

IF2211 Strategi Algoritma

Laporan Tugas Kecil 1



Disusun oleh:

Muhammad Kinan Arkansyaddad (13523152)

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
JL. GANESA 10, BANDUNG 40132
2025

Daftar Isi

Daftar Isi	2
BAB I	
DESKRIPSI MASALAH DAN ALGORITMA	4
1.1 IQ Puzzler Pro	4
1.2 Algoritma Brute Force	4
1.3 Algoritma Brute Force dalam penyelesaian IQ Puzzler Pro	5
BAB II	
IMPLEMENTASI PROGRAM DALAM BAHASA JAVA	7
2.1 App.java	7
2.2 PuzzleBoard.java	7
2.3 PuzzlePiece.java	9
2.4 PuzzleSolver.java	11
2.5 PuzzleSolverGUI.java	12
2.6 InputReader.java	14
BAB III	
SOURCE CODE PROGRAM	16
3.1 Repository Program	16
3.2 Source Code Algoritma Brute Force	16
BAB IV	
EKSPERIMEN	21
4.1 Test Case 1	21
4.2 Test Case 2	24
4.3 Test Case 3	29
4.4 Test Case 4	32
4.5 Test Case 5	36
4.6 Test Case 6	41
4.7 Test Case 7	43
4.8 Test Case 8	44
4.9 Test Case 9	48
4.10 Test Case 10	49
4.11 Test Case 11	51

4.12 Test Case 12	52
4.13 Test Case 13	54
4.14 Test Case 14	56
BAB V	
LAMPIRAN	59
Pustaka	60

BAB I

DESKRIPSI MASALAH DAN ALGORITMA

1.1 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:

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

1.2 Algoritma Brute Force

Algoritma brute force adalah pendekatan langsung dan sederhana dalam menyelesaikan suatu permasalahan. Pendekatan ini mencoba semua kemungkinan solusi tanpa mempertimbangkan optimasi, sehingga sering kali tidak efisien tetapi mudah dipahami dan diimplementasikan. Metode ini mengikuti prinsip yang jelas, yaitu menyelesaikan masalah dengan cara yang paling obvious atau langsung, tanpa menggunakan teknik yang lebih kompleks.

Keunggulan utama dari algoritma brute force adalah kesederhanaannya. Langkah-langkahnya mudah dipahami dan tidak memerlukan strategi khusus dalam penyelesaiannya. Oleh karena itu, pendekatan ini sering kali menjadi solusi pertama yang diuji sebelum beralih ke metode yang lebih efisien. Slogan seperti "Just do it!" atau "Just solve it!" menggambarkan bagaimana algoritma ini bekerja—langsung mencoba semua kemungkinan hingga solusi ditemukan.

Dalam praktiknya, algoritma brute force dapat dituliskan secara langsung berdasarkan pernyataan masalah yang diberikan dan definisi atau konsep yang terlibat. Hal ini menjadikannya sebagai pendekatan yang intuitif, terutama dalam kasus-kasus di mana ruang pencarian solusi masih dalam skala yang dapat ditangani secara komputasional. Namun, dalam permasalahan yang lebih kompleks dengan ruang pencarian yang besar, metode ini sering kali digantikan oleh algoritma yang lebih optimal.

1.3 Algoritma Brute Force dalam penyelesaian IQ Puzzler Pro

Penulis menerapkan pendekatan brute force dengan teknik backtracking untuk menyelesaikan IQ Puzzler Pro. Algoritma ini bekerja dengan mencoba semua kemungkinan penempatan kepingan puzzle hingga menemukan solusi yang valid. Berikut adalah langkah-langkah dalam penyelesaiannya:

1. Pengguna memasukkan file teks(.txt) yang berisi representasi papan dan kepingan puzzle.
2. Program membaca data papan permainan dan kepingan puzzle. Setiap kepingan puzzle memiliki beberapa variasi bentuk akibat rotasi atau pencerminan. Program menyimpan data papan permainan, tipe

permainan(*default* atau *custom*), dan semua variasi unik kepingan puzzle.

