

Tugas Kecil 1 IF2211 Strategi Algoritma
Penyelesaian IQ Puzzler Pro dengan Algoritma Brute Force



Disusun oleh :

Shannon Aurellius Anastasya Lie (13523019)

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2025

DAFTAR ISI

BAB I DESKRIPSI PERSOALAN DAN ALGORITMA.....	3
1.1. Algoritma Brute Force.....	3
1.2. IQ Puzzler Pro.....	3
1.3. Penyelesaian IQ Puzzler Pro dengan Algoritma Brute Force.....	4
BAB II IMPLEMENTASI ALGORITMA DALAM BAHASA JAVA.....	5
2.1. File Input.java.....	5
2.2. File Block.java.....	5
2.3. File Solve.java.....	6
2.4. File Output.java.....	7
2.5. File Main.java.....	8
BAB III SOURCE CODE PROGRAM.....	9
3.1. Repositori Github.....	9
3.2. Source Code Program.....	9
3.2.1. File Input.java.....	9
3.2.2. File Block.java.....	10
3.2.3. File Solve.java.....	11
3.2.4. File Output.java.....	12
3.2.5. File Main.java.....	15
BAB IV MASUKAN DAN LUARAN PROGRAM.....	16
4.1. Test Case 1.....	16
4.2. Test Case 2.....	18
4.3. Test Case 3.....	20
4.4. Test Case 4.....	21
4.5. Test Case 5.....	22
4.6. Test Case 6.....	24
4.7. Test Case 7.....	26
BAB V LAMPIRAN.....	29
DAFTAR PUSTAKA.....	30

BAB I

DESKRIPSI PERSOALAN DAN ALGORITMA

1.1. Algoritma Brute Force

Algoritma brute force adalah metode yang digunakan untuk memecahkan masalah atau mencari solusi dengan cara mencoba semua kemungkinan secara berurutan hingga menemukan solusi yang benar. Metode ini sering digunakan ketika tidak ada cara yang lebih efisien untuk menyelesaikan masalah tersebut. Algoritma ini merupakan metode yang sederhana dan kuat, karena dia tidak mengandalkan pengetahuan sebelumnya atau optimasi khusus.^[1] Algoritma brute force merupakan suatu metode pendekatan yang lurus atau lempang (straightforward) untuk memecahkan suatu persoalan. Algoritma brute force memecahkan persoalan secara sangat sederhana, langsung, jelas caranya (obvious way), mudah dipahami, *Just do it!* atau *Just Solve it!* Biasanya algoritma brute force dapat langsung ditulis berdasarkan pada pernyataan di dalam persoalan (problem statement) dan definisi/konsep yang dilibatkan.^[2]

1.2. IQ Puzzler Pro



Gambar 1 Permainan IQ Puzzler Pro

(Sumber: <https://www.smartgamesusa.com>)

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.^[3]

1.3. Penyelesaian IQ Puzzler Pro dengan Algoritma Brute Force

Penyelesaian permainan IQ Puzzler Pro dapat dilakukan menggunakan pendekatan algoritma brute force. Untuk lebih detailnya, langkah-langkah penyelesaian tertera sebagai berikut.

1. Program meminta pengguna untuk memasukkan nama file input dengan format .txt melalui kelas Main. Kemudian file input diakses dan divalidasi (memastikan file tersebut ada dan sesuai format yaitu .txt) dengan menggunakan fungsi `Input.isFileExist()`. Setelah itu, file akan dibaca dan diekstrak informasi dimensi papan (N, M) dan jumlah blok (P) dari baris pertama, serta memvalidasi bahwa baris kedua berisi kata "DEFAULT" menggunakan fungsi `Input.readInput(String fileName)`. Lalu, papan permainan (board) diinisialisasi dan setiap bentuk blok dibaca, dikonversi menjadi matriks, dan disimpan sebagai objek Block.
2. Setiap objek Block yang terbentuk dari string input diproses untuk menghasilkan berbagai transformasi (rotasi 90°, 180°, 270° dan pencerminan beserta rotasinya). Transformasi ini dihasilkan oleh metode `generateTransformations(char[][] baseShape)` di dalam kelas Block. Oleh karena itu, setiap blok memiliki beberapa kemungkinan orientasi yang nantinya akan diuji dalam proses pencarian solusi.
3. Setelah itu, proses pencarian solusi dimulai dengan pemanggilan metode `Solve.solve(0)`. Pada setiap langkah rekursif, algoritma mencoba menempatkan blok yang sedang diproses pada setiap posisi yang mungkin di papan. Fungsi `canPlace(char[][] shape, int x, int y)` digunakan untuk mengecek apakah blok dalam bentuk tertentu dapat ditempatkan di posisi (x, y) tanpa menabrak batas papan atau blok lain yang sudah ditempatkan. Jika blok dapat ditempatkan, fungsi `placeBlock(char[][] shape, int x, int y, char symbol)` menempatkan blok tersebut ke papan. Proses rekursif dilanjutkan untuk blok selanjutnya. Bila penempatan tidak menghasilkan solusi akhir (papan penuh), fungsi `removeBlock(char[][] shape, int x, int y)` menghapus dan mencoba penempatan lain. Pengecekan apakah papan sudah penuh (solusi ditemukan) dilakukan oleh fungsi `isBoardFull()`.
4. Lalu apabila solusi berhasil ditemukan (seluruh blok berhasil ditempatkan sehingga papan terisi penuh), program menampilkan solusi di terminal dengan pewarnaan melalui fungsi `Output.printBoard()`. Program juga mencatat waktu eksekusi dan jumlah iterasi yang telah dilakukan selama proses pencarian solusi. Pada akhir eksekusi, pengguna diberikan opsi untuk menyimpan solusi ke dalam file teks (menggunakan `Output.outputToText()`) maupun sebagai gambar (menggunakan `Output.outputToImage()`). Kedua metode ini meminta input nama file dari pengguna dan melakukan penyimpanan output sesuai format yang diinginkan.
5. Namun, jika tidak ditemukan solusi yang memenuhi syarat (papan tidak bisa terisi penuh), program akan menampilkan pesan yang menginformasikan kegagalan pencarian solusi dan mengakhiri proses.

