Laporan Tugas Kecil 1 IF2211 Strategi Algoritma

Semester II Tahun Akademik 2024/2025

Penyelesaian IQ Puzzler Pro dengan Algoritma Brute Force

Disusun Oleh:

Muhammad Adha Ridwan – 13523098

K-02

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2025

Daftar Isi

1. Deskripsi Tugas

Gambar 1. Permainan IQ Puzzler Pro

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

Komponen penting dari permainan IQ Puzzler Pro terdiri dari:

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

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

Tugas anda adalah menemukan cukup satu solusi dari permainan IQ Puzzler Pro dengan menggunakan algoritma Brute Force, atau menampilkan bahwa solusi tidak ditemukan jika tidak ada solusi yang mungkin dari puzzle.

2. Algoritma Brute Force

Pada laporan ini, algoritma yang digunakan oleh penulis adalah bruteforce dengan pendekatan *pure bruteforce.*

2.1 Pendekatan *pure bruteforce*

Pendekatan *pure bruteforce* yang penulis gunakan adalah, untuk setiap *piece* puzzle akan dicoba dipasangkan di *cell* papan yang kosong, kemudian akan dicari *cell* papan selanjutnya yang masih kosong. Setiap piece akan dicoba untuk setiap konfigurasi rotasi sebanyak 4 dan balikan sebanyak 2.

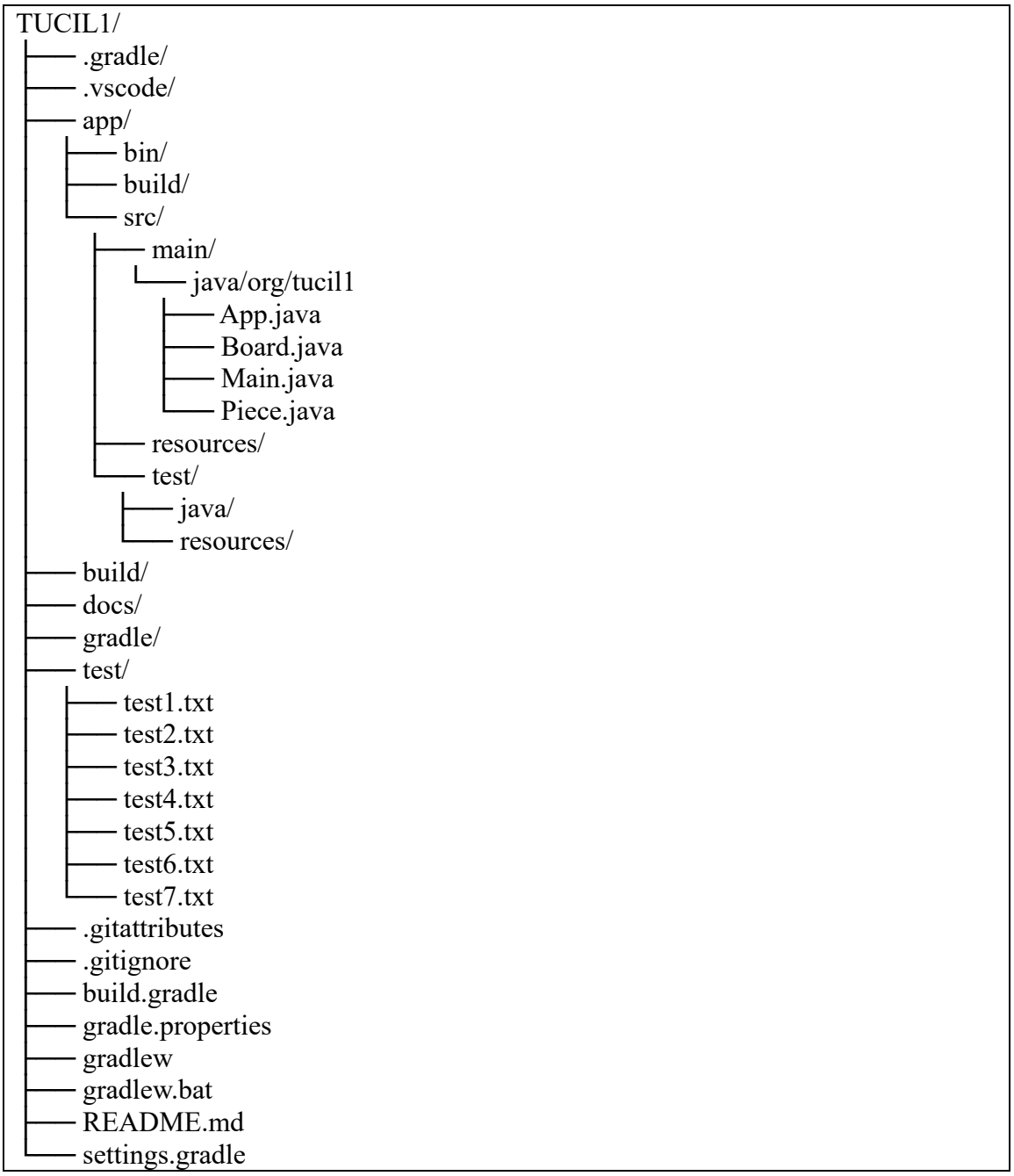
2.2 Langkah Langkah bruteforce

Berikut langkah-lankah penyelesaian

1. Setiap *piece* akan di parse dan disimpan kedalam list of piece.
2. Pemasangan *piece* pada papan selalu dimulai dari *cell* papan (1,1) dan mengarah dari kiri -kanan dan atas-bawah.
3. Untuk setiap *piece* akan dicoba dipasangkan ke papan, jika berhasil *piece* terpasang akan ditandai bahwa terpakai dan akan lanjut ke *piece* selanjutnya.
4. Akan dicari *cell* papan kosong selanjutnya dengan arah kiri-kanan dan atas-bawah, jika ditemukan *piece* akan dipasang.
5. Untuk setiap *piece* akan dicoba juga semua konfigurasi rotasi dan balikan.
6. Untuk setiap *piece* yang dipasang dan dilepas akan disimpan jumlah *cell* yang kosong untuk menentukan apakah Solusi sudah ditemukan atau tidak.
7. Jika jumlah *cell* yang kosong sama dengan 0, pencarian dihentikan dan konfigurasi *piece* saat itu menjadi Solusi.

