefault(cell, Color.BLACK));
                int x = PADDING + j * CELL_SIZE;
                int y = PADDING + i * CELL_SIZE;
                g.fillRect(x, y, CELL_SIZE, CELL_SIZE);
            }
        }
    }

    g.dispose();

    try {
        File outputfile = new File(filename);
        ImageIO.write(image, "png", outputfile);
        System.out.println("Solusi disimpan sebagai gambar di: " + filename);
    } catch (IOException e) {
        System.err.println("Gagal menyimpan gambar: " + e.getMessage());
    }
}
}

```

2.6 Main.java

```
import models.*;
import utils.*;

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

public class Main {
    Run | Debug
    public static void main(String[] args) {
        System.out.println(x:"\nSelamat datang di Program IQ Puzzler Pro\n");

        Scanner scanner = new Scanner(System.in);
        System.out.print(s:"Masukkan nama file input dengan format .txt: ");
        String fileName = scanner.nextLine().trim();

        String inputFile = "test/" + fileName;

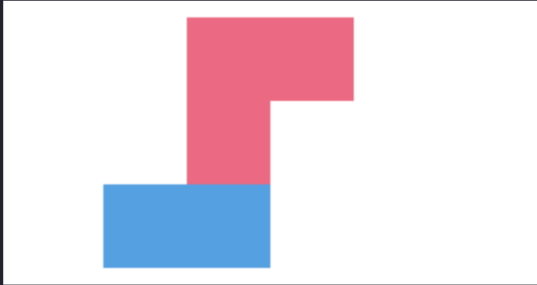
        try {
            // read input file
            InputHandler inputHandler = new InputHandler(inputFile);
            Board board = inputHandler.getBoard();
            List<Piece> pieces = inputHandler.getPieces();
            String mode = inputHandler.getMode();

            System.out.println("\nMode permainan: " + mode);
            System.out.println("Dimensi papan: " + board.getHeight() + " x " + board.getWidth());
            System.out.println("Jumlah blok: " + pieces.size());
            System.out.println(x:"\nMemulai pencarian solusi...\n");

            // solve the puzzle
            long startTime = System.currentTimeMillis();
            int[] iterations = {0};
            boolean solutionFound = Solver.solve(board, pieces, index:0, iterations);
            long endTime = System.currentTimeMillis();
            long searchTime = endTime - startTime;

            if (solutionFound) {
                System.out.println(x:"\nSolusi ditemukan!\n");
                SolutionHandler.displaySolution(board, searchTime, iterations[0]);
                SolutionHandler.saveSolution(board, searchTime, iterations[0], fileName);
            } else {
                System.out.println(x:"\nSolusi tidak ditemukan.");
                System.out.println("Waktu pencarian: " + searchTime + "ms");
                System.out.println("Banyak kasus yang ditinjau: " + iterations[0]);
            }
        } catch (IOException e) {
            System.err.println("Terjadi kesalahan saat membaca file: " + e.getMessage());
        } catch (IllegalArgumentException e) {
            System.err.println("Error input: " + e.getMessage());
        }
    }
}
```

3. Test Case

Input	Output
<pre>test > ≡ test_case_1.txt 1 3 6 2 2 CUSTOM 3 ..XX.. 4 ..X... 5 .XX... 6 A 7 AA 8 B 9 B</pre>	<pre>Selamat datang di Program IQ Puzzler Pro Masukkan nama file input dengan format .txt: test_case_1.txt Mode permainan: CUSTOM Dimensi papan: 3 x 6 Jumlah blok: 2 Memulai pencarian solusi... Solusi ditemukan! # # A A # # # # A # # # # B B # # # Waktu pencarian: 0 ms Banyak kasus yang ditinjau: 37 Apakah Anda ingin menyimpan solusi? (ya/tidak): ya Solusi berhasil disimpan di: test/test_case_1_solution.txt Solusi disimpan sebagai gambar di: test/test_case_1_solution.png</pre> 
	<pre>test > ≡ test_case_2.txt 1 3 4 4 2 DEFAULT 3 A 4 A 5 AA 6 BBB 7 BB 8 CC 9 D</pre>

```
test > ≡ test_case_1_solution.txt
1 1 /// ##AA##
2 2 /// ##A###
3 3 /// #BB###
4 4
5 5 Waktu pencarian: 0 ms
6 6 Banyak kasus yang ditinjau: 37
```

```
test > ≡ test_case_2.txt
```

```
1 3 4 4
2 DEFAULT
3 A
4 A
5 AA
6 BBB
7 BB
8 CC
9 D
```

```
Selamat datang di Program IQ Puzzler Pro

Masukkan nama file input dengan format .txt: test_case_2.txt

Mode permainan: DEFAULT
Dimensi papan: 3 x 4
Jumlah blok: 4

Memulai pencarian solusi...

