

Tugas Kecil 1

IF2211 Strategi Algoritma

Semester II Tahun 2024/2025

Penyelesaian IQ Puzzler Pro dengan Algoritma Brute Force



Disusun Oleh:

Grace Evelyn Simon

13523087

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2025

DAFTAR ISI

BAB I	3
DESKRIPSI MASALAH	3
BAB II	4
TEORI DASAR	4
2.1 Penjelasan Singkat	4
2.2 Penjabaran Algoritma	5
BAB III	7
IMPLEMENTASI	7
3.1 Github Repository	8
3.2 Implementasi Kode Program	8
BAB IV	22
EKSPERIMEN	22
4.1 Cara Menjalankan Program	22
4.1 Pengujian	24
4.1.1 Kasus Uji 1 – DEFAULT Solved	24
4.1.2 Kasus Uji 2 – DEFAULT Not Solved	25
4.1.3 Kasus Uji 3 – CUSTOM Solved	26
4.1.4 Kasus Uji 4 – CUSTOM Not Solved	27
4.1.5 Kasus Uji 5 – DEFAULT dengan Whitespace di Awal	27
4.1.6 Kasus Uji 6 – CUSTOM dengan Whitespace di Awal	28
4.1.7 Kasus Uji 7 – Tidak Menyimpan Solusi	29
BAB V	30
KESIMPULAN, SARAN, DAN REFLEKSI	30
5.1 Kesimpulan	30
5.2 Saran	30
5.2 Refleksi	30
BAB VI	32
LAMPIRAN	32

BAB I

DESKRIPSI MASALAH

IQ Puzzler Pro merupakan permainan teka-teki logika yang diciptakan oleh Smart Toys and Games, Inc., sebuah perusahaan yang memiliki spesialisasi di bidang permainan logika multi-level. IQ Puzzler Pro dapat dimainkan dalam dua mode, yakni secara 2D dan 3D. Dalam Tugas Kecil 1 IF2211 Strategi Algoritma, terdapat 3 jenis kasus permainan yang dapat ditinjau, yakni Default (2D), Custom (2D), dan Pyramid (3D). Untuk menemukan solusi dalam permainan IQ Puzzler Pro, diperlukan strategi penyelesaian yang logis dan sistematis. Permainan dilakukan dengan mencoba meletakkan potongan blok untuk mengisi papan hingga penuh. Potongan blok dapat dirotasikan maupun dicerminkan. Permainan dinyatakan selesai jika dan hanya jika papan terisi penuh dan seluruh potongan blok puzzle berhasil diletakkan.

Strategi Algoritma memiliki peranan penting dalam menyelesaikan permainan IQ Puzzler Pro karena permainan ini membutuhkan pendekatan yang sistematis untuk menempatkan potongan blok secara optimal. Dalam Tugas Kecil 1 IF2211 Strategi Algoritma, para mahasiswa mengimplementasikan algoritma Brute Force secara “murni” untuk mencari sebuah solusi permainan IQ Puzzler Pro menggunakan bahasa pemrograman Java. Pencarian solusi dilakukan dengan mencoba setiap kemungkinan susunan potongan blok hingga menemukan satu solusi yang valid.



Gambar 1. Permainan IQ Puzzler Pro

(Sumber: <https://www.smartgamesusa.com/one-player-games/iq-puzzler-pro>)

BAB II

TEORI DASAR

2.1 Penjelasan Singkat

Algoritma Brute Force merupakan algoritma yang menerapkan pendekatan yang lurus atau lempeng (*straightforward*) untuk memecahkan suatu persoalan. Algoritma Brute Force memecahkan persoalan secara sederhana, langsung, jelas, dan mudah dipahami, yakni dengan mencoba seluruh kemungkinan solusi secara sistematis sampai ditemukan sebuah solusi. Dalam permainan IQ Puzzler Pro, Brute Force digunakan untuk menempatkan potongan blok satu per satu pada papan dengan mencoba semua rotasi dan pencerminan potongan blok hingga ditemukan satu solusi yang memenuhi seluruh aturan permainan.