3. Source Code

Penulis membagi kode program menjadi 4 file .java yakni, Board.java, Piece.java, Main.java, dan App.java. Struktur *workspace* sebagai berikut



3.1 App.java

```
package org.tucil1;

import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.VBox;
import javafx.scene.layout.HBox;
import javafx.stage.FileChooser;
import javafx.stage.Stage;
import javafx.scene.layout.Pane;
import javafx.scene.image.WritableImage;
import javafx.scene.SnapshotParameters;

import java.io.BufferedReader;
import java.io.File;
import java.io.IOException;
import java.nio.file.Files;
import java.util.ArrayList;
import java.util.List;
import javafx.scene.control.Alert;
import javafx.scene.control.Alert.AlertType;
import java.awt.image.BufferedImage;
import javafx.scene.paint.Color;

public class App extends Application {
    private Board board;
    private GridPane boardDisplay;
    private Label statusLabel;
    private Button solveButton;
    private Button savePNGButton;
    private Button saveTXTButton;
    private Button resetButton;
    private Label metricsLabel;


    private static final String[] CSS_COLORS = {
        "", // Index 0 (unused)
        "rgb(255,0,0)",         // 1
        "rgb(255,128,0)",       // 2
        "rgb(255,255,0)",       // 3
        "rgb(128,255,0)",       // 4
        "rgb(0,255,0)",         // 5
        "rgb(0,255,128)",       // 6
        "rgb(0,255,255)",       // 7
        "rgb(0,128,255)",       // 8
        "rgb(0,0,255)",         // 9
        "rgb(128,0,255)",       // 10
        "rgb(255,0,255)",       // 11
        "rgb(255,0,128)",       // 12
        "rgb(255,128,128)",     // 13
        "rgb(255,191,0)",       // 14
        "rgb(191,255,0)",       // 15
        "rgb(0,255,191)",       // 16
        "rgb(0,191,255)",       // 17
        "rgb(128,128,255)",     // 18
        "rgb(255,128,255)",     // 19
        "rgb(255,69,0)",        // 20
        "rgb(0,128,128)",        // 21
        "rgb(128,0,128)",        // 22
        "rgb(128,128,0)",        // 23
        "rgb(255,215,0)",        // 24
        "rgb(169,169,169)",      // 25
        "rgb(0,191,191)"         // 26
    };
```

```
    public static String getColorforGUI(int id) {
        if (id < 1 || id > 26) {
            return "rgb(255,255,255)";
        }
        return CSS_COLORS[id];
    }

    public static boolean samePieceBlock(String a , String b){
        char charA = '!';
        char charB = '!';
        for(int i = 0; i < a.length(); i++){
            if(a.charAt(i) >= 'A' && a.charAt(i) <= 'Z'){
                charA = a.charAt(i);
            }
        }

        for(int i = 0 ; i < b.length(); i++){
            if(b.charAt(i) >= 'A' && b.charAt(i) <= 'Z'){
                charB = b.charAt(i);
            }
        }

        return charA == charB;
    }

    public static Piece getPieceFromString(List<String> rawPieces){
        int n = rawPieces.size();
        int m = -1;
        for(int i = 0 ; i < n;i++){
            m = Math.max(m, rawPieces.get(i).length());
        }

        int x = -1;
        int y = -1;
        boolean[][] pieceShape = new boolean[n][m];

        for(int i = 0 ; i < n;i++){
            for(int j = 0 ; j < m;j++){
                if(j >= rawPieces.get(i).length()){
                    pieceShape[i][j] = false;
                }else{
                            if(rawPieces.get(i).charAt(j)  >=  'A'  &&
rawPieces.get(i).charAt(j) <= 'Z'){
                        pieceShape[i][j] = true;
                        x = i;
                        y = j;
                    }
                }
            }
        }

        char id = getPieceID(rawPieces.get(0));
        int intID = id - 'A';
        if(x == -1 || y == -1){
            throw new IllegalArgumentException("Pieces broken!");
        }
        return new Piece(n, m, pieceShape, intID);

    }

    public static char getPieceID(String piece){
        for(int i = 0 ; i < piece.length(); i++){
            if(piece.charAt(i) >= 'A' && piece.charAt(i) <= 'Z'){
                return piece.charAt(i);
            }
        }
        return '!';
    }

      public  static  Piece[]  proccesPieceFromStringList(List<String>
rawPiecesList, int p){
```

```
            boolean[] usedChar = new boolean[26];
            for(int i = 0 ; i < 26 ; i++){
                usedChar[i] = false;
            }

            Piece[] pieces = new Piece[p+1];
            int cnt = 1;
            int len = rawPiecesList.size();
            for(int i = 0 ; i < len;i++){
                int pieceCount = i;
                                while(samePieceBlock(rawPiecesList.get(i),
    rawPiecesList.get(pieceCount))){
                    pieceCount++;
                    if(pieceCount == len){
                        break;
                    }
                }
                char pieceID = getPieceID(rawPiecesList.get(i));
                if(usedChar[pieceID - 'A']){
                       throw new IllegalArgumentException("Pieces Already
    Used!");
                }else{
                    usedChar[pieceID - 'A'] = true;
                }
               pieces[cnt++] = getPieceFromString(rawPiecesList.subList(i,
    pieceCount));

                i = pieceCount-1;

            }
            return pieces;
        }
        @Override
        public void start(Stage primaryStage) {
            VBox root = new VBox(20);
            root.setPadding(new Insets(20));
            root.setAlignment(Pos.CENTER);

            statusLabel = new Label("Please load a puzzle file");
            statusLabel.setStyle("-fx-font-size: 14px;");

            metricsLabel = new Label("");
               metricsLabel.setStyle("-fx-font-size: 12px; -fx-text-fill:
    #666666;");

            HBox buttonBox = new HBox(10);
            buttonBox.setAlignment(Pos.CENTER);

            Button loadButton = new Button("Load Puzzle");
            solveButton = new Button("Solve!");
            savePNGButton = new Button("Save as PNG");
            saveTXTButton = new Button("Save as TXT");
            resetButton = new Button("Reset");

            solveButton.setDisable(true);
            savePNGButton.setDisable(true);
            saveTXTButton.setDisable(true);
            resetButton.setDisable(true);

                 buttonBox.getChildren().addAll(resetButton,  loadButton,
    solveButton, savePNGButton, saveTXTButton);

            boardDisplay = new GridPane();
            boardDisplay.setAlignment(Pos.CENTER);
            boardDisplay.setHgap(1);
            boardDisplay.setVgap(1);
            boardDisplay.setStyle("-fx-background-color: white;");

           root.getChildren().addAll(statusLabel, metricsLabel, buttonBox,
    boardDisplay);

            loadButton.setOnAction(e -> {
```

```
                FileChooser fileChooser = new FileChooser();
                fileChooser.getExtensionFilters().add(
                    new FileChooser.ExtensionFilter("Text Files", "*.txt")
                );
                                                File      selectedFile      =
        fileChooser.showOpenDialog(primaryStage);

                if (selectedFile != null) {
                    try {
                                                 BufferedReader   reader   =
