LAPORAN TUGAS KECIL I

IF2211 STRATEGI ALGORITMA

Penyelesaian **IQ Puzzler Pro** dengan Algoritma Brute Force

Disusun oleh:

Steven Owen Liauw 13523103

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2025

Daftar Isi

BAGIAN I
ALGORITMA BRUTE FORCE

Algoritma *Brute Force* adalah algoritma *straight forward* yang memiliki metode penyelesaian masalah dengan mencoba semua kemungkinan secara berurutan sampai menemukan solusi yang benar. Pada permainan IQ Puzzle Pro algoritma *Brute Force* memiliki pendekatan :

- Rotasikan piece untuk setiap kemungkinannya, yaitu Rotasi 90, 180, dan 270 derajat beserta dengan pencerminannya untuk masing masing kemungkinan, sehingga terdapat 8 kemungkinan untuk masing masing piece.
- Coba tempatkan piece ke koordinat yang ada di Map.
- Apabila berhasil, maka ulangi langkah pertama untuk piece berikutnya.
- Jika gagal, maka hapus piece tersebut dan coba kemungkinan rotasi lainnya.
- Apabila masih gagal, maka coba kemungkinan koordinat yang lainnya.
- Jika sudah berhasil, ulangi dari langkah 1 untuk piece berikutnya.
- Jika piece sudah habis tapi papan belum penuh, maka dapat disimpulkan tidak memiliki jawaban.

Pseudo Code Algoritma *Brute Force* :

```
function solve(board : array of array of char, pieces : List of InputPiece, currentPiece : integer) → boolean
{ Menguji apakah semua potongan dapat ditempatkan pada board.
  Mengembalikan true jika solusi ditemukan (board terisi penuh), atau false jika tidak. }
Deklarasi:
   i, j       : integer
   rows, cols : integer
   piece      : InputPiece
   matrix     : List of array of char
   rotations  : array of List of array of char
   rotatedMatrix : List of array of char

Algoritma:
   if currentPiece = size(pieces) then
      return fit(board)    { Memeriksa apakah board sudah terisi penuh }
   else
      piece ← pieces[currentPiece]
      matrix ← piece.getMatrix()
      rows ← length(board)
      cols ← length(board[0])

      for i ← 0 to rows - 1 do
         for j ← 0 to cols - 1 do
            rotations ← { matrix,
                          mirrorHorizontal(matrix),
                          rotate90(matrix),
                          mirrorHorizontal(rotate90(matrix)),
                          rotate180(matrix),
                          mirrorHorizontal(rotate180(matrix)),
                          rotate270(matrix),
                          mirrorHorizontal(rotate270(matrix)) }

            for each rotatedMatrix in rotations do
```

```
                    attempts ← attempts + 1
                    if placePiece(board, rotatedMatrix, i, j) then
                       if solve(board, pieces, currentPiece + 1) then
                          return true
                       else
                          removePiece(board, rotatedMatrix, i, j)
                       endif
                    endif
                 endfor
              endfor
           endfor

           return false
  endif
```

Misalkan :

1. R adalah panjang papan.

2. C adalah lebar papan.

3. N adalah banyak piece.

Maka untuk setiap level rekursi akan dilakukan (8 x R x C) kali percobaan karena menggunakan Nested Loop untuk mencari kemungkinan koordinatnya. Sementara angka 8 adalah banyaknya kemungkinan rotasi dari setiap piece, Sehingga untuk keseluruhan dari rekursi akan dilakukan $(8xRxC)^N$ kali percobaan, sehingga kompleksitas Big(O) untuk program ini adalah $O((RxC)^N)$.

BAB II
SOURCE PROGRAM

Projek ini ditulis dalam Bahasa Java, menggunakan *library*:

1. java.awt
2. java.awt.image
3. javax.imageio
4. java.io
5. java.util
6. java.util.stream
7. javafx

Di program ini, setiap file memiliki fungsinya masing masing yaitu:

- Puzzle.java : program utama yang melakukan kalkulasi dan menghasilkan solusi.
- Colors.java : program yang berisi kelas warna.
- AnsiConverter : program parsing dari format warna Ansi ke Hex.
- Controller.java : backend dari Gui, menjalankan logika komponen yang ada di Gui.
- MainApp.java : inisiasi fxml.
- Ui.fxml : frontend aplikasi, mengatur tata letak komponen.

Berikut *source code*-nya:

Puzzle.Java

```
package solver;

import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.File;
import javax.imageio.ImageIO;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.stream.Collectors;
import java.util.Scanner;
import java.io.IOException;

class InputPiece {
    List<char[]> matrix;
    String color;

    public InputPiece() {
```

```
        this.matrix = new ArrayList<>();
    }

    public void setMatrix(char[] matrixs) {
        this.matrix.add(matrixs);
    }

    public void setColor(String color) {
        this.color = color;
    }

    public String getColor() {
        return this.color;
    }

    public List<char[]> getMatrix() {
        return this.matrix;
    }
}

public class Puzzle {

    public static boolean hasCommonElement(char[] arr1, char[] arr2) {
        for (char c : arr1) {
            for (char d : arr2) {
                if (c == d && c != ' ' && d != ' ') {
                    return true;
                }
            }
        }
        return false;
    }

    public static int lenPieceRow(String filepath) {
        int row = 0;
        try (BufferedReader reader = new BufferedReader(new FileReader(filepath))) {
            String[] dimensions = reader.readLine().split(" ");
            String S = reader.readLine();
            String line;
            while ((line = reader.readLine()) != null) {
                row++;
            }
        } catch (IOException e) {
            e.printStackTrace();
            return -1;
        }
        return row;
    }

    public static int longestPiece(String filepath) {
        int maxlen = 0;
        try (BufferedReader reader = new BufferedReader(new FileReader(filepath))) {
            String[] dimensions = reader.readLine().split(" ");
            String S = reader.readLine();
            String line;
            while ((line = reader.readLine()) != null) {
                if (line.length() > maxlen) {
                    maxlen = line.length();
```

```
            }
        }
    } catch (IOException e) {
        e.printStackTrace();
        return -1;
    }
    return maxlen;
}

public static List<char[]> padMatrix(List<char[]> matrix) {
    int maxLength = matrix.stream()
            .mapToInt(row -> row.length)
            .max()
            .orElse(0);

    List<char[]> paddedMatrix = new ArrayList<>();
    for (char[] row : matrix) {
        char[] paddedRow = new char[maxLength];
        System.arraycopy(row, 0, paddedRow, 0, row.length);
        for (int i = row.length; i < maxLength; i++) {
            paddedRow[i] = ' ';
        }
        paddedMatrix.add(paddedRow);
    }
    return paddedMatrix;
}

public static List<char[]> rotate90(List<char[]> matrix) {
    List<char[]> padded = padMatrix(matrix);
    int rows = padded.size();
    int cols = padded.get(0).length;

    List<char[]> rotated = new ArrayList<>();
    for (int j = 0; j < cols; j++) {
        char[] newRow = new char[rows];
        for (int i = 0; i < rows; i++) {
            newRow[i] = padded.get(rows - 1 - i)[j];
        }
        rotated.add(newRow);
    }
    return rotated;
}

public static List<char[]> rotate180(List<char[]> matrix) {
    List<char[]> padded = padMatrix(matrix);
    List<char[]> rotated = new ArrayList<>();
    for (int i = padded.size() - 1; i >= 0; i--) {
        char[] reversedRow = new char[padded.get(i).length];
        for (int j = 0; j < reversedRow.length; j++) {
            reversedRow[j] = padded.get(i)[reversedRow.length - 1 - j];
        }
        rotated.add(reversedRow);
    }
    return rotated;
}

public static List<char[]> rotate270(List<char[]> matrix) {
    List<char[]> padded = padMatrix(matrix);
```

```
        int rows = padded.size();
        int cols = padded.get(0).length;

        List<char[]> rotated = new ArrayList<>();
        for (int j = cols - 1; j >= 0; j--) {
            char[] newRow = new char[rows];
            for (int i = 0; i < rows; i++) {
                newRow[i] = padded.get(i)[j];
            }
            rotated.add(newRow);
        }
        return rotated;
    }

    public static List<char[]> mirrorHorizontal(List<char[]> matrix) {
        List<char[]> mirrored = new ArrayList<>();
        for (char[] row : matrix) {
            char[] newRow = new char[row.length];
            for (int i = 0; i < row.length; i++) {
                newRow[i] = row[row.length - 1 - i];
            }
            mirrored.add(newRow);
        }
        return mirrored;
    }

    // mengisi data
    public static Map<String, Object> readInput(String filepath) {
        Map<String, Object> result = new HashMap<>();
        try (BufferedReader reader = new BufferedReader(new FileReader(filepath))) {
            // baca baris pertama: N M P S
            String[] dimensions = reader.readLine().split(" ");

            int N = Integer.parseInt(dimensions[0]);
            int M = Integer.parseInt(dimensions[1]);
            int P = Integer.parseInt(dimensions[2]);
            String S = reader.readLine();

            int row = lenPieceRow(filepath);
            List<InputPiece> allPieces = new ArrayList<>(); // LIST of Object

            int currentColor = 0;
            char[] firstPieceData = reader.readLine().replaceAll("\\s+$", "").toCharArray();
            InputPiece currentPiece = new InputPiece();
            currentPiece.setMatrix(firstPieceData);
            currentPiece.setColor(Colors.COLORS[currentColor]);
            allPieces.add(currentPiece);

            char[] temp = firstPieceData;
            for (int i = 1; i < row; i++) {
                char[] piece = reader.readLine().replaceAll("\\s+$", "").toCharArray();
                if (hasCommonElement(temp, piece)) {
                    currentPiece.setMatrix(piece);
                } else {
                    currentColor++;
                    currentPiece = new InputPiece();
                    currentPiece.setMatrix(piece);
                    currentPiece.setColor(Colors.COLORS[currentColor]);
```

```
                allPieces.add(currentPiece);
                temp = piece;
            }
        }

        result.put("N", N);
        result.put("M", M);
        result.put("P", P);
        result.put("S", S);
        result.put("totalPiece", row);
        result.put("allPieces", allPieces);

        return result;
    } catch (

    IOException e) {
        e.printStackTrace();
        return null;
    }
}

public static boolean isValidPiece(char[] piece) {
    if (piece.length == 0) {
        return false;
    }

    int firstCharIdx = 0;
    for (int i = 0; i < piece.length; i++) {
        if (piece[i] != ' ') {
            firstCharIdx = i;
            break;
        }
    }

    char firstChar = piece[firstCharIdx];
    for (char c : piece) {
        if ((c != firstChar && c != ' ') || (c != ' ' && !String.valueOf(c).matches("[A-Z]"))) {
            System.out.println(c);
            return false;
        }
    }
    return true;
}

public static String returnThrowMessage(String filepath) {

    try (BufferedReader reader = new BufferedReader(new FileReader(filepath))) {
        // baca baris pertama: N M P S
        String[] dimensions = reader.readLine().split(" ");

        if (dimensions.length < 3 ||
                dimensions[0].trim().isEmpty() ||
                dimensions[1].trim().isEmpty() ||
                dimensions[2].trim().isEmpty()) {
            return "Error : Nilai N, M, dan P tidak boleh kosong.";
        }

        if (dimensions[0].trim().matches("\\d+") == false || dimensions[1].trim().matches("\\d+") == false
```

```
            || dimensions[2].trim().matches("\\d+") == false) {
        return "Error : Nilai N, M, dan P harus berupa angka POSITIF.";
    }

    String S = reader.readLine();

    if (S.trim().isEmpty() || !S.trim().equals("DEFAULT") || S == null) {
        return "Error : String S tidak boleh kosong atau selain DEFAULT.";
    }

    int row = lenPieceRow(filepath);
    List<InputPiece> allPieces = new ArrayList<>();

    char[] firstPieceData = reader.readLine().toCharArray();
    if (firstPieceData == null || firstPieceData.length == 0) {
        return "Error : Data piece tidak boleh kosong.";
    }
    if (!isValidPiece(firstPieceData)) {
        return "Error: Piece tidak valid. Terdapat karakter yang berbeda.";
    }
    InputPiece currentPiece = new InputPiece();
    currentPiece.setMatrix(firstPieceData);
    allPieces.add(currentPiece);

    char[] temp = firstPieceData;
    for (int i = 1; i < row; i++) {
        char[] piece = reader.readLine().toCharArray();
        if (piece == null || piece.length == 0) {
            return "Error : Data piece tidak lengkap.";
        }
        if (!isValidPiece(piece)) {
            return "Error: Piece tidak valid. Terdapat karakter yang berbeda.";
        }
        if (hasCommonElement(temp, piece)) {
            currentPiece.setMatrix(piece);
        } else {
            currentPiece = new InputPiece();
            currentPiece.setMatrix(piece);
            allPieces.add(currentPiece);
            temp = piece;
        }
    }

    for (int i = 0; i < allPieces.size(); i++) {
        for (int j = i + 1; j < allPieces.size(); j++) {

            if (Arrays.equals(allPieces.get(i).getMatrix().get(0), allPieces.get(j).getMatrix().get(0))) {
                return "Error: Terdapat piece yang sama.";
            }
        }
    }

    return null;
  } catch (IOException e) {
    return "Error : File tidak ditemukan.";
  }
}
```

```
public static void printOutput(Map<String, Object> result) {
    int N = (int) result.get("N");
    int M = (int) result.get("M");
    int P = (int) result.get("P");
    String S = (String) result.get("S");

    @SuppressWarnings("unchecked")
    List<InputPiece> allPieces = (List<InputPiece>) result.get("allPieces");
    List<InputPiece> paddedPieces = padAllPieces(allPieces);

    System.out.println("N: " + N);
    System.out.println("M: " + M);
    System.out.println("P: " + P);
    System.out.println("S: " + S);

    System.out.println("\nAll Pieces:");
    int count = 0;
    for (InputPiece piece : paddedPieces) {
        System.out.println("Piece " + (count) + ":");
        List<char[]> matrix = piece.getMatrix(); // list matriks dari instance piece

        String output = matrix.stream()
                .map(Arrays::toString)
                .collect(Collectors.joining(",\n  ", "[\n  ", "\n]"));

        System.out.println(output);
        System.out.println();
        count++;
    }
}

public static List<InputPiece> padAllPieces(List<InputPiece> allPieces) {
    List<InputPiece> paddedPieces = new ArrayList<>();
    for (InputPiece piece : allPieces) {
        List<char[]> paddedMatrix = padMatrix(piece.getMatrix());
        InputPiece paddedPiece = new InputPiece();
        paddedPiece.matrix = paddedMatrix;
        paddedPieces.add(paddedPiece);
    }
    return paddedPieces;
}

public static InputPiece checkObject(List<InputPiece> allPieces, char Element) {
    for (InputPiece pieces : allPieces) {
        int length = pieces.getMatrix().size();
        for (int i = 0; i < length; i++) {
            char[] matrix = pieces.getMatrix().get(i);
            for (char c : matrix) {
                if (c == Element) {
                    return pieces;
                }
            }
        }
    }
    return null;
}

// bagian Board
```

```
public static char[][] initializeBoard(int N, int M) {
    char[][] board = new char[N][M];

    for (int i = 0; i < N; i++) {
        Arrays.fill(board[i], '.');
    }

    return board;
}

public static boolean placePiece(char[][] board, List<char[]> piece, int x, int y) {
    int rows = piece.size();
    int cols = piece.get(0).length;

    // Cek potongan muat ga di board
    if (x + rows > board.length || y + cols > board[0].length) {
        return false;
    }

    // Cek ada konflik ?
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            if (piece.get(i)[j] != ' ' && board[x + i][y + j] != '.') {
                return false;
            }
        }
    }

    // Tempatkan potongan di board
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            if (piece.get(i)[j] != ' ') {
                board[x + i][y + j] = piece.get(i)[j];
            }
        }
    }

    return true;
}

public static boolean fit(char[][] board) {
    for (char[] row : board) {
        for (char cell : row) {
            if (cell == '.') {
                return false;
            }
        }
    }
    return true;
}

public static void removePiece(char[][] board, List<char[]> piece, int x, int y) {
    int r = piece.size();
    int c = piece.get(0).length;

    for (int i = 0; i < r; i++) {
        for (int j = 0; j < c; j++) {
            if (piece.get(i)[j] != ' ') {
```

```
                board[x + i][y + j] = '.';
            }
        }
    }
}

public static BufferedImage saveBoardAsImage(char[][] board, List<InputPiece> allPieces) {
    int rows = board.length;
    int cols = board[0].length;
    int cellSize = 50;
    int width = cols * cellSize;
    int height = rows * cellSize;

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

    g.setColor(Color.WHITE);
    g.fillRect(0, 0, width, height);
    g.setColor(Color.BLACK);
    g.setStroke(new BasicStroke(2));

    for (int i = 0; i <= rows; i++) {
        g.drawLine(0, i * cellSize, width, i * cellSize);
    }
    for (int j = 0; j <= cols; j++) {
        g.drawLine(j * cellSize, 0, j * cellSize, height);
    }

    g.setFont(new Font("Arial", Font.BOLD, 30));
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            char value = board[i][j];
            String text = String.valueOf(value);
            int textWidth = g.getFontMetrics().stringWidth(text);
            int textHeight = g.getFontMetrics().getAscent();
            int x = (j * cellSize) + (cellSize - textWidth) / 2;
            int y = (i * cellSize) + (cellSize + textHeight) / 2 - 5;

            InputPiece piece = checkObject(allPieces, value);

            g.setColor(AnsiConverter.colorFromANSI(piece.getColor()));
            g.drawString(text, x, y);
        }
    }

    g.dispose();
    return image;
}

public static File saveBoardAsTXT(char[][] board) throws IOException {
    File file = new File("output.txt");

    try (BufferedWriter writer = new BufferedWriter(new FileWriter(file))) {
        for (char[] row : board) {
            writer.write(row);
            writer.newLine();
        }
    }
```

```
    return file;
}
// bagian solusi

public static long attempts = 0;

public static boolean solve(char[][] board, List<InputPiece> pieces, int currentPiece) {

    if (currentPiece == pieces.size()) {
        return fit(board); // penuhh
    }

    InputPiece piece = pieces.get(currentPiece);
    List<char[]> matrix = piece.getMatrix();
    int rows = board.length;
    int cols = board[0].length;

    // Cobain kordinat
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            @SuppressWarnings("unchecked")
            List<char[]>[] rotations = new List[] { matrix, mirrorHorizontal(matrix), rotate90(matrix),
                    mirrorHorizontal(rotate90(matrix)),
                    rotate180(matrix),
                    mirrorHorizontal(rotate180(matrix)),
                    rotate270(matrix),
                    mirrorHorizontal(rotate270(matrix)) };

            for (List<char[]> rotatedMatrix : rotations) {
                attempts++;
                if (placePiece(board, rotatedMatrix, i, j)) {
                    if (solve(board, pieces, currentPiece + 1)) {
                        return true;
                    }
                    removePiece(board, rotatedMatrix, i, j);
                }
            }
        }
    }

    return false;
}

// bagian implementasi

public static Map<String, Object> MainGUI(String filepath) throws IOException, InterruptedException {

    Map<String, Object> result = readInput(filepath);
    if (result != null) {
        int N = (int) result.get("N");
        int M = (int) result.get("M");
        @SuppressWarnings("unchecked")
        List<InputPiece> allPieces = (List<InputPiece>) result.get("allPieces");
        List<InputPiece> paddedPieces = padAllPieces(allPieces);

        char[][] board = initializeBoard(N, M);
```

```
            long startTime = System.currentTimeMillis();

            boolean isSolved = solve(board, paddedPieces, 0);
            long endTime = System.currentTimeMillis();
            long duration = endTime - startTime;
            System.out.println(isSolved);

            if (!isSolved) {
                result.put("duration", duration);
                result.put("attempts", attempts);
                result.put("image", null);
                attempts = 0;
                return result;
            } else {

                BufferedImage image = saveBoardAsImage(board, allPieces); // Gambar + warnanya
                File file = saveBoardAsTXT(board);

                result.put("file", file);
                result.put("duration", duration);
                result.put("attempts", attempts);
                result.put("image", image);

                return result;
            }
        }
        return null;

    }

}
```