2.2 Penjabaran Algoritma

Permainan IQ Puzzler Pro adalah permainan puzzle dimana pemain harus menempatkan beberapa potongan blok ke dalam papan dengan ukuran tertentu. Setiap blok memiliki bentuk yang unik, dan tujuan permainan adalah menempatkan semua blok ke dalam papan tanpa tumpang tindih dan memenuhi semua aturan yang diberikan. Algoritma Brute Force yang digunakan dalam program ini mencoba semua kemungkinan penempatan blok pada papan hingga menemukan solusi yang valid. Berikut adalah penjelasan detail tentang bagaimana algoritma tersebut bekerja:

1. Inisialisasi Papan dan Blok

Program membaca *file input* yang berisi ukuran papan ($N \times M$), jumlah blok (P), konfigurasi papan (DEFAULT atau CUSTOM), dan bentuk blok. Papan direpresentasikan sebagai matriks 2D (`char[][] grid`) dengan ukuran $N \times M$. Setiap sel pada papan dapat berisi . (titik) untuk sel kosong (dalam mode DEFAULT) dan X untuk sel kosong (dalam mode CUSTOM). Kemudian, setiap blok direpresentasikan sebagai matriks 2D (`char[][] shape`) yang berisi bentuk blok. Setiap blok memiliki karakter unik yang digunakan untuk mengidentifikasinya.

2. Generasi Variasi Blok

Setiap blok dapat diputar (*rotate*) dan dibalik (*flip*) untuk menghasilkan variasi bentuk yang berbeda. Blok dapat diputar 90, 180, maupun 270 derajat searah jarum jam dan

dapat dibalik secara horizontal atau vertikal. Untuk setiap blok, program menghasilkan semua variasi bentuk yang mungkin dan menyimpannya dalam sebuah list (`List<char[][]> variations`).

3. Algoritma Brute Force

Algoritma Brute Force bekerja secara rekursif dengan mencoba semua kemungkinan penempatan blok pada papan. Berikut langkah-langkah algoritmanya:

a. Kasus Basis

Jika semua blok sudah ditempatkan (`index == puzzles.size()`), program memeriksa apakah papan sudah penuh (`board.isFull()`). Jika papan penuh, solusi ditemukan, dan program mengembalikan `true`. Jika tidak, program mengembalikan `false`.

b. Kasus Rekurens

Untuk setiap blok (`puzzles.get(index)`), program mencoba semua kemungkinan penempatan pada papan. Program mengiterasi semua posisi pada papan (`row` dan `col`) sebagai titik awal penempatan blok. Untuk setiap posisi, program mencoba semua variasi bentuk blok. Program memeriksa apakah blok dapat ditempatkan pada posisi tersebut dengan memanggil `board.canPlace(variation, row, col, isCustomMode)`. Jika blok dapat ditempatkan, program menempatkan blok tersebut pada papan dengan memanggil `board.place(variation, row, col, currentPuzzle.character)`. Kemudian, program memanggil dirinya sendiri (`bruteForce(index + 1)`) untuk mencoba menempatkan blok berikutnya. Jika rekursi mengembalikan `false`, program menghapus blok yang baru saja ditempatkan dengan memanggil `board.remove(variation, row, col)`. Selanjutnya, program melanjutkan iterasi untuk mencoba variasi atau posisi lain.

c. Pengembalian Nilai

Jika semua kemungkinan penempatan blok telah dicoba dan tidak ada solusi yang ditemukan, program mengembalikan `false`.

4. Fungsi Pendukung

Ada beberapa fungsi pendukung yang ditambahkan untuk membantu keberjalanan program, yakni:

a. `board.canPlace()`

Memeriksa apakah blok dapat ditempatkan pada posisi tertentu tanpa melanggar aturan (tidak tumpang tindih dengan blok lain atau melanggar batas papan).

b. `board.place()`

Menempatkan blok pada papan dan menandai sel-sel yang diisi dengan karakter blok.

c. `board.remove()`

Menghapus blok dari papan dan mengembalikan sel-sel yang diisi ke keadaan semula (. untuk DEFAULT atau X untuk CUSTOM).

d. `board.isFull()`

Memeriksa apakah semua sel pada papan sudah terisi (tidak ada sel kosong atau sel yang dilarang).

5. Output Solusi

Jika solusi ditemukan, program mencetak papan ke terminal dengan warna tertentu. Program juga menawarkan opsi untuk menyimpan solusi dalam file teks (.txt) atau gambar (.png).

6. Kompleksitas Algoritma

Algoritma Brute Force memiliki kompleksitas waktu yang tinggi karena mencoba semua kemungkinan penempatan blok. Jumlah kemungkinan yang harus dicoba bergantung pada jumlah blok (P), ukuran papan ($N \times M$), dan jumlah variasi bentuk setiap blok. Dalam kasus terburuk, kompleksitas waktu bisa mencapai $O((N * M)^P * V)$ dengan V adalah jumlah variasi bentuk blok.

