

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



Oleh:
Muhammad Fithra Rizki
13523049

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

DAFTAR ISI

DAFTAR ISI.....	2
BAB I.....	3
DESKRIPSI PROGRAM.....	3
1.1 Algoritma Brute Force.....	3
1.2 IQ Puzzler Pro.....	3
1.3 Algoritma Penyelesaian IQ Puzzler Pro dengan Pendekatan Brute Force.....	4
BAB II.....	6
ALGORITMA PROGRAM.....	6
2.1 PuzzleInput.java.....	6
2.2 PuzzleSolver.java.....	6
2.3 Main.java.....	7
2.4 PuzzleGUI.java.....	8
BAB III.....	9
SOURCE CODE.....	9
3.1 Repository Github.....	9
3.2 Source Code Program.....	9
3.2.1 PuzzleInput.java.....	9
3.2.2 PuzzleSolver.java.....	10
3.2.3 Main.java.....	14
3.2.4 PuzzleGUI.java.....	14
BAB IV.....	16
IMPLEMENTASI PROGRAM.....	16
4.1 Test Case 1.....	16
4.2 Test Case 2.....	18
4.3 Test Case 3.....	19
4.4 Test Case 4.....	22
4.5 Test Case 5.....	24
4.6 Test Case 6.....	26
4.7 Test Case 7.....	28
BAB V.....	31
LAMPIRAN.....	31
DAFTAR PUSTAKA.....	32

BAB I

DESKRIPSI PROGRAM

1.1 Algoritma Brute Force

Algoritma brute force merupakan pendekatan penyelesaian masalah yang terbilang lurus dan lempeng. Algoritma brute force sendiri dapat memecahkan persoalan dengan cara sederhana, langsung, mudah dipahami, dan jelas. Proses ini dilakukan sesuai dengan prinsip “*Just Do It!*” atau “*Just Solve It!*”. Algoritma brute force memiliki tujuan untuk mencari solusi secara menyeluruh atau satu persatu, lalu perlu dicari solusi mana yang menggunakan tenaga yang paling efisien. Algoritma ini tentunya memerlukan sumber daya komputasi yang besar dan waktu penyelesaian yang lama.

Pendekatan melalui metode brute force tentunya memiliki kelebihan sekaligus kekurangan yang dirasakan. Kelebihan dari algoritma ini diantaranya adalah algoritma ini dapat menyelesaikan banyak masalah yang ada (*wide applicability*), algoritma ini memiliki cara yang sederhana dan mudah dimengerti, algoritma ini memiliki beberapa solusi untuk permasalahan penting, seperti perkalian matriks, pencarian, dan lainnya, serta algoritma ini menghasilkan algoritma baku untuk tugas-tugas komputasi seperti penjumlahan/perkalian n buah bilangan, menentukan elemen minimum atau maksimum di dalam senarai (larik). Di sisi lain, kelemahan dari algoritma ini adalah algoritma ini jarang menghasilkan solusi yang mangkus, algoritma ini terbilang lambat untuk kasus yang besar, dan tidak sekreatif pemecahan masalah lainnya.

1.2 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.

Permainan dimulai dengan papan yang kosong. Pemain dapat meletakkan blok puzzle sedemikian sehingga tidak ada blok yang bertumpang tindih (kecuali dalam kasus 3D). Setiap blok puzzle dapat dirotasikan maupun dicerminkan. Puzzle dinyatakan selesai jika dan hanya jika papan terisi penuh dan seluruh blok puzzle berhasil diletakkan.



Gambar 1. Permainan IQ Puzzle Pro
(Sumber: <https://www.smartgamesusa.com>)

1.3 Algoritma Penyelesaian IQ Puzzler Pro dengan Pendekatan Brute Force

Masalah IQ Puzzler Pro dapat diselesaikan dengan metode brute force melalui beberapa langkah sebagai berikut:

1. Program yang sudah dijalankan akan meminta masukan/input berupa file yang berisi masalah terkait IQ Puzzler Pro. File ini memiliki format txt dan memiliki isi ukuran papan permainan, jumlah blok, jenis blok, dan puzzle-puzzle yang memiliki bentuk secara acak.
2. Program yang dijalankan akan mencatat waktu mulai dan variabel untuk menghitung iterasi yang dilakukan.
3. Untuk puzzle yang diberikan akan diklasifikasi sesuai dengan huruf alfabet yang ada dan disimpan dalam bentuk *list of list*, lalu puzzle yang ada akan dicari seluruh kemungkinan bentuknya dengan fungsi untuk melakukan transformasi puzzle, bisa melalui rotasi ataupun pencerminan.

4. Ketika diletakkan dalam papan, program akan mengecek selalu apakah papan tersebut sudah penuh atau masih kosong, lalu jika masih kosong, potongan puzzle akan diletakkan pada suatu posisi tanpa bertabrakan dan dilanjutkan ke potongan lain untuk diletakkan dan jika ujungnya tidak ada solusi, potongan akan dihapus dan dicari posisi lain yang berbeda dan sesuai.
5. Jika semua potongan berhasil ditempatkan dalam papan tanpa sisa, maka solusi berhasil ditemukan. Jika papan penuh tapi masih ada sisa potongan atau jika potongan tidak bersisa tapi papan belum terisi penuh, maka solusi tidak berhasil ditemukan dan program harus mencari konfigurasi lain untuk mencari solusi yang ada hingga semua kemungkinan teruji.

BAB II

ALGORITMA PROGRAM

2.1 PuzzleInput.java

File ini memiliki fungsi untuk menerima masukan/input dari file yang dipilih oleh pengguna.

Nama Fungsi	Deskripsi
readFromFile(String filename)	Fungsi untuk membaca file yang telah disimpan dalam folder “test/input/”, pengguna menuliskan nama file yang ingin dipecahkan
normalizePiece(List<String> piece)	Fungsi untuk menjaga bentuk potongan puzzle dari spasi yang tidak perlu dan memiliki bentuk konsisten
getRows()	Fungsi untuk mendapat baris dari papan
getCols()	Fungsi untuk mendapat kolom dari papan
getPieces()	Fungsi untuk mendapat jumlah potongan puzzle
getType()	Fungsi untuk mendapat tipe papan
getPuzzleShapes()	Fungsi untuk mendapat masing-masing puzzle yang ada

2.2 PuzzleSolver.java

File ini memiliki fungsi untuk menyelesaikan puzzle dengan algoritma brute force dan dapat melakukan save sekaligus generate gambar.

Nama Fungsi	Deskripsi
PuzzleSolver(int rows, int cols, List<List<String>> puzzle)	Fungsi untuk inisialisasi dan membuat papan permainan
solve()	Fungsi untuk mencari solusi dengan algoritma brute force dan mencatat waktu eksekusi, fungsi bersifat boolean yang akan mengembalikan true jika ada solusi dan false jika tidak
placePiece(int index)	Fungsi untuk menempatkan setiap potongan puzzle ke papan
getPieceChar(List<String> piece)	Fungsi untuk mengambil potongan puzzle berdasarkan karakter dalam alfabet
canPlace(List<String> piece, int row, int col)	Fungsi untuk mengecek apakah suatu potongan puzzle dapat diletakkan pada posisi tertentu, seperti tidak bertabrakan atau keluar dari papan

place(List<String> piece, int row, int col, char pieceChar)	Fungsi untuk meletakkan potongan puzzle pada papan
remove(List<String> piece, int row, int col)	Fungsi untuk menghapus potongan puzzle dari papan
isBoardFull()	Fungsi untuk mengecek apakah papan sudah terisi penuh oleh potongan puzzle
transform(List<String> piece)	Fungsi untuk menghasilkan semua kemungkinan bentuk puzzle berdasarkan rotasi dan pencerminan
rotate90(List<String> piece)	Fungsi untuk memutar potongan puzzle sebesar 90 derajat
flipHorizontally(List<String> piece)	Fungsi untuk memutar potongan puzzle secara horizontal
flipVertically(List<String> piece)	Fungsi untuk memutar potongan puzzle secara vertikal
printSolution(String filename)	Fungsi untuk mengeluarkan solusi dengan warna sesuai potongan dan menawarkan untuk menyimpan file dalam bentuk txt ataupun gambar
getColoredChar(char c)	Fungsi untuk mewarnai potongan puzzle
saveSolution(String solutionText, String fileName)	Fungsi untuk menyimpan solusi dalam format teks (txt)
saveSolutionAsImage(String fileName)	Fungsi untuk menyimpan solusi dalam bentuk gambar yang di generate (png)
getIterationCount()	Fungsi untuk mendapat banyak kasus yang dicek
getSolutionText()	Fungsi untuk mendapat solusi berbentuk text

2.3 Main.java

File ini berfungsi untuk menjalankan algoritma secara keseluruhan.

Nama Fungsi	Deskripsi
main(String[] args)	Fungsi untuk membaca input file dan menjalankan solver sekaligus print solusi yang didapat

2.4 PuzzleGUI.java

File ini memiliki fungsi untuk membuat GUI pada program yang telah dibuat. Dengan adanya GUI, pengguna tidak perlu melakukan proses pencarian solusi dari terminal lagi dan dapat dilakukan langsung di GUI.

