

**Laporan Tugas Kecil 1**  
**IF2211 Strategi Algoritma 2024/2025**  
**Penyelesaian IQ Puzzler Pro dengan Algoritma Brute Force**



Disusun oleh:  
Varel Tiara (13523008)

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

2024

## DAFTAR PUSTAKA

<b>DAFTAR PUSTAKA</b>	<b>2</b>
<b>DAFTAR GAMBAR</b>	<b>2</b>
<b>BAB I</b>	<b>4</b>
<b>DESKRIPSI MASALAH</b>	<b>4</b>
<b>BAB II</b>	<b>5</b>
<b>ALGORITMA BRUTE FORCE</b>	<b>5</b>
2.1 Pendahuluan	5
2.2 Penjelasan Algoritma	5
2.2.1 Iterasi Semua Transformasi Blok	5
2.2.2 Penempatan Blok pada Papan	5
2.2.3 Rekursi dan Backtracking	6
2.2.4 Kondisi Berhenti	6
2.3 Pemanggilan Algoritma	6
2.4 Klarifikasi Algoritma	6
<b>BAB III</b>	<b>7</b>
<b>IMPLEMENTASI</b>	<b>7</b>
3.1 Spek Bonus yang Dikerjakan	7
3.2 Struktur Kode Utama	7
3.3 Algoritma Brute Force (Solver)	8
3.4 Block	10
3.5 Board	11
3.6 Color	12
3.7 InputHandler	13
3.8 OutputHandler	15
3.9 GUI	15
<b>BAB IV</b>	<b>16</b>
<b>EKSPERIMEN</b>	<b>16</b>
<b>BAB V</b>	<b>23</b>
<b>KESIMPULAN, SARAN, KOMENTAR, DAN REFLEKSI</b>	<b>23</b>
5.1 Kesimpulan	23
5.2 Refleksi	23
<b>BAB VI</b>	<b>24</b>
<b>LAMPIRAN</b>	<b>24</b>
6.1 Tautan Repository	24

# **BAB I**

## **DESKRIPSI MASALAH**

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 ini dimulai dengan adanya papan dengan dimensi tertentu yang kosong. Lalu, pemain dapat meletakkan blok puzzle sedemikian sehingga tidak ada blok yang bertumpang tindih. Setiap blok puzzle dapat dirotasikan maupun dicerminkan. Puzzle dinyatakan selesai jika dan hanya jika papan terisi penuh dan seluruh blok puzzle berhasil diletakkan.

Pada tugas kecil 1 mata kuliah Strategi Algoritma, hanya diminta untuk menemukan cukup satu solusi dari permainan IQ Puzzler Pro dengan menggunakan *algoritma Brute Force*, atau menampilkan bahwa solusi tidak ditemukan jika tidak ada solusi yang mungkin dari puzzle.

## **BAB II**

### **ALGORITMA BRUTE FORCE**

#### **2.1 Pendahuluan**

Untuk menyelesaikan permainan IQ Puzzler Pro, algoritma Brute Force dengan teknik rekursi dan *backtracking* digunakan. Algoritma ini akan mengeksplorasi semua kemungkinan konfigurasi penempatan blok pada papan, termasuk rotasi dan pencerminan blok, hingga mencapai solusi yang valid, yaitu ketika papan permainan terisi penuh oleh blok dan seluruh blok berhasil diletakkan pada papan permainan. Pendekatan ini memastikan bahwa tidak ada kemungkinan yang akan terlewat, meskipun memerlukan waktu komputasi yang signifikan untuk kasus kompleks atau tidak ada solusi.

#### **2.2 Penjelasan Algoritma**

Algoritma Brute Force akan berjalan secara sistematis dengan mencoba semua kombinasi penempatan blok pada papan permainan. Berikut adalah langkah-langkahnya:

##### **2.2.1 Iterasi Semua Transformasi Blok**

Setiap blok memiliki 8 kemungkinan transformasi (4 kali rotasi dan 2 kali pencerminan). Untuk setiap blok algoritma akan melakukan rotasi blok  $0^\circ$ ,  $90^\circ$ ,  $180^\circ$ ,  $270^\circ$ , pencerminan blok setelah setiap rotasi yang dilakukan.

##### **2.2.2 Penempatan Blok pada Papan**

Untuk setiap bentuk hasil transformasi, algoritmanya akan menghitung batas maksimal posisi penempatan pada papan berdasarkan ukuran blok, mengiterasi semua kemungkinan koordinat awal (baris dan kolom) di papan, dan memeriksa apakah blok dapat ditempatkan pada posisi tersebut tanpa bertabrakan atau tumpang tindih dengan blok lain atau keluar dari batas papan permainan.

### 2.2.3 Rekursi dan Backtracking

Jika blok dapat ditempatkan pada papan permainan maka blok tersebut akan ditandai sebagai “sudah digunakan”. Lalu algoritma akan memanggil dirinya sendiri (rekursi) untuk menempatkan blok berikutnya pada papan permainan. Jika rekursi gagal menemukan solusi, blok tersebut dihapus dari papan (backtracking), dan pencarian akan dilanjutkan ke konfigurasi blok berikutnya.

### 2.2.4 Kondisi Berhenti

Algoritma akan berhenti ketika semua blok berhasil ditempatkan dan papan terisi penuh (solusi ditemukan) dan tidak ada konfigurasi valid yang tersedia (tidak ada solusi).

## 2.3 Pemanggilan Algoritma

Proses utama dari program yang telah saya buat melalui fungsi **solve(int blockIndex)** pada kelas Solver. Pemanggilan pertama dilakukan dengan `blockIndex = 0`, yaitu proses akan dimulai dari blok pertama, lalu untuk setiap blok, algoritma akan mencoba semua transformasi dan posisi. Jika berhasil, rekursi dilanjutkan ke `blockIndex + 1`, yaitu blok selanjutnya. Jika `blockIndex` telah mencapai jumlah total blok ( $P$ ), pemeriksaan akan dilakukan untuk memastikan papan terisi penuh oleh blok. Contoh pemanggilan:

**`solver.solve(0);`**

## 2.4 Klarifikasi Algoritma

Algoritma menjamin solusi yang ditemukan adalah solusi yang valid pertama yang memenuhi syarat sehingga tidak mencari solusi dengan jumlah langkah yang minimal. Jika terdapat banyak solusi, algoritma akan mengembalikan solusi yang muncul pertama kali. Kompleksitas waktu eksponensial akibat kombinasi rotasi, pencerminan, dan penempatan blok. Untuk blok berjumlah  $P$ , kompleksitasnya akan mencapai  $O(8^P \times N \times M)$  dengan  $N$  dan  $M$  adalah dimensi dari papan.

## **BAB III**

### **IMPLEMENTASI**

#### **3.1 Spek Bonus yang Dikerjakan**

1. Output berupa Gambar (2 poin)  
Pengguna dapat menyimpan solusi suatu puzzle dalam bentuk file gambar. Setiap blok puzzle yang ditampilkan akan memiliki warna berbeda.
2. Graphical User Interface (8 poin)  
Di tugas kecil ini saya menggunakan JavaFX sebagai GUI untuk program yang telah saya buat. Interface ini dapat memvisualisasikan papan yang sudah terisi oleh blok puzzle berwarna.

