

Tugas Kecil 1 IF2211 Strategi Algoritma
Penyelesaian IQ Puzzler Pro dengan Algoritma Brute Force
Semester II tahun 2024/2025



Oleh:

Michael Alexander Angkawijaya 13523102

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
JL. GANESA 10, BANDUNG 40132
2024

Algoritma Brute Force untuk IQ Puzzler Pro Solver

1. Tempatkan blok pertama pada sel kosong pertama yang tersedia. Periksa apakah penempatan blok tersebut valid (dengan memastikan bahwa seluruh bagian blok berada dalam batas papan dan tidak bertumpang tindih dengan blok lain).
2. Jika tidak ada pelanggaran, lanjutkan dengan mencoba menempatkan blok berikutnya, mulai dari sel paling kiri atas yang masih kosong di papan.
3. Jika pada pemeriksaan ditemukan pelanggaran, yaitu penempatan blok tidak dibolehkan, maka coba dengan merotasi dan mencerminkan blok tersebut.
4. Jika seluruh kemungkinan rotasi dan cerminan dari blok tidak bisa ditempatkan di papan, maka kembali ke blok sebelumnya dan coba cari posisi lain untuk blok tersebut (backtrack).
5. Ulangi langkah 1 sampai seluruh sel terisi atau sampai semua kemungkinan diuji dan tidak ditemukan solusi.

Kode Algoritma

```
public boolean solve(int pieceIndex) {  
    iterationCount++;  
    if (pieceIndex == pieces.size()) {  
        if (board.getRemainingEmptyCells() == 0) {  
            solved = true;  
            return true;  
        }  
        return false;  
    }  
    Piece piece = pieces.get(pieceIndex);  
    for (int rotation = 0; rotation < 4; rotation++) {  
        for (int mirror = 0; mirror < 2; mirror++) {  
            for (int x = 0; x < board.getRows(); x++) {  
                for (int y = 0; y < board.getCols(); y++) {  
                    if (board.canPlacePiece(piece, x, y)) {  
                        board.placePiece(piece, x, y);  
                        if (solve(pieceIndex + 1)) {  
                            return true; // Solution found  
                        }  
                        board.removePiece(piece, x, y); // Backtrack  
                    }  
                }  
            }  
            piece.mirrorHorizontal();  
        }  
        piece.rotateClockwise();  
    }  
    return false; // No valid placement found for this piece  
}
```

Link Repository Program:

<https://github.com/angkaberapa/Tucil1-Stima-IQPuzzlerPro>

Source code Program:

Board.java

```
package sourcecode;
import java.io.FileWriter;
import java.io.IOException;

public class Board {
    private char[][] grid;
    private int rows, cols;
    private int remainingEmptyCells;
    private int piecePlacedCount;
    private int totalPiece;
    private static final String[] COLORS = {
        "\u001B[31m", // A - Red
        "\u001B[32m", // B - Green
        "\u001B[34m", // C - Blue
        "\u001B[33m", // D - Yellow
        "\u001B[35m", // E - Magenta
        "\u001B[36m", // F - Cyan
        "\u001B[91m", // G - Light Red
        "\u001B[92m", // H - Light Green
        "\u001B[94m", // I - Light Blue
        "\u001B[95m", // J - Light Magenta
        "\u001B[96m", // K - Light Cyan
        "\u001B[97m", // L - White
        "\u001B[90m", // M - Dark Gray
        "\u001B[33m", // N - Orange
        "\u001B[32m", // O - Light Green
        "\u001B[34m", // P - Blue
        "\u001B[35m", // Q - Magenta
        "\u001B[36m", // R - Cyan
        "\u001B[91m", // S - Light Red
        "\u001B[92m", // T - Light Green
        "\u001B[94m", // U - Light Blue
        "\u001B[95m", // V - Light Magenta
        "\u001B[96m", // W - Light Cyan
        "\u001B[97m", // X - White
        "\u001B[90m", // Y - Dark Gray
        "\u001B[31m", // Z - Red
    };

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

    public Board(char[][] grid, int totalPiece) {
        this.rows = grid.length;
        this.cols = grid[0].length;
        this.grid = new char[rows][cols];
        this.remainingEmptyCells = 0;
        this.piecePlacedCount = 0;
        this.totalPiece = totalPiece;
        // '*' represents empty cells
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                if (grid[i][j] == '*') {
                    remainingEmptyCells++;
                }
                this.grid[i][j] = grid[i][j];
            }
        }
    }
}
```

```

    public int getRows() {
        return rows;
    }
    public int getCols() {
        return cols;
    }
    public char[][] getGrid() {
        return grid;
    }
    public int getRemainingEmptyCells() {
        return remainingEmptyCells;
    }
    public int piecePlacedCount(){
        return piecePlacedCount;
    }
    public int getTotalPiece(){
        return totalPiece;
    }
    public void printBoard() {
        for (char[] row : grid) {
            for (char cell : row) {
                int index = cell - 'A';
                if (index >= 0 && index < COLORS.length) {
                    System.out.print(COLORS[index] + cell + RESET);
                } else {
                    System.out.print(cell);
                }
            }
            System.out.println();
        }
    }
    public boolean canPlacePiece(Piece piece, int x, int y) {
        char[][] shape = piece.getShape();
        int pieceRows = shape.length;
        int pieceCols = shape[0].length;

        for (int i = 0; i < pieceRows; i++) {
            for (int j = 0; j < pieceCols; j++) {
                if (shape[i][j] != ' ' && (x + i >= rows || y + j >= cols || grid[x + i][y + j] != '*')) {
                    return false;
                }
            }
        }
        return true;
    }
}