3. Program akan mencoba menempatkan kepingan puzzle satu per satu menggunakan metode rekursif dan backtracking. Program akan mencoba semua posisi di papan dan semua variasi kepingan puzzle. Jika suatu kepingan puzzle tidak bisa ditempatkan di posisi tertentu, program akan langsung mencoba posisi atau kepingan lainnya.
4. Jika seluruh kepingan berhasil ditempatkan dan memenuhi seluruh papan, solusi ditemukan. Jika tidak ada solusi yang ditemukan dengan kombinasi saat ini, program akan kembali ke langkah sebelumnya (backtracking) dan mencoba konfigurasi lainnya.
5. Program akan mencari satu kemungkinan solusi yang valid, kemudian berhenti.
6. Setelah solusi ditemukan, program akan menampilkan hasilnya kepada pengguna. Solusi akan divisualisasikan dalam bentuk papan permainan yang telah terisi dengan kepingan puzzle. Program juga akan menampilkan jumlah langkah yang diperlukan (jumlah percobaan) dan waktu eksekusi dalam ms. Pengguna akan diberikan opsi untuk menyimpan solusi dalam format teks (.txt) atau gambar (.png).

BAB II

IMPLEMENTASI PROGRAM DALAM BAHASA JAVA

2.1 App.java

App.java	
Class App adalah class utama yang menjalankan program dan menginisialisasi GUI	
Atribut	
-	
Konstruktor	
-	-
Metode	
main	Metode utama yang menjalankan aplikasi dan memulai GUI

2.2 PuzzleBoard.java

PuzzleBoard.java	
Class PuzzleBoard adalah class yang merepresentasikan papan permainan, menyediakan metode untuk menempatkan, menghapus, dan memeriksa kepingan puzzle.	
Atribut	
<ol style="list-style-type: none">1. private char[][] board2. private int row, col	

3. private Map<Character, String> pieceColors 4. private static final String RESET	
Konstruktor	
public PuzzleBoard(int rows, int cols)	Konstruktor yang membuat papan permainan dengan ukuran rows × cols dan mengisinya dengan tanda '#' (kosong).
public PuzzleBoard(char[][] customBoard)	Konstruktor yang membuat papan permainan berdasarkan matriks yang diberikan (customBoard), mengganti karakter 'X' dengan '#' agar tidak sama ketika ada kepingan puzzle dengan id 'X'..
Metode	
public int getRow()	Mengembalikan jumlah baris papan.
public int getCol()	Mengembalikan jumlah kolom papan.
public char[][] getBoard()	Mengembalikan matriks papan permainan saat ini.
public boolean isEmpty(int x, int y)	Memeriksa apakah sel pada posisi (x, y) kosong ('#').
public boolean canPlacePiece(PuzzlePiece	Mengecek apakah kepingan piece dapat ditempatkan di

piece, int x, int y)	posisi (x, y).
public void placePiece(PuzzlePiece piece, int x, int y)	Menempatkan kepingan piece di posisi (x, y) pada papan.
public void removePiece(PuzzlePiece piece, int x, int y)	Menghapus kepingan piece dari papan di posisi (x, y).
public boolean isSolved()	Mengecek apakah seluruh papan sudah terisi oleh kepingan puzzle (tidak ada '#' yang tersisa).
public void printBoard()	Mencetak papan permainan ke terminal dengan pewarnaan berdasarkan kepingan yang telah ditempatkan.

2.3 PuzzlePiece.java

PuzzlePiece.java
Class PuzzlePiece adalah class yang merepresentasikan kepingan puzzle, termasuk bentuk, rotasi, dan variasi yang memungkinkan.
Atribut
<ol style="list-style-type: none"> 1. private int[][] shape 2. private int size

3. private char id 4. private String color 5. public PuzzlePiece[] variations	
Konstruktor	
public PuzzlePiece(int[][] shape, int size, char id, String color)	Konstruktor yang menginisialisasi kepingan puzzle dengan bentuk, ukuran, ID, dan warna.
Metode	
public int[][] getShape()	Mengembalikan bentuk kepingan puzzle dalam koordinat.
public int getSize()	Mengembalikan ukuran kepingan puzzle.
public char getId()	Mengembalikan karakter ID kepingan puzzle.
public String getColor()	Mengembalikan warna kepingan puzzle.
public PuzzlePiece rotate()	Menghasilkan kepingan baru yang merupakan rotasi 90° dari kepingan saat ini.
public PuzzlePiece flipH()	Menghasilkan kepingan baru yang merupakan hasil pencerminan horizontal dari kepingan saat ini.
public PuzzlePiece normalize()	Mengubah koordinat bentuk

	kepingan agar selalu dimulai dari (0,0).
public void getAllUniqueVariations()	Menghasilkan semua variasi unik dari kepingan puzzle melalui rotasi dan pencerminan, lalu menyimpannya di array variations.