#### **3.2 Struktur Kode Utama**

Kode yang telah saya buat terdiri dari beberapa kelas utama, yaitu:

- Block: Merepresentasikan blok dengan bentuk, rotasi, dan pencerminan.
- Board: Merepresentasikan papan permainan dan metode untuk menempatkan atau menghapus blok.
- Solver: Mengimplementasikan algoritma Brute Force dengan rekursi dan backtracking.
- App dan Controller: Menangani antarmuka pengguna (GUI) dan interaksi dari pengguna.

### 3.3 Algoritma Brute Force (Solver)

```
1 public boolean solve(int blockIndex) {
2     if (blockIndex == blocks.size()) {
3         return isBoardFullyFilled() && allBlocksPlaced();
4     }
5
6     Block block = blocks.get(blockIndex);
7
8     // iterasi dari orientasi blok
9     for (List<List<Character>> orientation : block.getOrientations()) {
10        block.setShape(orientation);
11        int blockRows = block.getRow();
12        int blockCols = block.getCol();
13        int maxRow = board.getRows() - blockRows;
14        int maxCol = board.getCols() - blockCols;
15
16        if (maxRow < 0 || maxCol < 0) continue;
17
18        for (int i = 0; i <= maxRow; i++) {
19            for (int j = 0; j <= maxCol; j++) {
20                iterationCount++;
21                if (board.canPlaceBlock(block, i, j)) {
22                    board.placeBlock(block, i, j, block.getSymbol());
23                    block.setPlaced(true);
24                    // rekursi ke next blok
25                    if (solve(blockIndex + 1)) {
26                        return true;
27                    }
28                    board.removeBlock(block, i, j);
29                    block.setPlaced(false);
30                }
31            }
32        }
33    }
34    return false;
35 }
```

Gambar 1. Fungsi Solver

Fungsi ini akan memeriksa apakah semua blok telah ditempatkan (`blockIndex == blocks.size()`). Jika ya, metode akan memverifikasi apakah papan terisi penuh dan semua blok telah digunakan. Jika kedua kondisi terpenuhi, solusi ditemukan, dan metode mengembalikan `true`. Jika tidak, algoritma melanjutkan dengan mencoba semua orientasi (rotasi dan pencerminan) dari blok saat ini. Untuk setiap orientasi, algoritma menghitung batas maksimal posisi penempatan blok pada papan dan mengiterasi semua koordinat yang valid. Pada setiap posisi, algoritma memeriksa apakah blok dapat ditempatkan tanpa melanggar aturan

menggunakan metode `canPlaceBlock()`. Jika penempatan valid, blok ditempatkan pada papan, dan algoritma melanjutkan ke blok berikutnya secara rekursif. Jika rekursi gagal menemukan solusi, algoritma melakukan backtracking dengan menghapus blok dari papan dan mencoba konfigurasi berikutnya. Proses ini diulang hingga semua kemungkinan penempatan blok telah dicoba atau solusi ditemukan. Meskipun algoritma ini menjamin solusi akan ditemukan jika ada, kompleksitas waktunya bersifat eksponensial karena mencoba semua kombinasi orientasi dan posisi blok. Untuk meningkatkan efisiensi, orientasi blok diprekomputasi, dan duplikasi dihilangkan untuk mengurangi jumlah iterasi yang tidak perlu.

```
1 private boolean isBoardFullyFilled() {
2     for (int i = 0; i < board.getRows(); i++) {
3         for (int j = 0; j < board.getCols(); j++) {
4             if (board.getGrid().get(i).get(j) == '.') {
5                 return false;
6             }
7         }
8     }
9     return true;
10 }
```

Gambar 2. Fungsi `isBoardFullyFilled`

Memeriksa apakah semua sel pada papan telah terisi oleh blok.

```
1 private boolean allBlocksPlaced() {
2     for (Block block : blocks) {
3         if (!block.isPlaced()) {
4             return false;
5         }
6     }
7     return true;
8 }
```

Gambar 3. Fungsi `allBlockPlaced`

Memeriksa apakah semua blok telah ditempatkan.



### 3.4 Block

```
1 private Character findSymbol(List<List<Character>> shape) {
2     for (List<Character> row : shape) {
3         for (Character c : row) {
4             if (c != '.') {
5                 return c;
6             }
7         }
8     }
9     return '.';
10 }
```

Gambar 4. Fungsi findSymbol

Mencari simbol yang mewakili blok dari bentuk (shape) yang diberikan.

```
1 private void generateOrientations() {
2     List<List<Character>> temp = new ArrayList<>(originalShape);
3
4     for (int rotation = 0; rotation < 4; rotation++) {
5         if (isUnique(temp)) {
6             orientations.add(new ArrayList<>(temp));
7         }
8         temp = rotate90(temp);
9     }
10
11     temp = flip(originalShape);
12     for (int rotation = 0; rotation < 4; rotation++) {
13         if (isUnique(temp)) {
14             orientations.add(new ArrayList<>(temp));
15         }
16         temp = rotate90(temp);
17     }
18 }
```

Gambar 5. Fungsi generateOrientations

Menghasilkan semua orientasi unik dari sebuah blok dengan melakukan rotasi dan pencerminan.

```
1 public boolean isPlaced() {
2     return isPlaced;
3 }
```

Gambar 6. Fungsi isPlaced

Mengembalikan status apakah blok telah ditempatkan pada papan atau belum.

```
1 public void setPlaced(boolean placed) {
2     this.isPlaced = placed;
3 }
```

Gambar 7. Fungsi setPlaced

Mengubah status penempatan blok.

```

1 private List<List<Character>> flip(List<List<Character>> shape) {
2     List<List<Character>> flippedShape = new ArrayList<>();
3
4     for (int i = shape.size() - 1; i >= 0; i--) {
5         flippedShape.add(new ArrayList<>(shape.get(i)));
6     }
7     return flippedShape;
8 }

```

Gambar 8. Fungsi flip

Mencerminkan bentuk balok secara vertikal dengan membalik urutan baris.

```

1 private List<List<Character>> rotate90(List<List<Character>> shape) {
2     int maxCol = shape.get(0).size();
3     int rowCount = shape.size();
4     List<List<Character>> rotated = new ArrayList<>();
5
6     for (int i = 0; i < maxCol; i++) {
7         List<Character> newRow = new ArrayList<>();
8         for (int j = rowCount - 1; j >= 0; j--) {
9             if (1 < shape.get(j).size()) {
10                 newRow.add(shape.get(j).get(i));
11             } else {
12                 newRow.add('.');
13             }
14         }
15         rotated.add(newRow);
16     }
17     return rotated;
18 }

```

Gambar 9. Fungsi rotate90

Memutar bentuk balok sebesar 90° searah jarum jam.

```

1 private boolean isUnique(List<List<Character>> shape) {
2     for (List<List<Character>> orientation : orientations) {
3         if (orientation.equals(shape)) {
4             return false;
5         }
6     }
7     return true;
8 }

```

Gambar 10. Fungsi isUnique

Mengembalikan apakah blok unik.