BAB II

IMPLEMENTASI ALGORITMA DALAM BAHASA JAVA

2.1. File Input.java

- **Atribut**

Atribut	Deskripsi
public static int N, M, P	N : Menyimpan jumlah baris papan permainan. M : Menyimpan jumlah kolom papan permainan. P : Menyimpan jumlah blok yang akan digunakan dalam permainan.
public static char[][] board	Inisialisasi Matriks 2D yang merepresentasikan papan permainan.
public static final List<Block> blocks = new ArrayList<>()	Daftar untuk menyimpan objek Block yang telah diproses dari file input.

- **Konstruktor**

Tidak terdapat konstruktor.

- **Metode**

Metode	Deskripsi
public static boolean isFileExist(String fileName)	Fungsi untuk mengecek apakah file dengan nama tersebut sudah pernah ada sebelumnya.
public static void readInput(String fileName) throws IOException	Fungsi untuk membuka dan membaca input berupa file.txt.

2.2. File Block.java

- **Atribut**

Atribut	Deskripsi
char symbol	Menyimpan karakter simbol yang merepresentasikan blok.
List<char[][]> shapes	Daftar yang menyimpan berbagai bentuk transformasi dari blok ini (rotasi dan cerminan).

- **Konstruktor**

Konstruktor	Deskripsi
Block(String shape)	Konstruktor untuk membuat objek Block, mengambil bentuk awal dalam bentuk string, mengonversinya menjadi matriks, dan menghasilkan transformasi rotasi dan cerminan.

- **Metode**

Metode	Deskripsi
private char[][] convertToMatrix(String shape)	Mengonversi string bentuk blok menjadi matriks 2D karakter.
private void generateTransformations(char[][] baseShape)	Menghasilkan semua kemungkinan transformasi dari bentuk dasar blok (rotasi dan pencerminan) dan menyimpannya dalam daftar shapes.
private char[][] rotate(char[][] shape)	Melakukan rotasi 90 derajat terhadap matriks bentuk blok.
private char[][] mirror(char[][] shape)	Membalik (mencerminkan) bentuk blok secara vertikal.
List<char[][]> getAllShapes()	Mengembalikan daftar semua bentuk transformasi blok.

2.3. File Solve.java

- **Atribut**

Atribut	Deskripsi
private static int iterationCount = 0	Menghitung jumlah iterasi yang dilakukan dalam proses pencarian solusi.

- **Konstruktor**

Tidak terdapat konstruktor.

- **Metode**

Metode	Deskripsi
public static boolean solve(int index)	Metode rekursif yang mencoba menempatkan semua blok dalam daftar Input.blocks ke papan permainan dengan backtracking.

private static boolean canPlace(char[][] shape, int x, int y)	Mengecek apakah suatu blok dapat ditempatkan pada posisi (x, y) di papan permainan tanpa menabrak batas atau blok lain.
private static void placeBlock(char[][] shape, int x, int y, char symbol)	Menempatkan blok ke dalam papan permainan pada posisi (x, y) dengan simbol tertentu.
private static void removeBlock(char[][] shape, int x, int y)	Menghapus blok yang telah ditempatkan dari papan permainan untuk keperluan backtracking.
private static boolean isBoardFull()	Mengecek apakah papan permainan sudah penuh, yang berarti semua blok telah berhasil ditempatkan.
public static int getIterationCount()	Mengembalikan jumlah iterasi yang telah dilakukan dalam proses penyelesaian.

2.4. File Output.java

- **Atribut**

Atribut	Deskripsi
private static final String RESET	Kode ANSI untuk mengatur ulang warna teks di terminal.
private static final String[] TEXT_COLORS	Array warna ANSI yang digunakan untuk menampilkan teks berwarna di terminal.
private static final String[] IMAGE_COLORS	Array warna dalam format heksadesimal yang digunakan untuk warna blok pada gambar output.
private static final Map<Character, String> textColorMap	Peta (map) yang menyimpan warna teks untuk setiap karakter blok di terminal.
private static final Map<Character, Color> imageColorMap	Peta (map) yang menyimpan warna blok dalam gambar berdasarkan karakter blok.

- **Konstruktor**

Tidak terdapat konstruktor.

- **Metode**

Metode	Deskripsi
private static void assignTextColors()	Mengatur warna teks untuk setiap blok berdasarkan TEXT_COLORS.
private static void assignImageColors()	Mengatur warna gambar untuk setiap blok berdasarkan IMAGE_COLORS.
public static void printBoard()	Mencetak papan permainan ke terminal dengan warna berdasarkan blok yang ditempatkan.
public static void outputToText() throws IOException	Menyimpan papan permainan ke dalam file teks (.txt) setelah meminta nama file dari pengguna.
public static void outputToImage()	Menyimpan papan permainan sebagai gambar .png setelah meminta nama file dari pengguna.

2.5. File Main.java

- **Atribut**

Atribut	Deskripsi
public static final Scanner sc	Scanner global untuk membaca input dari pengguna.

- **Konstruktor**

Tidak terdapat konstruktor.