Colors.java

```
package solver;

public class Colors {
    public static final String[] COLORS = {
            "\033[38;2;255;0;0m", // Merah
            "\033[38;2;255;127;0m", // Oranye
            "\033[38;2;255;255;0m", // Kuning
            "\033[38;2;127;255;0m", // Chartreuse
            "\033[38;2;0;255;0m", // Hijau
            "\033[38;2;0;255;127m", // Hijau Muda
            "\033[38;2;0;255;255m", // Cyan
            "\033[38;2;0;127;255m", // Azure
            "\033[38;2;0;0;255m", // Biru
            "\033[38;2;127;0;255m", // Violet
            "\033[38;2;255;0;255m", // Magenta
            "\033[38;2;255;0;127m", // Rose
            "\033[38;2;128;128;128m", // Abu-abu
            "\033[38;2;192;192;192m", // Perak
            "\033[38;2;128;0;0m", // Maroon
            "\033[38;2;128;128;0m", // Zaitun
            "\033[38;2;0;128;0m", // Hijau Tua
            "\033[38;2;128;0;128m", // Ungu
```

```
        "\033[38;2;0;128;128m", // Teal
        "\033[38;2;0;0;128m", // Biru Tua
        "\033[38;2;255;165;0m", // Orange
        "\033[38;2;255;192;203m", // Pink
        "\033[38;2;173;216;230m", // Biru Muda
        "\033[38;2;240;230;140m", // Khaki
        "\033[38;2;75;0;130m", // Indigo
        "\033[38;2;60;179;113m" // Medium Sea Green
    };

    public static final String RESET = "\033[0m";
}
```

AnsiConverter.java

```
package solver;

import java.awt.Color;

public class AnsiConverter {
    public static Color colorFromANSI(String ansiCode) {
        // Pastikan ANSI code mengandung "38;2;" untuk true color
        String marker = "38;2;";
        int index = ansiCode.indexOf(marker);
        if (index == -1) {
            // Jika format gak sesuai
            return Color.BLACK;
        }

        // Ambil substring setelah "38;2;"
        String rgbPart = ansiCode.substring(index + marker.length());
        // Hapus karakter 'm' di akhir (jika ada)
        if (rgbPart.endsWith("m")) {
            rgbPart = rgbPart.substring(0, rgbPart.length() - 1);
        }

        String[] parts = rgbPart.split(";");
        if (parts.length != 3) {
            return Color.BLACK;
        }

        try {
            int r = Integer.parseInt(parts[0]);
            int g = Integer.parseInt(parts[1]);
            int b = Integer.parseInt(parts[2]);
            return new Color(r, g, b);
        } catch (NumberFormatException e) {
            e.printStackTrace();
            return Color.BLACK;
        }
    }
}
```

Controller.java

```
package GUI;

import javafx.application.Platform;
import javafx.embed.swing.SwingFXUtils;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.Hyperlink;
import javafx.scene.control.Label;
import javafx.scene.layout.AnchorPane;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import java.awt.image.BufferedImage;
import javafx.stage.FileChooser;
import solver.Colors;
import solver.Puzzle;
import java.io.File;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.StandardCopyOption;
import java.util.Map;

public class Controller {

    @FXML
    private Button FileButton;

    @FXML
    private Label fileLabel;

    @FXML
    private Button solveButton;

    @FXML
    private ImageView resultImage;

    @FXML
    private Label attempts;

    @FXML
    private Label time;

    @FXML
    private Label LabelHasil;

    @FXML
    private Label LabelDownload;

    @FXML
    private Hyperlink linkDownload;

    private File inputFile;

    @FXML
```

```
public void initialize() {
    if (FileButton != null) {
        FileButton.setOnAction(event -> handleUploadButton());
    }
}

@FXML
private void handleUploadButton() {
    FileChooser fileChooser = new FileChooser();
    fileChooser.setTitle("Pilih File");

    // Filter file agar hanya menampilkan TXT
    FileChooser.ExtensionFilter extFilter = new FileChooser.ExtensionFilter("Text Files (*.txt)", "*.txt");
    fileChooser.getExtensionFilters().add(extFilter);

    // Buka dialog file chooser
    File selectedFile = fileChooser.showOpenDialog(FileButton.getScene().getWindow());

    if (selectedFile != null) {
        inputFile = selectedFile;
        if (fileLabel != null) {
            fileLabel.setText("File selected: " + selectedFile.getName());
        }
        System.out.println("File yang dipilih: " + selectedFile.getAbsolutePath());
    } else {
        if (fileLabel != null) {
            fileLabel.setText("No file selected");
        }
        System.out.println("Tidak ada file yang dipilih.");
    }
}

@FXML
private void handleDownloadLink(File outputFile) {
    if (outputFile == null) {
        return;
    }
    FileChooser fileChooser = new FileChooser();
    fileChooser.setTitle("Save File");

    // Filter file agar hanya menampilkan TXT
    FileChooser.ExtensionFilter extFilter = new FileChooser.ExtensionFilter("Text Files (*.txt)", "*.txt");
    fileChooser.getExtensionFilters().add(extFilter);

    // Buka dialog file chooser
    File selectedFile = fileChooser.showSaveDialog(FileButton.getScene().getWindow());

    if (selectedFile != null) {
        try {
            File destination = new File(selectedFile.getAbsolutePath());
            Files.copy(outputFile.toPath(), destination.toPath(), StandardCopyOption.REPLACE_EXISTING);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

@FXML
```

```
    private void handleSolveButton() {
        try {
            if (inputFile == null) {
                LabelHasil.setText("Tidak ada file input. Silahkan pilih file terlebih dahulu.");
                return;
            }

            if (Puzzle.returnThrowMessage(inputFile.getAbsolutePath()) != null) {
                LabelHasil.setText(Puzzle.returnThrowMessage(inputFile.getAbsolutePath()));
                Platform.runLater(() -> resultImage.setImage(null));
                return;
            }
            Map<String, Object> result = Puzzle.MainGUI(inputFile.getAbsolutePath());

            if (result != null) {
                BufferedImage bufferedImage = (BufferedImage) result.get("image");
                if (bufferedImage != null) {

                    Image fxImage = SwingFXUtils.toFXImage(bufferedImage, null);
                    Platform.runLater(() -> resultImage.setImage(fxImage));
                    Platform.runLater(() -> linkDownload.setText((String) "Output.txt"));
                    attempts.setText(" : " + result.get("attempts"));
                    time.setText(" : " + result.get("duration") + " ms");

                    LabelHasil.setText("Hasil Solusi : ");
                    Platform.runLater(() -> LabelDownload.setText("File Hasil : "));
                    linkDownload.setOnMouseClicked(event -> {
                        if (event.getClickCount() == 2) { // double-click
                            handleDownloadLink((File) result.get("file"));
                        }
                    });

                } else {
                    Platform.runLater(() -> LabelDownload.setText(null));
                    Platform.runLater(() -> linkDownload.setText(null));
                    LabelHasil.setText("Tidak Ada Solusi Ditemukan");
                    attempts.setText(" : " + result.get("attempts"));
                    time.setText(" : " + result.get("duration") + " ms");
                    Platform.runLater(() -> resultImage.setImage(null));
                }
            }

        } catch (IOException | InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

MainApp.java

```
package GUI;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
```

```
import javafx.stage.Stage;

public class MainApp extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception {
        java.net.URL fxmlLocation = getClass().getResource("Ui.fxml");
        if (fxmlLocation == null) {
            System.err.println("FXML file not found!");
        } else {
            System.out.println("FXML file loaded: " + fxmlLocation);
        }
        FXMLLoader loader = new FXMLLoader(getClass().getResource("/GUI/Ui.fxml"));

        if (fxmlLocation == null) {
            System.err.println("FXML file not found!");
        } else {
            System.out.println("FXML file loaded: " + fxmlLocation);
        }

        Parent root = loader.load();

        primaryStage.setTitle("File Upload Example");
        primaryStage.setScene(new Scene(root, 900, 600)); // Sesuaikan ukuran window
        primaryStage.show();
    }