### 3.5 Board

```

1 private void initializeGrid() {
2     grid = new ArrayList<>();
3     for (int i = 0; i < rows; i++) {
4         List<Character> row = new ArrayList<>();
5         for (int j = 0; j < cols; j++) {
6             row.add('.');
7         }
8         grid.add(row);
9     }
10 }



```

```

1 public boolean canPlaceBlock(Block block, int row, int col) {
2     List<List<Character>> shape = block.getShape();
3     for (int i = 0; i < shape.size(); i++) {
4         for (int j = 0; j < shape.get(i).size(); j++) {
5             if (shape.get(i).get(j) != '.' && (row + i >= rows || col + j >= cols || grid.get(row + i).get(col + j) != '.')) {
6                 return false;
7             }
8         }
9     }
10     return true;
11 }

```

Gambar 12. Fungsi canPlaceBlock

Gambar 11. Fungsi initializeGrid	
<p>Menginisialisasi papan permainan dengan ukuran rows dan cols yang telah diberikan di input. Setiap sel pada papan akan diisi dengan karakter '.' yang menandakan sel kosong.</p>	<p>Memeriksa apakah blok dapat ditempatkan pada posisi (row, col) di papan tanpa bertabrakan atau tumpang tindih dengan blok lain atau keluar batas papan.</p>
 <pre> 1 public void placeBlock(Block block, int row, int col, char symbol) { 2     List&lt;List&lt;Character&gt;&gt; shape = block.getShape(); 3     for (int i = 0; i &lt; shape.size(); i++) { 4         for (int j = 0; j &lt; shape.get(i).size(); j++) { 5             if (shape.get(i).get(j) != '.') { 6                 grid.get(row + i).set(col + j, symbol); 7             } 8         } 9     } 10 } </pre> <p>Gambar 13. Fungsi placeBlock</p>	 <pre> 1 public void removeBlock(Block block, int row, int col) { 2     List&lt;List&lt;Character&gt;&gt; shape = block.getShape(); 3     for (int i = 0; i &lt; shape.size(); i++) { 4         for (int j = 0; j &lt; shape.get(i).size(); j++) { 5             if (shape.get(i).get(j) != '.') { 6                 grid.get(row + i).set(col + j, '.'); 7             } 8         } 9     } 10 } </pre> <p>Gambar 14. Fungsi removeBlock</p>
<p>Menempatkan blok pada papan di posisi (row, col) dengan symbol tertentu.</p>	<p>Menghapus blok dari papan di posisi (row, col) dengan mengembalikan sel-sel yang ditempati blok ke kondisi kosong ('.').</p>

### 3.6 Color



Gambar 15. Fungsi getBlockColor

Menyediakan warna untuk setiap huruf.



Gambar 16. Fungsi getHexColor

Menyediakan warna untuk setiap huruf dalam bentuk hex.

## 3.7 InputHandler

```
private void readFile(String filePath) throws FileNotFoundException {
    try (Scanner scanner = new Scanner(new File(filePath))) {
        // Baca konfigurasi papan
        if (scanner.hasNextInt()) {
            scanner.close();
            throw new IllegalArgumentException("Error: Nilai N tidak ditemukan dalam file atau bukan bilangan bulat.");
        }
        N = scanner.nextInt();
        if (scanner.hasNextInt()) {
            scanner.close();
            throw new IllegalArgumentException("Error: Nilai M tidak ditemukan dalam file atau bukan bilangan bulat.");
        }
        M = scanner.nextInt();
        // Cek apakah positif
        if (N < 1 || M < 1) {
            scanner.close();
            throw new IllegalArgumentException("Error: Nilai N atau M tidak valid (positif).");
        }
        if (scanner.hasNextInt()) {
            scanner.close();
            throw new IllegalArgumentException("Error: Nilai P tidak ditemukan dalam file atau bukan bilangan bulat.");
        }
        P = scanner.nextInt();
        // Cek P harus lebih kecil dari 27
        if (P > 26) {
            scanner.close();
            throw new IllegalArgumentException("Error: Jumlah blok (' + P + ') melebihi batas maksimum (26).");
        }
        scanner.nextLine();
        caseType = scanner.nextLine();
        // Cek apakah valid
        if (!caseType.equals("DEFAULT")) {
            scanner.close();
            throw new IllegalArgumentException("Error: Case type yang valid hanya 'DEFAULT'");
        }
        // Inisialisasi variabel
        int blockCount = 0;
        List<List<Character>> currentBlock = new ArrayList<>();
        List<List<List<Character>>> blockShape = new ArrayList<>();
        char currentSymbol = 'W';
        // Proses input
        while (scanner.hasNext()) {
            String line = scanner.nextLine();
            if (line.isEmpty()) {
                if (currentBlock.isEmpty()) {
                    blockShape.add(new ArrayList<>(currentBlock));
                    currentBlock.clear();
                    blockCount++;
                }
                continue;
            }
            for (char c : line.toCharArray()) {
                if (!isValidChar(c)) {
                    throw new IllegalArgumentException("Error: Karakter yang ditemukan (' + c + ') bukan huruf kapital atau spasi.");
                }
                char firstLetter = ' ';
                for (char e : line.toCharArray()) {
                    if (Character.isUpperCase(e)) {
                        firstLetter = e;
                        break;
                    }
                }
                List<Character> processedRow = processLine(line);
                if (firstLetter != ' ') {
                    if (currentSymbol == 'W') {
                        currentSymbol = firstLetter;
                        currentBlock.add(processedRow);
                    } else if (currentSymbol == firstLetter) {
                        currentBlock.add(processedRow);
                    } else {
                        if (currentBlock.isEmpty()) {
                            blockShape.add(new ArrayList<>(currentBlock));
                            currentBlock.clear();
                            blockCount++;
                        }
                        currentSymbol = firstLetter;
                        currentBlock.add(processedRow);
                    }
                }
            }
            if (currentBlock.isEmpty()) {
                blockShape.add(new ArrayList<>(currentBlock));
                blockCount++;
            }
        }
        if (blockCount < P) {
            scanner.close();
            throw new IllegalArgumentException("Error: Jumlah blok yang terbaca (' + blockCount + ') kurang dari jumlah yang diharapkan (' + P + ').");
        }
        // Buat grid square
        for (List<List<Character>> block : blockShape) {
            List<List<Character>> squareBlock = makeSquare(block);
            block newBlock = new Block(squareBlock);
            blocks.add(newBlock);
        }
        scanner.close();
    }
}
```

Gambar 17. Fungsi readFile

Fungsi ini membaca file input yang berisi konfigurasi papan (N, M, P), jenis permainan, dan bentuk blok.

```

1 private boolean isValidChar(char c) {
2     return (Character.isUpperCase(c)) || c == ' ';
3 }