- **Metode**

Metode	Deskripsi
public static void main(String[] args)	Driver utama yang menjalankan program IQ PUZZLER PRO Solver.

BAB III

SOURCE CODE PROGRAM

3.1. Repositori Github

Repositori program dapat diakses melalui tautan GitHub berikut :

https://github.com/shanlie20/Tucil1_13523019

3.2. Source Code Program

3.2.1. File Input.java

```
1  import java.io.*;
2  import java.util.*;
3
4  class Input {
5      public static int N, M, P;
6      public static char[][] board;
7      public static final List<Block> blocks = new ArrayList<>();
8
9      public static boolean isFileExist(String fileName) {
10         File file = new File(fileName);
11         return file.exists() && file.isFile();
12     }
13
14     public static void readInput(String fileName) throws IOException {
15         try (BufferedReader br = new BufferedReader(new FileReader(fileName))) {
16             String[] dimensions = br.readLine().split(regex:" ");
17             if (dimensions.length < 3) throw new IOException(message:"Format file tidak valid.");
18
19             N = Integer.parseInt(dimensions[0]);
20             M = Integer.parseInt(dimensions[1]);
21             P = Integer.parseInt(dimensions[2]);
22
23             String formatCheck = br.readLine();
24             if (!"DEFAULT".equals(formatCheck)) {
25                 throw new IOException(message:"Format file tidak valid: Baris kedua harus berisi DEFAULT.");
26             }
27
28             board = new char[N][M];
29             for (char[] row : board) Arrays.fill(row, val:'.');
30
31             List<String> blockShapes = new ArrayList<>();
32             String line;
33             while ((line = br.readLine()) != null) {
34                 if (line.isEmpty()) continue;
35
36                 if (blockShapes.isEmpty() || blockShapes.get(blockShapes.size() - 1).charAt(index:0) != line.charAt(index:0)) {
37                     blockShapes.add(line);
38                 } else {
39                     blockShapes.set(blockShapes.size() - 1, blockShapes.get(blockShapes.size() - 1) + " " + line);
40                 }
41             }
42
43             P = blockShapes.size();
44             for (String shape : blockShapes) {
45                 blocks.add(new Block(shape));
46             }
47
48             System.out.println(x:"File input berhasil dibaca");
49         } catch (IOException | NumberFormatException e) {
50             System.out.println("Gagal membaca file input: " + e.getMessage());
51             System.exit(status:0);
52         }
53     }
54 }
```

3.2.2. File Block.java

```
1  import java.util.*;
2
3  class Block {
4      char symbol;
5      List<char[][]> shapes = new ArrayList<>();
6
7      Block(String shape) {
8          this.symbol = shape.charAt(index:0);
9          char[][] baseShape = convertToMatrix(shape);
10         generateTransformations(baseShape);
11     }
12
13     private char[][] convertToMatrix(String shape) {
14         String[] rows = shape.split(regex:" ");
15         int rowCount = rows.length;
16         int colCount = 0;
17
18         for (String row : rows) {
19             colCount = Math.max(colCount, row.length());
20         }
21
22         char[][] matrix = new char[rowCount][colCount];
23
24         for (int i = 0; i < rowCount; i++) {
25             Arrays.fill(matrix[i], val:'.');
26             for (int j = 0; j < rows[i].length(); j++) {
27                 matrix[i][j] = rows[i].charAt(j);
28             }
29         }
30         return matrix;
31     }
32
33     private void generateTransformations(char[][] baseShape) {
34         shapes.add(baseShape);
35         shapes.add(rotate(baseShape));
36         shapes.add(rotate(rotate(baseShape)));
37         shapes.add(rotate(rotate(rotate(baseShape))));
38         shapes.add(mirror(baseShape));
39         shapes.add(rotate(mirror(baseShape)));
40         shapes.add(rotate(rotate(mirror(baseShape))));
41         shapes.add(rotate(rotate(rotate(mirror(baseShape)))));
42     }
43
44     private char[][] rotate(char[][] shape) {
45         int rows = shape.length;
46         int cols = 0;
47
48         for (char[] row : shape) {
49             cols = Math.max(cols, row.length);
50         }
51
52         char[][] rotated = new char[cols][rows];
53         for (int i = 0; i < rows; i++) {
54             for (int j = 0; j < shape[i].length; j++) {
55                 rotated[j][rows - 1 - i] = shape[i][j];
56             }
57         }
58         return rotated;
59     }
60 }
```

```

61     private char[][] mirror(char[][] shape) {
62         int rows = shape.length;
63         char[][] mirrored = new char[rows][];
64
65         for (int i = 0; i < rows; i++) {
66             mirrored[i] = shape[rows - 1 - i].clone();
67         }
68         return mirrored;
69     }
70
71     List<char[][]> getAllShapes() {
72         return shapes;
73     }
74 }
75

```

3.2.3. File Solve.java

```

1  public class Solve {
2      private static int iterationCount = 0;
3
4      public static boolean solve(int index) {
5          if (index == Input.blocks.size()) return isBoardFull();
6          Block block = Input.blocks.get(index);
7
8          for (char[][] shape : block.getAllShapes()) {
9              for (int i = 0; i < Input.N; i++) {
10                 for (int j = 0; j < Input.M; j++) {
11                     iterationCount++;
12                     if (canPlace(shape, i, j)) {
13                         placeBlock(shape, i, j, block.symbol);
14                         if (solve(index + 1)) return true;
15                         removeBlock(shape, i, j);
16                     }
17                 }
18             }
19         }
20         return false;
21     }
22
23     private static boolean canPlace(char[][] shape, int x, int y) {
24         for (int i = 0; i < shape.length; i++) {
25             for (int j = 0; j < shape[0].length; j++) {
26                 if (shape[i][j] != '.' && (x + i >= Input.N || y + j >= Input.M || Input.board[x + i][y + j] != '.')) {
27                     return false;
28                 }
29             }
30         }
31         return true;
32     }
33 }