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

Ui.fxml

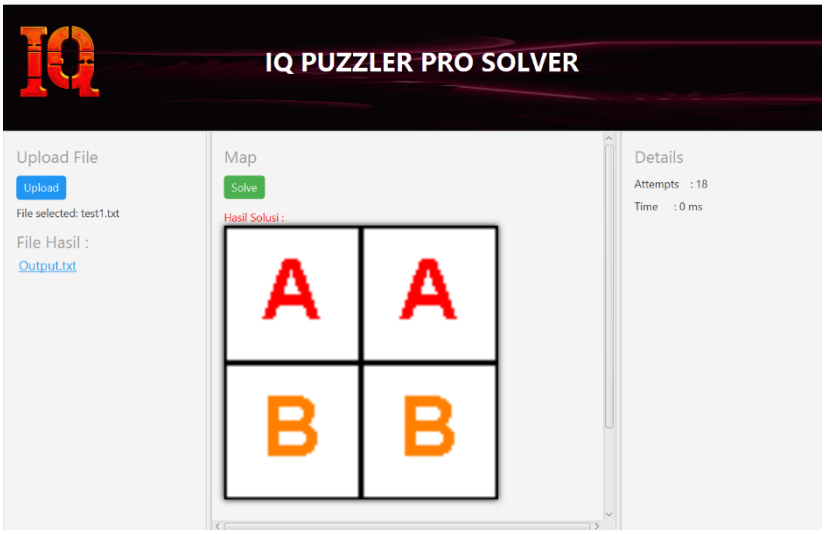
```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Hyperlink?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.ScrollPane?>
<?import javafx.scene.control.SplitPane?>
<?import javafx.scene.effect.DropShadow?>
<?import javafx.scene.image.Image?>
<?import javafx.scene.image.ImageView?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.HBox?>
<?import javafx.scene.layout.Pane?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.paint.Color?>
<?import javafx.scene.text.Font?>
<?import javafx.scene.text.Text?>
<?import javafx.css.CssParser?>
```

```
<VBox prefHeight="600.0" prefWidth="900.0" style="-fx-background-image: url('@/lib/fluidbg.jpg'); -fx-background-repeat: no-repeat; -fx-background-size: cover;" xmlns="http://javafx.com/javafx/23.0.1" xmlns:fx="http://javafx.com/fxml/1" fx:controller="GUI.Controller">
  <children>
    <AnchorPane prefHeight="200.0" prefWidth="200.0" style="-fx-background-color: #333333; -fx-alignment: CENTER_LEFT;">
      <children>
        <ImageView blendMode="SCREEN" fitHeight="132.0" fitWidth="125.0" pickOnBounds="true" preserveRatio="true">
          <image>
            <Image url="@../../lib/LogoCrop.png" />
          </image>
        </ImageView>
        <Label layoutX="287.0" layoutY="44.0" style="-fx-text-fill: #ffffff; -fx-font-size: 29px; -fx-font-family: 'Segoe UI'; -fx-font-weight: bold;" text="IQ PUZZLER PRO SOLVER" textFill="#efeaea">
          <font>
            <Font size="29.0" />
          </font>
          <effect>
            <DropShadow color="rgba(0,0,0,0.5)" offsetX="3" offsetY="3" radius="5" />
          </effect>
        </Label>
        <ImageView blendMode="OVERLAY" fitHeight="138.0" fitWidth="2000.0" layoutX="1.0" layoutY="-6.0" pickOnBounds="true" AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="0.0">
          <image>
            <Image url="@../../lib/blacktrg.jpg" />
          </image>
        </ImageView>
      </children>
    </AnchorPane>
    <SplitPane dividerPositions="0.2505567928730512, 0.7505567928730512" focusTraversable="true" prefHeight="-1.0" prefWidth="-1.0" VBox.vgrow="ALWAYS">
      <items>
        <AnchorPane>
          <children>
            <Label alignment="CENTER" layoutX="14.0" layoutY="14.0" minWidth="60.0" prefWidth="-1.0" style="&#10;" text="Upload File" textAlignment="CENTER" wrapText="false">
              <font>
                <Font size="18.0" fx:id="x1" />
              </font>
              <textFill>
                <Color red="0.624" green="0.624" blue="0.624" fx:id="x2" />
              </textFill>
            </Label>
                <Button fx:id="FileButton" layoutX="14.0" layoutY="48.0" mnemonicParsing="false" onAction="#handleUploadButton" style="-fx-background-color: #2196F3; -fx-text-fill: white; -fx-background-radius: 4; -fx-cursor: hand;" text="Upload" />
                <Label fx:id="fileLabel" layoutX="14.0" layoutY="80.0" text="No file selected" />
                <Label fx:id="LabelDownload" alignment="CENTER" layoutX="14.0" layoutY="107.0" minWidth="60.0" prefWidth="-1.0" style="&#10;" textAlignment="CENTER" wrapText="false">
                  <font>
                    <Font size="18.0" fx:id="x11" />
                  </font>
                  <textFill>
                    <Color red="0.624" green="0.624" blue="0.624" fx:id="x21" />
                  </textFill>
                </Label>
                <Hyperlink fx:id="linkDownload" layoutX="12.0" layoutY="134.0" style="-fx-font-size: 14px; -fx-text-fill: #2196F3; -fx-underline: true; -fx-font-family: 'Segoe UI';" />
```

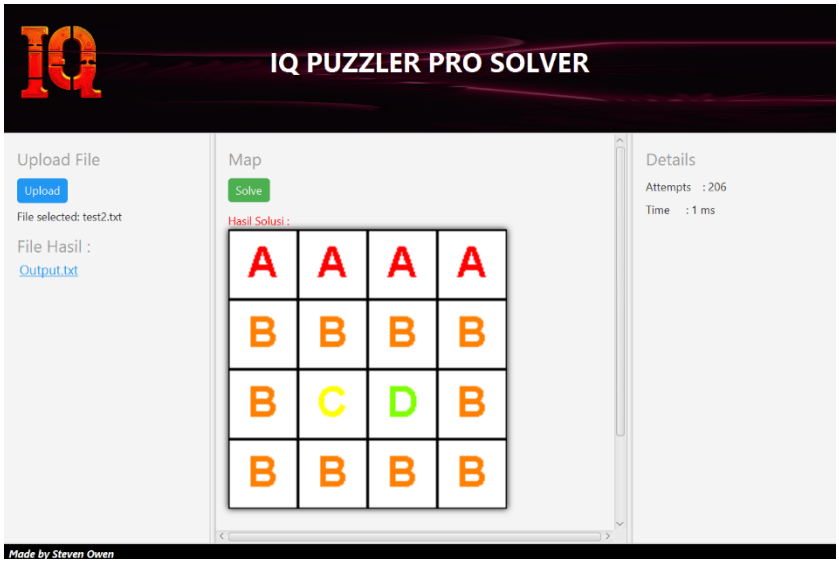
```
        </children>
      </AnchorPane>
      <ScrollPane prefHeight="-1.0" prefWidth="-1.0">
        <content>
          <AnchorPane id="Content" minHeight="-1.0" minWidth="-1.0" prefHeight="545.0" prefWidth="430.0">
            <children>
              <Label alignment="CENTER" font="$x1" layoutX="14.0" layoutY="14.0" style="&#10;" text="Map"
textAlignment="CENTER" textFill="$x2" wrapText="false" />
                    <Button fx:id="solveButton" layoutX="14.0" layoutY="48.0" mnemonicParsing="false"
onAction="#handleSolveButton" style="-fx-background-color: #4CAF50; -fx-text-fill: white; -fx-background-radius: 4; -fx-cursor:
hand;" text="Solve" />
                        <Label fx:id="LabelHasil" layoutX="14.0" layoutY="85.0" style="-fx-text-fill: red;" />
                        <ImageView fx:id="resultImage" fitHeight="300" fitWidth="300" layoutX="14.0" layoutY="103.0" style="-fx-
effect: dropshadow(gaussian, rgba(0,0,0,0.5), 10, 0.5, 0, 0);" />
            </children>
          </AnchorPane>
        </content>
      </ScrollPane>
      <AnchorPane>
        <children>
          <Label alignment="CENTER" font="$x1" layoutX="14.0" layoutY="14.0" style="&#10;" text="Details"
textAlignment="CENTER" textFill="$x2" wrapText="false" />
              <Label layoutX="14.0" layoutY="48.0" text="Attempts" />
              <Label layoutX="14.0" layoutY="73.0" text="Time" />
              <Label fx:id="attempts" layoutX="72.0" layoutY="48.0" />
              <Label fx:id="time" layoutX="54.0" layoutY="73.0" />
        </children>
      </AnchorPane>
    </items>
  </SplitPane>
  <HBox id="HBox" alignment="CENTER_LEFT" spacing="5.0" style="-fx-background-color:rgb(0, 0, 0); -fx-padding: 3.0;"
VBox.vgrow="NEVER">
    <children>
      <Label maxHeight="1.7976931348623157E308" maxWidth="-1.0" style="-fx-font-size: 11px; -fx-text-fill: rgb(255, 255, 255); -
fx-font-family: 'Segoe UI'; -fx-font-style: italic; -fx-font-weight: bold;" text=" Made by Steven Owen" HBox.hgrow="ALWAYS">
        <font>
          <Font size="11.0" fx:id="x3" />
        </font>
        <textFill>
          <Color red="0.625" green="0.625" blue="0.625" fx:id="x4" />
        </textFill>
      </Label>
      <Pane prefHeight="-1.0" prefWidth="-1.0" HBox.hgrow="ALWAYS" />
      <Label font="$x3" maxWidth="-1.0" style="-fx-font-size: 11px; -fx-text-fill: #757575; -fx-font-family: 'Segoe UI';"
textFill="$x4" HBox.hgrow="NEVER" />
    </children>
    <padding>
      <Insets bottom="3.0" left="3.0" right="3.0" top="3.0" />
    </padding>
  </HBox>
 </children>
</VBox>
```

BAGIAN III
SCREENSHOT HASIL TEST

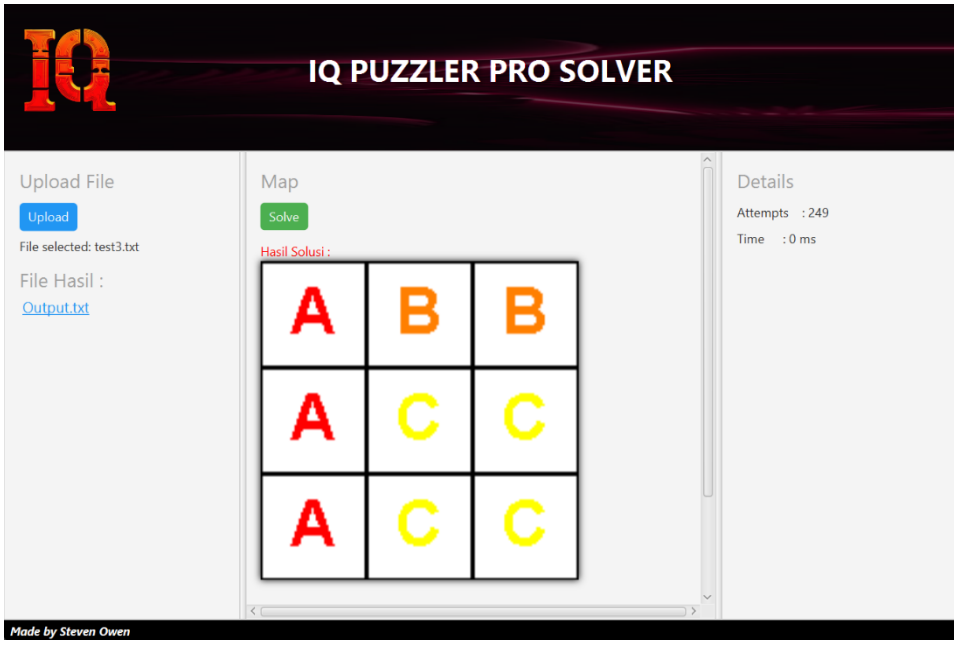
Hasil 1:



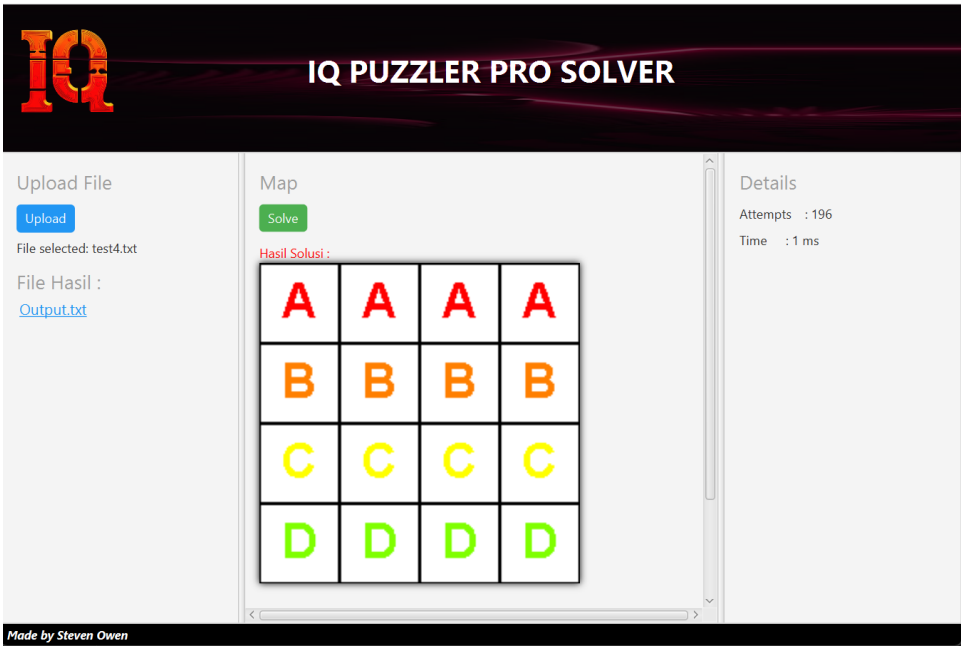
Hasil 2:



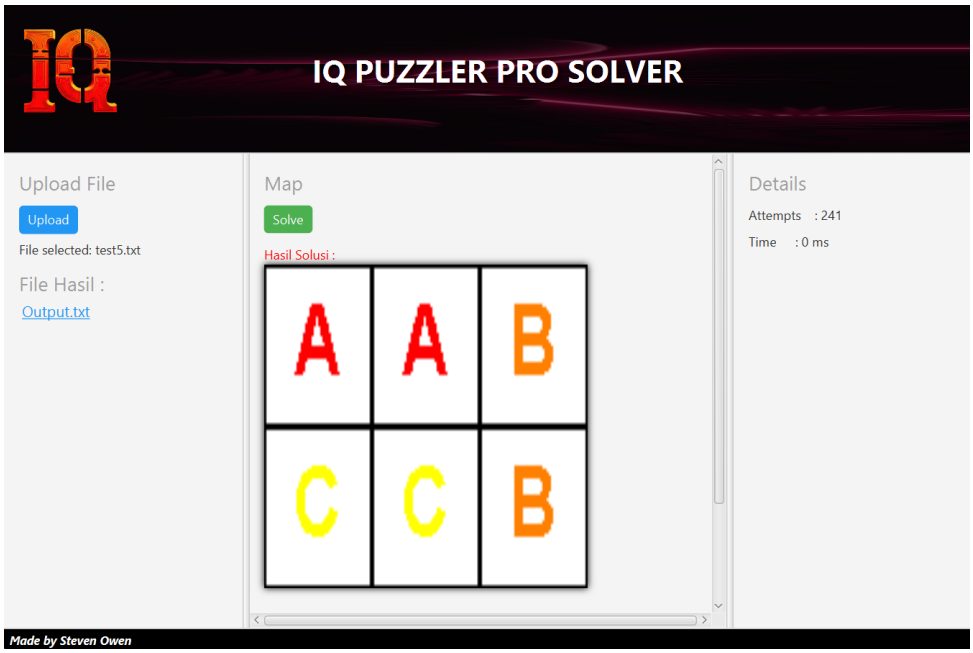
Hasil 3:


IQ PUZZLER PRO SOLVER
Upload File
Upload
File selected: test3.txt
File Hasil :
Output.txt
Map
Solve
Hasil Solusi :
A B B
A C C
A C C
Details
Attempts : 249
Time : 0 ms
Made by Steven Owen

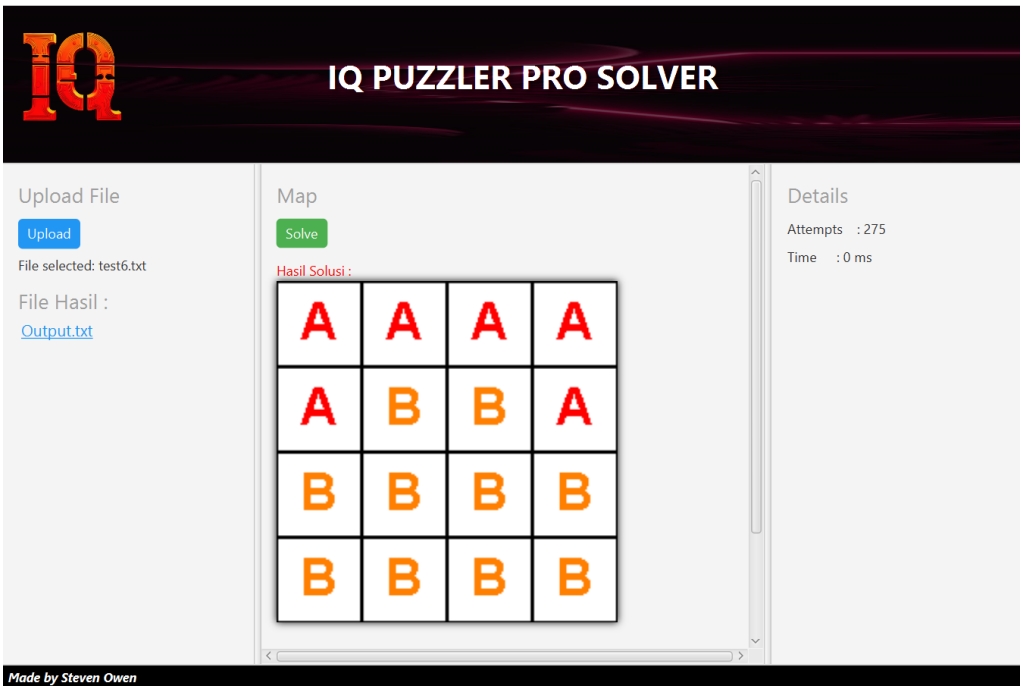
Hasil 4:


IQ PUZZLER PRO SOLVER
Upload File
Upload
File selected: test4.txt
File Hasil :
Output.txt
Map
Solve
Hasil Solusi :
A A A A
B B B B
C C C C
D D D D
Details
Attempts : 196
Time : 1 ms
Made by Steven Owen

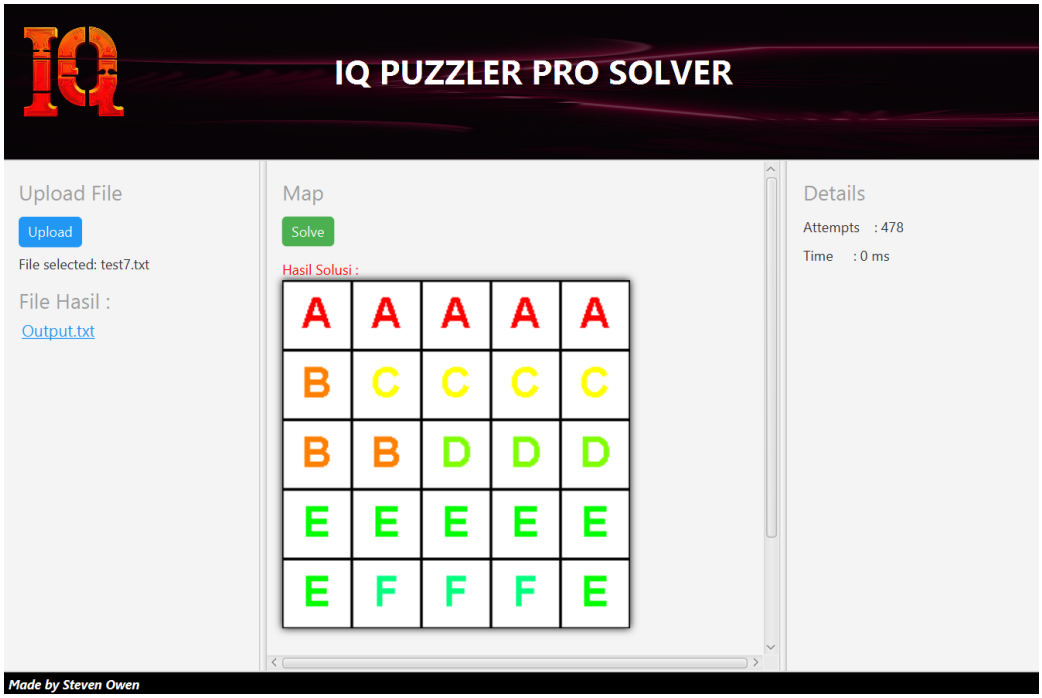
Hasil 5:



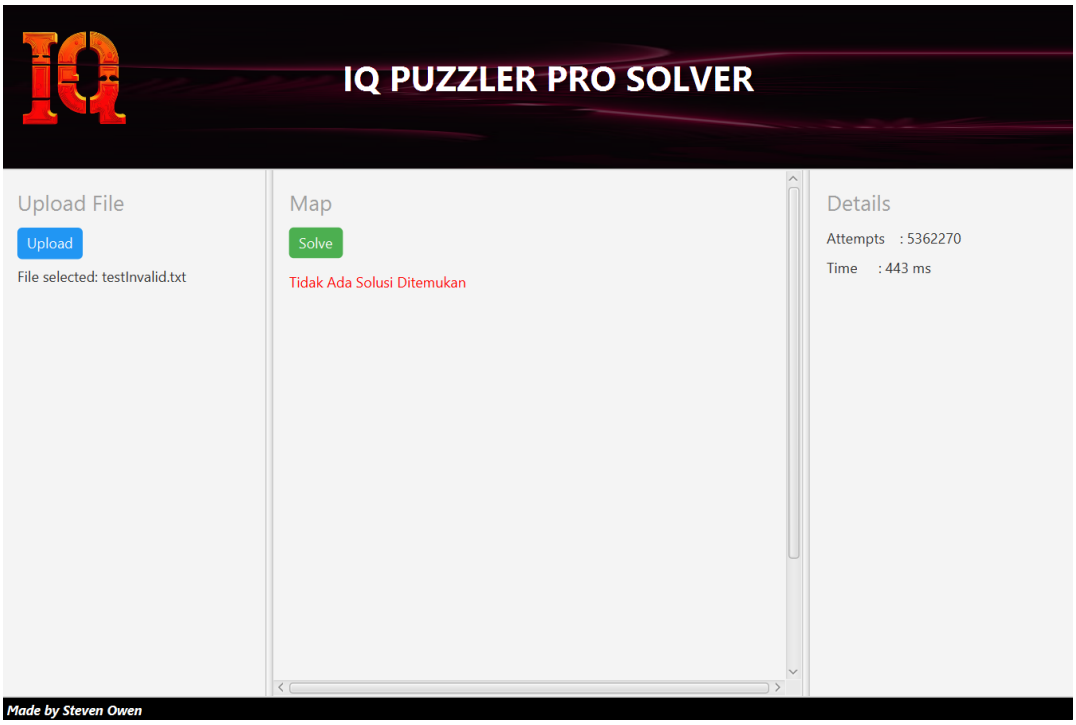
Hasil 6:



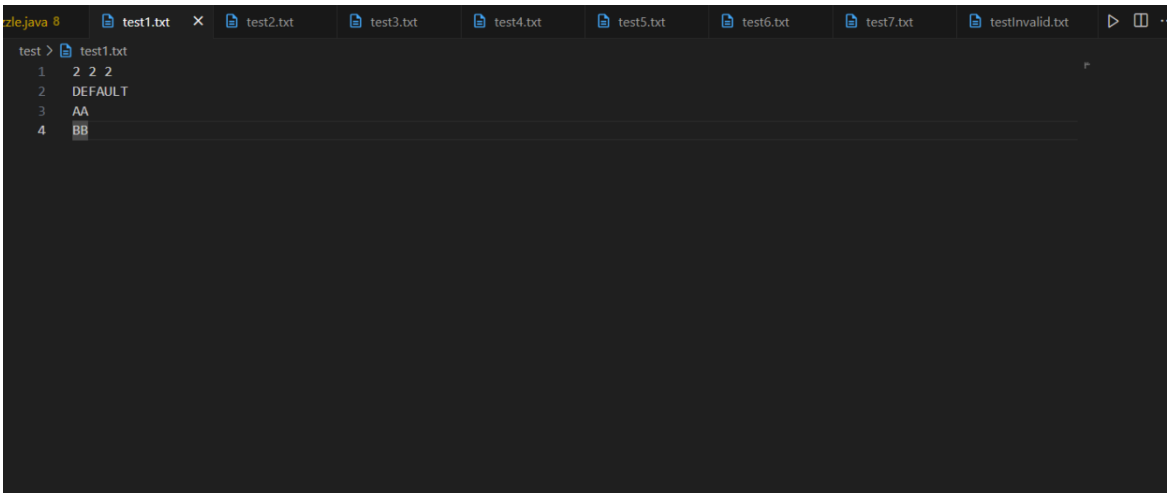
Hasil 7:



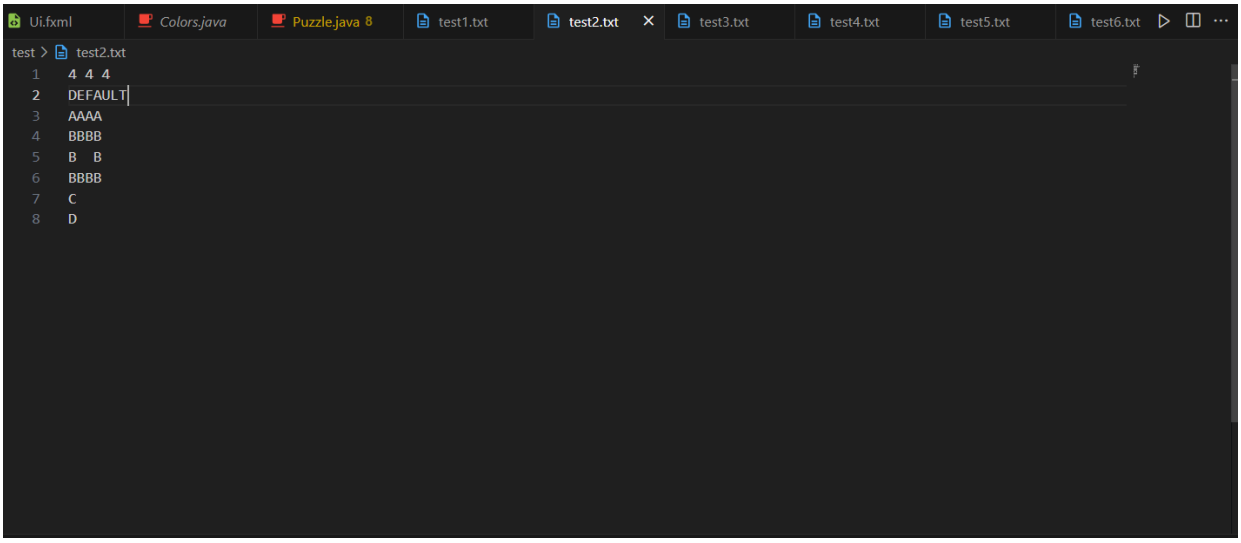
Hasil Invalid:



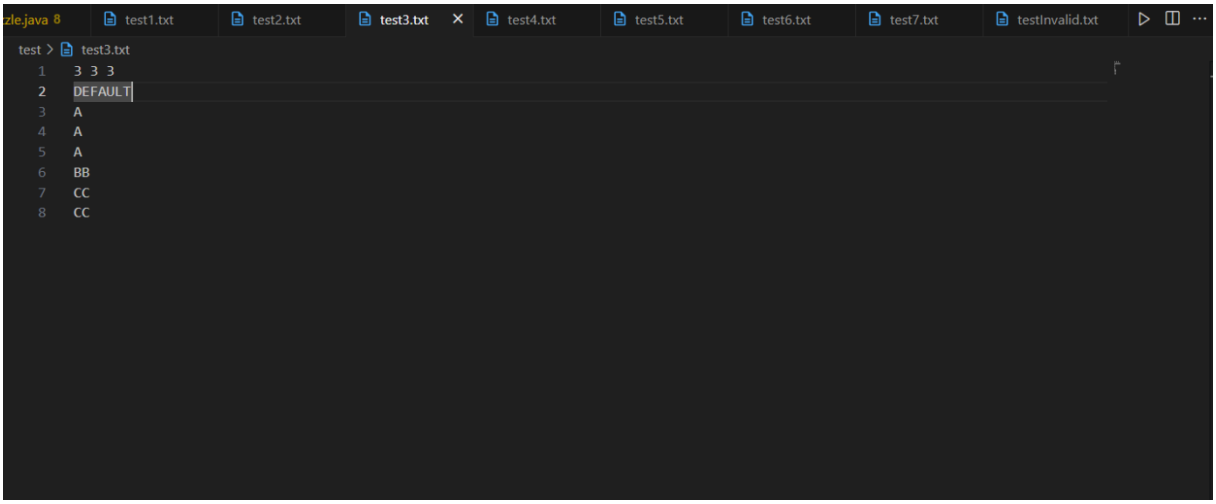
Input 1 :



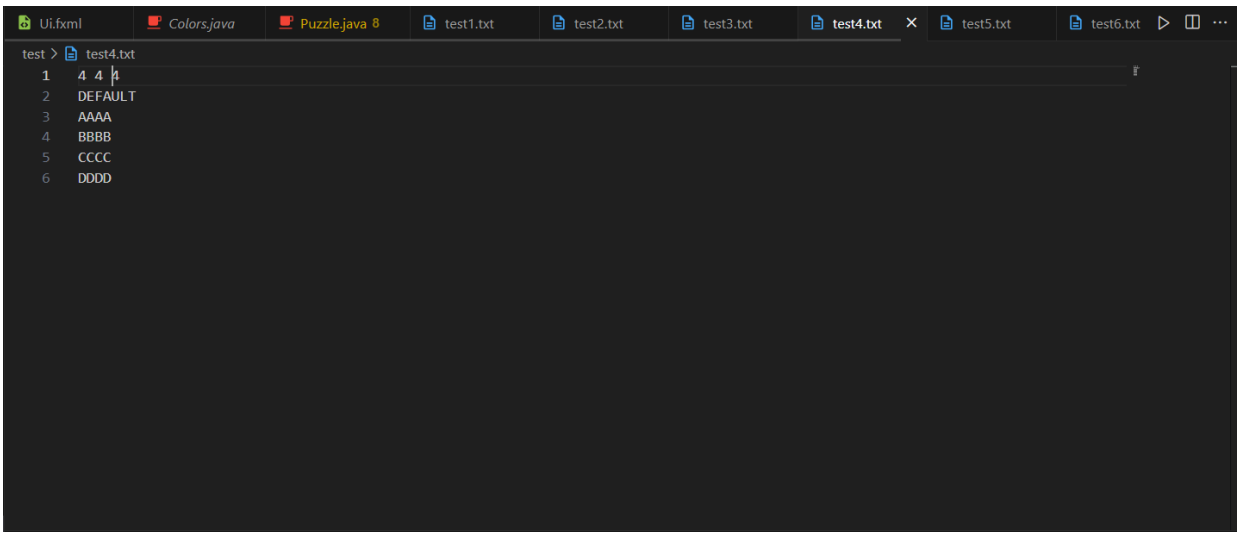
Input 2 :