2.4 PuzzleSolver.java

PuzzleSolver.java	
Class PuzzleSolver adalah class yang mengimplementasikan algoritma brute force dengan backtracking untuk menyusun puzzle hingga menemukan solusi.	
Atribut	
<ol style="list-style-type: none"> 1. public PuzzleBoard board 2. private PuzzlePiece[] pieces 3. public boolean solved 4. public long executionTime 5. public long turn 6. private String type 	
Konstruktor	
public PuzzleSolver(PuzzleBoard board, PuzzlePiece[] pieces,	Konstruktor untuk menginisialisasi solver dengan papan permainan, daftar

String type)	kepingan puzzle, dan jenis papan.
Metode	
public boolean solvePuzzle(int piecePlaced)	Algoritma brute force dengan backtracking untuk mencoba menempatkan kepingan puzzle satu per satu.
public void startSolving()	Memulai proses pencarian solusi dan mencatat waktu eksekusi algoritma.

2.5 PuzzleSolverGUI.java

PuzzleSolverGUI.java
Class PuzzleSolverGUI adalah class yang menyediakan GUI bagi pengguna untuk berinteraksi dengan solver, menampilkan solusi dalam bentuk visual, dan menyimpan solusi dalam bentuk teks dan gambar.
Atribut
<ol style="list-style-type: none"> 1. private File selectedFile; 2. private JButton selectFileButton; 3. private JFrame frame; 4. private JLabel loadingLabel; 5. private char[][] solutionBoard; 6. private boolean solved; 7. private long timesTaken; 8. private long turnsTaken;

	dipilih pengguna.
private void saveSolutionAsImage()	Menyimpan solusi puzzle dalam bentuk gambar PNG.
private Color getColorForPiece(char piece)	Mengembalikan warna Color berdasarkan karakter unik dari setiap bagian puzzle.
private void styleButton(JButton button)	Memberikan style dan efek hover pada tombol dalam GUI.
private void showErrorDialog(String message)	Menampilkan dialog error jika terjadi kesalahan selama proses solving.
private void restartApp()	Memulai ulang aplikasi dengan membuat instance baru dari PuzzleSolverGUI.

2.6 InputReader.java

PuzzleSolver.java
Class PuzzleSolver adalah class yang mengimplementasikan algoritma brute force dengan backtracking untuk menyusun puzzle hingga menemukan solusi.
Atribut
<ol style="list-style-type: none"> 1. public int N, M, P; 2. public String type; 3. public List<List<int[]>> puzzlePieces; 4. public char[] ids;

5. public char[][] customBoard;	
Konstruktor	
public InputReader(String fileName)	Konstruktor yang menginisialisasi puzzlePieces dan memanggil readInputFile(fileName) untuk membaca dan memproses file input.
Metode	
private void readInputFile(String fileName)	Membaca file input dan mengisi atribut N, M, P, type, board, puzzlePieces, dan ids.
public void startSolving()	Memulai proses pencarian solusi dan mencatat waktu eksekusi algoritma. Menangani validasi format input, seperti jumlah elemen pada baris pertama.
private int parsePositiveInt(String value, String errorMessage)	Mengonversi string ke bilangan bulat, jika value adalah bilangan bulat. Melempar Error jika value bukan bilangan positif atau bukan angka.
private List<int[]> parsePiece(List<String> shape)	bentuk potongan puzzle dari representasi teks menjadi daftar koordinat (x, y).

BAB III

SOURCE CODE PROGRAM

3.1 Repository Program

Repository Program dapat diakses melalui tautan Github berikut: https://github.com/kin-ark/Tucill_13523152