        Files.newBufferedReader(selectedFile.toPath());
                        String line = reader.readLine();

                        String[] nm = line.split(" ");
                        int n = Integer.parseInt(nm[0]);
                        int m = Integer.parseInt(nm[1]);
                        int p = Integer.parseInt(nm[2]);

                        String boardType = reader.readLine();

                        List<String> pieces = new ArrayList<>();
                        while ((line = reader.readLine()) != null) {
                            pieces.add(line);
                        }
                        reader.close();

                                                    Piece[]    pieceList    =
        proccesPieceFromStringList(pieces, p);
                        int pieceCount = p;

                        board = new Board(n, m, pieceList, pieceCount);
                                  statusLabel.setText("File  loaded:  "  +
        selectedFile.getName());
                        metricsLabel.setText("");
                        solveButton.setDisable(false);
                        updateBoardDisplay();

                         } catch (IOException | NumberFormatException |
        IndexOutOfBoundsException ex) {
                            showAlert("Error", "Failed to parse file: " +
        ex.getMessage());
                        }
                    }
                });

                solveButton.setOnAction(e -> {
                    if (board != null) {
                        long startTime = System.currentTimeMillis();
                        Board.resetIterationCount();
                        board.solve();
                        long endTime = System.currentTimeMillis();
                        long executionTime = endTime - startTime;

                        updateBoardDisplay();
                        savePNGButton.setDisable(false);
                        saveTXTButton.setDisable(false);
                        resetButton.setDisable(false);

                        if (board.foundSolution) {
                            statusLabel.setText("Solution found!");
                           metricsLabel.setText(String.format("Execution time:
        %d ms | Iterations: %d",
                                executionTime, Board.getIterationCount()));
                        } else {
                            statusLabel.setText("No solution found!");
                           metricsLabel.setText(String.format("Execution time:
        %d ms | Iterations: %d (No solution)",
                                executionTime, Board.getIterationCount()));
                        }
                    }
                });
```

```
                saveTXTButton.setOnAction(e -> {
                    FileChooser fileChooser = new FileChooser();
                    fileChooser.setTitle("Save Solution");
                    fileChooser.getExtensionFilters().add(
                        new FileChooser.ExtensionFilter("Text Files", "*.txt")
                    );

                    fileChooser.setInitialFileName("puzzle_solution");

                    File file = fileChooser.showSaveDialog(primaryStage);
                    if (file != null) {
                        String filename = file.getAbsolutePath();
                        if (filename.toLowerCase().endsWith(".txt")) {
                            filename = filename.substring(0, filename.length()
        - 4);
                        }
                        board.saveSolution(filename);
                    }
                });

                resetButton.setOnAction(e -> {
                    board.pieces = null;
                    board.grid = null;
                    board = null;
                    updateBoardDisplay();
                    solveButton.setDisable(true);
                    savePNGButton.setDisable(true);
                    saveTXTButton.setDisable(true);
                    resetButton.setDisable(true);
                    statusLabel.setText("Please load a puzzle file");
                    metricsLabel.setText("");
                });

                savePNGButton.setOnAction(e -> {
                    FileChooser fileChooser = new FileChooser();
                    fileChooser.getExtensionFilters().add(
                        new FileChooser.ExtensionFilter("PNG Files", "*.png")
                    );
                    File file = fileChooser.showSaveDialog(primaryStage);

                    if (file != null) {
                        try {
                            WritableImage snapshot = boardDisplay.snapshot(new
        SnapshotParameters(), null);

                            BufferedImage bufferedImage = new BufferedImage(
                                (int) snapshot.getWidth(),
                                (int) snapshot.getHeight(),
                                BufferedImage.TYPE_INT_ARGB
                            );

                            for (int x = 0; x < snapshot.getWidth(); x++) {
                                for (int y = 0; y < snapshot.getHeight(); y++)
        {
                                                                Color   color   =
        snapshot.getPixelReader().getColor(x, y);
                                        int argb = ((int) (color.getOpacity() *
        255) << 24) |
                                                ((int) (color.getRed() * 255) <<
        16) |
                                                 ((int) (color.getGreen() * 255)
        << 8) |
                                                ((int) (color.getBlue() * 255));
                                    bufferedImage.setRGB(x, y, argb);
                                }
                            }

                             javax.imageio.ImageIO.write(bufferedImage, "png",
        file);
                            showAlert("Success", "Board saved successfully!");
                        } catch (IOException ex) {
```

```
                        showAlert("Error", "Failed to save the image: " +
ex.getMessage());
                    }
                }
            });


            Scene scene = new Scene(root, 600, 500);
            primaryStage.setTitle("Pentomino Solver");
            primaryStage.setScene(scene);
            primaryStage.show();
        }

        private void updateBoardDisplay() {
            boardDisplay.getChildren().clear();
            if (board == null) return;

            for (int i = 1; i <= board.n; i++) {
                for (int j = 1; j <= board.m; j++) {
                    Pane cell = new Pane();
                    cell.setPrefSize(30, 30);

                    char piece = board.grid[i][j];
                    if (piece != '.') {
                        String color = getColorforGUI(piece - 'A' + 1);
                         cell.setStyle("-fx-background-color: " + color +
";");
                    } else {
                        cell.setStyle("-fx-background-color: black;");
                    }

                    boardDisplay.add(cell, j-1, i-1);
                }
            }
        }

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

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

3.2 Board.java