```

```

34     private static void placeBlock(char[][] shape, int x, int y, char symbol) {
35         for (int i = 0; i < shape.length; i++) {
36             for (int j = 0; j < shape[0].length; j++) {
37                 if (shape[i][j] != '.') {
38                     Input.board[x + i][y + j] = symbol;
39                 }
40             }
41         }
42     }
43
44     private static void removeBlock(char[][] shape, int x, int y) {
45         for (int i = 0; i < shape.length; i++) {
46             for (int j = 0; j < shape[0].length; j++) {
47                 if (shape[i][j] != '.') {
48                     Input.board[x + i][y + j] = '.';
49                 }
50             }
51         }
52     }
53
54     private static boolean isBoardFull() {
55         for (char[] row : Input.board) {
56             for (char cell : row) {
57                 if (cell == '.') return false;
58             }
59         }
60         return true;
61     }
62
63     public static int getIterationCount() {
64         return iterationCount;
65     }
66 }

```

3.2.4. File Output.java

```

1  import java.awt.*;
2  import java.awt.image.BufferedImage;
3  import java.io.*;
4  import java.util.HashMap;
5  import java.util.Map;
6  import javax.imageio.ImageIO;
7
8  class Output {
9      private static final String RESET = "\u001B[0m";
10     private static final String[] TEXT_COLORS = {
11         "\u001B[31m", "\u001B[32m", "\u001B[33m", "\u001B[34m",
12         "\u001B[35m", "\u001B[36m", "\u001B[91m", "\u001B[92m",
13         "\u001B[93m", "\u001B[94m", "\u001B[95m", "\u001B[96m",
14         "\u001B[97m"
15     };
16
17     private static final String[] IMAGE_COLORS = {
18         "#FF0000", "#00FF00", "#0000FF", "#FFFF00", "#FF00FF", "#00FFFF",
19         "#FFA500", "#800080", "#008000", "#808000", "#800000", "#008080",
20         "#000080"
21     };
22
23     private static final Map<Character, String> textColorMap = new HashMap<>();
24     private static final Map<Character, Color> imageColorMap = new HashMap<>();
25
26     private static void assignTextColors() {
27         int index = 0;
28         for (char[] row : Input.board) {
29             for (char cell : row) {
30                 if (cell != '.' && !textColorMap.containsKey(cell)) {
31                     textColorMap.put(cell, TEXT_COLORS[index % TEXT_COLORS.length]);
32                     index++;
33                 }
34             }
35         }
36     }
37 }

```

```

38     private static void assignImageColors() {
39         int index = 0;
40         for (char[] row : Input.board) {
41             for (char cell : row) {
42                 if (cell != '.' && !imageColorMap.containsKey(cell)) {
43                     imageColorMap.put(cell, Color.decode(IMAGE_COLORS[index % IMAGE_COLORS.length]));
44                     index++;
45                 }
46             }
47         }
48     }
49
50     public static void printBoard() {
51         assignTextColors();
52         for (char[] row : Input.board) {
53             for (char cell : row) {
54                 if (cell == '.') {
55                     System.out.print(cell);
56                 } else {
57                     System.out.print(textColorMap.get(cell) + cell + RESET);
58                 }
59             }
60             System.out.println();
61         }
62     }

```

```

63
64     public static void outputToText() throws IOException {
65         String outputTextFileName, outputTextFilePath;
66         System.out.print(s: "Apakah kamu mau menyimpan solusi dalam teks?? (ya/tidak): ");
67         String option = Main.sc.nextLine().trim().toLowerCase();
68         if (!option.equals(anObject: "ya")) return;
69
70         while (true) {
71             System.out.print(s: "Filanya mau disimpan dengan nama apaa? (berformat .txt): ");
72             outputTextFileName = Main.sc.nextLine().trim();
73             if (outputTextFileName.isEmpty()) {
74                 System.out.println(x: "Nama file ga boleh kosong! Coba lagi yaa~");
75                 continue;
76             }
77             if (!outputTextFileName.toLowerCase().endsWith(suffix: ".txt")) {
78                 System.out.println(x: "Nama file harus diakhiri dengan '.txt' nii! Coba lagi yaa~");
79                 continue;
80             }
81             outputTextFilePath = "test/output/text/" + outputTextFileName;
82             if (Input.isFileExist(outputTextFilePath)) {
83                 System.out.println(x: "File yang nama itu udah ada nii. Coba masukkin nama lain yaa~");
84                 continue;
85             }
86             break;
87         }
88         try (PrintWriter writer = new PrintWriter(new FileWriter(outputTextFilePath))) {
89             for (char[] row : Input.board) {
90                 for (char cell : row) {
91                     writer.print(cell);
92                 }
93                 writer.println();
94             }
95             System.out.println("Solusi udah disimpan di " + outputTextFilePath);
96             System.out.println();
97         } catch (IOException e) {
98             System.err.println("Gagal menulis ke dalam file! " + e.getMessage());
99         }
100     }