```

```

public void placePiece(Piece piece, int x, int y) {
    char[][] shape = piece.getShape();
    for (int i = 0; i < shape.length; i++) {
        for (int j = 0; j < shape[0].length; j++) {
            if (shape[i][j] != ' ') {
                grid[x + i][y + j] = shape[i][j];
                remainingEmptyCells--;
                piecePlacedCount++;
            }
        }
    }
}

public void removePiece(Piece piece, int x, int y) {
    char[][] shape = piece.getShape();
    for (int i = 0; i < shape.length; i++) {
        for (int j = 0; j < shape[0].length; j++) {
            if (shape[i][j] != ' ') {
                grid[x + i][y + j] = '*';
                remainingEmptyCells++;
                piecePlacedCount--;
            }
        }
    }
}

public void saveBoardToTxtFile(String txtSolutionPath) {
    try (FileWriter writer = new FileWriter(txtSolutionPath)) {
        for (char[] row : grid) {
            writer.write(row);
            writer.write(str: "\n"); // New line for each row
        }
        // System.out.println("Board saved to " + txtSolutionPath);
    } catch (IOException e) {
        e.printStackTrace();
    }
}

```

Piece.java

```

package sourcecode;
import java.util.List;

public class Piece {
    private char[][] shape;
    private static int pieceCount = 0;

    public Piece(List<String> shape) {
        int rows = shape.size();
        int cols = 0;
        for (String line : shape) {
            if (line.length() > cols) {
                cols = line.length();
            }
        }
    }
}

```

```

        this.shape = new char[rows][cols];
        for (int i = 0; i < rows; i++) {
            String line = shape.get(i);
            for (int j = 0; j < cols; j++) {
                this.shape[i][j] = (j < line.length()) ? line.charAt(j) : ' ';
            }
        }
        pieceCount++;
    }
}

```

```

public static int getPieceCount() {
    return pieceCount;
}

public char[][] getShape() {
    return shape;
}

public void rotateClockwise() {
    int rows = shape.length;
    int cols = shape[0].length;
    char[][] rotated = new char[cols][rows];

    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            rotated[j][rows - 1 - i] = shape[i][j];
        }
    }
    shape = rotated;
}

public void mirrorHorizontal() {
    int rows = shape.length;
    int cols = shape[0].length;
    char[][] mirrored = new char[rows][cols];

    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            mirrored[i][cols - 1 - j] = shape[i][j];
        }
    }
    shape = mirrored;
}

public void printPiece() {
    for (char[] row : shape) {
        System.out.println(new String(row));
    }
}
}