BAB III

IMPLEMENTASI

Implementasi merujuk pada cara suatu algoritma, struktur data, maupun fungsionalitas yang dirancang untuk menyelesaikan masalah, dalam program ini yakni menyelesaikan permainan IQ Puzzler Pro. Implementasi mencakup seluruh langkah teknis, mulai dari pembacaan input, pemrosesan data, algoritma Brute Force, hingga menampilkan maupun menyimpan solusi. Berikut merupakan implementasi dari program yang diciptakan:

3.1 Github Repository

https://github.com/graceevelyns/Tucil1_13523087.git

3.2 Implementasi Kode Program

```
package src;

// import library
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics2D;
import java.awt.image.BufferedImage;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.InputMismatchException;
import java.util.List;
import java.util.Scanner;
import javax.imageio.ImageIO;
import javax.swing.JFileChooser;
import javax.swing.SwingUtilities;
import javax.swing.UIManager;
```

```

import javax.swing.UnsupportedLookAndFeelException;
import javax.swing.filechooser.FileNameExtensionFilter;

public class IQPuzzlerPro {
    static Board board;
    static List<Puzzle> puzzles = new ArrayList<>();
    static int iterationCount = 0;
    static boolean hasSolution = false;
    static boolean isCustomMode = false;

    // array of colors
    static final Color[] COLORS = {
        Color.RED, Color.GREEN, Color.BLUE, Color.MAGENTA,
Color.CYAN,
        Color.YELLOW, Color.ORANGE, Color.PINK, Color.GRAY,
Color.DARK_GRAY
    };

    public static void main(String[] args) {
        try {

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
        } catch (ClassNotFoundException | InstantiationException |
IllegalAccessException | UnsupportedLookAndFeelException e) {
            System.out.println("Error: Gagal mengatur tampilan
Windows.");
            e.printStackTrace();
        }

        SwingUtilities.invokeLater(() -> {
            if (args.length == 0) {
                System.out.println("Usage: java IQPuzzlerPro
<input_file.txt>");
                return;
            }
            String inputFile = args[0];
            try (BufferedReader reader = new BufferedReader(new
FileReader(inputFile))) {
                String[] firstLine =
reader.readLine().trim().split("\\s+");
                if (firstLine.length < 3) {
                    throw new IllegalArgumentException("Format input

```



```

tidak valid. Harap berikan tiga angka dipisah oleh spasi (N, M,
P).");

    }

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

    if (N <= 0 || M <= 0 || P <= 0) {
        throw new IllegalArgumentException("Ukuran papan
dan jumlah blok harus lebih dari 0.");
    }

    String config = reader.readLine().trim();
    if (!config.equals("DEFAULT") &&
!config.equals("CUSTOM")) {
        throw new IllegalArgumentException("Hanya
konfigurasi 'DEFAULT' atau 'CUSTOM' yang diperbolehkan.");
    }

    if (config.equals("DEFAULT")) {
        board = new Board(N, M, false, puzzles);
        isCustomMode = false;
    } else {
        char[][] customGrid = new char[N][M];
        for (int i = 0; i < N; i++) {
            String line = reader.readLine().trim();
            if (line.length() != M) {
                throw new IllegalArgumentException("Baris
konfigurasi tidak sesuai dengan ukuran papan.");
            }
            customGrid[i] = line.toCharArray();
        }
        board = new Board(N, M, true, customGrid,
puzzles);

        isCustomMode = true;
    }

    List<char[]> blockLines = new ArrayList<>();
    String line;
    while ((line = reader.readLine()) != null) {
        if (line.trim().isEmpty()) {
            break;

```

```

        }
        blockLines.add(line.toCharArray());
    }

    List<List<char[]>> blocks = new ArrayList<>();
    List<char[]> currentBlock = new ArrayList<>();
    char previousChar = '\0';

    for (char[] lineChars : blockLines) {
        char firstChar = '\0';
        for (char c : lineChars) {
            if (c != ' ') {
                firstChar = c;
                break;
            }
        }
        if (firstChar != previousChar &&
!currentBlock.isEmpty()) {
            blocks.add(currentBlock);
            currentBlock = new ArrayList<>();
        }
        currentBlock.add(lineChars);
        previousChar = firstChar;
    }

    if (!currentBlock.isEmpty()) {
        blocks.add(currentBlock);
    }

    if (blocks.size() != P) {
        throw new IllegalArgumentException("Jumlah puzzle
yang dibaca tidak sesuai dengan nilai P.");
    }

    for (List<char[]> block : blocks) {
        puzzles.add(new Puzzle(block));
    }

    // time
    long startTime = System.nanoTime();
    hasSolution = bruteForce(0);
    long endTime = System.nanoTime();