```

```

101
102     public static void outputToImage() {
103         String outputImageFileName, outputImageFilePath;
104         System.out.print(s:"Apakah kamu mau menyimpan solusi sebagai gambar?? (ya/tidak): ");
105         String option = Main.sc.nextLine().trim().toLowerCase();
106         if (!option.equals(anObject:"ya")) return;
107
108         while (true) {
109             System.out.print(s:"Filanya mau disimpan dengan nama apaa? (berformat .png): ");
110             outputImageFileName = Main.sc.nextLine().trim();
111             if (outputImageFileName.isEmpty()) {
112                 System.out.println(x:"Nama file ga boleh kosong! Coba lagi yaa~");
113                 continue;
114             }
115             if (!outputImageFileName.toLowerCase().endsWith(suffix:".png")) {
116                 System.out.println(x:"Nama file harus diakhiri dengan '.png'niii! Coba lagi yaa~");
117                 continue;
118             }
119             outputImageFilePath = "test/output/image/" + outputImageFileName;
120             if (Input.isFileExist(outputImageFilePath)) {
121                 System.out.println(x:"File yang nama itu udah ada nii. Coba masukkin nama lain yaa~");
122                 continue;
123             }
124             break;
125         }
126

```

```

127         try {
128             int rows = Input.board.length;
129             int cols = Input.board[0].length;
130             int cellSize = 50;
131             int width = cols * cellSize;
132             int height = rows * cellSize;
133
134             BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_ARGB);
135             Graphics2D g = image.createGraphics();
136
137             g.setColor(Color.WHITE);
138             g.fillRect(x:0, y:0, width, height);
139             g.setStroke(new BasicStroke(width:2));
140
141             assignImageColors();
142
143             for (int i = 0; i < rows; i++) {
144                 for (int j = 0; j < cols; j++) {
145                     char cell = Input.board[i][j];
146                     if (cell != '.') {
147                         g.setColor(imageColorMap.get(cell));
148                         g.fillRect(j * cellSize, i * cellSize, cellSize, cellSize);
149                         g.setColor(Color.BLACK);
150                         g.drawRect(j * cellSize, i * cellSize, cellSize, cellSize);
151                         g.drawString(String.valueOf(cell), j * cellSize + 20, i * cellSize + 30);
152                     }
153                 }
154             }
155
156             g.dispose();
157             File outputFile = new File(outputImageFilePath);
158             ImageIO.write(image, formatName:"png", outputFile);
159             System.out.println("Solusi udah disimpan di " + outputFile);
160             System.out.println();
161         } catch (IOException e) {
162             System.err.println("Terjadi kesalahan saat menyimpan gambar: " + e.getMessage());
163         }
164     }
165 }

```

3.2.5. File Main.java

```
1  import java.io.*;
2  import java.util.Scanner;
3
4  public class Main {
5      public static final Scanner sc = new Scanner(System.in);
6      public static void main(String[] args) {
7          String inputFileName, inputFilePath;
8
9          System.out.println();
10         System.out.println(x:"----- Selamat datang di program IQ PUZZLER PRO Solver! -----");
11         System.out.println();
12         System.out.println(x:"IQ PUZZLER PRO");
13         System.out.println(x:"");
14         System.out.println(x:"");
15         System.out.println(x:"");
16         System.out.println(x:"");
17         System.out.println(x:"");
18         System.out.println(x:"-----");
19         System.out.println(x:"Shannon Aurellius Anastasya Lie");
20         System.out.println(x:"13523019 - K-01");
21         System.out.println(x:"-----");
22         System.out.println();
23
24         while (true) {
25             System.out.print(s:"Yukk masukkan nama file inputnyaa (berformat .txt): ");
26             inputFileName = sc.nextLine().trim();
27
28             if (inputFileName.isEmpty()) {
29                 System.out.println(x:"Nama file ga boleh kosong! Coba lagi yaa~");
30                 continue;
31             }
32             if (!inputFileName.toLowerCase().endsWith(suffix:".txt")) {
33                 System.out.println(x:"Nama file harus diakhiri dengan '.txt' niiii! Coba lagi yaa~");
34                 continue;
35             }
36             inputFilePath = "test/input/" + inputFileName;
37             if (!Input.isFileExist(inputFilePath)) {
38                 System.out.println(x:"File ga ditemukan nii! Coba lagi yaa~");
39                 continue;
40             }
41             break;
42         }
43
44         try {
45             Input.readInput(inputFilePath);
46             long startTime = System.currentTimeMillis();
47             boolean solved = Solve.solve(index:0);
48             long endTime = System.currentTimeMillis();
49
50             if (solved) {
51                 System.out.println();
52                 System.out.println(x:"Solusinya ketemu nii:");
53                 Output.printBoard();
54                 System.out.println();
55             } else {
56                 System.out.println(x:"Ga ketemu solusinya nii.");
57             }
58
59             System.out.println("Waktu Eksekusi: " + (endTime - startTime) + " ms");
60             System.out.println();
61             System.out.println("Banyak kasus yang ditinjau: " + Solve.getIterationCount());
62             System.out.println();
63
64             Output.outputToText();
65             Output.outputToImage();
66
67             System.out.println(x:"Terima kasih udah pake program ini! See you di lain kesempatan yaa~");
68         } catch (IOException e) {
69             System.err.println("Gagal membaca file: " + e.getMessage());
70         }
71     }
72 }
73
```

BAB IV

MASUKAN DAN LUARAN PROGRAM

4.1. Test Case 1

Isi case1.txt

```
test > input > case1.txt
1 5 5 8
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 pertama saat program baru dijalankan dan masukkan input nama file

```
----- Selamat datang di program IQ PUZZLER PRO Solver! -----
IQ PUZZLER PRO
-----
Shannon Aurellius Anastasya Lie
13523019 - K-01
-----
Yukk masukkan nama file inputnyaa (berformat .txt): case1.txt
```


Output penyelesaian program case1.txt pada terminal

```
Yukk masukkan nama file inputnyaa (berformat .txt): case1.txt
File input berhasil dibaca!