Nama Fungsi	Deskripsi
loadPuzzle()	Fungsi untuk mendapat informasi-informasi seputar test case yang ada untuk diolah pada bagian GUI
solvePuzzle()	Fungsi untuk melakukan solve puzzle berdasarkan fungsi yang telah dibuat pada file PuzzleSolver
saveSolutionAsText()	Fungsi untuk melakukan save dengan format text (txt) berdasarkan fungsi yang ada pada PuzzleSolver
saveSolutionAsImage()	Fungsi untuk melakukan save dengan bentuk gambar (png) berdasarkan fungsi yang ada pada PuzzleSolver
drawBoard(Graphics g)	Fungsi untuk melakukan visualisasi papan permainan

BAB III

SOURCE CODE

3.1 Repository Github

Berikut adalah link repository github dari algoritma yang telah dibuat:

https://github.com/fithrarkz/Tucill_13523049

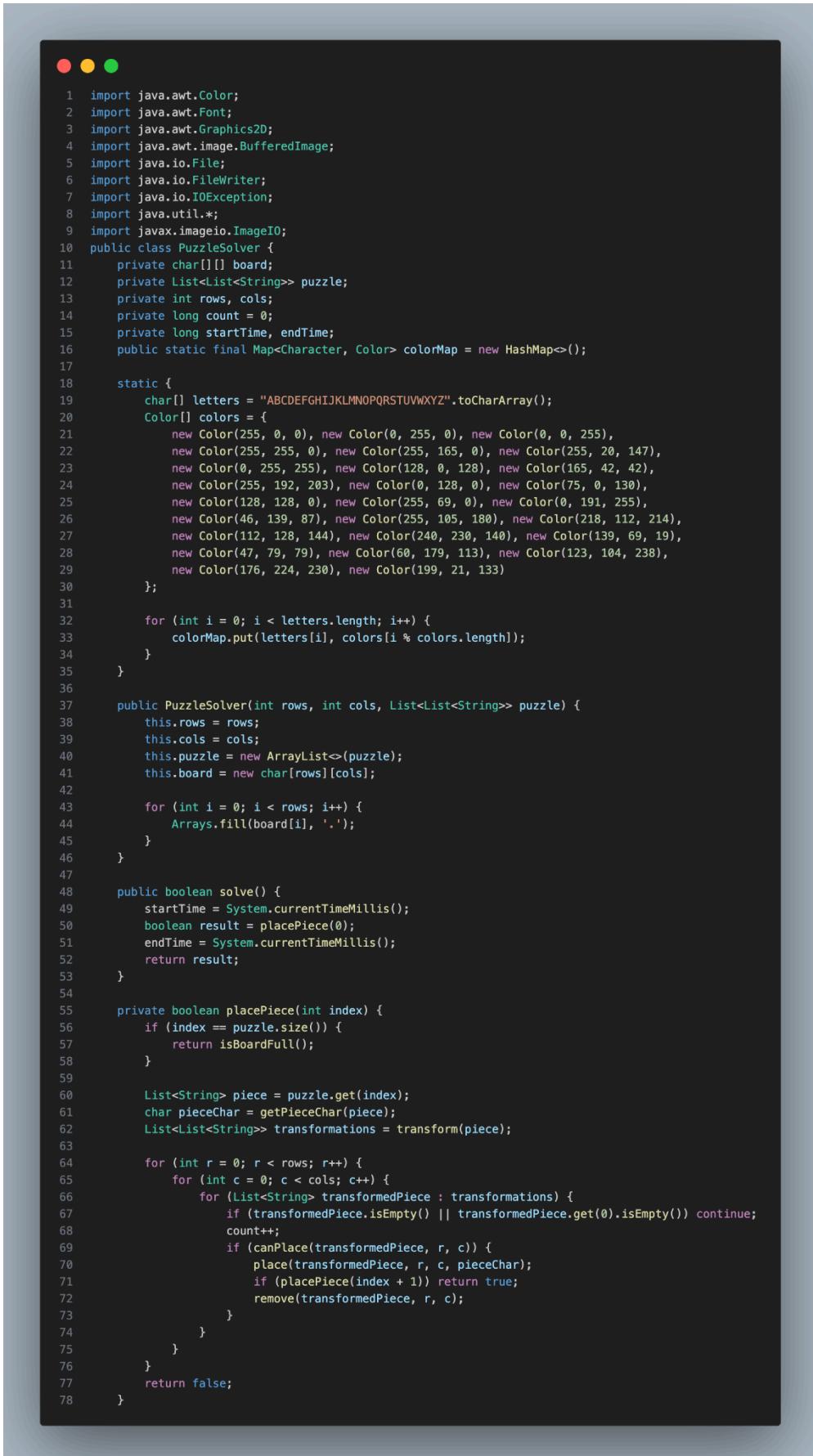
3.2 Source Code Program

3.2.1 PuzzleInput.java



```
1 import java.io.*;
2 import java.util.*;
3
4 public class PuzzleInput {
5     private int rows, cols, pieces;
6     private String type;
7     private String filename;
8     private List<List<String>> puzzleShapes = new ArrayList<>();
9
10    public PuzzleInput(String filename) throws IOException {
11        this.filename = filename;
12        readFromFile(filename);
13    }
14
15    private void readFromFile(String filename) throws IOException {
16        String inputPath = "test/input/" + filename;
17        BufferedReader br = new BufferedReader(new FileReader(inputPath));
18
19        // Baca baris pertama (N M P)
20        String[] firstLine = br.readLine().trim().split(" ");
21        if (firstLine.length != 3) {
22            throw new IllegalArgumentException("Format input salah! Baris pertama harus berisi 3 angka (N M P).");
23        }
24
25        rows = Integer.parseInt(firstLine[0]);
26        cols = Integer.parseInt(firstLine[1]);
27        pieces = Integer.parseInt(firstLine[2]);
28        type = br.readLine().trim();
29
30        List<String> currentPiece = new ArrayList<>();
31        String prevFirstChar = null;
32        String line;
33
34        while ((line = br.readLine()) != null) {
35            if (line.trim().isEmpty()) {
36                continue; // Lewati baris kosong
37            }
38
39            if (line.length() <= cols && line.chars().allMatch(Character::isWhitespace)) {
40                continue;
41            }
42            String trimmedLine = line.replaceAll("^\\s+", "");
43            if (trimmedLine.isEmpty()) {
44                continue;
45            }
46            String firstChar = trimmedLine.substring(0, 1);
47
48            if (prevFirstChar != null && !prevFirstChar.equals(firstChar) && !currentPiece.isEmpty()) {
49                puzzleShapes.add(normalizePiece(currentPiece));
50            }
51            currentPiece.add(line);
52            prevFirstChar = firstChar;
53        }
54
55        if (!currentPiece.isEmpty()) {
56            puzzleShapes.add(normalizePiece(currentPiece));
57        }
58        br.close();
59
60        if (puzzleShapes.size() != pieces) {
61            throw new IllegalArgumentException(
62                "Jumlah puzzle = " + pieces + ", tidak sesuai dengan jumlah puzzle yang ditemukan = " + puzzleShapes.size()
63            );
64        }
65    }
66
67    private List<String> normalizePiece(List<String> piece) {
68        int minLeadingSpaces = Integer.MAX_VALUE;
69        for (String row : piece) {
70            int leadingSpaces = row.length() - row.stripLeading().length();
71            if (!row.strip().isEmpty()) {
72                minLeadingSpaces = Math.min(minLeadingSpaces, leadingSpaces);
73            }
74        }
75        List<String> normalized = new ArrayList<>();
76        for (String row : piece) {
77            normalized.add(row.substring(Math.min(minLeadingSpaces, row.length())));
78        }
79        return normalized;
80    }
81
82
83    public int getRows() { return rows; }
84    public int getCols() { return cols; }
85    public int getPieces() { return pieces; }
86    public String getType() { return type; }
87    public List<List<String>> getPuzzleShapes() { return puzzleShapes; }
88 }
```