```

Gambar 18. Fungsi isValidChar

Memeriksa apakah karakter yang diberikan valid (huruf atau spasi).

```

1 private List<Character> processLine(String line) {
2     List<Character> processedRow = new ArrayList<>();
3     boolean foundFirstLetter = false;
4
5     for (char c : line.toCharArray()) {
6         if (Character.isUpperCase(c)) {
7             foundFirstLetter = true;
8             processedRow.add(c);
9         } else if (!foundFirstLetter && c == ' ') {
10            processedRow.add(' ');
11        } else {
12            processedRow.add('.');
13        }
14    }
15
16    return processedRow;
17 }

```

Gambar 19. Fungsi processLine

Memproses setiap baris dari file input untuk membuat representasi bentuk blok.

```

1 private static List<List<Character>> makeSquare(List<List<Character>> block) {
2     int rows = block.size();
3     int cols = 0;
4
5     for (List<Character> row : block) {
6         cols = Math.max(cols, row.size());
7     }
8
9     int max = Math.max(rows, cols);
10
11     List<List<Character>> squareBlock = new ArrayList<>();
12     for (List<Character> row : block) {
13         List<Character> newRow = new ArrayList<>(row);
14         while (newRow.size() < max) {
15             newRow.add('.');
16         }
17         squareBlock.add(newRow);
18     }
19
20     while (squareBlock.size() < max) {
21         List<Character> row = new ArrayList<>();
22         while (row.size() < max) {
23             row.add('.');
24         }
25         squareBlock.add(row);
26     }
27
28     return squareBlock;
29 }

```

Gambar 20. Fungsi makeSquare

Mengubah bentuk blok menjadi persegi dengan menambahkan '.' jika diperlukan.

### 3.8 OutputHandler

```
1 public static void saveToFile(Board board, String filePath) throws IOException {
2     FileWriter fileWriter = new FileWriter("test/output/" + filePath);
3     List<List<Character>> grid = board.getGrid();
4     for (List<Character> row : grid) {
5         for (Character cell : row) {
6             fileWriter.write(cell);
7         }
8         fileWriter.write("\n");
9     }
10    fileWriter.close();
11 }
12 }
```

Gambar 21. Fungsi saveToFile

Menyimpan papan permainan yang telah diselesaikan dalam bentuk file ‘.txt’.

```
1 public static void saveToImage(Board board, String filePath) {
2     int cellSize = 50;
3     List<List<Character>> grid = board.getGrid();
4     int rows = grid.size();
5     int cols = grid.get(0).size();
6
7     BufferedImage image = new BufferedImage(cols * cellSize, rows * cellSize, BufferedImage.TYPE_INT_ARGB);
8     Graphics2D g = image.createGraphics();
9
10    for (int i = 0; i < rows; i++) {
11        for (int j = 0; j < cols; j++) {
12            char cell = grid.get(i).get(j);
13            if (cell == ".") {
14                g.setColor(Color.WHITE);
15            } else {
16                g.setColor(Color.decode(ColorManager.getHexColor(cell)));
17            }
18            g.fillRect(j * cellSize, i * cellSize, cellSize, cellSize);
19
20            g.setColor(Color.BLACK);
21            g.drawRect(j * cellSize, i * cellSize, cellSize, cellSize);
22            g.setColor(Color.WHITE);
23            g.drawString(String.valueOf(cell), j * cellSize + 20, i * cellSize + 30);
24        }
25    }
26    g.dispose();
27    try {
28        ImageIO.write(image, "png", new File("test/output/" + filePath + ".png"));
29    } catch (java.io.IOException e) {
30        e.printStackTrace();
31    }
32 }
```

Gambar 22. Fungsi saveToImage

Menyimpan papan permainan yang telah diselesaikan dalam bentuk file gambar ‘.png’.