```
    package org.tucil1;

    import java.io.FileWriter;
    import java.io.IOException;
    import java.time.LocalDateTime;
    import java.time.format.DateTimeFormatter;
    import java.util.Arrays;

    public class Board {
        char[][] grid;
        int n;
        int m;
        Piece[] pieces;
        int pieceCount;
        static int iterationCount = 0;
        boolean foundSolution = false;
        int emptyCellCount;
```

```
    public static final String RESET = "\u001B[0m";
    private static final String[] ANSI_COLORS = {
        "", //empty index-0
        "\u001B[38;2;255;0;0m",
        "\u001B[38;2;255;128;0m",
        "\u001B[38;2;255;255;0m",
        "\u001B[38;2;128;255;0m",
        "\u001B[38;2;0;255;0m",
        "\u001B[38;2;0;255;128m",
        "\u001B[38;2;0;255;255m",
        "\u001B[38;2;0;128;255m",
        "\u001B[38;2;0;0;255m",
        "\u001B[38;2;128;0;255m",
        "\u001B[38;2;255;0;255m",
        "\u001B[38;2;255;0;128m",
        "\u001B[38;2;255;128;128m",
        "\u001B[38;2;255;191;0m",
        "\u001B[38;2;191;255;0m",
        "\u001B[38;2;0;255;191m",
        "\u001B[38;2;0;191;255m",
        "\u001B[38;2;128;128;255m",
        "\u001B[38;2;255;128;255m",
        "\u001B[38;2;255;69;0m",
        "\u001B[38;2;0;128;128m",
        "\u001B[38;2;128;0;128m",
        "\u001B[38;2;128;128;0m",
        "\u001B[38;2;255;215;0m",
        "\u001B[38;2;169;169;169m",
        "\u001B[38;2;0;191;191m"
    };

    public static String getColor(int id) {
        if (id < 1 || id > 26) {
            return RESET;
        }
        return ANSI_COLORS[id];
    }

    public Board(int n, int m, Piece[] pieces, int pieceCount) {
        this.n = n;
        this.m = m;
        this.pieces = pieces;
        this.pieceCount = pieceCount;
        this.emptyCellCount = m * n;
        this.grid = new char[n + 2][m + 2];
        initializeGrid();
    }

    private void initializeGrid() {
        for (int i = 1; i <= n; i++) {
            Arrays.fill(grid[i], 1, m + 1, '.');
        }
    }

    public void printBoard() {
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= m; j++) {
                System.out.print(getColor(grid[i][j] - 'A' + 1) +
grid[i][j] + RESET);
            }
            System.out.println();
        }
    }

    private boolean placePiece(int x, int y, Piece piece) {
        if (x + piece.n - 1 > n || y + piece.m - 1 > m) return false;

        for (int i = 1; i <= piece.n; i++) {
            for (int j = 1; j <= piece.m; j++) {
```

```
                    if (piece.shape[i][j] && grid[x + i - 1][y + j - 1]
!= '.') {
                        return false;
                    }
                }
            }

            for (int i = 1; i <= piece.n; i++) {
                for (int j = 1; j <= piece.m; j++) {
                    if (piece.shape[i][j]) {
                        grid[x + i - 1][y + j - 1] = (char) ('A' +
piece.id);
                        emptyCellCount--;
                    }
                }
            }
            return true;
        }

        private void removePiece(int x, int y, Piece piece) {
            for (int i = 1; i <= piece.n; i++) {
                for (int j = 1; j <= piece.m; j++) {
                    if (piece.shape[i][j]) {
                        grid[x + i - 1][y + j - 1] = '.';
                        emptyCellCount++;
                    }
                }
            }
        }

        public static int getIterationCount() {
            return iterationCount;
        }

        public static void resetIterationCount() {
            iterationCount = 0;
        }

        public void saveSolution(String filename) {
            if (!foundSolution) {
                System.out.println("No solution to save!");
                return;
            }

            try {
                LocalDateTime now = LocalDateTime.now();
                DateTimeFormatter formatter =
DateTimeFormatter.ofPattern("yyyy-MM-dd_HH-mm-ss");
                String timestamp = now.format(formatter);
                String fullFilename = filename + "_" + timestamp +
".txt";

                FileWriter writer = new FileWriter(fullFilename);

                writer.write("Puzzle Solution\n");
                writer.write("Grid size: " + n + "x" + m + "\n");
                writer.write("Number of pieces: " + pieceCount + "\n");
                writer.write("Iterations: " + iterationCount + "\n\n");

                writer.write("Solution:\n");
                for (int i = 1; i <= n; i++) {
                    for (int j = 1; j <= m; j++) {
                        writer.write(grid[i][j]);
                    }
                    writer.write("\n");
                }

                writer.close();
                System.out.println("Solution saved to: " + fullFilename);

            } catch (IOException e) {
```

```
                System.err.println("Error saving solution: " +
        e.getMessage());
            }
        }

        public void findCombination(int x, int y, boolean[] usedPieces) {
            if (foundSolution) return;

            iterationCount++;

            if (emptyCellCount == 0) {
                foundSolution = true;
                return;
            }

            int nextX = x, nextY = y;
            boolean found = false;

            for (int i = 1; i <= n && !found; i++) {
                for (int j = 1; j <= m && !found; j++) {
                    if (grid[i][j] == '.') {
                        nextX = i;
                        nextY = j;
                        found = true;
                    }
                }
            }

            if (!found) return;

            for (int i = 1; i <= pieceCount && !foundSolution; i++) {
                if (usedPieces[i]) continue;