3.2.2 PuzzleSolver.java



```
1 import java.awt.Color;
2 import java.awt.Font;
3 import java.awt.Graphics2D;
4 import java.awt.image.BufferedImage;
5 import java.io.File;
6 import java.io.FileWriter;
7 import java.io.IOException;
8 import java.util.*;
9 import javax.imageio.ImageIO;
10 public class PuzzleSolver {
11     private char[][] board;
12     private List<List<String>> puzzle;
13     private int rows, cols;
14     private long count = 0;
15     private long startTime, endTime;
16     public static final Map<Character, Color> colorMap = new HashMap<>();
17
18     static {
19         char[] letters = "ABCDEFGHIJKLMNPQRSTUVWXYZ".toCharArray();
20         Color[] colors = {
21             new Color(255, 0, 0), new Color(0, 255, 0), new Color(0, 0, 255),
22             new Color(255, 255, 0), new Color(255, 165, 0), new Color(255, 20, 147),
23             new Color(0, 255, 255), new Color(128, 0, 128), new Color(165, 42, 42),
24             new Color(255, 192, 203), new Color(0, 128, 0), new Color(75, 0, 130),
25             new Color(128, 128, 0), new Color(255, 69, 0), new Color(0, 191, 255),
26             new Color(46, 139, 87), new Color(255, 105, 180), new Color(228, 112, 214),
27             new Color(112, 128, 144), new Color(240, 230, 140), new Color(139, 69, 19),
28             new Color(47, 79, 79), new Color(60, 179, 113), new Color(123, 104, 238),
29             new Color(176, 224, 230), new Color(199, 21, 133)
30         };
31
32         for (int i = 0; i < letters.length; i++) {
33             colorMap.put(letters[i], colors[i % colors.length]);
34         }
35     }
36
37     public PuzzleSolver(int rows, int cols, List<List<String>> puzzle) {
38         this.rows = rows;
39         this.cols = cols;
40         this.puzzle = new ArrayList<>(puzzle);
41         this.board = new char[rows][cols];
42
43         for (int i = 0; i < rows; i++) {
44             Arrays.fill(board[i], '.');
45         }
46     }
47
48     public boolean solve() {
49         startTime = System.currentTimeMillis();
50         boolean result = placePiece(0);
51         endTime = System.currentTimeMillis();
52         return result;
53     }
54
55     private boolean placePiece(int index) {
56         if (index == puzzle.size()) {
57             return isBoardFull();
58         }
59
60         List<String> piece = puzzle.get(index);
61         char pieceChar = getPieceChar(piece);
62         List<List<String>> transformations = transform(piece);
63
64         for (int r = 0; r < rows; r++) {
65             for (int c = 0; c < cols; c++) {
66                 for (List<String> transformedPiece : transformations) {
67                     if (transformedPiece.isEmpty() || transformedPiece.get(0).isEmpty()) continue;
68                     count++;
69                     if (canPlace(transformedPiece, r, c)) {
70                         place(transformedPiece, r, c, pieceChar);
71                         if (placePiece(index + 1)) return true;
72                         remove(transformedPiece, r, c);
73                     }
74                 }
75             }
76         }
77         return false;
78     }
}
```

```
1  private char getPieceChar(List<String> piece) {
2      return piece.stream().filter(row -> !row.isEmpty()).map(row -> row.charAt(0)).findFirst().orElse('?');
3  }
4
5  private boolean canPlace(List<String> piece, int row, int col) {
6      for (int r = 0; r < piece.size(); r++) {
7          for (int c = 0; c < piece.get(r).length(); c++) {
8              if (piece.get(r).charAt(c) != ' ' && (row + r >= rows || col + c >= cols || board[row + r][col + c] != '.')) {
9                  return false;
10             }
11         }
12     }
13     return true;
14 }
15
16 private void place(List<String> piece, int row, int col, char pieceChar) {
17     for (int r = 0; r < piece.size(); r++) {
18         for (int c = 0; c < piece.get(r).length(); c++) {
19             if (piece.get(r).charAt(c) != ' ') {
20                 board[row + r][col + c] = pieceChar;
21             }
22         }
23     }
24 }
25
26 private void remove(List<String> piece, int row, int col) {
27     for (int r = 0; r < piece.size(); r++) {
28         for (int c = 0; c < piece.get(r).length(); c++) {
29             if (piece.get(r).charAt(c) != ' ') {
30                 board[row + r][col + c] = '.';
31             }
32         }
33     }
34 }
35
36 private boolean isBoardFull() {
37     for (char[] row : board) {
38         for (char c : row) {
39             if (c == '.') return false;
40         }
41     }
42     return true;
43 }
44
45 public long getIterationCount() {
46     return count;
47 }
48
49 public String getSolutionText() {
50     StringBuilder sb = new StringBuilder();
51     for (int r = 0; r < rows; r++) {
52         for (int c = 0; c < cols; c++) {
53             sb.append(board[r][c]).append(" ");
54         }
55         sb.append("\n");
56     }
57     return sb.toString();
58 }
59
60 private List<List<String>> transform(List<String> piece) {
61     Set<List<String>> transformations = new LinkedHashSet<>();
62     transformations.add(piece);
63     transformations.add(flipHorizontally(piece));
64     transformations.add(flipVertically(piece));
65     List<String> rotated = piece;
66     for (int i = 0; i < 3; i++) {
67         rotated = rotate90(rotated);
68         transformations.add(rotated);
69         transformations.add(flipHorizontally(rotated));
70         transformations.add(flipVertically(rotated));
71     }
72     return new ArrayList<>(transformations);
73 }
```

```
 1  private List<String> rotate90(List<String> piece) {
 2      if (piece.isEmpty()) return new ArrayList<>();
 3      int height = piece.size();
 4      int width = piece.stream().mapToInt(String::length).max().orElse(0);
 5      char[][] rotated = new char[width][height];
 6
 7      for (int r = 0; r < height; r++) {
 8          for (int c = 0; c < piece.get(r).length(); c++) {
 9              rotated[c][height - 1 - r] = piece.get(r).charAt(c);
10          }
11      }
12      List<String> rotatedStrings = new ArrayList<>();
13      for (char[] row : rotated) {
14          rotatedStrings.add(new String(row).replace('0', ' '));
15      }
16      return rotatedStrings;
17  }
18
19  private List<String> flipHorizontally(List<String> piece) {
20      List<String> flipped = new ArrayList<>();
21      for (String row : piece) {
22          flipped.add(new StringBuilder(row).reverse().toString());
23      }
24      return flipped;
25  }
26
27  private List<String> flipVertically(List<String> piece) {
28      List<String> flipped = new ArrayList<>(piece);
29      Collections.reverse(flipped);
30      return flipped;
31  }
32
33  public void printSolution(String filename) {
34      StringBuilder output = new StringBuilder();
35      for (char[] row : board) {
36          for (char c : row) {
37              System.out.print(getColoredChar(c));
38              output.append(c);
39          }
40          System.out.println();
41          output.append("\n");
42      }
43      System.out.println("Waktu Pencarian: " + (endTime - startTime) + " ms");
44      System.out.println("Banyak kasus yang ditinjau: " + count);
45
46      Scanner scanner = new Scanner(System.in);
47      System.out.print("Apakah Anda ingin menyimpan solusi dalam txt? (ya/tidak): ");
48      String response = scanner.nextLine().trim().toLowerCase();
49
50      if (response.equals("ya")) {
51          saveSolution(output.toString(), filename);
52      }
53
54      Scanner scan = new Scanner(System.in);
55      System.out.print("Apakah Anda ingin menyimpan solusi dalam gambar? (ya/tidak): ");
56      String respon = scan.nextLine().trim().toLowerCase();
57
58      if (respon.equals("ya")) {
59          saveSolutionAsImage(filename);
60      }
61  }
```

```
1  private String getColoredChar(char c) {
2      Map<Character, String> colorMap = new HashMap<>();
3
4      char[] letters = "ABCDEFGHIJKLMNPQRSTUVWXYZ".toCharArray();
5      String[] colors = {
6          "\u001B[31m", "\u001B[32m", "\u001B[33m", "\u001B[34m", "\u001B[35m",
7          "\u001B[36m", "\u001B[91m", "\u001B[92m", "\u001B[93m", "\u001B[94m",
8          "\u001B[95m", "\u001B[96m", "\u001B[97m", "\u001B[90m", "\u001B[100m",
9          "\u001B[101m", "\u001B[102m", "\u001B[103m", "\u001B[104m", "\u001B[105m",
10         "\u001B[106m", "\u001B[107m", "\u001B[37m", "\u001B[95m", "\u001B[94m",
11         "\u001B[92m"
12     };
13
14     for (int i = 0; i < letters.length; i++) {
15         colorMap.put(letters[i], colors[i % colors.length]);
16     }
17     return colorMap.getOrDefault(c, "\u001B[0m") + c + "\u001B[0m";
18 }
19
20 public void saveSolution(String solutionText, String fileName) {
21     String folderPath = "test/output/";
22     String filePath = folderPath + "solusi_" + fileName + ".txt";
23
24     try (FileWriter writer = new FileWriter(filePath)) {
25         writer.write(solutionText);
26         writer.write("Waktu Pencarian: " + (endTime - startTime) + " ms\n");
27         writer.write("Banyak kasus yang ditinjau: " + count + "\n\n");
28     } catch (IOException e) {
29         System.out.println("Gagal menyimpan solusi: " + e.getMessage());
30     }
31 }
32 public void saveSolutionAsImage(String fileName) {
33     int cellSize = 50;
34     int width = cols * cellSize;
35     int height = rows * cellSize;
36     BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
37     Graphics2D g2d = image.createGraphics();
38     g2d.setFont(new Font("Arial", Font.BOLD, cellSize / 2));
39
40     for (int r = 0; r < rows; r++) {
41         for (int c = 0; c < cols; c++) {
42             char pieceChar = board[r][c];
43             g2d.setColor(colorMap.getOrDefault(pieceChar, Color.WHITE));
44             g2d.fillRect(c * cellSize, r * cellSize, cellSize, cellSize);
45             g2d.setColor(Color.BLACK);
46             g2d.drawRect(c * cellSize, r * cellSize, cellSize, cellSize);
47             if (pieceChar != '.') {
48                 g2d.setColor(Color.BLACK);
49                 int x = c * cellSize + cellSize / 3;
50                 int y = r * cellSize + 2 * cellSize / 3;
51                 g2d.drawString(String.valueOf(pieceChar), x, y);
52             }
53         }
54     }
55     g2d.dispose();
56
57     try {
58         File outputFile = new File("test/output/solusi_" + fileName + ".png");
59         ImageIO.write(image, "png", outputFile);
60         System.out.println("Solusi berhasil disimpan sebagai gambar di " + outputFile.getAbsolutePath());
61     } catch (IOException e) {
62         System.out.println("Gagal menyimpan gambar: " + e.getMessage());
63     }
64 }
65
66 public char[][] getBoard() { return board; }
67 }
```