```

FileReader.java

```
package sourcecode;
import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class FileReader {
    private Scanner sc;

    public FileReader(String filename) throws FileNotFoundException {
        sc = new Scanner(new File(filename));
    }

    public Board readBoard() {
        int N = sc.nextInt();
        int M = sc.nextInt();
        int P = sc.nextInt();
        sc.nextLine();

        String boardType = sc.nextLine();
        char[][] boardGrid = new char[N][M];

        switch (boardType) {
            case "DEFAULT":
                for (int i = 0; i < N; i++) {
                    for (int j = 0; j < M; j++) {
                        boardGrid[i][j] = '*';
                    }
                }
                break;
            case "CUSTOM":
                for (int i = 0; i < N; i++) {
                    String line = sc.nextLine();
                    for (int j = 0; j < M; j++) {
                        boardGrid[i][j] = line.charAt(j);
                        if (boardGrid[i][j] == 'X') {
                            boardGrid[i][j] = '*';
                        }
                    }
                }
                break;
            default:
                System.out.println("Invalid board type!");
                return null;
        }

        return new Board(boardGrid, P);
    }
}
```

```

public List<Piece> readPieces() {
    List<Piece> pieces = new ArrayList<>();
    List<String> currentPiece = new ArrayList<>();
    char currentPieceChar = ' ';

    while (sc.hasNextLine()) {
        String tempLine = sc.nextLine();
        if (tempLine.isEmpty()) break;

        int idx = 0;
        int idx2 = tempLine.length();
        char tempChar = ' ';
        while((tempChar == ' ') && (idx < idx2)){
            tempChar = tempLine.charAt(idx);
            idx++;
        }
        if(tempChar == ' '){
            break;
        }
        if (currentPieceChar == tempChar) {
            currentPiece.add(tempLine);
        } else {
            if (!currentPiece.isEmpty()) {
                pieces.add(new Piece(currentPiece));
                currentPiece.clear();
            }
            currentPieceChar = tempChar;
            currentPiece.add(tempLine);
        }
    }

    if (!currentPiece.isEmpty()) {
        pieces.add(new Piece(currentPiece));
    }

    return pieces;
}

public void close() {
    sc.close();
}
}

```


Solver.java

```
package sourcecode;
import java.util.List;

public class Solver {
    private Board board;
    private List<Piece> pieces;
    private boolean solved;
    private int iterationCount = 0;

    public Solver(Board board, List<Piece> pieces) {
        this.board = board;
        this.pieces = pieces;
        this.solved = false;
    }

    public int getIterationCount() {
        return iterationCount;
    }

    public boolean solve(int pieceIndex) {
        iterationCount++;
        if (pieceIndex == pieces.size()) {
            if (board.getRemainingEmptyCells() == 0) {
                solved = true;
                return true;
            }
            return false;
        }
        Piece piece = pieces.get(pieceIndex);
        for (int rotation = 0; rotation < 4; rotation++) {
            for (int mirror = 0; mirror < 2; mirror++) {

                for (int x = 0; x < board.getRows(); x++) {
                    for (int y = 0; y < board.getCols(); y++) {
                        if (board.canPlacePiece(piece, x, y)) {
                            board.placePiece(piece, x, y);

                            if (solve(pieceIndex + 1)) {
                                return true; // Solution found
                            }

                            board.removePiece(piece, x, y); // Backtrack
                        }
                    }
                }
                piece.mirrorHorizontal();
            }
            piece.rotateClockwise();
        }
        return false; // No valid placement found for this piece
    }

    public void run() {
        if (solve(pieceIndex:0)) {
            System.out.println(x:"Solution found:");
            board.printBoard();
        } else {
            System.out.println(x:"No solution found.");
        }
    }
}
```