Solusinya ketemu nii:
AGGGD
AABDD
CCBBE
CFFEE
FFFE

Waktu Eksekusi: 100 ms


Banyak kasus yang ditinjau: 5480381

Apakah kamu mau menyimpan solusi dalam teks?? (ya/tidak): ya
Filenya mau disimpan dengan nama apaa? (berformat .txt): solusiCase1.txt
Solusi udah disimpan di test/output/text/solusiCase1.txt


Apakah kamu mau menyimpan solusi sebagai gambar?? (ya/tidak): ya
Filenya mau disimpan dengan nama apaa? (berformat .png): solusiCase1.png
Solusi udah disimpan di test/output/image/solusiCase1.png

Terima kasih udah pake program ini! See you di lain kesempatan yaa~
```

Output solusi program case1.txt berupa text yang telah disimpan pada file solusiCase1.txt

```
test > output > text >  solusiCase1.txt
1  AGGGD
2  AABDD
3  CCBBE
4  CFFEE
5  FFFEE
```

Output solusi program case1.txt berupa image yang telah disimpan pada file solusiCase1.png

```
test > output > image >  solusiCase1.png
```

A	G	G	G	D
A	A	B	D	D
C	C	B	B	E
C	F	F	E	E
F	F	F	E	E

4.2. Test Case 2

Isi case2.txt

```
test > input > case2.txt
1 6 6 7
2 DEFAULT
3 A
4 AAAA
5 B
6 BBBB
7 C
8 CCCC
9 D
10 DDDD
11 E
12 EEEE
13 F
14 FFFF
15 GGGGGG
```

Tampilan pertama saat program baru dijalankan dan masukkan input nama file

```
----- Selamat datang di program IQ PUZZLER PRO Solver! -----
IQ PUZZLER PRO
-----
Shannon Aurellius Anastasya Lie
13523019 - K-01
-----
Yukk masukkan nama file inputnyaa (berformat .txt): case2.txt
```

Output penyelesaian program case2.txt pada terminal

```
Yukk masukkan nama file inputnyaa (berformat .txt): case2.txt
File input berhasil dibaca!

Solusinya ketemu nii:
ADDDDG
AAAADG
BEEEEG
BBBBEG
CFFFFG
CCCCFG

Waktu Eksekusi: 0 ms


Banyak kasus yang ditinjau: 3795

Apakah kamu mau menyimpan solusi dalam teks?? (ya/tidak): ya
Filanya mau disimpan dengan nama apaa? (berformat .txt): solusiCase2.txt
Solusi udah disimpan di test/output/text/solusiCase2.txt

Apakah kamu mau menyimpan solusi sebagai gambar?? (ya/tidak): ya
Filanya mau disimpan dengan nama apaa? (berformat .png): solusiCase2.png
Solusi udah disimpan di test\output\image\solusiCase2.png

Terima kasih udah pake program ini! See you di lain kesempatan yaa~
```

Output solusi program case2.txt berupa text yang telah disimpan pada file solusiCase2.txt

```
test > output > text >  solusiCase2.txt
1  ADDDDG
2  AAAADG
3  BEEEEG
4  BBBBEG
5  CFFFFG
6  CCCCCG
```

Output solusi program case2.txt berupa image yang telah disimpan pada file solusiCase2.png



4.3. Test Case 3

Isi case3.txt (Uji kasus yang tidak valid)

```
test > input > case3.txt
1 6 6 3
2 DEFAULT
3 A
4 AA
5 B
6 BB
7 C
8 CC
```

Tampilan pertama saat program baru dijalankan dan masukkan input nama file

```
----- Selamat datang di program IQ PUZZLER PRO Solver! -----
IQ PUZZLER PRO
-----
Shannon Aurellius Anastasya Lie
13523019 - K-01
-----
Yukk masukkan nama file inputnyaa (berformat .txt): case3.txt
```

Output penyelesaian program case3.txt pada terminal

```
Yukk masukkan nama file inputnyaa (berformat .txt): case3.txt
File input berhasil dibaca!
Ga ketemu solusinya nii.
Waktu Eksekusi: 320 ms

Banyak kasus yang ditinjau: 9163296

Apakah kamu mau menyimpan solusi dalam teks?? (ya/tidak): tidak
Apakah kamu mau menyimpan solusi sebagai gambar?? (ya/tidak): tidak
Terima kasih udah pake program ini! See you di lain kesempatan yaa~
```

4.4. Test Case 4

Isi case4.txt

```
test > input > case4.txt
1 4 3 4
2 DEFAULT
3 A
4 AA
5 B
6 BB
7 C
8 CC
9 D
10 DD
```

Tampilan pertama saat program baru dijalankan dan masukkan input nama file

```
----- Selamat datang di program IQ PUZZLER PRO Solver! -----
IQ PUZZLER PRO
-----
Shannon Aurellius Anastasya Lie
13523019 - K-01
-----
Yukk masukkan nama file inputnyaa (berformat .txt): case4.txt
```

Output penyelesaian program case4.txt pada terminal

```
Yukk masukkan nama file inputnyaa (berformat .txt): case4.txt
File input berhasil dibaca!