3.2.3 Main.java



```
1 public class Main {
2     public static void main(String[] args) {
3         javax.swing.SwingUtilities.invokeLater(() -> {
4             PuzzleGUI gui = new PuzzleGUI();
5             gui.setVisible(true);
6         });
7     }
8 }
```

3.2.4 PuzzleGUI.java



```
1 import java.awt.*;
2 import java.util.List;
3 import javax.swing.*;
4
5 public class PuzzleGUI extends JFrame {
6     private JTextField fileInputField;
7     private JPanel boardPanel;
8     private JTextArea outputArea;
9     private JButton solveButton, saveTextButton, saveImageButton;
10    private PuzzleSolver solver;
11    private int rows, cols;
12    private List<List<String>> puzzleShapes;
13
14    public PuzzleGUI() {
15        setTitle("IO Puzzler Solver");
16        setSize(600, 700);
17        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
18        setLayout(new BorderLayout());
19
20        // input
21        JPanel inputPanel = new JPanel();
22        fileInputField = new JTextField(20);
23        inputPanel.add(new JLabel("Masukkan nama file: "));
24        inputPanel.add(fileInputField);
25        add(inputPanel, BorderLayout.NORTH);
26
27        // board
28        boardPanel = new JPanel() {
29            protected void paintComponent(Graphics g) {
30                super.paintComponent(g);
31                drawBoard(g);
32            }
33        };
34        boardPanel.setPreferredSize(new Dimension(400, 400));
35        JPanel centerPanel = new JPanel();
36        centerPanel.setLayout(new BorderLayout());
37        centerPanel.add(boardPanel, BorderLayout.NORTH);
38
39        // info
40        outputArea = new JTextArea(5, 40);
41        outputArea.setEditable(false);
42        JScrollPane scrollPane = new JScrollPane(outputArea);
43        centerPanel.add(scrollPane, BorderLayout.CENTER);
44        add(centerPanel, BorderLayout.CENTER);
45
46        // solve
47        solveButton = new JButton("Solve");
48        saveTextButton = new JButton("Simpan sebagai Teks");
49        saveImageButton = new JButton("Simpan sebagai Gambar");
50        saveTextButton.setVisible(false);
51        saveImageButton.setVisible(false);
52
53        JPanel buttonPanel = new JPanel();
54        buttonPanel.add(solveButton);
55        buttonPanel.add(saveTextButton);
56        buttonPanel.add(saveImageButton);
57        add(buttonPanel, BorderLayout.SOUTH);
58        solveButton.addActionListener(e -> loadPuzzle());
59        saveTextButton.addActionListener(e -> saveSolutionAsText());
60        saveImageButton.addActionListener(e -> saveSolutionAsImage());
61    }
62}
```

```
1  private void loadPuzzle() {
2      String fileName = fileInputField.getText().trim();
3      if (fileName.isEmpty()) {
4          JOptionPane.showMessageDialog(this, "Masukkan nama file terlebih dahulu!", "Error", JOptionPane.ERROR_MESSAGE);
5          return;
6      }
7
8      try {
9          PuzzleInput puzzleInput = new PuzzleInput(fileName);
10         this.rows = puzzleInput.getRows();
11         this.cols = puzzleInput.getCols();
12         this.puzzleShapes = puzzleInput.getPuzzleShapes();
13         this.solver = new PuzzleSolver(rows, cols, puzzleShapes);
14         boardPanel.setPreferredSize(new Dimension(cols * 50, rows * 50));
15         boardPanel.revalidate();
16         boardPanel.repaint();
17         saveTextButton.setVisible(false);
18         saveImageButton.setVisible(false);
19         solvePuzzle();
20     } catch (Exception ex) {
21         JOptionPane.showMessageDialog(this, "Error: " + ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
22     }
23 }
24
25 private void solvePuzzle() {
26     outputArea.setText("");
27     long startTime = System.currentTimeMillis();
28     boolean solved = solver.solve();
29     long endTime = System.currentTimeMillis();
30     double executionTime = endTime - startTime;
31     boardPanel.repaint();
32
33     if (solved) {
34         outputArea.append(
35             "Waktu eksekusi: " + executionTime + " ms\n" +
36             "Jumlah iterasi: " + solver.getIterationCount() + "\n");
37
38         saveTextButton.setVisible(true);
39         saveImageButton.setVisible(true);
40     } else {
41         JOptionPane.showMessageDialog(this, "Tidak ada solusi yang ditemukan!");
42     }
43 }
44
45
46 private void saveSolutionAsText() {
47     if (solver == null) return;
48     String fileName = fileInputField.getText().trim();
49     solver.saveSolution(solver.getSolutionText(), fileName);
50 }
51
52 private void saveSolutionAsImage() {
53     if (solver == null) return;
54     String fileName = fileInputField.getText().trim();
55     solver.saveSolutionAsImage(fileName);
56 }
57
58 private void drawBoard(Graphics g) {
59     if (solver == null) return;
60     int cellSize = 50;
61     FontMetrics fm = g.getFontMetrics();
62     for (int r = 0; r < rows; r++) {
63         for (int c = 0; c < cols; c++) {
64             char piece = solver.getBoard()[r][c];
65             g.setColor(PuzzleSolver.colorMap.getOrDefault(piece, Color.WHITE));
66             g.fillRect(c * cellSize, r * cellSize, cellSize, cellSize);
67             g.setColor(Color.BLACK);
68             g.drawRect(c * cellSize, r * cellSize, cellSize, cellSize);
69             if (piece != ' ') {
70                 String pieceText = String.valueOf(piece);
71                 int textWidth = fm.stringWidth(pieceText);
72                 int textHeight = fm.getAscent();
73                 int x = c * cellSize + (cellSize - textWidth) / 2;
74                 int y = r * cellSize + (cellSize + textHeight) / 2 - 4;
75                 g.drawString(pieceText, x, y);
76             }
77         }
78     }
79 }
80 public static void main(String[] args) {
81     SwingUtilities.invokeLater(() -> new PuzzleGUI().setVisible(true));
82 }
83 }
```