3.2 Source Code Algoritma Brute Force

PuzzleSolver.java:

```
package com.kinan.iqpuzzlerpro.solver;

import com.kinan.iqpuzzlerpro.game.*;

public class PuzzleSolver {
    public PuzzleBoard board;
    private PuzzlePiece[] pieces;
    public boolean solved;
    public long executionTime;
    public long turn;
    private String type;

    public PuzzleSolver(PuzzleBoard board, PuzzlePiece[] pieces,
String type){
        this.board = board;
        this.pieces = pieces;
        this.solved = false;
        this.turn = 0;
        this.type = type;
    }

    public boolean solvePuzzle(int piecePlaced) {
        if (piecePlaced == pieces.length) {
            if (board.isSolved()) {
                board.printBoard();
                solved = true;
                return solved;
            }
        }
    }
}
```



```

        return solved;
    }

    PuzzlePiece piece = pieces[piecePlaced];

    for (PuzzlePiece variation : piece.variations) {
        for (int y = 0; y < board.getRow(); y++) {
            for (int x = 0; x < board.getCol(); x++) {
                turn++;

                if (!board.canPlacePiece(variation, x, y)) continue;

                board.placePiece(variation, x, y);
                if (solvePuzzle(piecePlaced + 1)) {
                    return true;
                }
                board.removePiece(variation, x, y);
            }
        }
    }
    return false;
}

public void startSolving() {
    solved = false;

    long startTime = System.nanoTime();

    if (!solvePuzzle(0)) {
        System.out.println("No solution found.");
    }

    System.out.println("Turn: " + turn);
    long endTime = System.nanoTime();

    executionTime = (endTime - startTime) / 1000000;
    System.out.println(executionTime + "ms");
}

```

```
}  
}
```

PuzzleSolverGUI.java:

```
private void runSolver() {  
    SwingWorker<Void, Void> solverWorker = new  
    SwingWorker<>() {  
        @Override  
        protected Void doInBackground() {  
            try {  
                InputReader reader = new  
                InputReader(selectedFile.getAbsolutePath());  
  
                PuzzleBoard board = null;  
                if ("DEFAULT".equals(reader.type)){  
                    board = new PuzzleBoard(reader.N, reader.M);  
                } else if ("CUSTOM".equals(reader.type)){  
                    board = new PuzzleBoard(reader.customBoard);  
                }  
  
                PuzzlePiece[] pieces = new PuzzlePiece[reader.P];  
  
                String[] colors = {  
                    "\u001B[31m", "\u001B[32m", "\u001B[33m",  
                    "\u001B[34m", "\u001B[35m", "\u001B[36m",  
                    "\u001B[1;31m", "\u001B[1;32m", "\u001B[1;33m",  
                    "\u001B[1;34m", "\u001B[1;35m", "\u001B[1;36m",  
                    "\u001B[4;31m", "\u001B[4;32m", "\u001B[4;33m",  
                    "\u001B[4;34m", "\u001B[4;35m", "\u001B[4;36m",  
                    "\u001B[41m", "\u001B[42m", "\u001B[43m",  
                    "\u001B[44m", "\u001B[45m", "\u001B[46m",  
                }  
            }  
        }  
    }  
}
```

```

        "\u001B[1;4;31m", "\u001B[1;4;32m"
    };

    for (int i = 0; i < reader.P; i++) {
        int[][] shape =
reader.puzzlePieces.get(i).toArray(new int[0][]);
        int size = reader.puzzlePieces.get(i).size();
        pieces[i] = new PuzzlePiece(shape, size,
reader.ids[i], colors[i]);
        pieces[i].getAllUniqueVariations();
    }

    PuzzleSolver solver = new PuzzleSolver(board, pieces,
reader.type);
    solver.startSolving();

    if (solver.solved){
        solutionBoard = solver.board.getBoard();
        solved = true;
    }

    timesTaken = solver.executionTime;
    turnsTaken = solver.turn;

    } catch (Error e) {
        showErrorDialog("Input Error: " + e.getMessage());
        return null;
    } catch (Exception e) {
        showErrorDialog("Unexpected Error: " +
e.getMessage());
        return null;
    }
}

```

```
        return null;
    }

    @Override
    protected void done() {
        SwingUtilities.invokeLater(() -> showFinalScreen());
    }
};

solverWorker.execute();
}
```

BAB IV EKSPERIMEN

4.1 Test Case 1

Masukan
5 5 7 DEFAULT A AA B BB C CC D DD EE EE E FF FF F GGG
Keluaran
Terminal:


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

Turn: 5194981
93ms

GUI:



File Teks(.txt)

```
test > output >  output1.txt
```

```
1 AGGGD
```

```
2 AABDD
```

```
3 CCBBE
```

```
4 CFFEE
```