Solusinya ketemu nii:
ACC
AAC
BDD
BBD

Waktu Eksekusi: 4 ms


Banyak kasus yang ditinjau: 66

Apakah kamu mau menyimpan solusi dalam teks?? (ya/tidak): ya
Filanya mau disimpan dengan nama apaa? (berformat .txt): solusiCase4.txt
Solusi udah disimpan di test/output/text/solusiCase4.txt

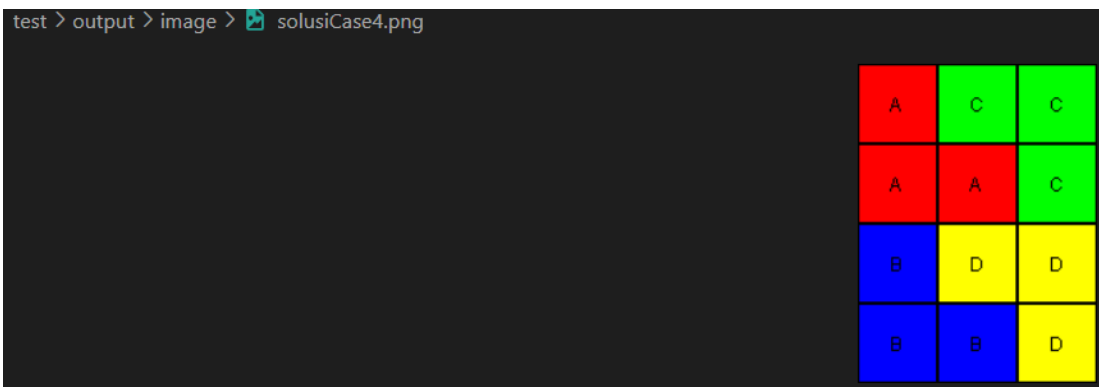
Apakah kamu mau menyimpan solusi sebagai gambar?? (ya/tidak): ya
Filanya mau disimpan dengan nama apaa? (berformat .png): solusiCase4.png
Solusi udah disimpan di test/output/image/solusiCase4.png

Terima kasih udah pake program ini! See you di lain kesempatan yaa~
```

Output solusi program case4.txt berupa text yang telah disimpan pada file solusiCase4.txt


```
test > output > text >  solusiCase4.txt  
1 ACC  
2 AAC  
3 BDD  
4 BBD
```

Output solusi program case4.txt berupa image yang telah disimpan pada file solusiCase4.png



4.5. Test Case 5

Isi case5.txt

```
test > input >  case5.txt  
1 5 4 7  
2 DEFAULT  
3 AA  
4 AA  
5 BB  
6 C  
7 CC  
8 D  
9 EE  
10 FFFF  
11 GGG  
12 G
```


Tampilan pertama saat program baru dijalankan dan masukkan input nama file

```
----- Selamat datang di program IQ PUZZLER PRO Solver! -----  
IQ PUZZLER PRO  
-----  
Shannon Aurellius Anastasya Lie  
13523019 - K-01  
-----  
Yukk masukkan nama file inputnyaa (berformat .txt): case5.txt
```

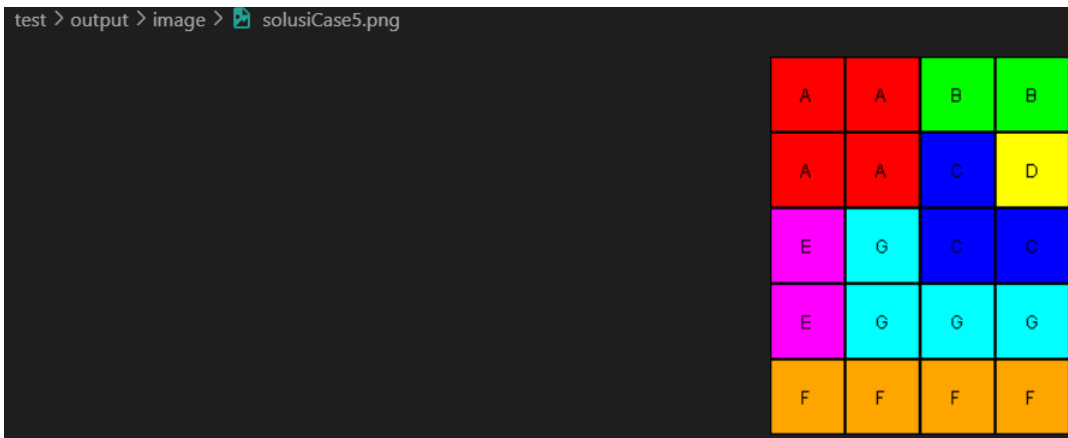
Output penyelesaian program case5.txt pada terminal

```
Yukk masukkan nama file inputnyaa (berformat .txt): case5.txt  
File input berhasil dibaca!  
  
Solusinya ketemu nii:  
AABB  
AACD  
EGCC  
EGGG  
FFFF  
  
Waktu Eksekusi: 0 ms  
  
Banyak kasus yang ditinjau: 6395  
  
Apakah kamu mau menyimpan solusi dalam teks?? (ya/tidak): ya  
Filanya mau disimpan dengan nama apaa? (berformat .txt): solusiCase5.txt  
Solusi udah disimpan di test/output/text/solusiCase5.txt  
  
Apakah kamu mau menyimpan solusi sebagai gambar?? (ya/tidak): ya  
Filanya mau disimpan dengan nama apaa? (berformat .png): solusiCase5.png  
Solusi udah disimpan di test/output/image/solusiCase5.png  
  
Terima kasih udah pake program ini! See you di lain kesempatan yaa~
```

Output solusi program case5.txt berupa text yang telah disimpan pada file solusiCase5.txt