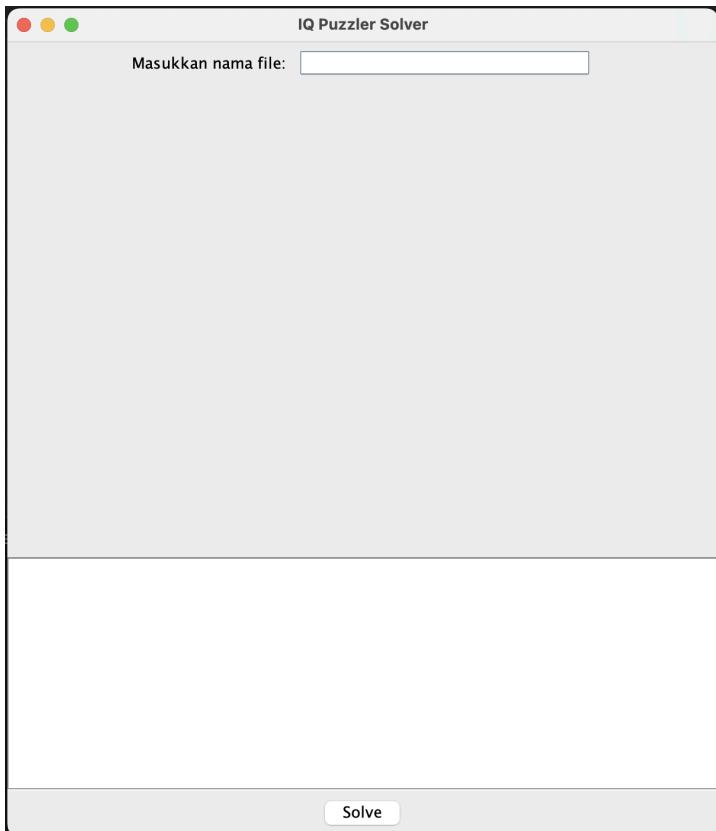
BAB IV

IMPLEMENTASI PROGRAM

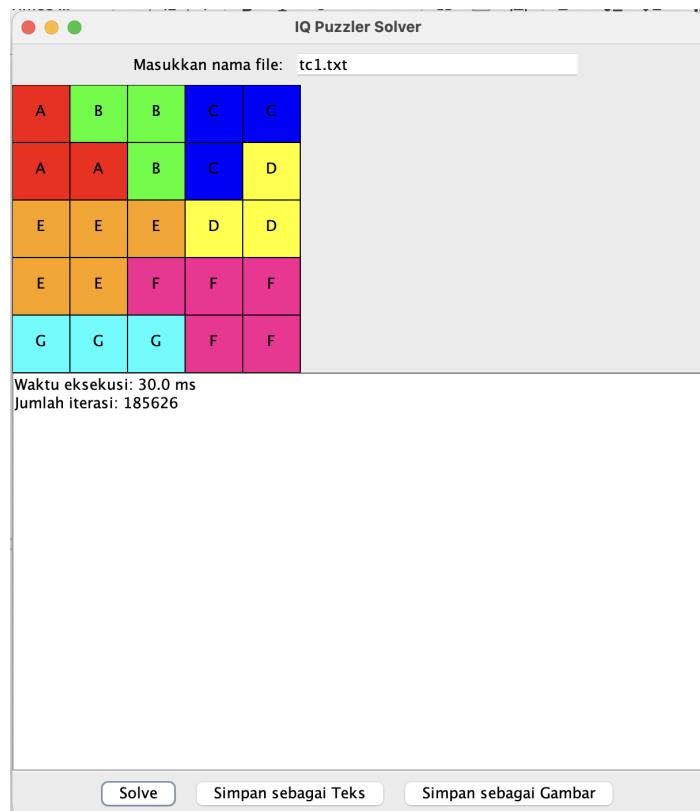
4.1 Test Case 1

Test case pertama berisi input sesuai dengan spesifikasi pada dokumen yang diberikan oleh asisten.

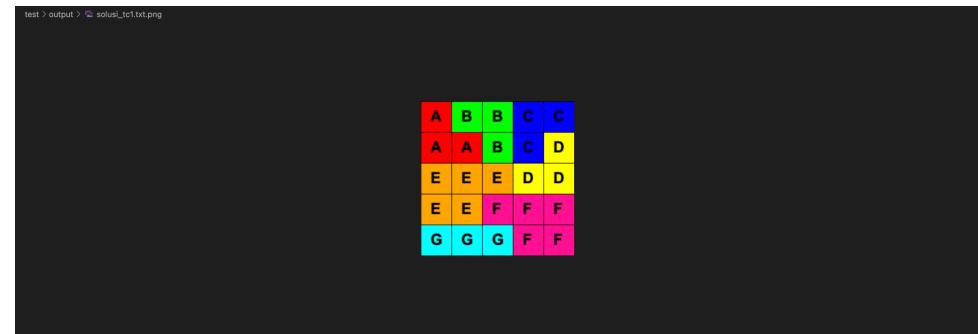
Input test case	
1	5 5 7
2	DEFAULT
3	A
4	AA
5	B
6	BB
7	C
8	CC
9	D
10	DD
11	EE
12	EE
13	E
14	FF
15	FF
16	F
17	GGG

Tampilan awal GUI	
 A screenshot of a window titled "IQ Puzzler Solver". At the top, there are three colored window control buttons (red, yellow, green). Below the title bar is a text input field with the placeholder text "Masukkan nama file:". At the bottom of the window is a horizontal button labeled "Solve".	

Output test case



Hasil penyimpanan teks dan gambar

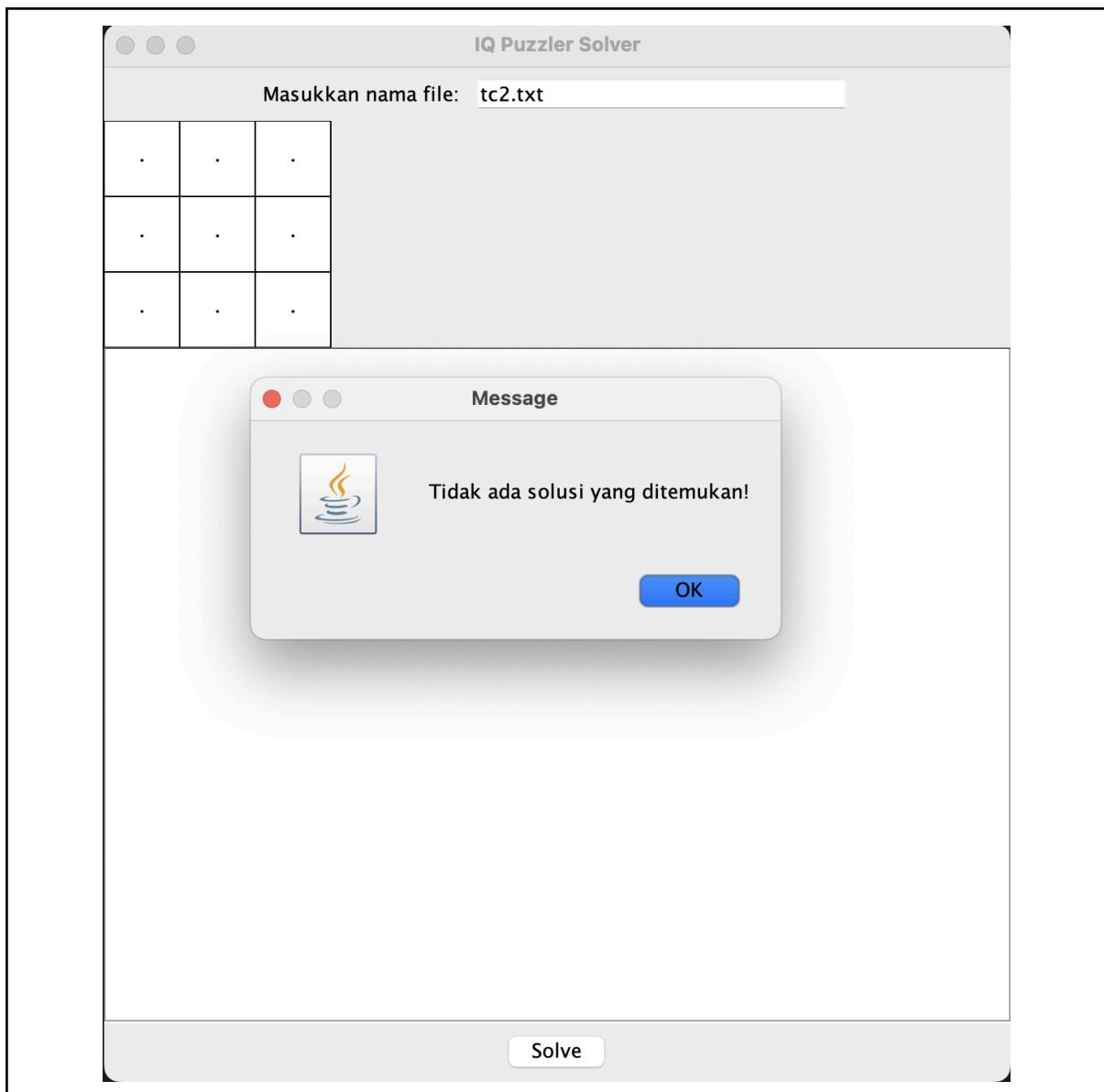


```
test > output > solusi_tc1.txt.txt
1  A B B C C
2  A A B C D
3  E E E D D
4  E E F F F
5  G G G F F
6  Waktu Pencarian: 30 ms
7  Banyak kasus yang ditinjau: 185626
8
9
```

4.2 Test Case 2

Test case ini memiliki jenis permasalahan dengan input yang tidak memiliki solusi di akhir.