PuzzleImageSaver.java

```
package sourcecode;
import java.awt.Color;
import java.awt.Font;
import java.awt.FontMetrics;
import java.awt.Graphics2D;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;

import javax.imageio.ImageIO;

public class PuzzleImageSaver {
    private static final int CELL_SIZE = 50; // Ukuran setiap blok
    private static final int PADDING = 5; // Jarak antar blok
    private static final Color[] COLORS = {
        new Color(r:255, g:0, b:0), // Red
        new Color(r:0, g:255, b:0), // Green
        new Color(r:0, g:0, b:255), // Blue
        new Color(r:255, g:255, b:0), // Yellow
        new Color(r:255, g:165, b:0), // Orange
        new Color(r:128, g:0, b:128), // Purple
        new Color(r:0, g:255, b:255), // Cyan
        new Color(r:255, g:192, b:203), // Pink
        new Color(r:165, g:42, b:42), // Brown
        new Color(r:0, g:128, b:0), // Dark Green
        new Color(r:128, g:128, b:128), // Gray
        new Color(r:255, g:0, b:255), // Magenta
        new Color(r:0, g:0, b:128), // Navy
        new Color(r:128, g:128, b:0), // Olive
        new Color(r:255, g:69, b:0), // Red-Orange
        new Color(r:75, g:0, b:130), // Indigo
        new Color(r:139, g:69, b:19), // Saddle Brown
        new Color(r:210, g:105, b:30), // Chocolate
        new Color(r:244, g:164, b:96), // Sandy Brown
        new Color(r:255, g:228, b:181), // Moccasin
        new Color(r:173, g:216, b:230), // Light Blue
        new Color(r:60, g:179, b:113), // Medium Sea Green
        new Color(r:47, g:79, b:79), // Dark Slate Gray
        new Color(r:154, g:205, b:50), // Yellow Green
        new Color(r:70, g:130, b:180), // Steel Blue
        new Color(r:199, g:21, b:133), // Medium Violet Red
    };
};
```

```

public static BufferedImage savePuzzleImage(Board board, String filePath) {
    char[][] grid = board.getGrid();
    int rows = grid.length;
    int cols = grid[0].length;

    int width = cols * (CELL_SIZE + PADDING) + PADDING;
    int height = rows * (CELL_SIZE + PADDING) + PADDING;

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

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

    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            char piece = grid[i][j];

            if (piece != '.') {
                int index = piece - 'A';
                g2d.setColor(COLORS[index]);
                g2d.fillRect(j * (CELL_SIZE + PADDING) + PADDING, i * (CELL_SIZE + PADDING) + PADDING, CELL_SIZE, CELL_SIZE);

                g2d.setColor(Color.BLACK);
                g2d.setFont(new Font(name:"Arial", Font.BOLD, size:20));
                FontMetrics fm = g2d.getFontMetrics();
                int x = j * (CELL_SIZE + PADDING) + PADDING + (CELL_SIZE - fm.charWidth(piece)) / 2;
                int y = i * (CELL_SIZE + PADDING) + PADDING + (CELL_SIZE + fm.getAscent()) / 2 - fm.getDescent();
                g2d.drawString(String.valueOf(piece), x, y);
            }
            else {
                g2d.setColor(new Color(r:64, g:64, b:64)); // Dark Grey
                g2d.fillRect(j * (CELL_SIZE + PADDING) + PADDING, i * (CELL_SIZE + PADDING) + PADDING, CELL_SIZE, CELL_SIZE);
            }
        }
    }

    g2d.dispose();
    return image;
}

public static void saveImageToPdf(BufferedImage image, String filePath){
    try {
        ImageIO.write(image, formatName:"png", new File(filePath));
        // System.out.println("Puzzle saved to " + filePath);
    } catch (IOException e) {
        e.printStackTrace();
    }
}
}

```

Main.java (untuk CLI)

```
package sourcecode;
import java.io.FileNotFoundException;
import java.util.List;
import java.util.Scanner;

public class Main {
    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
        System.out.print(s:"Masukkan nama file: ");
        Scanner sc = new Scanner(System.in);
        String filename = sc.nextLine();
        try {
            FileReader reader = new FileReader("../test/" + filename);

            Board board = reader.readBoard();

            List<Piece> pieces = reader.readPieces();

            long startTime = System.currentTimeMillis();

            Solver solver = new Solver(board, pieces);
            Boolean solved = solver.solve(pieceIndex:0);
            long endTime = System.currentTimeMillis();
            long elapsedTime = endTime - startTime;
            if (solved) {
                System.out.println(x:"Solution found:");
                board.printBoard();
                System.out.println("Waktu pencarian: " + elapsedTime + " ms");
                System.out.println("Banyak kasus yang ditinjau: " + solver.getIterationCount());
                System.out.println(x:"Apakah anda ingin menyimpan solusi? (ya/tidak)");
                String save = sc.nextLine();
                if(save.equals(anObject:"ya")){
                    board.saveBoardToTxtFile("../test/" + filename.substring(beginIndex:0,filename.lastIndexOf(str:".")) + "_solution.txt");
                    PuzzleImageSaver.savePuzzleImage(board, "../test/" + filename.substring(beginIndex:0,filename.lastIndexOf(str:".")) + "_solution.png");
                }
            } else {
                System.out.println(x:"No solution found.");
                System.out.println("Waktu pencarian: " + elapsedTime + " ms");
                System.out.println("Banyak kasus yang ditinjau: " + solver.getIterationCount());
            }
            sc.close();
        } catch (FileNotFoundException e) {
            System.out.println(x:"File not found.");
        }
    }
}
```