```

```

        long durationInMillis = (endTime - startTime) /
1_000_000;

        if (!hasSolution) {
            System.out.println("Tidak ada solusi!");
        } else {
            System.out.println("Solusi ditemukan:");
            board.print();
        }

        // iteration
        System.out.println("Waktu pencarian: " +
durationInMillis + " ms");
        System.out.println("Banyak kasus yang ditinjau: " +
iterationCount);

        // solution
        System.out.println("\nApakah Anda ingin menyimpan
solusi dalam file .txt? (ya/tidak)");
        try (Scanner scanner = new Scanner(System.in)) {
            String saveTxtChoice =
scanner.nextLine().trim().toLowerCase();
            if (saveTxtChoice.equals("ya")) {
                saveSolutionToFile(inputFile);
            } else if (!saveTxtChoice.equals("tidak")) {
                System.out.println("Error: Pilihan jawaban
hanya ya atau tidak.");
            }

            System.out.println("\nApakah Anda ingin menyimpan
solusi dalam file .png? (ya/tidak)");
            String savePngChoice =
scanner.nextLine().trim().toLowerCase();
            if (savePngChoice.equals("ya")) {
                saveSolutionToImage(inputFile);
            } else if (savePngChoice.equals("tidak")) {
                System.out.println("Penyimpanan dibatalkan
oleh pengguna.");
                System.exit(0);
            } else {
                System.out.println("Error: Pilihan jawaban
hanya ya atau tidak.");
            }
        }
    }
}

```

```

        System.exit(0);
    }
}

} catch (IOException e) {
    System.out.println("Error: File tidak ditemukan atau
tidak dapat dibaca.");
} catch (InputMismatchException e) {
    System.out.println("Error: Input awal harus berupa
angka (baris, kolom, jumlah blok).");
} catch (IllegalArgumentException e) {
    System.out.println("Error: " + e.getMessage());
} catch (Exception e) {
    System.out.println("Terjadi kesalahan: " +
e.getMessage());
}
});
}

// brute-force algorithm
public static boolean bruteForce(int index) {
    if (index == puzzles.size()) {
        if (board.isFull()) {
            return true;
        }
        return false;
    }

    Puzzle currentPuzzle = puzzles.get(index);
    for (int row = 0; row < board.N; row++) {
        for (int col = 0; col < board.M; col++) {
            List<char[][]> variations =
currentPuzzle.generateVariations();
            for (char[][] variation : variations) {
                iterationCount++;
                if (board.canPlace(variation, row, col,
isCustomMode)) {
                    board.place(variation, row, col,
currentPuzzle.character);
                    if (bruteForce(index + 1)) {
                        return true;
                    }
                    board.remove(variation, row, col);
                }
            }
        }
    }
}

```

```

        }
    }
}

return false;
}

public static void saveSolutionToFile(String inputFileName) {
    JFileChooser fileChooser = new JFileChooser();
    fileChooser.setDialogTitle("Simpan Solusi sebagai File
Teks");

    FileNameExtensionFilter filter = new
FileNameExtensionFilter("Text Files (*.txt)", "txt");
    fileChooser.setFileFilter(filter);

    int userSelection = fileChooser.showSaveDialog(null);

    if (userSelection == JFileChooser.APPROVE_OPTION) {
        File fileToSave = fileChooser.getSelectedFile();
        String outputFileName = fileToSave.getAbsolutePath();

        if (!outputFileName.toLowerCase().endsWith(".txt")) {
            outputFileName += ".txt";
        }

        try (PrintWriter writer = new PrintWriter(new
FileWriter(outputFileName))) {
            if (hasSolution) {
                for (char[] row : board.grid) {
                    for (char cell : row) {
                        if (cell == '.' || cell == 'X') {
                            writer.print(' ');
                        } else {
                            writer.print(cell);
                        }
                    }
                    writer.println();
                }
                System.out.println("Solusi berhasil disimpan di:
" + outputFileName);
            } else {
                writer.println("Tidak ada solusi!");
                System.out.println("Solusi berhasil disimpan di:

```

```

" + outputFileName);
    }
    } catch (IOException e) {
        System.out.println("Error: Gagal menyimpan solusi ke
file.");
    }
    } else {
        System.out.println("Penyimpanan dibatalkan oleh
pengguna.");
    }
}

public static void saveSolutionToImage(String inputFileName) {
    JFileChooser fileChooser = new JFileChooser();
    fileChooser.setDialogTitle("Simpan Solusi sebagai Gambar");

    FileNameExtensionFilter filter = new
FileNameExtensionFilter("PNG Images (*.png)", "png");
    fileChooser.setFileFilter(filter);

    int userSelection = fileChooser.showSaveDialog(null);

    if (userSelection == JFileChooser.APPROVE_OPTION) {
        File fileToSave = fileChooser.getSelectedFile();
        String outputFileName = fileToSave.getAbsolutePath();
        if (!outputFileName.toLowerCase().endsWith(".png")) {
            outputFileName += ".png";
        }
        int cellSize = 50;
        int width = board.M * cellSize;
        int height = board.N * cellSize;
        BufferedImage image = new BufferedImage(width, height,
BufferedImage.TYPE_INT_RGB);
        Graphics2D g2d = image.createGraphics();

        g2d.setColor(Color.WHITE);
        g2d.fillRect(0, 0, width, height);

        for (int row = 0; row < board.N; row++) {
            for (int col = 0; col < board.M; col++) {
                char cell = board.grid[row][col];
                if (cell != '.' && cell != 'X') {