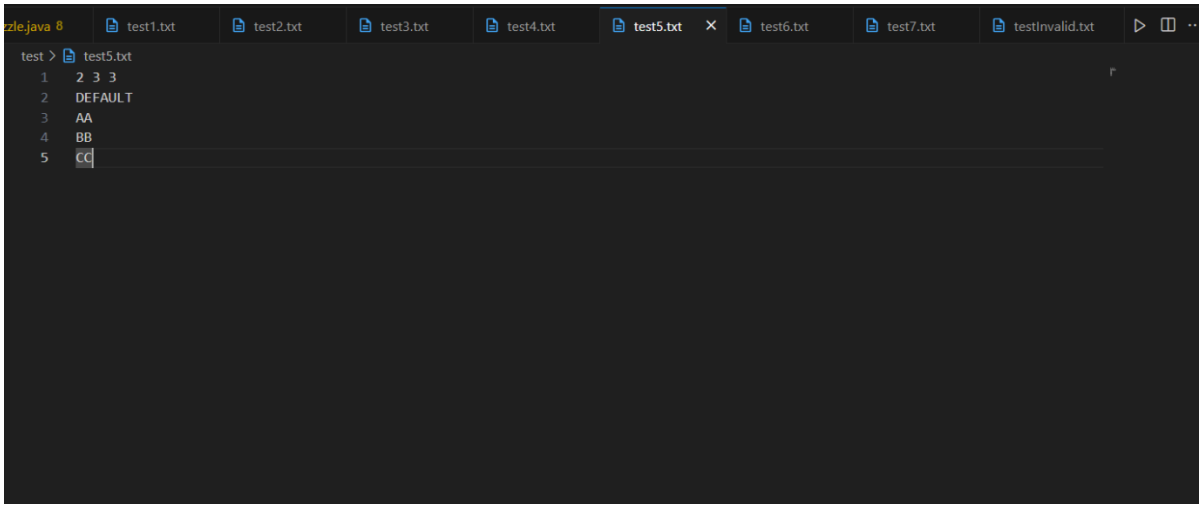
Input 3 :



Input 4 :



Input 5 :



Input 6 :

```
4 4 2
DEFAULT
AAAA
A  A
 BB
BBBB
BBBB
```

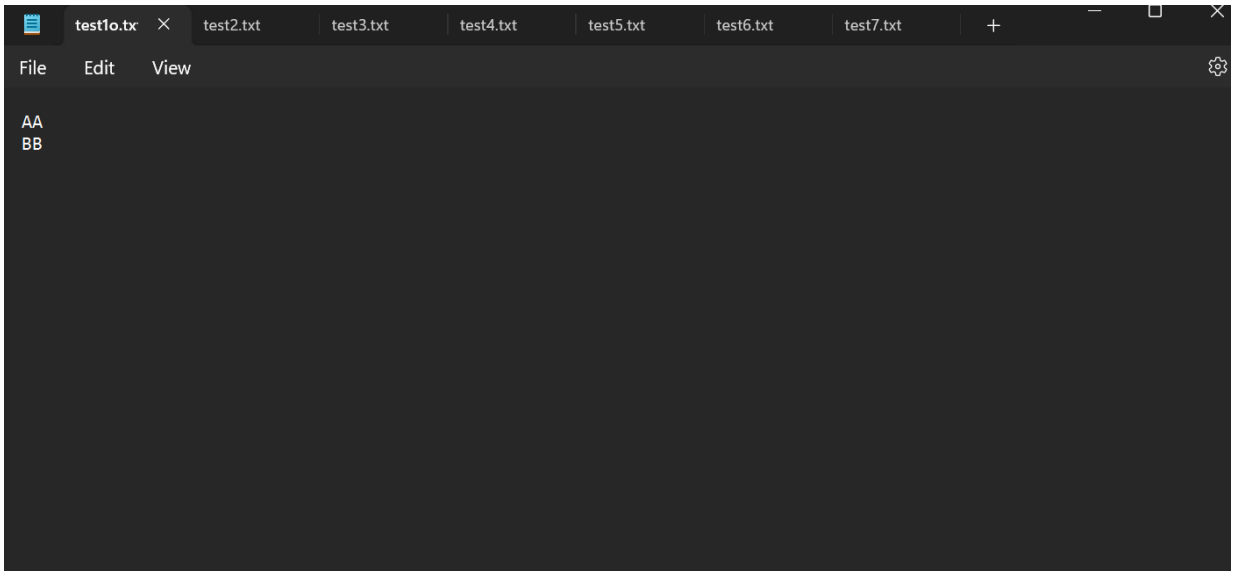
Input 7 :

```
5 5 5
DEFAULT
AAAAA
B
BB
CCCC
DDD
EEEEE
E    E
FFF
```

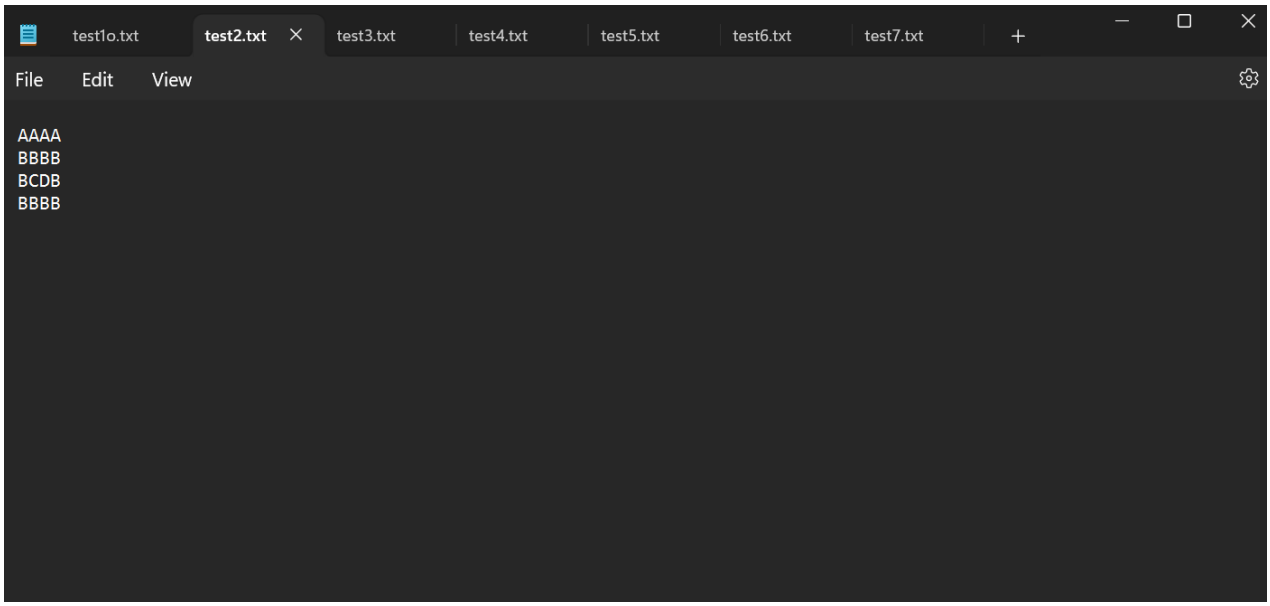
Input Invalid :

```
4 4 4
DEFAULT
AAAA
BB
D
DD
EEE
```

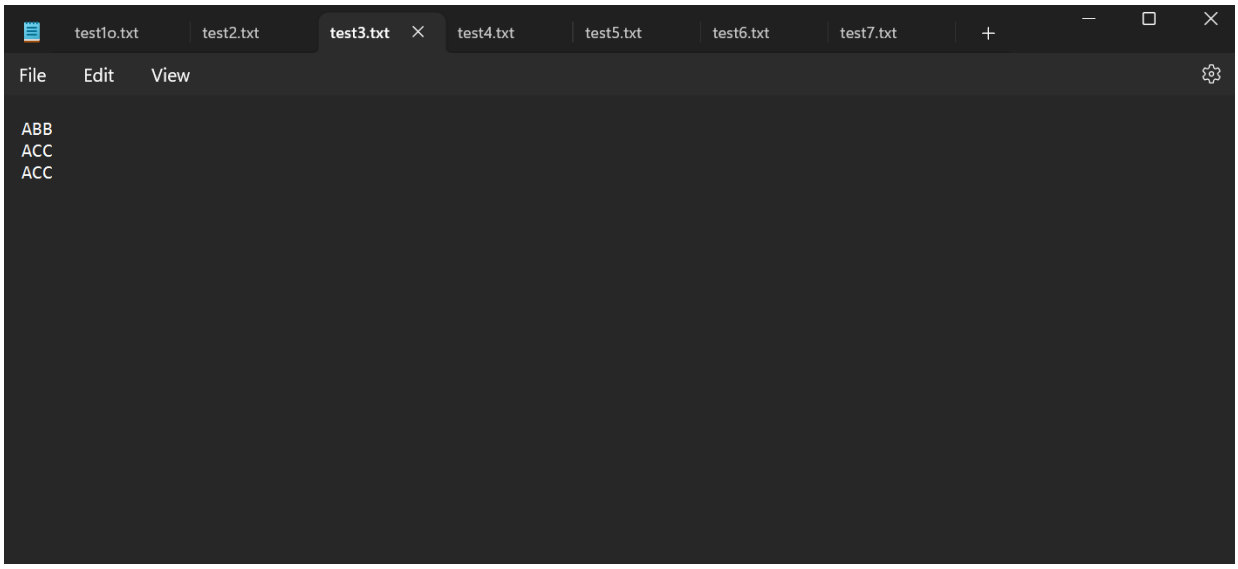
Output File 1 :