### 3.9 GUI

```
gui
├── App.java
├── Controller.java
```

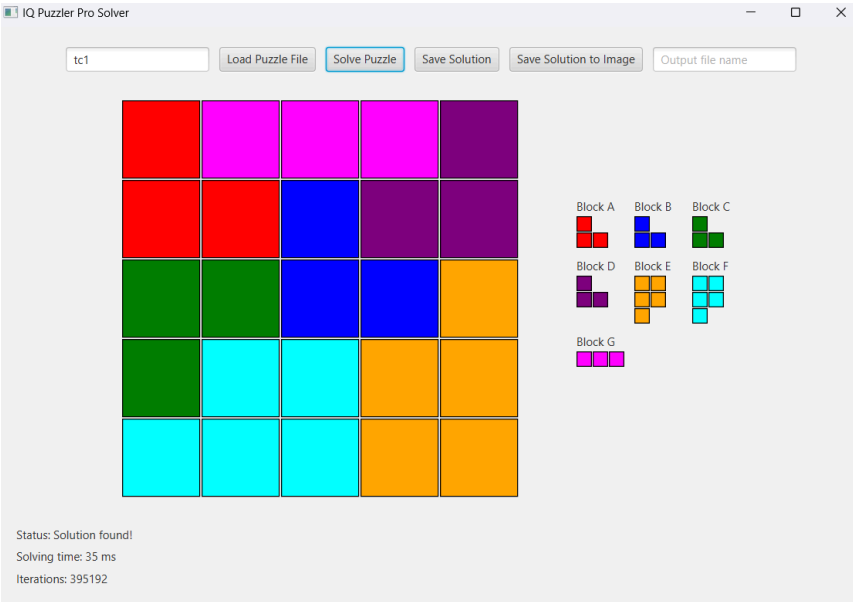

Gambar 23. Folder GUI

```
resources\gui
├── primary.fxml
```

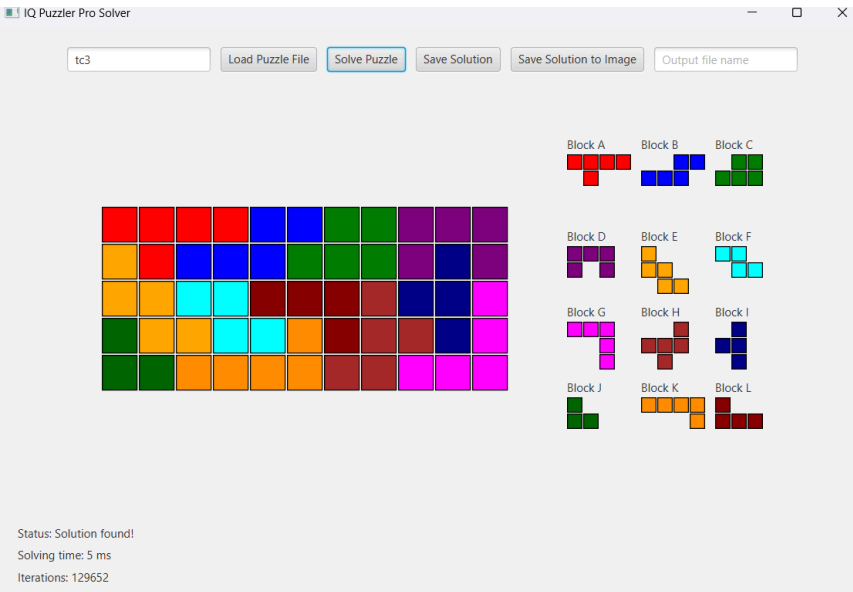
Gambar 24. Folder GUI

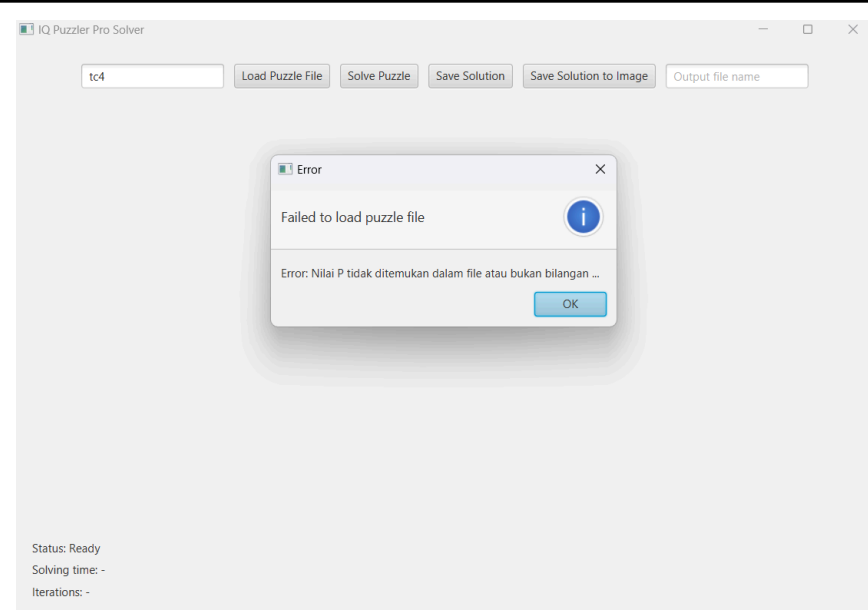
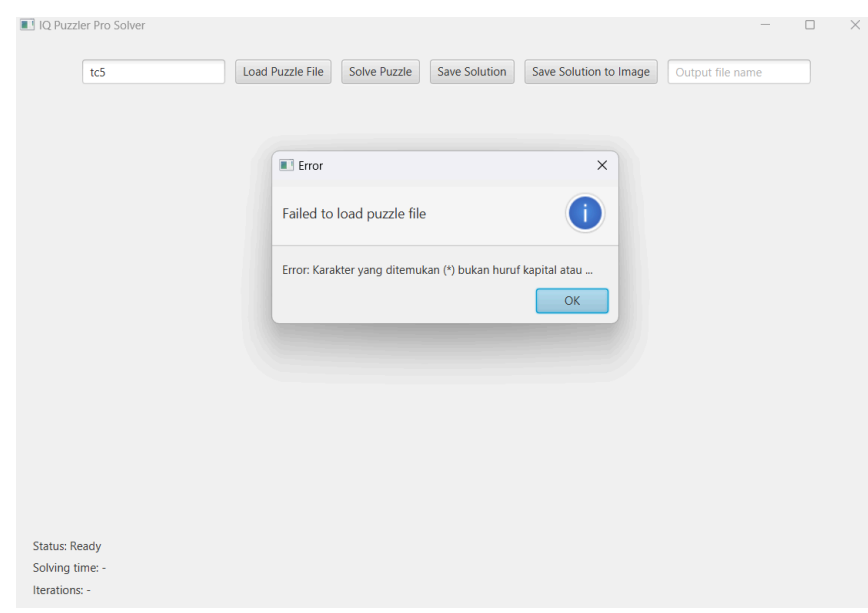
# BAB IV

## EKSPERIMEN

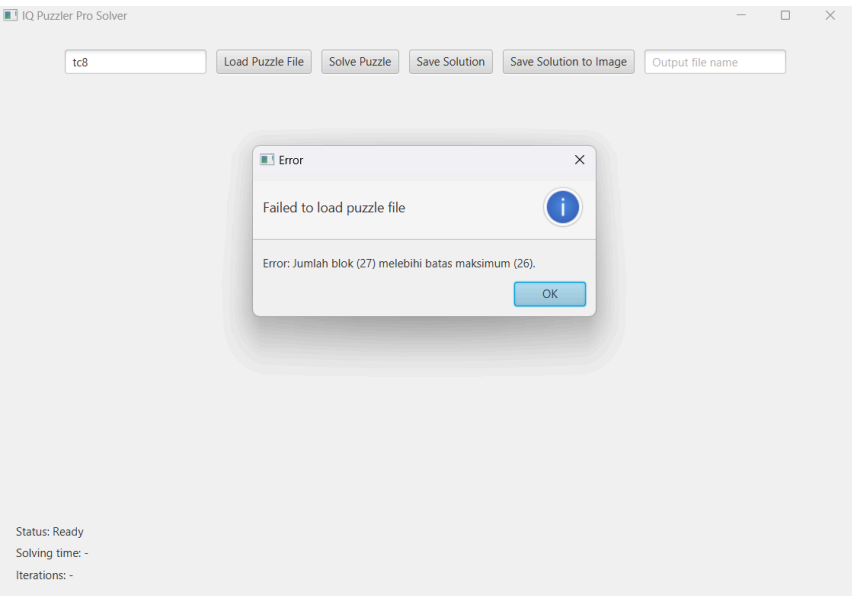
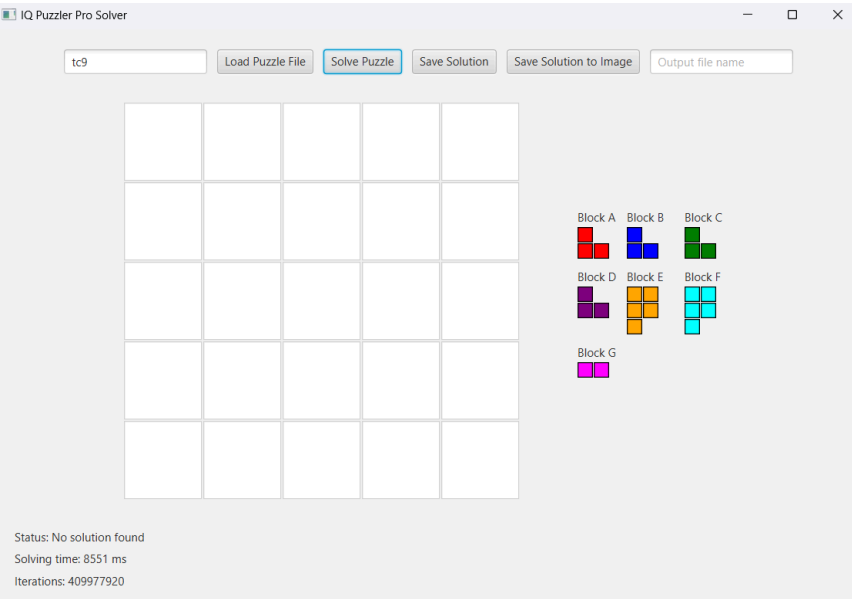
1.	5 5 7 DEFAULT A AA B BB C CC D DD EE EE E FF FF F GGG	
2.	6 6 9 DEFAULT A A A A A BBB B B CC CC D D D DD DD E EE	