```

```

        int colorIndex = cell - 'A';
        g2d.setColor(COLORS[colorIndex %
COLORS.length]);

        g2d.fillRect(col * cellSize, row * cellSize,
cellSize, cellSize);

        g2d.setColor(Color.BLACK);
        g2d.setFont(new Font("Arial", Font.BOLD,
20));

        g2d.drawString(String.valueOf(cell), col *
cellSize + 20, row * cellSize + 30);
    }
}

try {
    ImageIO.write(image, "PNG", new
File(outputFileName));
    System.out.println("Solusi berhasil disimpan di: " +
outputFileName);
} catch (IOException e) {
    System.out.println("Error: Gagal menyimpan solusi
sebagai gambar.");
}
} else {
    System.out.println("Penyimpanan dibatalkan oleh
pengguna.");
}

System.exit(0);
}
}

class Board {
    int N, M;
    char[][] grid;
    boolean isCustomMode;
    List<Character> placedPuzzles = new ArrayList<>();
    List<Puzzle> puzzles;

    static final String RESET = "\u001B[0m";
    static final String UNDERLINE = "\u001B[4m";

    static final String[] TEXT_COLORS = {
        "\u001B[31m", // A - Red

```

```

        "\u001B[32m", // B - Green
        "\u001B[34m", // C - Blue
        "\u001B[35m", // D - Magenta
        "\u001B[36m", // E - Cyan
        "\u001B[33m", // F - Yellow
        "\u001B[90m", // G - Gray
        "\u001B[4;31m", // H - Red (Underlined)
        "\u001B[4;32m", // I - Green (Underlined)
        "\u001B[4;34m", // J - Blue (Underlined)
        "\u001B[4;35m", // K - Magenta (Underlined)
        "\u001B[4;36m", // L - Cyan (Underlined)
        "\u001B[4;33m", // M - Yellow (Underlined)
        "\u001B[4;90m", // N - Gray (Underlined)
        "\u001B[40;30m", // O - Black text, Gray bg
        "\u001B[41;30m", // P - Black text, Red bg
        "\u001B[42;30m", // Q - Black text, Green bg
        "\u001B[43;30m", // R - Black text, Yellow bg
        "\u001B[44;30m", // S - Black text, Blue bg
        "\u001B[45;30m", // T - Black text, Magenta bg
        "\u001B[46;30m", // U - Black text, Cyan bg
        "\u001B[41;93m", // V - Yellow text, Red bg
        "\u001B[42;93m", // W - Yellow text, Green bg
        "\u001B[43;93m", // X - Yellow text, Yellow bg
        "\u001B[44;93m", // Y - Yellow text, Blue bg
        "\u001B[45;93m" // Z - Yellow text, Magenta bg
    };

    // default constructor
    public Board(int N, int M, boolean isCustomMode, List<Puzzle>
puzzles) {
        this.N = N;
        this.M = M;
        this.isCustomMode = isCustomMode;
        this.puzzles = puzzles;
        grid = new char[N][M];
        for (char[] row : grid) {
            Arrays.fill(row, '.');
        }
    }

    // custom constructor
    public Board(int N, int M, boolean isCustomMode, char[][]

```



```

customGrid, List<Puzzle> puzzles) {
    this.N = N;
    this.M = M;
    this.isCustomMode = isCustomMode;
    this.grid = customGrid;
    this.puzzles = puzzles;
}

public boolean isFull() {
    if (puzzles.size() != placedPuzzles.size()) {
        return false;
    }
    for (char[] row : grid) {
        for (char cell : row) {
            if (isCustomMode && cell == 'X') {
                return false;
            } else if (!isCustomMode && cell == '.') {
                return false;
            }
        }
    }
    return true;
}

public boolean canPlace(char[][] block, int row, int col, boolean
isCustomMode) {
    int bN = block.length;
    int bM = block[0].length;
    if (row + bN > N || col + bM > M) return false;
    for (int i = 0; i < bN; i++) {
        for (int j = 0; j < bM; j++) {
            if (block[i][j] != ' ') {
                if (isCustomMode && grid[row + i][col + j] !=
'X') {
                    return false;
                } else if (!isCustomMode && grid[row + i][col +
j] != '.') {
                    return false;
                }
            }
        }
    }
}