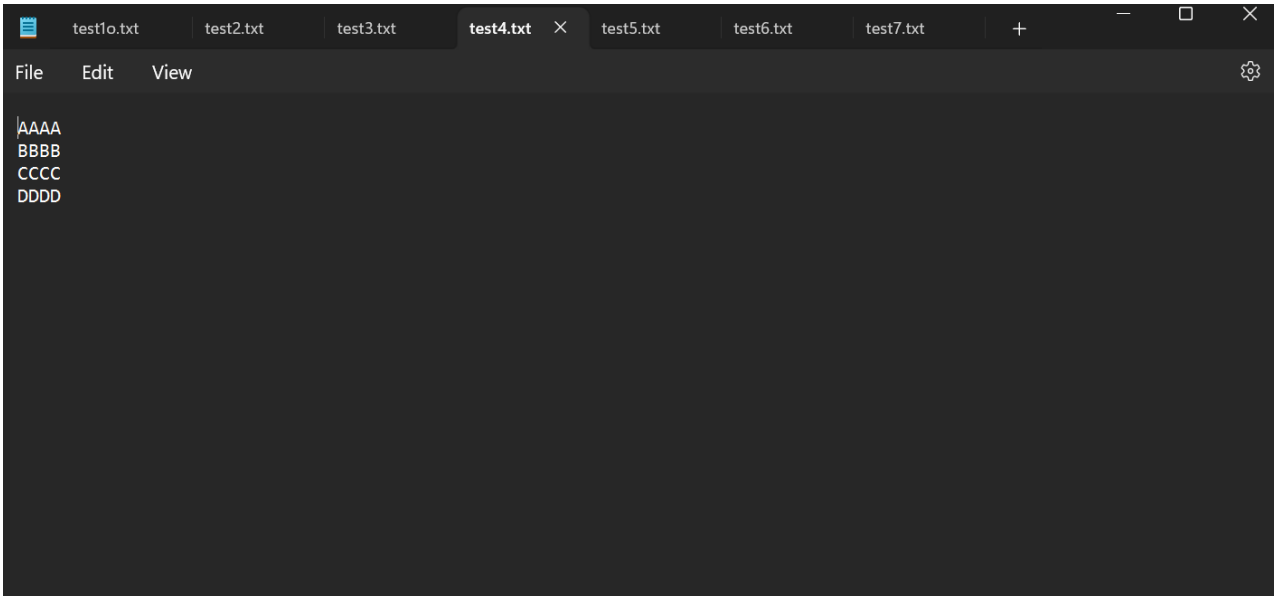
Output File 2 :



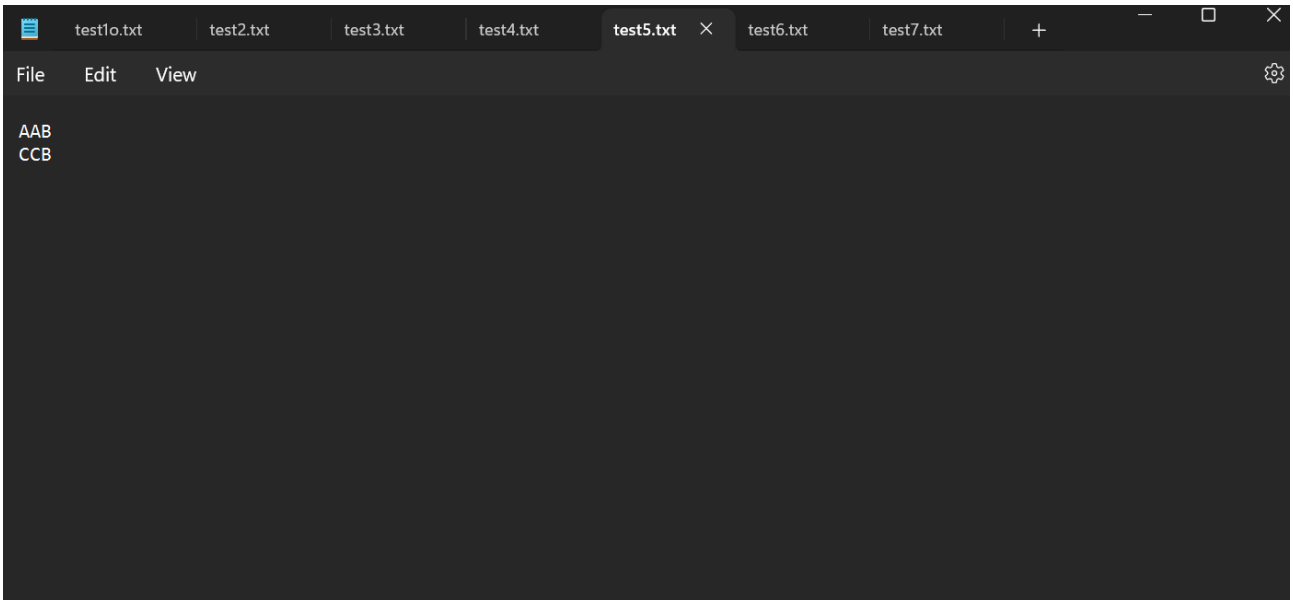
Output File 3 :



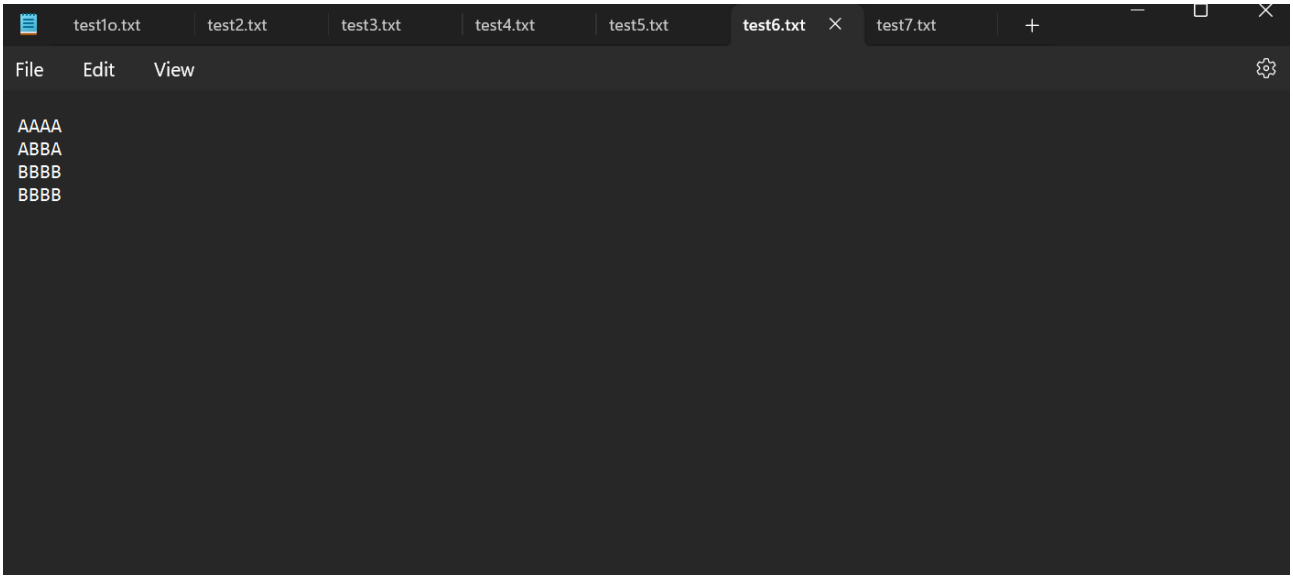
Output File 4 :



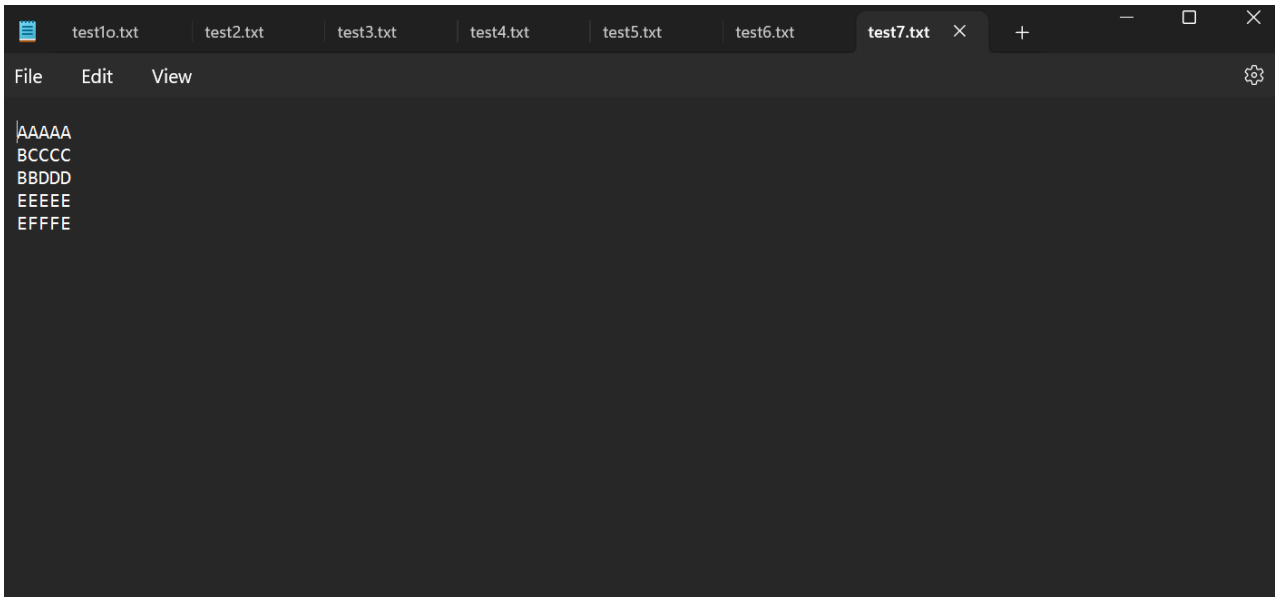
Output File 5 :



Output File 6 :



Output File 7 :



LINK REPOSITORY

https://github.com/stevennowen/Tucil_13523103.git

CHECKLIST

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 *Graphical User Interface* (GUI)	✓	
6	Program dapat menyimpan solusi dalam bentuk file gambar	✓	
7	Program dapat menyelesaikan kasus konfigurasi *custom*		✓
8	Program dapat menyelesaikan kasus konfigurasi Piramida (3D)		✓
9	Program dibuat oleh saya sendiri	✓	