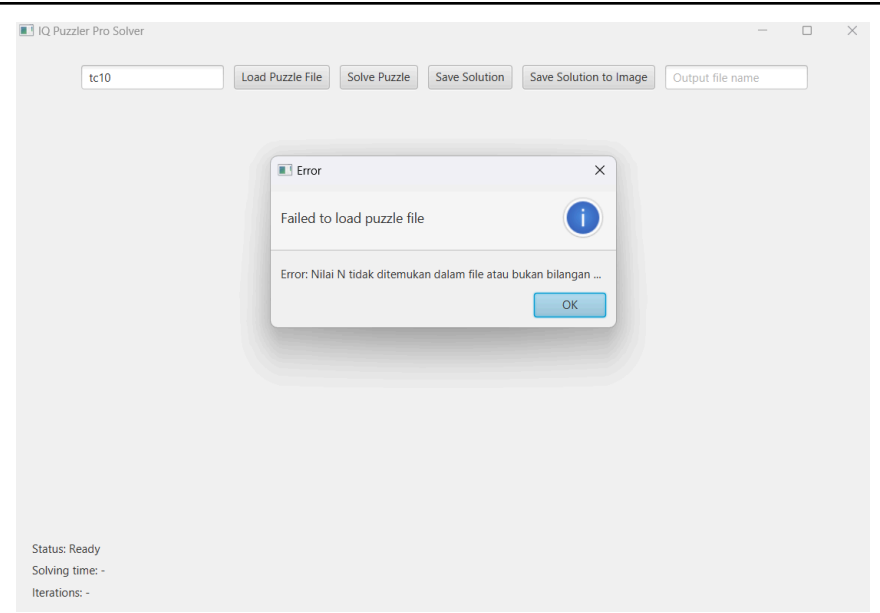
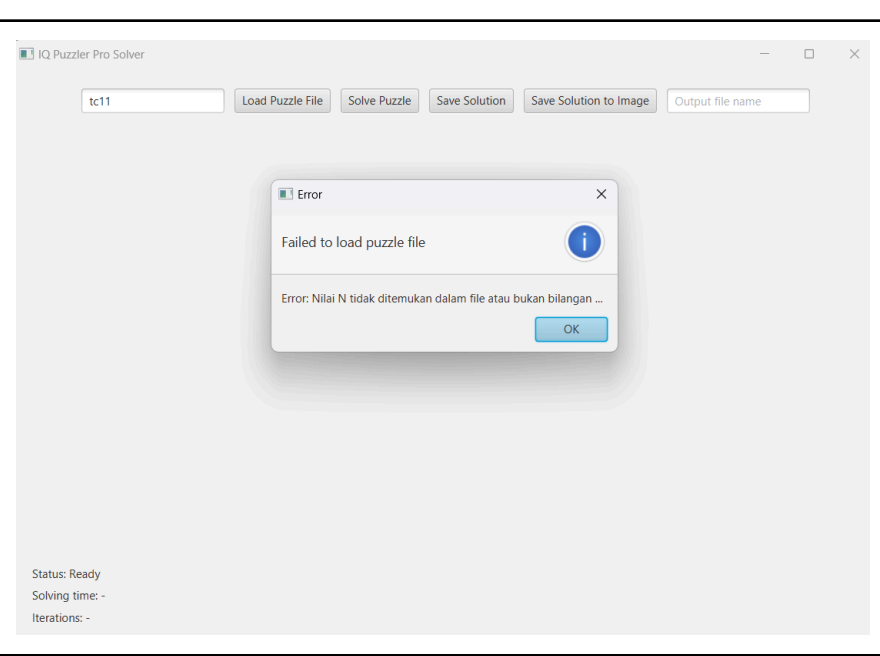


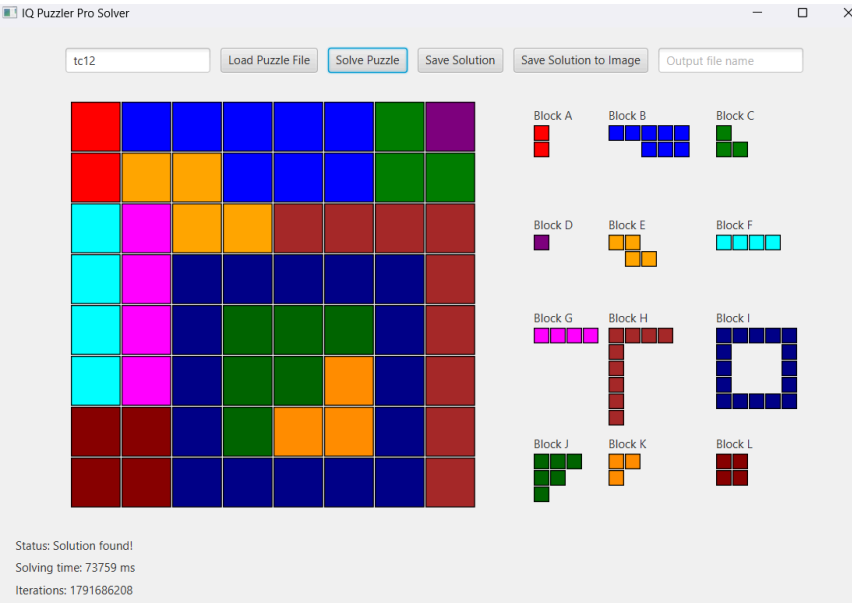
	F GG GG HHH III	
3.	5 11 12 DEFAULT AAAA A BB BBB CC CCC DDD D D E EE EE FF FF GGG G G H HHH H I II I J JJ KKKK K L LLL	