```

```

        return true;
    }

    public void place(char[][] block, int row, int col, char c) {
        int bN = block.length;
        int bM = block[0].length;
        for (int i = 0; i < bN; i++) {
            for (int j = 0; j < bM; j++) {
                if (block[i][j] != ' ') {
                    grid[row + i][col + j] = c;
                }
            }
        }
        placedPuzzles.add(c);
    }

    public void remove(char[][] block, int row, int col) {
        int bN = block.length;
        int bM = block[0].length;
        for (int i = 0; i < bN; i++) {
            for (int j = 0; j < bM; j++) {
                if (block[i][j] != ' ') {
                    grid[row + i][col + j] = isCustomMode ? 'X' :
'.';
                }
            }
        }
        placedPuzzles.remove(placedPuzzles.size() - 1);
    }

    public void print() {
        for (char[] row : grid) {
            for (char cell : row) {
                if (cell == '.' || cell == 'X') {
                    System.out.print(' ');
                } else {
                    int index = cell - 'A';
                    String style = (index >= 7 && index <= 13) ?
UNDERLINE : "";
                    System.out.print(style + TEXT_COLORS[index] +
cell + RESET);
                }
            }
        }
    }

```

```

        }
        System.out.println();
    }
    System.out.println();
}

class Puzzle {
    char[][] shape;
    char character;

    public Puzzle(List<char[]> blockLines) {
        this.shape = convertListToArray(blockLines);
        this.character = findCharacter(shape);
    }

    private char[][] convertListToArray(List<char[]> list) {
        int maxCols = 0;
        for (char[] row : list) {
            maxCols = Math.max(maxCols, row.length);
        }

        char[][] array = new char[list.size()][maxCols];
        for (int i = 0; i < list.size(); i++) {
            char[] row = list.get(i);
            Arrays.fill(array[i], ' ');
            System.arraycopy(row, 0, array[i], 0, row.length);
        }
        return array;
    }

    private char findCharacter(char[][] block) {
        for (char[] row : block) {
            for (char cell : row) {
                if (cell != ' ') return cell;
            }
        }
        throw new IllegalArgumentException("Puzzle harus memiliki karakter huruf.");
    }

    public List<char[][]> generateVariations() {

```

```

        List<char[][]> variations = new ArrayList<>();
        char[][] rotated = shape;

        for (int i = 0; i < 4; i++) {
            rotated = rotateClockwise(rotated);
            variations.add(rotated);
            variations.add(flipHorizontally(rotated));
            variations.add(flipVertically(rotated));
        }
        return variations;
    }

    private char[][] rotateClockwise(char[][] block) {
        int bN = block.length;
        int bM = block[0].length;
        char[][] rotated = new char[bM][bN];
        for (int i = 0; i < bN; i++) {
            for (int j = 0; j < bM; j++) {
                rotated[j][bN - 1 - i] = block[i][j];
            }
        }
        return rotated;
    }

    private char[][] flipHorizontally(char[][] block) {
        int bN = block.length;
        int bM = block[0].length;
        char[][] flipped = new char[bN][bM];
        for (int i = 0; i < bN; i++) {
            for (int j = 0; j < bM; j++) {
                flipped[i][bM - 1 - j] = block[i][j];
            }
        }
        return flipped;
    }

    private char[][] flipVertically(char[][] block) {
        int bN = block.length;
        int bM = block[0].length;
        char[][] flipped = new char[bN][bM];
        for (int i = 0; i < bN; i++) {
            flipped[i] = block[bN - 1 - i];
        }
    }

```

```
    }  
    return flipped;  
  }  
}
```

BAB IV

EKSPERIMEN

4.1 Cara Menjalankan Program

Berikut struktur folder pada program ini:

Tucil1_13523087/

```
|
|— bin/
|   └─ (file .class hasil kompilasi)
|
|— doc/
|   └─ Tucil1_K2_13523087_Grace Evelyn Simon.pdf
|
|— src/
|   └─ IQPuzzlerPro.java
|
|— test/
|   └─ testcase/
|       └─ case1.txt
|       └─ case2.txt
|       └─ case3.txt
|       └─ case4.txt
|       └─ case5.txt
|       └─ case6.txt
|       └─ case7.txt
|
└─ README.md
```

Berikut langkah-langkah untuk menjalankan program:

1. *Clone Repository*

Lakukan *clone* pada repository Github dengan mengetikkan:

```
git clone https://github.com/graceevelyns/Tucill1_13523087.git
```

2. Pastikan Java Development Kit (JDK) Telah Terinstal

```
java -version
```

```
javac -version
```

Jika perintah ini menampilkan versi Java, maka JDK telah terinstal. Jika belum, silahkan *download* JDK terlebih dahulu sebelum menjalankan program.