                Piece curPiece = pieces[i];
                usedPieces[i] = true;

                for (int rot = 0; rot < 4 && !foundSolution; rot++) {
                    for (int flip = 0; flip < 2 && !foundSolution;
        flip++) {
                        if (placePiece(nextX, nextY, curPiece)) {
                            findCombination(nextX, nextY, usedPieces);
                            if (!foundSolution) {
                                removePiece(nextX, nextY, curPiece);
                            }
                        }
                        curPiece = curPiece.flip();
                    }
                    curPiece = curPiece.rotate();
                }

                if (!foundSolution) {
                    usedPieces[i] = false;
                }
            }
        }

        public void solve() {
            boolean[] usedPieces = new boolean[pieceCount + 1];
            long startTime = System.currentTimeMillis();
            findCombination(1,1,usedPieces);
            long endTime = System.currentTimeMillis();

            System.out.println("Time Elapsed: " + (endTime - startTime) +
        " ms");
            System.out.println("Iteration Count: " + iterationCount);

            if (foundSolution) {
                System.out.println("Solution Found!");
                printBoard();
            } else {
                System.out.println("No Solution Found!");
            }
```

```
        }

    }
```

3.3 Main.java

```
    package org.tucil1;

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

    public class Main {
        public static boolean samePieceBlock(String a , String b){
            char charA = '!';
            char charB = '!';
            for(int i = 0; i < a.length(); i++){
                if(a.charAt(i) >= 'A' && a.charAt(i) <= 'Z'){
                    charA = a.charAt(i);
                }
            }

            for(int i = 0 ; i < b.length(); i++){
                if(b.charAt(i) >= 'A' && b.charAt(i) <= 'Z'){
                    charB = b.charAt(i);
                }
            }

            return charA == charB;
        }

        public static Piece getPieceFromString(List<String> rawPieces){
            int n = rawPieces.size();
            int m = -1;
            for(int i = 0 ; i < n;i++){
                m = Math.max(m, rawPieces.get(i).length());
            }

            int x = -1;
            int y = -1;
            boolean[][] pieceShape = new boolean[n][m];

            for(int i = 0 ; i < n;i++){
                for(int j = 0 ; j < m;j++){
                    if(j >= rawPieces.get(i).length()){
                        pieceShape[i][j] = false;
                    }else{
                        if(rawPieces.get(i).charAt(j) >= 'A' &&
    rawPieces.get(i).charAt(j) <= 'Z'){
                            pieceShape[i][j] = true;
                            x = i;
                            y = j;
                        }
                    }
                }
            }

            if(x == -1 || y == -1){
                throw new IllegalArgumentException("Pieces broken!");
            }
            return new Piece(n, m, pieceShape);

        }

        public static char getPieceID(String piece){
            for(int i = 0 ; i < piece.length(); i++){
                if(piece.charAt(i) >= 'A' && piece.charAt(i) <= 'Z'){
                    return piece.charAt(i);
                }
            }
            return '!';
```

```
        }

        public static Piece[] proccesPieceFromStringList(List<String>
    rawPiecesList, int p){
            boolean[] usedChar = new boolean[26];
            for(int i = 0 ; i < 26 ; i++){
                usedChar[i] = false;
            }

            Piece[] pieces = new Piece[p+1];
            int cnt = 1;
            int len = rawPiecesList.size();
            for(int i = 0 ; i < len;i++){
                int pieceCount = i;
                while(samePieceBlock(rawPiecesList.get(i),
    rawPiecesList.get(pieceCount))){
                    pieceCount++;
                    if(pieceCount == len){
                        break;
                    }
                }
                char pieceID = getPieceID(rawPiecesList.get(i));
                if(usedChar[pieceID - 'A']){
                    throw new IllegalArgumentException("Pieces Already
    Used!");
                }else{
                    usedChar[pieceID - 'A'] = true;
                }
                pieces[cnt++] =
    getPieceFromString(rawPiecesList.subList(i, pieceCount));

                i = pieceCount-1;

            }
            return pieces;
        }

        public static void main(String[] args) {
            Scanner scanner = new Scanner(System.in);
            System.out.print("Enter test case file path: ");
            String filePath = scanner.nextLine();
            int n = -1, m = -1, p = -1;
            String boardType = "";
            List<String> pieces = new ArrayList<>();

            try{
                BufferedReader reader = new BufferedReader(new
    FileReader(filePath));
                String line = null;

                // parse n m p
                line = reader.readLine();
                String[] nm = line.split(" ");
                n = Integer.parseInt(nm[0]);
                m = Integer.parseInt(nm[1]);
                p = Integer.parseInt(nm[2]);


                // parse board
                line = reader.readLine();
                boardType = line;


                // parse pieces
                while((line = reader.readLine()) != null){
                    pieces.add(line);
                }
                reader.close();
            }
            catch(IOException e){
                System.out.println("Error reading file");
            }finally{
```

```
                scanner.close();
            }

            Piece[] finalPieces = proccesPieceFromStringList(pieces,p);

            System.out.println("n = " + n);
            System.out.println("m = " + m);
            System.out.println("p = " + p);

            System.out.println("Board type: " + boardType);

            System.out.println("Pieces: ");
            for(int i = 1; i <= p; i++){
                System.out.println("Piece ID = " + (char)('A' +
    finalPieces[i].id));
                System.out.println("n = " + finalPieces[i].n);
                System.out.println("m = " + finalPieces[i].m);
                System.out.println("Shape: ");
                for(int j = 0 ; j < finalPieces[i].n + 2; j++){
                    for(int k = 0 ; k < finalPieces[i].m + 2; k++){
                        System.out.print(finalPieces[i].shape[j][k] ? "#"
    : ".");
                    }
                    System.out.println();
                }
            }

            String filename;
            System.out.print("Enter output file name: ");
            filename = scanner.nextLine();

            Board board = new Board(n, m, finalPieces, p);
            System.out.println("Board: ");
            board.printBoard();
            System.out.println("Start placing pieces: ");
            board.solve();
            System.out.println("Save to txt? (y/n)");
            String save = scanner.nextLine();
            if(save.equals("y")){
                board.saveSolution(filename);
            }
        }
    }
```

3.4 Piece.java

```
    package org.tucil1;

    // import java.util.List;

    public class Piece {
        int id;
        int n,m;
        boolean[][] shape;
        static int counter = 0;

        public Piece(int n, int m, boolean[][] shape){
            this.n = n;
            this.m = m;
            this.shape = new boolean[n+2][m+2];
            this.id = counter++;

            for(int i = 1; i <= n; i++){
                for(int j = 1; j <= m; j++){
                    this.shape[i][j] = shape[i-1][j-1];
                }
            }
        }
```