Input test case
<pre>test > input > tc2.txt 1 3 3 4 2 DEFAULT 3 AAA 4 A 5 B 6 C 7 DD</pre>
Tampilan awal GUI
 A screenshot of a Mac OS X application window titled "IQ Puzzler Solver". The window has three main sections: a top header bar with red, yellow, and green close buttons; a middle section containing a label "Masukkan nama file:" followed by an empty text input field; and a bottom section containing a single "Solve" button.
Output test case



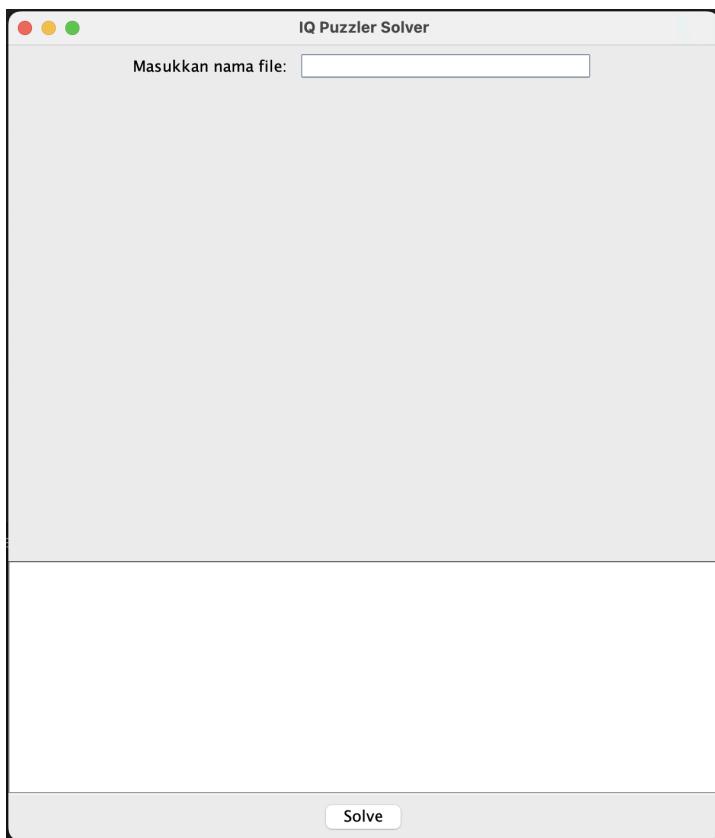
4.3 Test Case 3

Test case ini memiliki bentuk potongan puzzle yang sangat beragam dan juga terdapat potongan puzzle yang diawali dengan spasi terlebih dahulu.

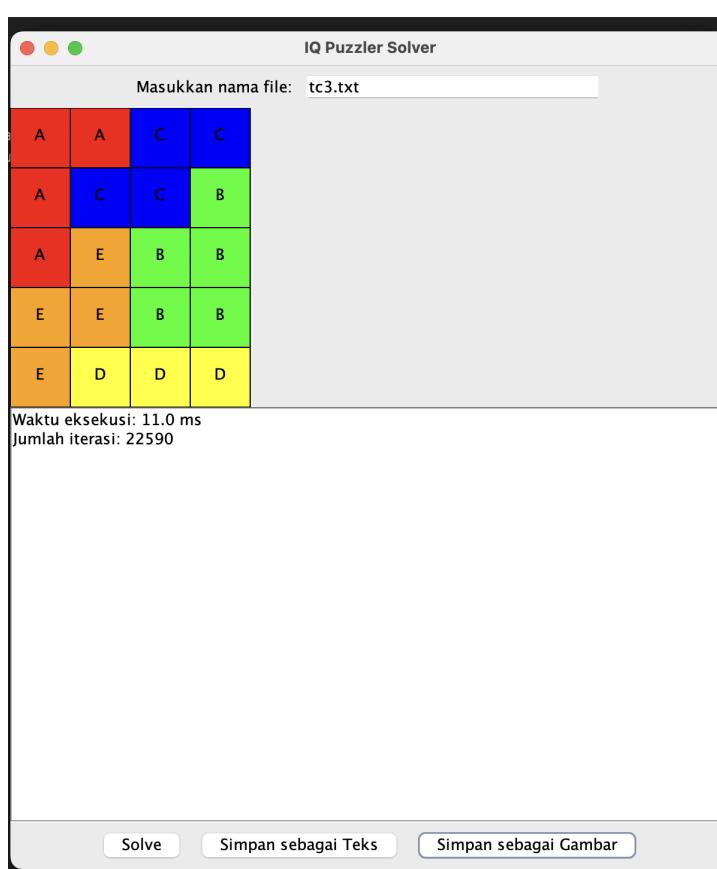
Input test case

```
test > input > tc3.txt
1 5 4 5
2 DEFAULT
3 AA
4 A
5 A
6 BB
7 BBB
8 | C
9 CC
10 C
11 DDD
12 E
13 EE
14 || E
```

Tampilan awal GUI

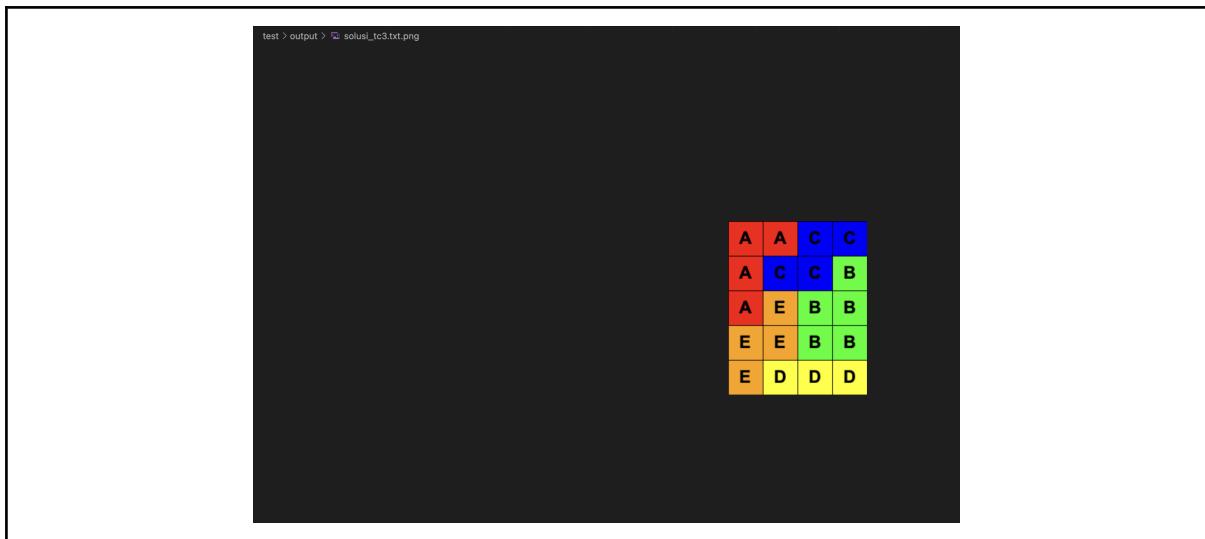


Output test case



Hasil penyimpanan teks dan gambar

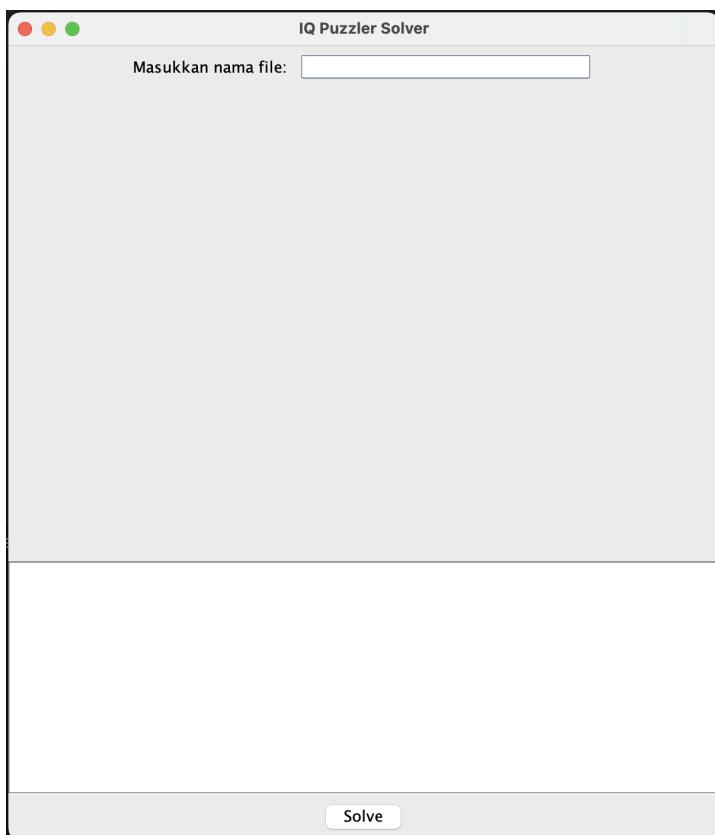
```
test > output > solusi_tc3.txt.txt
1 A A C C
2 A C C B
3 A E B B
4 E E B B
5 E D D D
6 Waktu Pencarian: 11 ms
7 Banyak kasus yang ditinjau: 22590
8
9
```



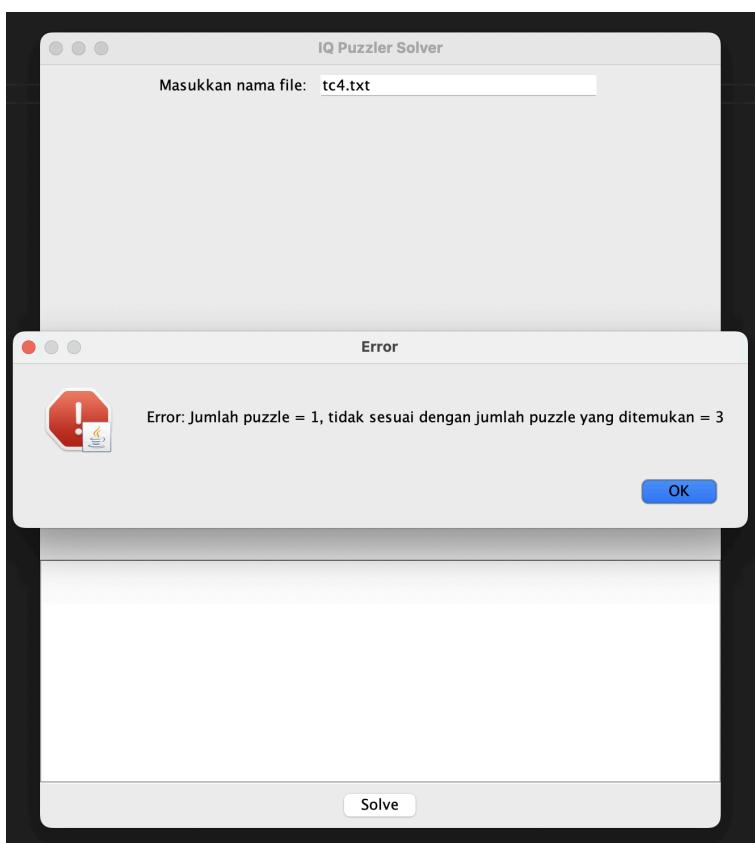
4.4 Test Case 4

Test case ini menguji apakah jumlah potongan puzzle sesuai dengan input P yang dimasukan pengguna.