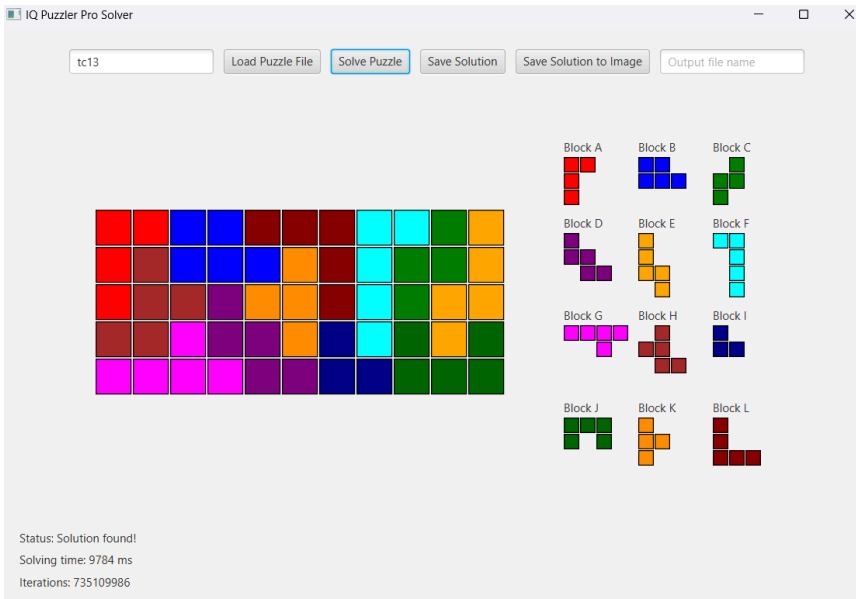
4.	2 3 DEFAULT A AA B B C	 <p>IQ Puzzler Pro Solver</p> <p>tc4 Load Puzzle File Solve Puzzle Save Solution Save Solution to Image Output file name</p> <p>Error</p> <p>Failed to load puzzle file</p> <p>Error: Nilai P tidak ditemukan dalam file atau bukan bilangan ...</p> <p>OK</p> <p>Status: Ready Solving time: - Iterations: -</p>
5.	3 3 3 DEFAULT A AA B BB ***	 <p>IQ Puzzler Pro Solver</p> <p>tc5 Load Puzzle File Solve Puzzle Save Solution Save Solution to Image Output file name</p> <p>Error</p> <p>Failed to load puzzle file</p> <p>Error: Karakter yang ditemukan (*) bukan huruf kapital atau ...</p> <p>OK</p> <p>Status: Ready Solving time: - Iterations: -</p>

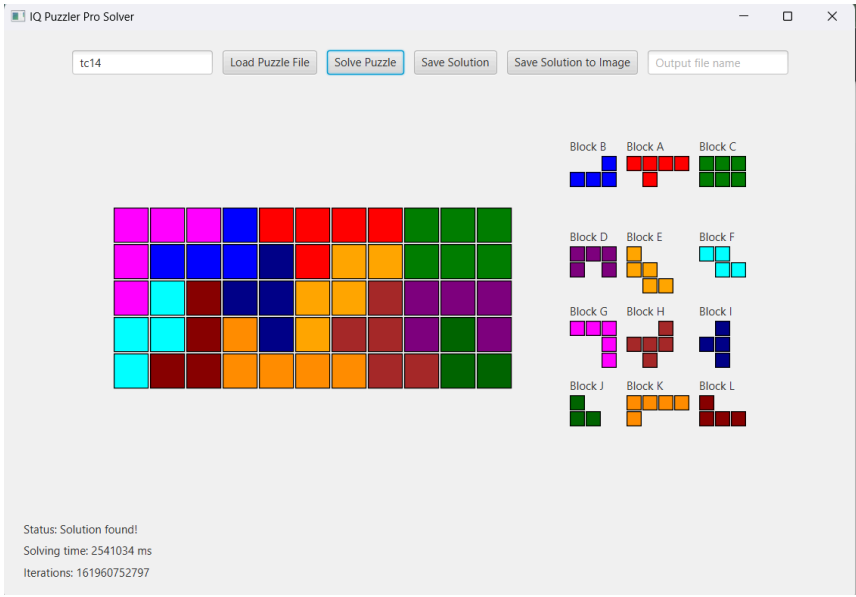
6.	3 3 3 DEFAULT A AA B BB aaa	<p>The screenshot shows the IQ Puzzler Pro Solver interface. The puzzle name is 'tc6'. An error dialog box is displayed with the message: 'Failed to load puzzle file. Error: Karakter yang ditemukan (a) bukan huruf kapital atau ...'. The status at the bottom is 'Ready', 'Solving time: -', and 'Iterations: -'.</p>
7.	3 3 3 A AA B BB CCC	<p>The screenshot shows the IQ Puzzler Pro Solver interface. The puzzle name is 'tc7'. An error dialog box is displayed with the message: 'Failed to load puzzle file. Error: Case type tidak ditemukan dalam file.'. The status at the bottom is 'Ready', 'Solving time: -', and 'Iterations: -'.</p>

8.	3 3 27 DEFAULT A AA B BB CCC	 <p>The screenshot shows the IQ Puzzler Pro Solver interface. An error dialog box is displayed in the center, stating "Failed to load puzzle file" and "Error: Jumlah blok (27) melebihi batas maksimum (26)." (Error: Number of blocks (27) exceeds the maximum limit (26)). The background window shows the "Solve Puzzle" button highlighted, and the status at the bottom indicates "Status: Ready", "Solving time: -", and "Iterations: -".</p>
9.	5 5 7 DEFAULT A AA B BB C CC D DD EE EE E FF FF F GG	 <p>The screenshot shows the IQ Puzzler Pro Solver interface with a 5x5 grid. The "Solve Puzzle" button is highlighted. To the right of the grid is a legend for the blocks: Block A (red), Block B (blue), Block C (green), Block D (purple), Block E (orange), Block F (cyan), and Block G (pink). The status at the bottom indicates "Status: No solution found", "Solving time: 8551 ms", and "Iterations: 409977920".</p>

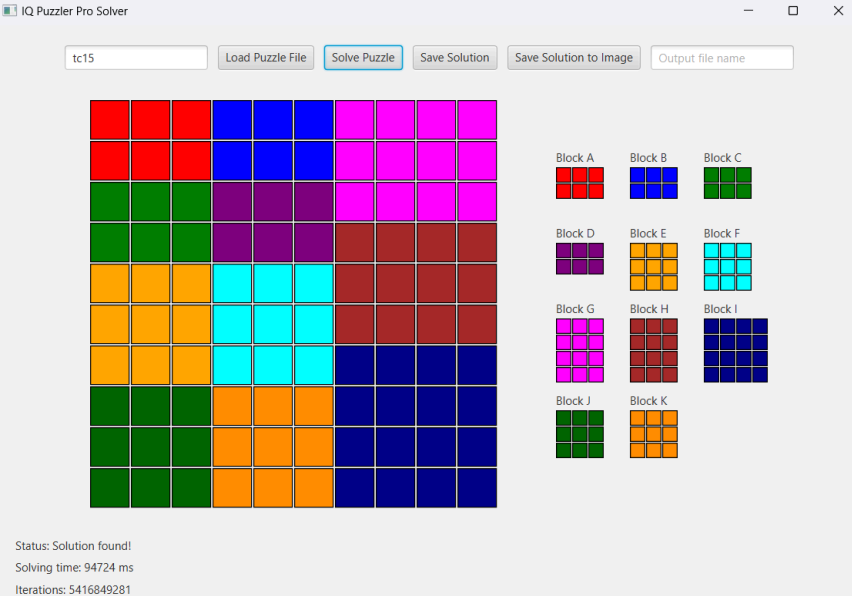
10.	halo	 <p>The screenshot shows the IQ Puzzler Pro Solver application window. At the top, there is a text input field containing 'tc10' and four buttons: 'Load Puzzle File', 'Solve Puzzle', 'Save Solution', and 'Save Solution to Image'. To the right of these buttons is an 'Output file name' text input field. In the center, an 'Error' dialog box is displayed with the title 'Error' and a close button. The dialog contains the message 'Failed to load puzzle file' and a detailed error message: 'Error: Nilai N tidak ditemukan dalam file atau bukan bilangan ...'. An 'OK' button is at the bottom of the dialog. At the bottom of the main window, the status is 'Ready', solving time is '-', and iterations are '-'. The application title bar reads 'IQ Puzzler Pro Solver'.</p>
11.		 <p>This screenshot is identical to the one in row 10, but the text input field at the top now contains 'tc11'. All other elements, including the error dialog and status information, remain the same.</p>