3. Buka Terminal/Command Prompt

Buka terminal maupun command prompt pada direktori utama (Tucill1_13523087/).

4. Kompilasi Program

Kompilasi file IQPuzzlerPro.java yang terdapat pada folder src/ dengan mengetikkan:

```
javac -d bin src/IQPuzzlerPro.java
```

5. Jalankan Program

Setelah dikompilasi, jalankan program dengan mengetikkan case yang ingin diuji:

```
java -cp bin src.IQPuzzlerPro test/testcase/case1.txt
```

```
java -cp bin src.IQPuzzlerPro test/testcase/case2.txt
```

```
java -cp bin src.IQPuzzlerPro test/testcase/case3.txt
```

```
java -cp bin src.IQPuzzlerPro test/testcase/case4.txt
```

```
java -cp bin src.IQPuzzlerPro test/testcase/case5.txt
```

```
java -cp bin src.IQPuzzlerPro test/testcase/case6.txt
```

```
java -cp bin src.IQPuzzlerPro test/testcase/case7.txt
```

6. Lihat Output

Jika solusi ditemukan, program akan menampilkan solusi di terminal dan menawarkan opsi untuk menyimpan solusi ke file teks atau gambar.

4.1 Pengujian

4.1.1 Kasus Uji 1 – DEFAULT Solved

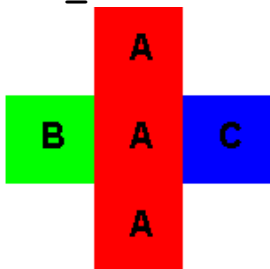
input.txt	output
2 3 2 DEFAULT AA A BB B	<pre>Solusi ditemukan: AAB ABB Banyak kasus yang ditinjau: 89 Apakah Anda ingin menyimpan solusi dalam file .txt? (ya/tidak) ya Solusi berhasil disimpan di: test/solution_case1.txt Apakah Anda ingin menyimpan solusi dalam file .png? (ya/tidak) ya Solusi berhasil disimpan di: test/solution_case1.png solution_case1.txt: AAB ABB solution_case1.png: A A B A B B</pre>

4.1.2 Kasus Uji 2 – DEFAULT Not Solved

input.txt	output
2 2 2 DEFAULT AA A BB B	<pre>Tidak ada solusi! Waktu pencarian: 0 ms Banyak kasus yang ditinjau: 624 Apakah Anda ingin menyimpan solusi dalam file .txt? (ya/tidak) ya Solusi berhasil disimpan di: test/solution_case2.txt Apakah Anda ingin menyimpan solusi dalam file .png? (ya/tidak) ya Solusi berhasil disimpan di: test/solution_case2.png</pre>

	solution_case2.txt: Tidak ada solusi! solution_case2.png:
--	---

4.1.3 Kasus Uji 3 – CUSTOM Solved

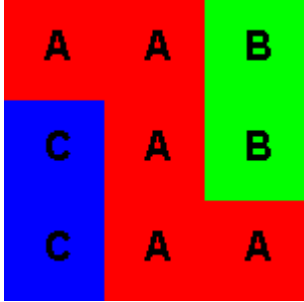
input.txt	output
3 5 3 CUSTOM ..X.. .XXX. ..X.. AAA B C	<pre> Solusi ditemukan: A BAC A Waktu pencarian: 0 ms Banyak kasus yang ditinjau: 195 Apakah Anda ingin menyimpan solusi dalam file .txt? (ya/tidak) ya Solusi berhasil disimpan di: test/solution_case3.txt Apakah Anda ingin menyimpan solusi dalam file .png? (ya/tidak) ya Solusi berhasil disimpan di: test/solution_case3.png </pre> <p>solution_case3.txt:</p> <pre> A BAC A </pre> <p>solution_case3.png:</p> 

4.1.4 Kasus Uji 4 – CUSTOM Not Solved

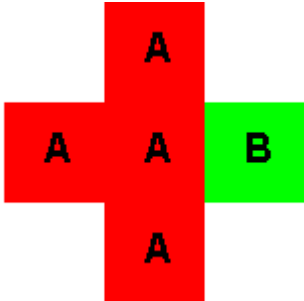
input.txt	output
4 4 2 CUSTOM X..X X..X AA A BB B	<pre>Tidak ada solusi! Waktu pencarian: 0 ms Banyak kasus yang ditinjau: 192 Apakah Anda ingin menyimpan solusi dalam file .txt? (ya/tidak) ya Solusi berhasil disimpan di: test/solution_case4.txt Apakah Anda ingin menyimpan solusi dalam file .png? (ya/tidak) ya Solusi berhasil disimpan di: test/solution_case4.png solution_case4.txt: Tidak ada solusi! solution_case4.png:</pre>