Input test case
<pre>test > input > tc4.txt 1 3 3 1 2 DEFAULT 3 AAA 4 BBB 5 CCC</pre>
Tampilan awal GUI

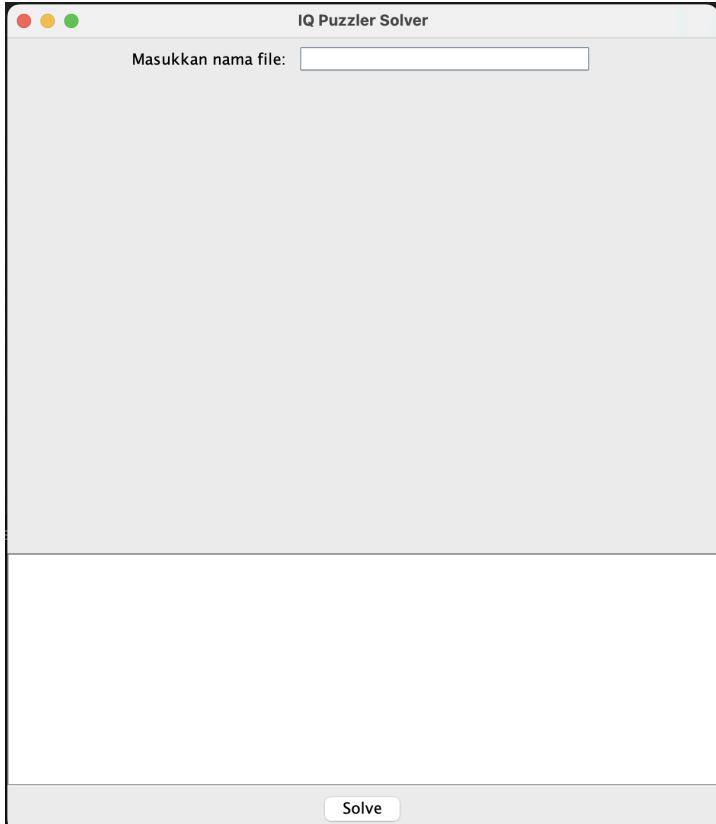


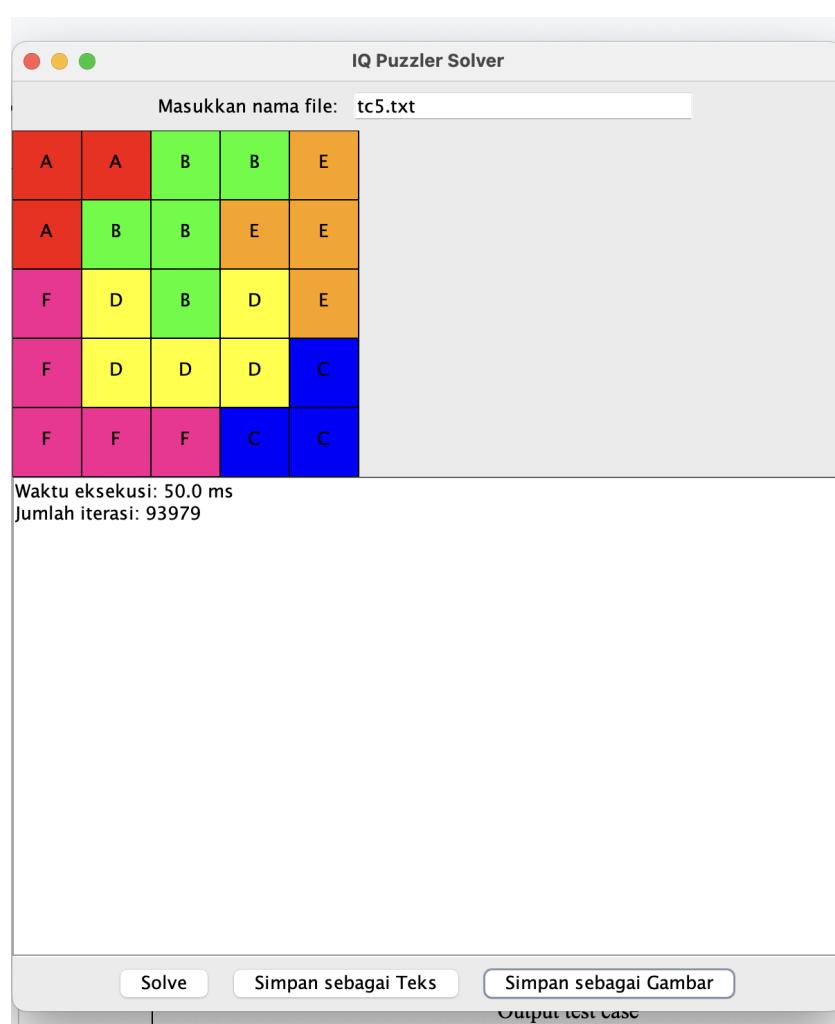
Output test case



4.5 Test Case 5

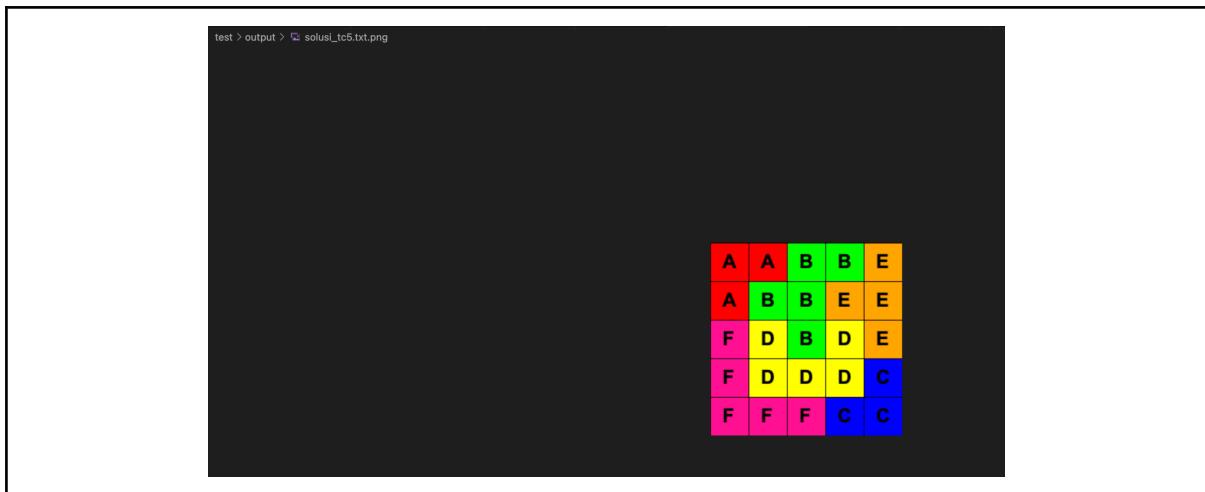
Test case ini menguji program dengan bentuk-bentuk potongan puzzle yang sangat rumit dan banyak yang diawali dengan spasi.