```
            public Piece(int n, int m, boolean[][] shape, int id){
                this.n = n;
                this.m = m;
                this.shape = new boolean[n+2][m+2];
                this.id = id;

                for(int i = 1; i <= n; i++){
                    for(int j = 1; j <= m; j++){
                        this.shape[i][j] = shape[i-1][j-1];
                    }
                }
            }

            public void printPiece(){
                for(int i = 1; i <= n; i++){
                    for(int j = 1; j <= m; j++){
                        if(shape[i][j]){
                            System.out.print((char)('A' + id));
                        }else{
                            System.out.print(".");
                        }
                    }
                    System.out.println();
                }
            }

            public Piece rotate() {
                boolean[][] newShape = new boolean[this.m][this.n];
                for (int i = 1; i <= this.m; i++) {
                  for (int j = 1; j <= this.n; j++) {
                    newShape[i-1][j-1] = this.shape[j][this.m-i+1];
                  }
                }
                return new Piece(this.m, this.n, newShape, this.id);
              }

            public Piece flip() {
                boolean[][] newShape = new boolean[this.n][this.m];
                for (int i = 1; i <= this.n; i++) {
                  for (int j = 1; j <= this.m; j++) {
                    newShape[i-1][j-1] = this.shape[i][this.m-j+1];
                  }
                }
                return new Piece(this.n, this.m, newShape, this.id);
              }

            public boolean isValidPosition(int x, int y){
                return x >= 1 && x <= this.n && y >= 1 && y <= this.m;
            }

        }
```

4. How to Use

Berikut merupakan Langkah-langkah untuk menjalankan program

Untuk menggunakan CLI

1. Buka directory terminal workspace di bin/main.
2. Karena .java sudah dicompile, bisa langsung run dengan command `java org.tucil1. Main.`
3. Berikut tampilan program dijalankan dengan CLI

```
PS D:\Kuliah\Akademik\Semester 4\Strategi Algoritma\Tucil1\app\src\main\java> java org.tucil1.Main
Enter test case file path: test/test1.txt
Start placing pieces:
Time Elapsed: 31 ms
Iteration Count: 1675
Solution Found!
ABBCDD
AABCCD
EFFGHH
EEFGGH
IJJKLL
IIJKKL
MNNOPP
MMNOOP
QRRSTT
QQRSST
UUVVWW
XUYVZW
XXYYZZ
Save to txt? (y/n)
y
Solution saved to: solve1 2025-02-24 10-03-33.txt
```

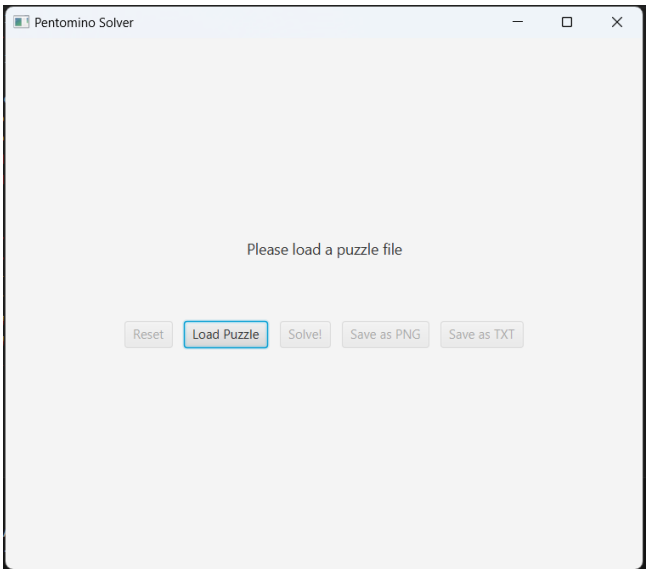
app > src > main > java > solve1_2025-02-24_10-03-33.txt

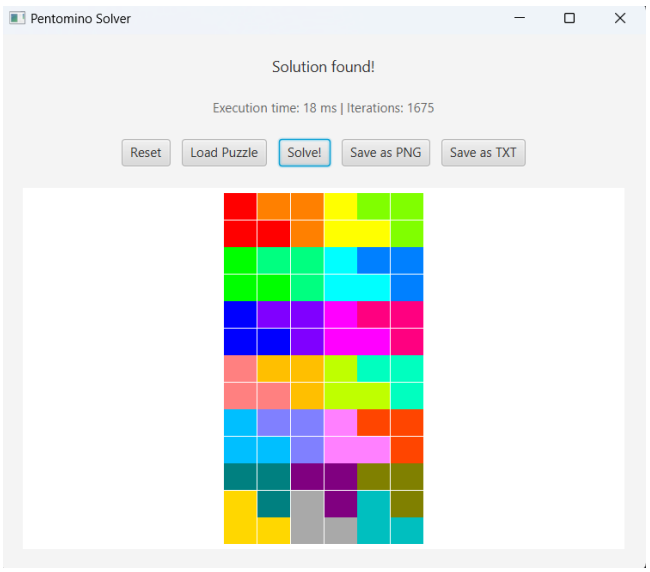
```
1   Puzzle Solution
2   Grid size: 13x6
3   Number of pieces: 26
4   Iterations: 1675
5
6   Solution:
7   ABBCDD
8   AABCCD
9   EFFGHH
10  EEFGGH
11  IJJKLL
12  IIJKKL
13  MNNOPP
14  MMNOOP
15  QRRSTT
16  QQRSST
17  UUVVWW
18  XUYVZW
19  XXYYZZ
```

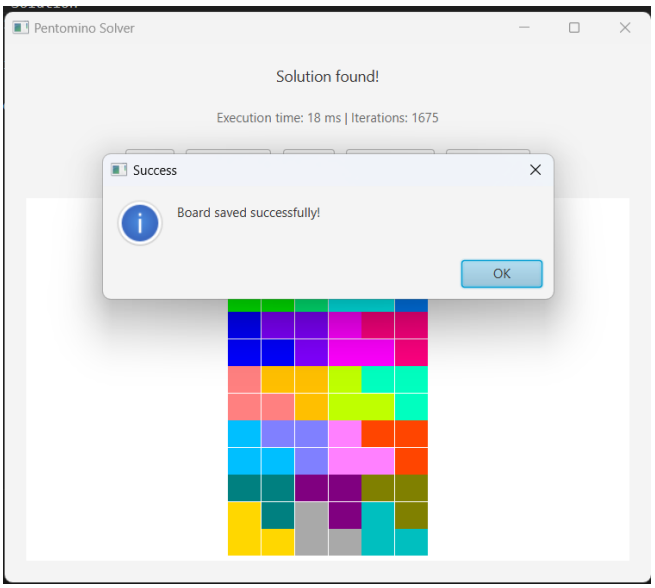
Untuk menggunakan GUI dengan Gradle

1. Pastikan sudah melakukan setup Gradle (https://gradle.org/install/)

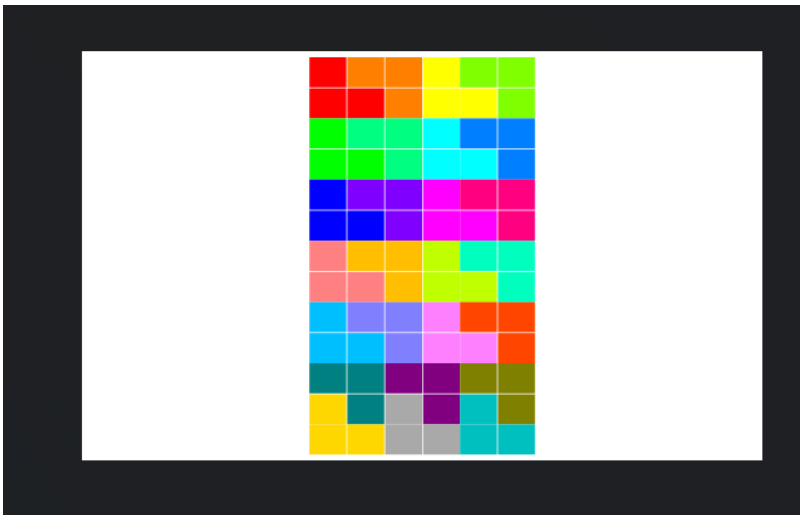
2. Pada folder src jalankan GUI dengan command `./gradlew run`
3. Berikut tampilan program dijalankan dengan GUI







4. Berikut Output Gambar PNG



5. Test Case

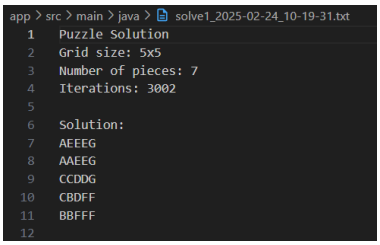
5.1 test1

Input:

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

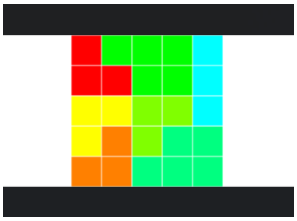
Output:

Txt:



```
app > src > main > java > solve1_2025-02-24_10-19-31.txt
1   Puzzle Solution
2   Grid size: 5x5
3   Number of pieces: 7
4   Iterations: 3002
5
6   Solution:
7   AEEEG
8   AAEEG
9   CCDDG
10  CBDFF
11  BBFFF
12
```

PNG:



5.2 test2

Input:

```
13 6 26
DEFAULT
A
AA
B
BB
C
CC
D
DD
E
EE
F
FF
G
GG
H
HH
I
II
J
JJ
K
KK
L
LL
M
MM
N
NN
O
OO
P
PP
Q
QQ
R
RR
S
SS
T
TT
U
UU
V
VV
W
WW
X
XX
Y
YY
Z
ZZ
```

Output:

Txt:

src > app > src > main > java > solve2_2025-02-24_10-33-01.txt

```
Puzzle Solution
Grid size: 13x6
Number of pieces: 26
Iterations: 1675

Solution:
ABBCDD
AABCCD
EFFGHH
EEFGGH
IJJKLL
IIJKKL
MNNOPP
MMNOOP
QRRSTT
QQRSST
UUVVWW
XUYVZW
XXYYZZ
```

PNG:

5.3 test3

Input:

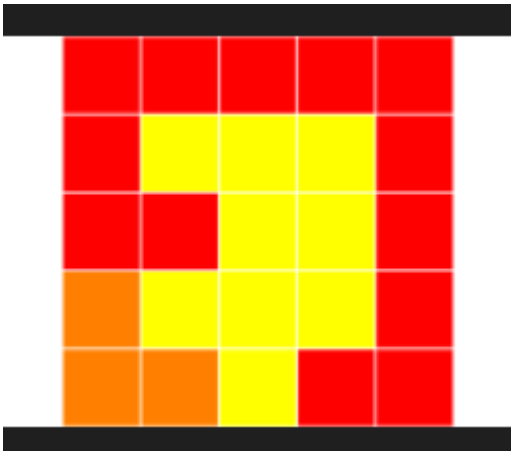
```
5 5 3
DEFAULT
AAAAA
A   A
    A
  A A
  AAA
B
BB
 CCC
CCCC
 C C
```

Output:

Txt:

```
1   Puzzle Solution
2   Grid size: 5x5
3   Number of pieces: 3
4   Iterations: 35
5
6   Solution:
7   AAAAA
8   ACCCA
9   AACCA
10  BCCCA
11  BBCAA
```

PNG:



5.4 test4

Input:

```
5 9 11
DEFAULT
AAA
 A
 A
 B
BB
B
B
CC
 CC
DD
 DD
  D
EE
E
E
FF
F
G
G
G
G
HH
H
```

```
I
II
II
JJ
KKKKK
```

Output:

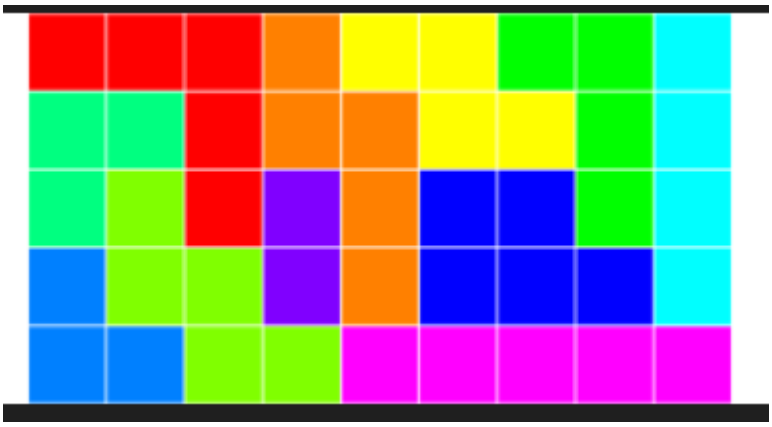
Txt:

test > solve4_2025-02-24_10-51-36.txt

```
Puzzle Solution
Grid size: 5x9
Number of pieces: 11
Iterations: 1213

Solution:
AAABCCEEG
FFABBCCEG
FDAJBIIEG
HDDJBIIIG
HHDDKKKKK
```

PNG:



5.5 test5

Input:

```
10 12 8
DEFAULT
 AAAAAAAAA
 AAAAAAAAA
    AAAA
    AAAAAA
    AAAAAA
 AAAAAAAAAAA
 AAAAAAAAAAA
 AAAAAAAAA
 AAA  AAA
 AAA  AAA
HHH
H
  BBB
 BBBBB
```

```
BBBBBB
RR
KKKKK
VVV
VVV
III
III
NNN
NNN
```

Output:

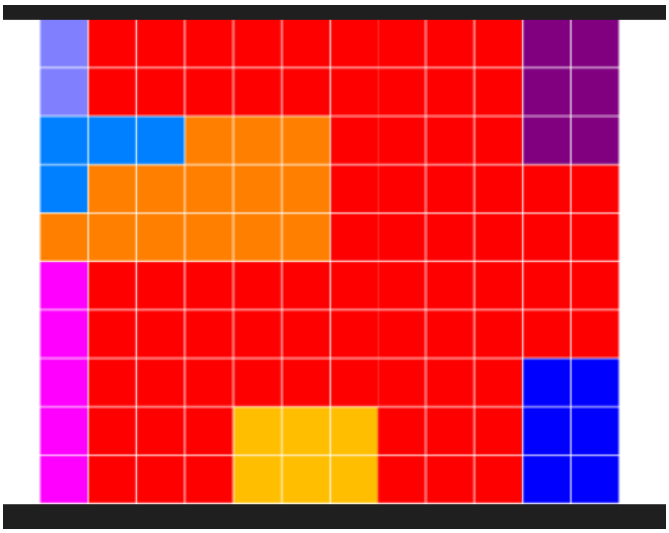
Txt:

test > solve5_2025-02-24_10-57-56.txt

```
Puzzle Solution
Grid size: 10x12
Number of pieces: 8
Iterations: 34106

Solution:
RAAAAAAAAAVV
RAAAAAAAAAVV
HHHBBBAAAAVV
HBBBBBAAAAAA
BBBBBBAAAAAA
KAAAAAAAAAAA
KAAAAAAAAAAA
KAAAAAAAAAII
KAAANNNAAAII
KAAANNNAAAII
```

PNG:



5.6 test6

Input:

```
4 4 3
DEFAULT
AAAA
A  A
A  A
AAAA
FF
F
B
```

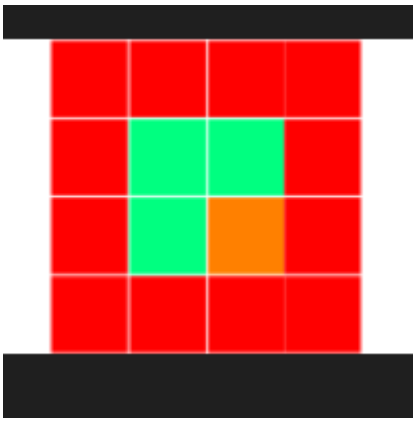
Output:

Txt:

```
Puzzle Solution
Grid size: 4x4
Number of pieces: 3
Iterations: 4

Solution:
AAAA
AFFA
AFBA
AAAA
```

PNG:



5.7 test7

Input

```
5 5 6
DEFAULT
AAA
A
A
 B
BB
B
B
CC
 CC
```

```
DD
 DD
  D
EE
E
E
FF
```

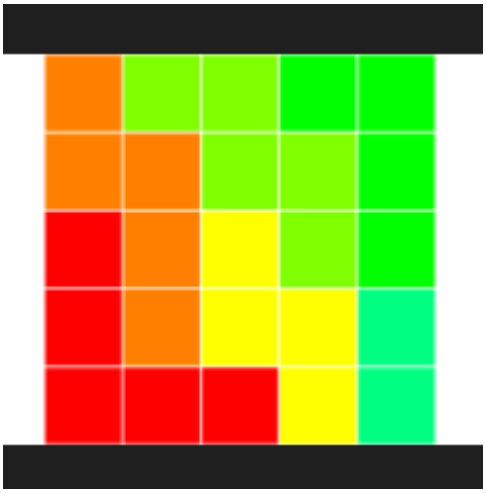
Output:

Txt:

test > solve7_2025-02-24_11-17-06.txt

```
1   Puzzle Solution
2   Grid size: 5x5
3   Number of pieces: 6
4   Iterations: 3992
5
6   Solution:
7   BDDEE
8   BBDDE
9   ABCDE
10  ABCCF
11  AAACF
12
```

PNG:



LAMPIRAN

No	Poin	Ya	Tidak
1	Program berhasil dikompilasi tanpa kesalahan	V	
2	Program berhasil dijalankan	V	
3	Solusi yang diberikan program benar dan mematuhi aturan permainan	V	
4	Program dapat membaca masukan berkas .txt serta menyimpan solusi dalam berkas .txt	V	
5	Program memiliki Graphical User Interface (GUI)	V	
6	Program dapat menyimpan solusi dalam bentuk file gambar	V	
7	Program dapat menyelesaikan kasus konfigurasi custom		V
8	Program dapat menyelesaikan kasus konfigurasi Piramida (3D)		V
9	Program dibuat oleh saya sendiri	V	
10			