4.1.5 Kasus Uji 5 – DEFAULT dengan Whitespace di Awal

input.txt	output
3 3 3 DEFAULT A AAA A BB CC	<pre>Solusi ditemukan: AAB CAB CAA Waktu pencarian: 0 ms Banyak kasus yang ditinjau: 63 Apakah Anda ingin menyimpan solusi dalam file .txt? (ya/tidak) ya Solusi berhasil disimpan di: test/solution_case5.txt Apakah Anda ingin menyimpan solusi dalam file .png? (ya/tidak) ya Solusi berhasil disimpan di: test/solution_case5.png solution_case5.txt: AAB CAB</pre>

	CAA solution_case5.png: 
--	--

4.1.6 Kasus Uji 6 – CUSTOM dengan Whitespace di Awal

input.txt	output
3 5 2 CUSTOM ..X.. .XXX. ..X.. A AAA B	<pre> Solusi ditemukan: A AAB A Waktu pencarian: 0 ms Banyak kasus yang ditinjau: 111 Apakah Anda ingin menyimpan solusi dalam file .txt? (ya/tidak) ya Solusi berhasil disimpan di: test/solution_case6.txt Apakah Anda ingin menyimpan solusi dalam file .png? (ya/tidak) ya Solusi berhasil disimpan di: test/solution_case6.png </pre> solution_case6.txt: <pre> A AAB A </pre> solution_case6.png: 

4.1.7 Kasus Uji 7 – Tidak Menyimpan Solusi

input.txt	output
2 2 1 CUSTOM X. XX AA A	<pre>Solusi ditemukan: A AA Waktu pencarian: 0 ms Banyak kasus yang ditinjau: 5 Apakah Anda ingin menyimpan solusi dalam file .txt? (ya/tidak) tidak Apakah Anda ingin menyimpan solusi dalam file .png? (ya/tidak) tidak</pre>

BAB V

KESIMPULAN, SARAN, DAN REFLEKSI

5.1 Kesimpulan

Program IQ Puzzler Pro yang dikembangkan menggunakan algoritma Brute Force telah berhasil menyelesaikan masalah penempatan blok puzzle pada papan dengan berbagai ukuran dan konfigurasi. Berdasarkan pengujian yang dilakukan, dapat disimpulkan bahwa algoritma Brute Force bekerja dengan baik untuk puzzle berukuran kecil (misalnya, papan 3x3 atau 4x4) karena jumlah kemungkinan penempatan blok masih terbatas. Namun, untuk puzzle yang lebih besar, algoritma ini memerlukan waktu yang sangat lama karena kompleksitasnya yang tinggi. Program ini mampu menangani dua mode konfigurasi papan, yaitu DEFAULT dan CUSTOM. Program tidak hanya menampilkan solusi dalam bentuk teks, tetapi juga menyediakan opsi untuk menyimpan solusi dalam bentuk gambar (PNG). Fitur ini memudahkan pengguna untuk memahami solusi secara visual. Program juga telah dilengkapi dengan validasi *input* untuk memastikan bahwa *file input* sesuai dengan format yang diharapkan. Jika *input* tidak valid, program akan memberikan pesan *error* yang jelas.

5.2 Saran

Berdasarkan pengalaman selama mengembangkan dan menguji program ini, penambahan fitur seperti GUI (*Graphical User Interface*) akan memudahkan pengguna dalam memasukkan input dan melihat solusi. Begitu pula dengan penambahan animasi untuk menunjukkan proses penempatan blok secara bertahap.

5.2 Refleksi

Selama mengerjakan program ini, beberapa hal yang dapat direfleksikan oleh penulis adalah:

1. Pemahaman Tentang Algoritma Brute Force

Program ini memberikan pemahaman yang mendalam tentang cara kerja algoritma Brute Force, termasuk kelebihan dan keterbatasannya. Meskipun sederhana, algoritma ini memerlukan waktu yang lama untuk puzzle yang kompleks.

2. Pentingnya Validasi Input

Penting untuk memastikan bahwa program dapat menangani berbagai kasus input yang tidak terduga.

3. Manajemen Waktu dan Kompleksitas

Program ini mengajarkan pentingnya mempertimbangkan kompleksitas algoritma sejak awal pengembangan. Untuk proyek yang lebih besar, pemilihan algoritma yang tepat akan sangat memengaruhi kinerja program.

BAB VI

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	✓	