App.java (GUI)

```
package com.myapp;

import java.io.IOException;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

/**
 * JavaFX App
 */
public class App extends Application {

    private static Scene scene;

    @Override
    public void start(Stage stage) throws IOException {
        scene = new Scene(loadFXML("main"));
        stage.setScene(scene);
        stage.show();
    }

    static void setRoot(String fxml) throws IOException {
        scene.setRoot(loadFXML(fxml));
    }

    private static Parent loadFXML(String fxml) throws IOException {
        FXMLLoader fxmlLoader = new FXMLLoader(App.class.getResource(fxml + ".fxml"));
        return fxmlLoader.load();
    }

    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
        launch();
    }
}
```

Controller.java

```
package com.myapp;

import java.awt.image.BufferedImage;
import java.io.File;
import java.io.FileNotFoundException;
import java.util.List;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.ScrollPane;
import javafx.scene.control.TextArea;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.image.PixelWriter;
import javafx.scene.image.WritableImage;
import javafx.scene.layout.StackPane;
import javafx.stage.FileChooser;
import javafx.stage.Stage;
import sourcecode.Board;
import sourcecode.FileReader;
import sourcecode.Piece;
import sourcecode.PuzzleImageSaver;
import sourcecode.Solver;

public class Controller {

    @FXML
    private ScrollPane Canvas;

    @FXML
    private Button SavePngButton;

    @FXML
    private Button SaveTxtButton;

    @FXML
    private Button SolveButton;

    @FXML
    private Button UploadFileButton;

    @FXML
    private TextArea textArea; // Added TextArea for output messages

    private File selectedFile;
    private Board board;
    private List<Piece> pieces;
    private Solver solver;
    private String solutionImagePath;
    private BufferedImage image;
```

```

@FXML
void uploadTxtFile(ActionEvent event) {
    FileChooser fileChooser = new FileChooser();
    fileChooser.getExtensionFilters().add(new FileChooser.ExtensionFilter(description:"Text Files", ...extensions:"*.txt"));
    fileChooser.setTitle(value:"Select a Puzzle File");

    File file = fileChooser.showOpenDialog(new Stage());
    if (file != null) {
        selectedFile = file;
        textArea.appendText("File selected: " + selectedFile.getAbsolutePath() + "\n");

        try {
            FileReader reader = new FileReader(selectedFile.getAbsolutePath());
            board = reader.readBoard();
            pieces = reader.readPieces();
        } catch (FileNotFoundException e) {
            showAlert(title:"Error", message:"File not found.", Alert.AlertType.ERROR);
        }
    }
}

```

```

@FXML
void solvePuzzle(ActionEvent event) {
    if (selectedFile == null || board == null || pieces == null) {
        showAlert(title:"Error", message:"No file selected or invalid puzzle data.", Alert.AlertType.ERROR);
        return;
    }

    solver = new Solver(board, pieces);

    long startTime = System.currentTimeMillis();
    boolean solved = solver.solve(pieceIndex:0);
    long endTime = System.currentTimeMillis();
    long elapsedTime = endTime - startTime;

    if (solved) {
        textArea.appendText("Solution found!\n");
        textArea.appendText("Waktu pencarian: " + elapsedTime + " ms\n");
        textArea.appendText("Banyak kasus yang ditinjau: " + solver.getIterationCount() + "\n");

        // Save solution image path
        solutionImagePath = selectedFile.getParent() + "/" + selectedFile.getName().replace(target:".txt", replacement:"_solution.png");
        image = PuzzleImageSaver.savePuzzleImage(board, solutionImagePath);
        Image fxImage = convertBufferedImageToFXImage(image);
        ImageView imageView = new ImageView(fxImage);

        // Show image in ScrollPane
        imageView.setPreserveRatio(value:true);
        imageView.setFitWidth(Canvas.getWidth() - 20);
        imageView.setFitHeight(Canvas.getHeight() - 20);
        imageView.setImage(fxImage);
        // Use StackPane to center the image
        StackPane container = new StackPane(imageView);
        container.setPrefSize(Canvas.getPrefWidth(), Canvas.getPrefHeight());

        // Set content to ScrollPane
        Canvas.setContent(container);
    } else {
        textArea.appendText("No solution found.\n");
        textArea.appendText("Waktu pencarian: " + elapsedTime + " ms\n");
        textArea.appendText("Banyak kasus yang ditinjau: " + solver.getIterationCount() + "\n");
    }
}