Input test case
<pre>test > input > tc5.txt 1 5 5 6 2 DEFAULT 3 AA 4 A 5 B 6 BB 7 BB 8 C 9 CC 10 DDD 11 D D 12 E 13 EE 14 E 15 F 16 F 17 FFF</pre>
Tampilan awal GUI
 A screenshot of a window titled "IQ Puzzler Solver". At the top left are three colored window control buttons (red, yellow, green). The main title bar says "IQ Puzzler Solver". Below the title bar is a text input field with the placeholder "Masukkan nama file:" followed by a small rectangular input box. At the bottom of the window is a horizontal button labeled "Solve".
Output test case



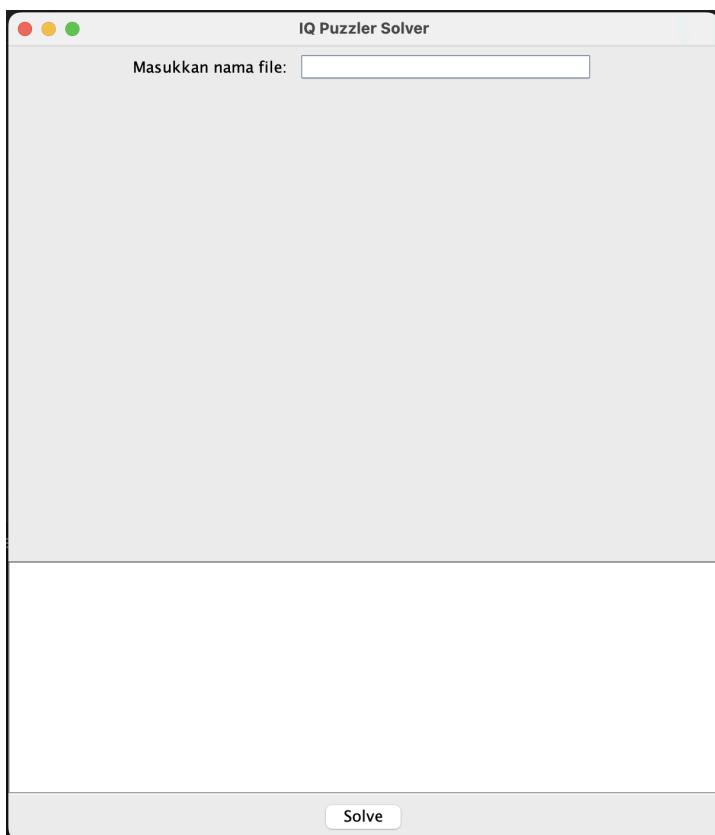
Hasil penyimpanan teks dan gambar

```
test > output > solusi_tc5.txt.txt
1 A A B B E
2 A B B E E
3 F D B D E
4 F D D D C
5 F F F C C
6 Waktu Pencarian: 50 ms
7 Banyak kasus yang ditinjau: 93979
8
9
```



4.6 Test Case 6

Input test case	
test > input > tc6.txt	<pre>1 5 5 7 2 DEFAULT 3 AAA 4 B 5 B 6 C 7 CC 8 D 9 DD 10 EEE 11 E E 12 E 13 FF 14 F 15 F 16 GGG 17 G</pre>
Tampilan awal GUI	



Output test case

IQ Puzzler Solver				
Masukkan nama file: tc6.txt				
A	A	A	B	E
C	C	E	B	E
F	C	E	E	E
F	G	G	G	D
F	F	G	D	D

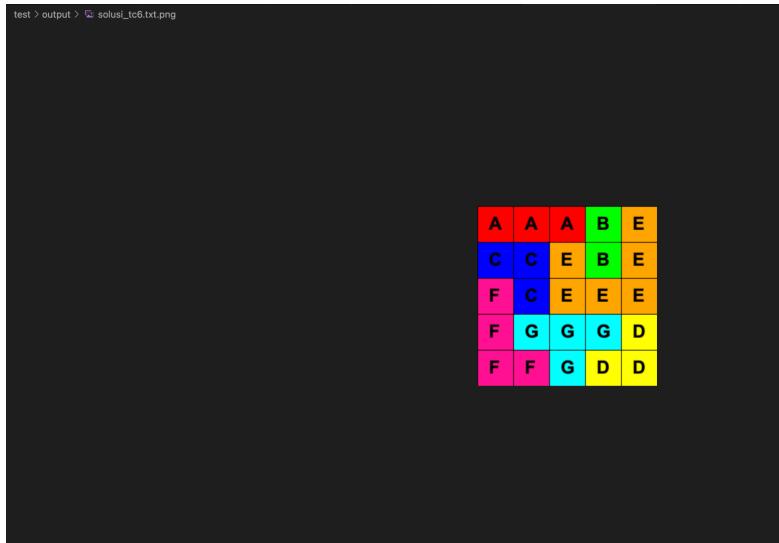
Waktu eksekusi: 41.0 ms
Jumlah iterasi: 361060

Solve

Simpan sebagai Teks

Simpan sebagai Gambar

Hasil penyimpanan teks dan gambar



test > output > solusi_tc6.txt.txt

```
1 A A A B E
2 C C E B E
3 F C E E E
4 F G G G D
5 F F G D D
6 Waktu Pencarian: 41 ms
7 Banyak kasus yang ditinjau: 361060
8
9
```

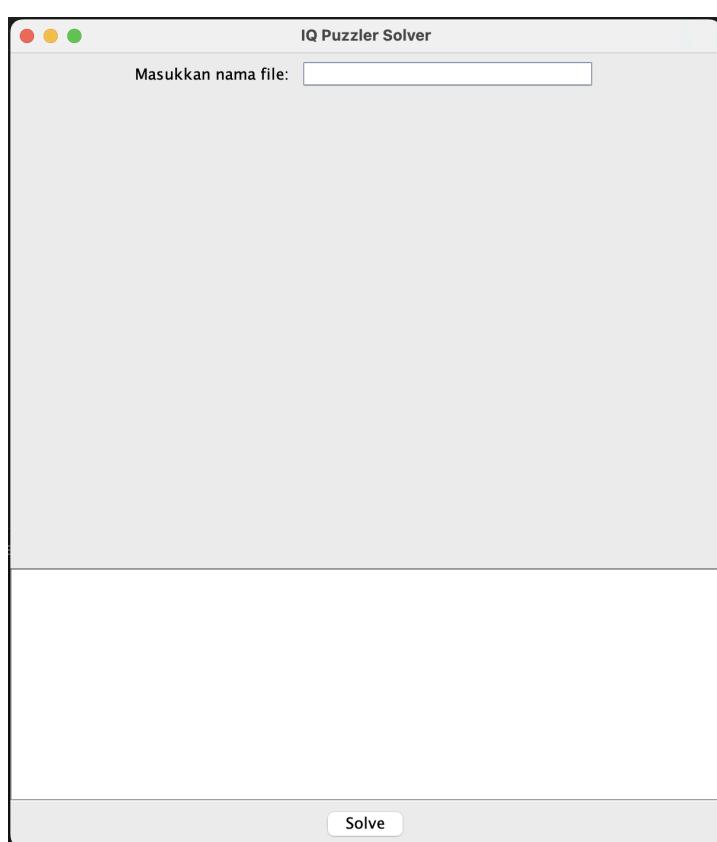
4.7 Test Case 7

Test case ini menguji keberadaan baris kosong dalam input.

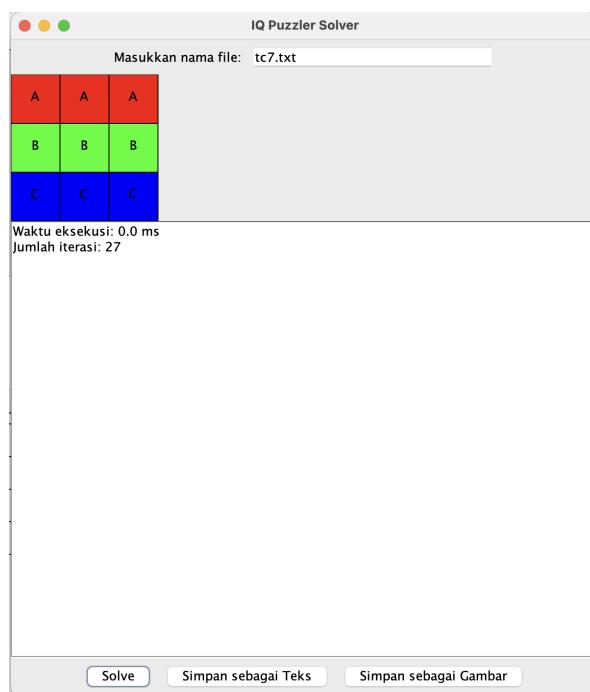
Input test case

```
test > input > tc7.txt
1 3 3
2 DEFAULT
3 AAA
4
5 BBB
6
7 CCC
```

Tampilan awal GUI



Output test case



Hasil penyimpanan teks dan gambar

test > output > 

A	A	A
B	B	B
C	C	C

test > output > solusi_tc7.txt.txt

```
1 A A A
2 B B B
3 C C C
4 Waktu Pencarian: 0 ms
5 Banyak kasus yang ditinjau: 27
6
7
```

BAB V
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 <i>Graphical User Interface</i> (GUI)	V	
6	Program dapat menyimpan solusi dalam bentuk file gambar	V	
7	Program dapat menyelesaikan kasus konfigurasi <i>custom</i>		V
8	Program dapat menyelesaikan kasus konfigurasi Piramida (3D)		V
9	Program dibuat oleh saya sendiri	V	

DAFTAR PUSTAKA

[1]

[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/02-Algoritma-Brute-Force-\(2025\)-Bag1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/02-Algoritma-Brute-Force-(2025)-Bag1.pdf)

[2] <https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/Tucil1-Stima-2025.pdf>