12.	8 8 12 DEFAULT A A BBBBB BBB C CC D EE EE FFFF GGGG HHHH H H H H H IIII I I I I I I IIII JJJ JJ J KK K LL LL	 <p>The screenshot shows the IQ Puzzler Pro Solver interface. The main puzzle is a 12x12 grid with various colored blocks placed. The blocks are labeled A through L, each with a unique shape and color. The interface includes buttons for 'Load Puzzle File', 'Solve Puzzle', 'Save Solution', 'Save Solution to Image', and 'Output file name'. The status bar at the bottom indicates 'Status: Solution found!', 'Solving time: 73759 ms', and 'Iterations: 1791686208'.</p>
-----	--	---

13.	5 11 12 DEFAULT AA A A BB BBB C CC C D DD DD E E EE E FF F F F GGGG G H HH HH I II JJJ J J K KK K L L LLL	
-----	--	--

14.	5 11 12 DEFAULT B BBB AAAA A CCC CCC DDD D D E EE EE FF FF GGG G G H HHH H I II I J JJ KKKK K L LLL	 <p>IQ Puzzler Pro Solver</p> <p>tc14 Load Puzzle File Solve Puzzle Save Solution Save Solution to Image Output file name</p> <p>Status: Solution found!  Solving time: 2541034 ms  Iterations: 161960752797</p>
-----	--	--



15.	10 10 11 DEFAULT AAA AAA BBB BBB CCC CCC DDD DDD EEE EEE FFF FFF FFF GGG GGG GGG GGG HHH HHH HHH HHH III III III III JJJ JJJ JJJ KKK KKK KKK	
-----	--	--

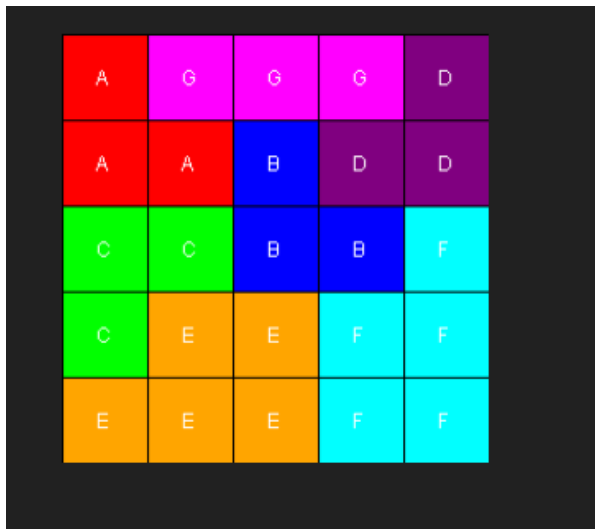
**Output Text Solution Test Case 1:**

```
AGGGD
AABDD
CCBBF
CEEFF
EEEFF
```

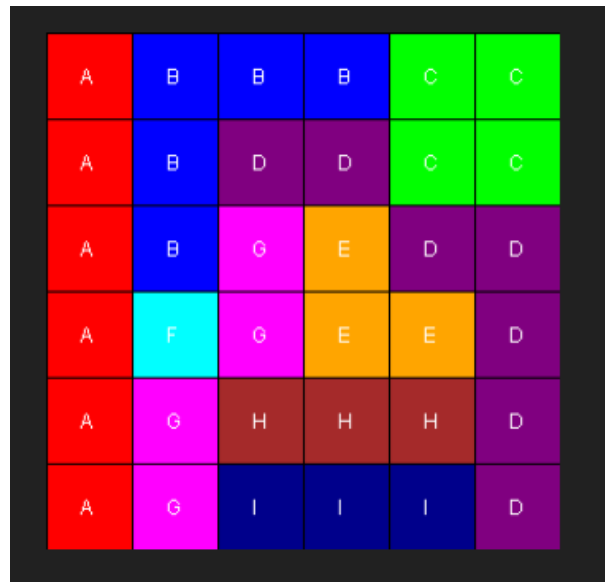
**Output Text Solution Test Case 2:**

```
ABBBCC
ABDDCC
ABGEDD
AFGEED
AGHHH
AGIIID
```

**Output Image Solution Test Case 1:**



**Output Image Solution Test Case 2:**



## **BAB V**

### **KESIMPULAN, SARAN, KOMENTAR, DAN REFLEKSI**

#### **5.1 Kesimpulan**

Melalui pengerjaan tugas kecil ini, saya telah berhasil mengimplementasikan algoritma Brute Force dengan teknik rekursi dan backtracking untuk menyelesaikan puzzle IQ Puzzler Pro. Dalam tugas ini, saya mengembangkan sebuah program yang dapat memuat konfigurasi papan dan blok dari file input, menempatkan blok-blok pada papan dengan semua kemungkinan rotasi dan pencerminan, serta menemukan solusi yang valid. Program ini juga dilengkapi dengan antarmuka grafis (GUI) yang memudahkan pengguna untuk memvisualisasikan papan, blok, dan solusi yang ditemukan. Selain itu, program ini dapat menyimpan solusi ke dalam file teks atau gambar.

#### **5.2 Refleksi**

Pengerjaan tugas kecil ini memberikan pengalaman yang sangat berharga dalam mengaplikasikan algoritma Brute Force. Tantangan utama yang telah saya hadapi adalah memastikan bahwa algoritma berjalan dengan baik dan efisien. Saya juga belajar mengenai pentingnya validasi input dan penanganan error yang baik, terutama saat membaca file dan memproses bentuk blok. Selain itu, pengembangan antarmuka grafis menggunakan JavaFX memberikan wawasan baru bagi saya tentang bagaimana menghubungkan logika dengan tampilan visual yang interaktif. Pengalaman ini tentunya akan sangat berguna dalam proyek-proyek teknologi yang lebih kompleks di masa depan.

**Untuk Tuhan, Bangsa, dan Almamater. Hidup Informatika!**

## BAB VI

### LAMPIRAN

#### 6.1 Tautan Repository

Link repository dari Tugas Kecil 1 IF2211 Strategi Algoritma adalah sebagai berikut:

[https://github.com/varel183/Tucil1\\_13523008](https://github.com/varel183/Tucil1_13523008)

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	✓	