```

```

@FXML
void saveToPng(ActionEvent event) {
    if (solutionImagePath != null) {
        PuzzleImageSaver.saveImageToPdf(image, solutionImagePath);
        showAlert(title:"Success", "Solution saved as PNG: " + solutionImagePath, Alert.AlertType.INFORMATION);
    } else {
        showAlert(title:"Error", message:"No solution to save.", Alert.AlertType.ERROR);
    }
}

@FXML
void saveToTxt(ActionEvent event) {
    if (selectedFile != null && board != null) {
        String txtSolutionPath = selectedFile.getParent() + "/" +
            selectedFile.getName().replace(target:".txt", replacement:"_solution.txt");

        board.saveBoardToTxtFile(txtSolutionPath);
        showAlert(title:"Success", "Solution saved as TXT: " + txtSolutionPath, Alert.AlertType.INFORMATION);
    } else {
        showAlert(title:"Error", message:"No solution to save.", Alert.AlertType.ERROR);
    }
}

private void showAlert(String title, String message, Alert.AlertType alertType) {
    Alert alert = new Alert(alertType);
    alert.setTitle(title);
    alert.setHeaderText(null);
    alert.setContentText(message);
    alert.showAndWait();
}

public static Image convertBufferedImageToFXImage(BufferedImage bufferedImage) {
    int width = bufferedImage.getWidth();
    int height = bufferedImage.getHeight();
    WritableImage writableImage = new WritableImage(width, height);
    PixelWriter pixelWriter = writableImage.getPixelWriter();

    for (int y = 0; y < height; y++) {
        for (int x = 0; x < width; x++) {
            int argb = bufferedImage.getRGB(x, y);
            pixelWriter.setArgb(x, y, argb);
        }
    }

    return writableImage;
}
}

```


Uji Coba Test Case

1. Input:

5 5 7
DEFAULT
A
AA
B
BB
C
CC
D
DD
EE
EE
E
FF
FF
F
GGG

Output:

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: 223 ms

Banyak kasus yang ditinjau: 27024

2. Input:

5 7 5
CUSTOM
...X...
.XXXXX.
XXXXXXXX
.XXXXX.
...X...
A
AAA
BB
BBB
CCCC
C
D
EEE
E

Output:

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

Waktu pencarian: 3 ms

Banyak kasus yang ditinjau: 218

3. Input:

1 3 2
DEFAULT
AAA
BBB

Output:

No solution found.
Waktu pencarian: 0 ms
Banyak kasus yang ditinjau: 5

4. Input:

4 5 5
DEFAULT
AA
A
AA
BB
B
CCC
CC
DDD
E
EE
E

Output:

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

Waktu pencarian: 11 ms
Banyak kasus yang ditinjau: 1408

5. Input:

4 5 5
DEFAULT
AA
A
AA
BB
B
CCC
C
DDD
E
EEE
E

Output:

No solution found.
Waktu pencarian: 292 ms
Banyak kasus yang ditinjau: 75761

6. **Input:**

5 12 9
CUSTOM
...X...X...
.XXXXX.XXXX.
XXXXXXXXXXXXXXXXX
.XXXXX.XXXX.
...X...X...
A
AAA
BB
BBB
CCCC
C
D
EEE
E
FFF
FFF
FFF
G
GG
G
H
I

7. **Input:**

5 7 5
CUSTOM
...X...
.XXXXX.
XXXXXXXXX
.XXXXX.
...X...
A
AAA
BB
BBB
CCCC
C
D
EEE
E

Output:

			A					D			
	B	B	A	A	A		F	F	F	G	
B	B	B	C	C	C	C	F	F	F	G	G
	E	E	E	C	H		F	F	F	G	
			E					I			

Waktu pencarian: 2034 ms

Banyak kasus yang ditinjau: 389404

Output:

No solution found.

Waktu pencarian: 62 ms

Banyak kasus yang ditinjau: 9589

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	✓	