```
test > output > text >  solusiCase5.txt  
1   AABB  
2   AACD  
3   EGCC  
4   EGGG  
5   FFFF
```

Output solusi program case5.txt berupa image yang telah disimpan pada file solusiCase5.png

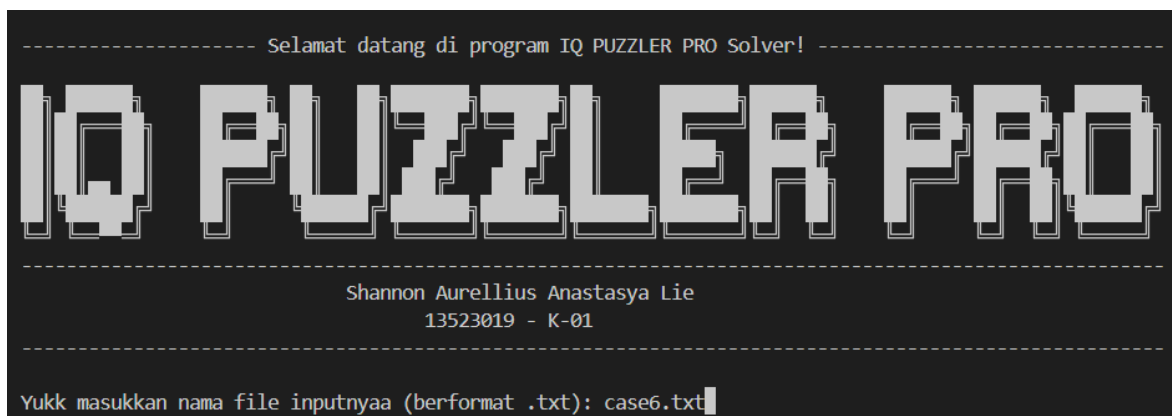


4.6. Test Case 6

Isi case6.txt

```
test > input > case6.txt
1 4 6 6
2 DEFAULT
3 AAA
4 A
5 BB
6 BB
7 CC
8 DD
9 EEE
10 E
11 E
12 FFFFFF
13 F
```

Tampilan pertama saat program baru dijalankan dan masukkan input nama file



Output penyelesaian program case6.txt pada terminal

```
Yukk masukkan nama file inputnyaa (berformat .txt): case6.txt
File input berhasil dibaca!

Solusinya ketemu nii:
AAABBE
ACCBBE
FDDEEE
FFFFFF

Waktu Eksekusi: 0 ms


Banyak kasus yang ditinjau: 1148

Apakah kamu mau menyimpan solusi dalam teks?? (ya/tidak): ya
Filenya mau disimpan dengan nama apaa? (berformat .txt): solusiCase6.txt
Solusi udah disimpan di test/output/text/solusiCase6.txt


Apakah kamu mau menyimpan solusi sebagai gambar?? (ya/tidak): ya
Filenya mau disimpan dengan nama apaa? (berformat .png): solusiCase6.png
Solusi udah disimpan di test/output/image/solusiCase6.png

Terima kasih udah pake program ini! See you di lain kesempatan yaa~
```

Output solusi program case6.txt berupa text yang telah disimpan pada file solusiCase6.txt

```
test > output > text >  solusiCase6.txt
1  AAABBE
2  ACCBBE
3  FDDEEE
4  FFFFFFF
```

Output solusi program case6.txt berupa image yang telah disimpan pada file solusiCase6.png

```
test > output > image >  solusiCase6.png
```

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

4.7. Test Case 7

Isi case7.txt

```
test > input > case7.txt
1 7 5 7
2 DEFAULT
3 AAAAA
4 A
5 A
6 A
7 A
8 BBBBB
9 B
10 CCC
11 DDDD
12 DD
13 EEE
14 E
15 FF
16 FF
17 GG
18 G
```

Tampilan pertama saat program baru dijalankan dan masukkan input nama file

```
----- Selamat datang di program IQ PUZZLER PRO Solver! -----
IQ PUZZLER PRO
-----
Shannon Aurellius Anastasya Lie
13523019 - K-01
-----
Yukk masukkan nama file inputnyaa (berformat .txt): case7.txt
```

Output penyelesaian program case7.txt pada terminal

Yukk masukkan nama file inputnyaa (berformat .txt): case7.txt
File input berhasil dibaca!

Solusinya ketemu nii:

AAAAA

ABBFF

ACBFF

ACBDD

ACBDD

GGBED

GEEED

Waktu Eksekusi: 121 ms

Banyak kasus yang ditinjau: 1990796

Apakah kamu mau menyimpan solusi dalam teks?? (ya/tidak): ya

Filenya mau disimpan dengan nama apaa? (berformat .txt): solusiCase7.txt

Solusi udah disimpan di test/output/text/solusiCase7.txt

Apakah kamu mau menyimpan solusi sebagai gambar?? (ya/tidak): ya

Filenya mau disimpan dengan nama apaa? (berformat .png): solusiCase7.png

Solusi udah disimpan di test\output\image\solusiCase7.png

Terima kasih udah pake program ini! See you di lain kesempatan yaa~

Output solusi program case7.txt berupa text yang telah disimpan pada file
solusiCase7.txt

test > output > text >  solusiCase7.txt

1 AAAAA

2 ABBFF

3 ACBFF

4 ACBDD


5 ACBDD

6 GGBED

7 GEEED

Output solusi program case7.txt berupa image yang telah disimpan pada file

solusiCase7.png

test > output > image >  solusiCase7.png

A	A	A	A	A
A	B	B	F	F
A	C	B	F	F
A	C	B	D	D
A	C	B	D	D
G	G	B	E	D
G	E	E	E	D

BAB V LAMPIRAN

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

DAFTAR PUSTAKA

- [1]<https://www.cloudeka.id/id/berita/web-sec/cara-kerja-algoritma-brute-force/>
- [2][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)
- [3]<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/Tucil1-Stima-2025.pdf>