Solusi ditemukan!
A A B D
C A B B
C A B B

Waktu pencarian: 0 ms
Banyak kasus yang ditinjau: 17
Apakah Anda ingin menyimpan solusi? (ya/tidak): ya
Solusi berhasil disimpan di: test/test_case_2_solution.txt
Solusi disimpan sebagai gambar di: test/test_case_2_solution.png
```



```
test > ≡ test_case_2_solution.txt
1 1 /// AABD
2 2 /// CABB
3 3 /// CABB
4 4
5 5 Waktu pencarian: 0 ms
6 6 Banyak kasus yang ditinjau: 17
```



```
test > ≡ test_case_3.txt
```

```
1 4 4 5
2 DEFAULT
3 A
4 AAA
5 B
6 BBB
7 C
8 CC
9 D
10 DD
11 EE
```

Selamat datang di Program IQ Puzzler Pro

Masukkan nama file input dengan format .txt: test_case_3.txt

Mode permainan: DEFAULT

Dimensi papan: 4 x 4

Jumlah blok: 5

Memulai pencarian solusi...

Solusi ditemukan!

```
A A A B
D D A B
D C B B
C C E E
```

Waktu pencarian: 1 ms

Banyak kasus yang ditinjau: 222

Apakah Anda ingin menyimpan solusi? (ya/tidak): ya

Solusi berhasil disimpan di: test/test_case_3_solution.txt

Solusi disimpan sebagai gambar di: test/test_case_3_solution.png



```
test > ≡ test_case_3_solution.txt
```

```
1 AAAB
2 DDAB
3 DCBB
4 CCEE
5
6 Waktu pencarian: 1 ms
7 Banyak kasus yang ditinjau: 222
```

```
test > test_case_4.txt
```

```
1 5 5 5
2 DEFAULT
3 ACCCD
4 ACCBD
5 ACBBB
6 AABBB
7 AEEDB
8
9 A
10 A
11 A
12 AA
13 A
14 C
15 CC
16 CCC
17 BB
18 BBBB
19 BB
20 DDD
21 EE
```

Selamat datang di Program IQ Puzzler Pro

Masukkan nama file input dengan format .txt: test_case_4.txt

Mode permainan: DEFAULT

Dimensi papan: 5 x 5

Jumlah blok: 5

Memulai pencarian solusi...

Solusi tidak ditemukan.

Waktu pencarian: 0ms

Banyak kasus yang ditinjau: 0

```
test > test_case_5.txt
```

```
1 3 3 3
2 DEFAULT
3 A
4 AA
5 B
6 B
7 BB
8 CC
```

Selamat datang di Program IQ Puzzler Pro

Masukkan nama file input dengan format .txt: test_case_5.txt

Mode permainan: DEFAULT

Dimensi papan: 3 x 3

Jumlah blok: 3

Memulai pencarian solusi...

Solusi ditemukan!

B B B

B A A

C C A

Waktu pencarian: 0 ms

Banyak kasus yang ditinjau: 76

Apakah Anda ingin menyimpan solusi? (ya/tidak): ya

Solusi berhasil disimpan di: test/test_case_5_solution.txt

Solusi disimpan sebagai gambar di: test/test_case_5_solution.png



	<pre> test > ≡ test_case_5_solution.txt 1 BBB 2 BAA 3 CCA 4 5 Waktu pencarian: 0 ms 6 Banyak kasus yang ditinjau: 76 </pre>
<pre> test > ≡ test_case_6.txt 1 4 3 2 2 DEFAULT 3 A 4 A 5 A 6 AA 7 BBB 8 BBBB </pre>	<pre> Selamat datang di Program IQ Puzzler Pro Masukkan nama file input dengan format .txt: test_case_6.txt Mode permainan: DEFAULT Dimensi papan: 4 x 3 Jumlah blok: 2 Memulai pencarian solusi... Solusi tidak ditemukan. Waktu pencarian: 1ms Banyak kasus yang ditinjau: 20 </pre>
<pre> test > ≡ test_case_7.txt 1 4 4 6 2 DEFAULT 3 A 4 AA 5 A 6 BBB 7 B 8 C 9 DD 10 E 11 FFFF </pre>	<pre> Selamat datang di Program IQ Puzzler Pro Masukkan nama file input dengan format .txt: test_case_7.txt Mode permainan: DEFAULT Dimensi papan: 4 x 4 Jumlah blok: 6 Memulai pencarian solusi... Solusi ditemukan! C A D F A A D F E A B F B B B F Waktu pencarian: 0 ms Banyak kasus yang ditinjau: 43 Apakah Anda ingin menyimpan solusi? (ya/tidak): ya Solusi berhasil disimpan di: test/test_case_7_solution.txt Solusi disimpan sebagai gambar di: test/test_case_7_solution.png </pre>



4. Pranala Repository

https://github.com/adndax/Tucil1_13523071

5. Lampiran

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	✓	

6. Daftar Pustaka

[1] R. Munir, 'Algoritma Brute Force', Institut Teknologi Bandung, 2025.