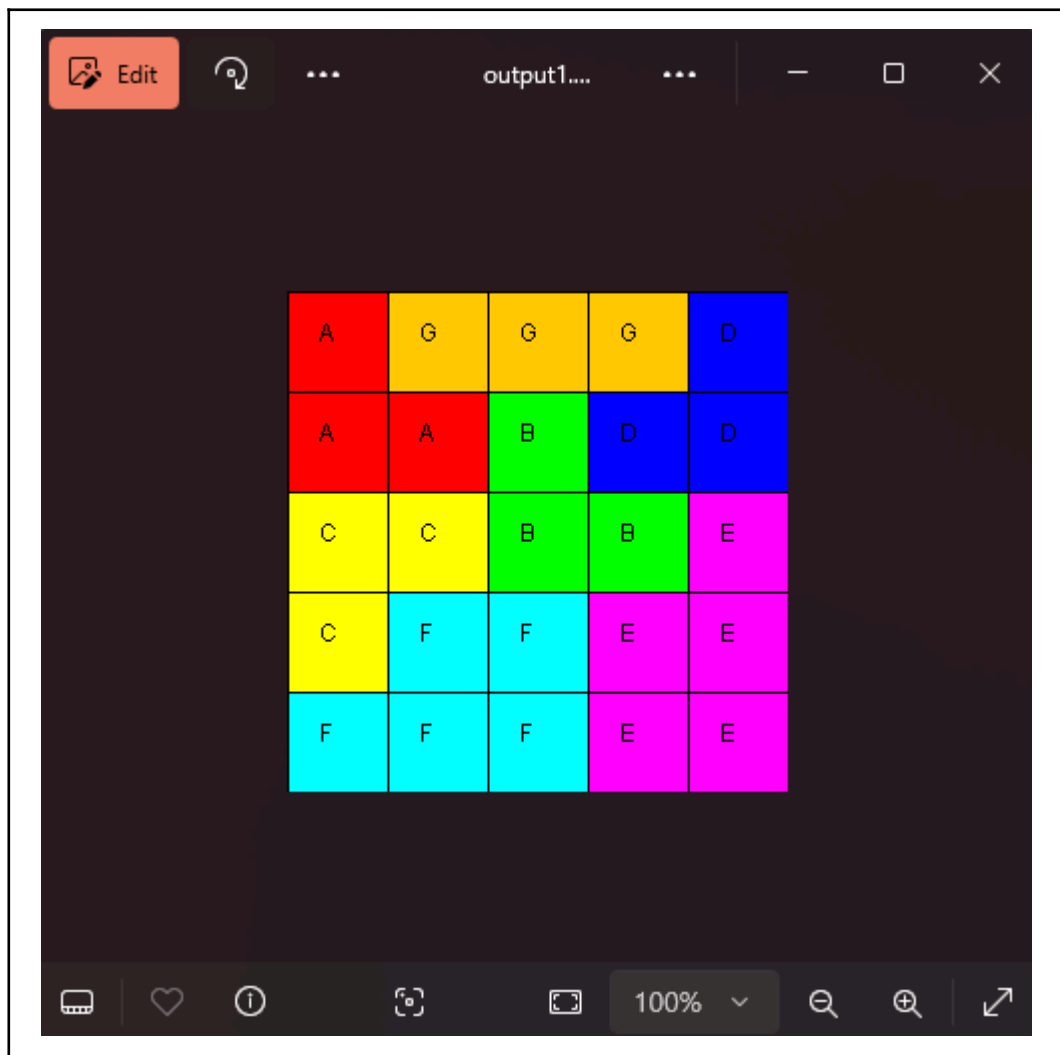
```
5 FFFEE
```

```
6 Times taken: 93 ms
```

```
7 Turns taken: 5194981 turns
```

```
8
```

File Gambar(.png)



4.2 Test Case 2

Masukan
13 2 26 DEFAULT A B C

D	
E	
F	
G	
H	
I	
J	
K	
L	
M	
N	
O	
P	
Q	
R	
S	
T	
U	
V	
W	
X	
Y	
Z	
Keluaran	
Terminal:	

A B
C D
E F
G H
I J
K L
M N
O P
Q R
S T
U V
W X
Y Z
Turn: 351
7ms

GUI:

IQ Puzzle Pro Solver

Puzzle Result!

Turns: 351Time: 7 ms

A	B
C	D
E	F
G	H
I	J
K	L
M	N
O	P
Q	R
S	T
U	V
W	X
Y	Z

Save as Text

Save as Image

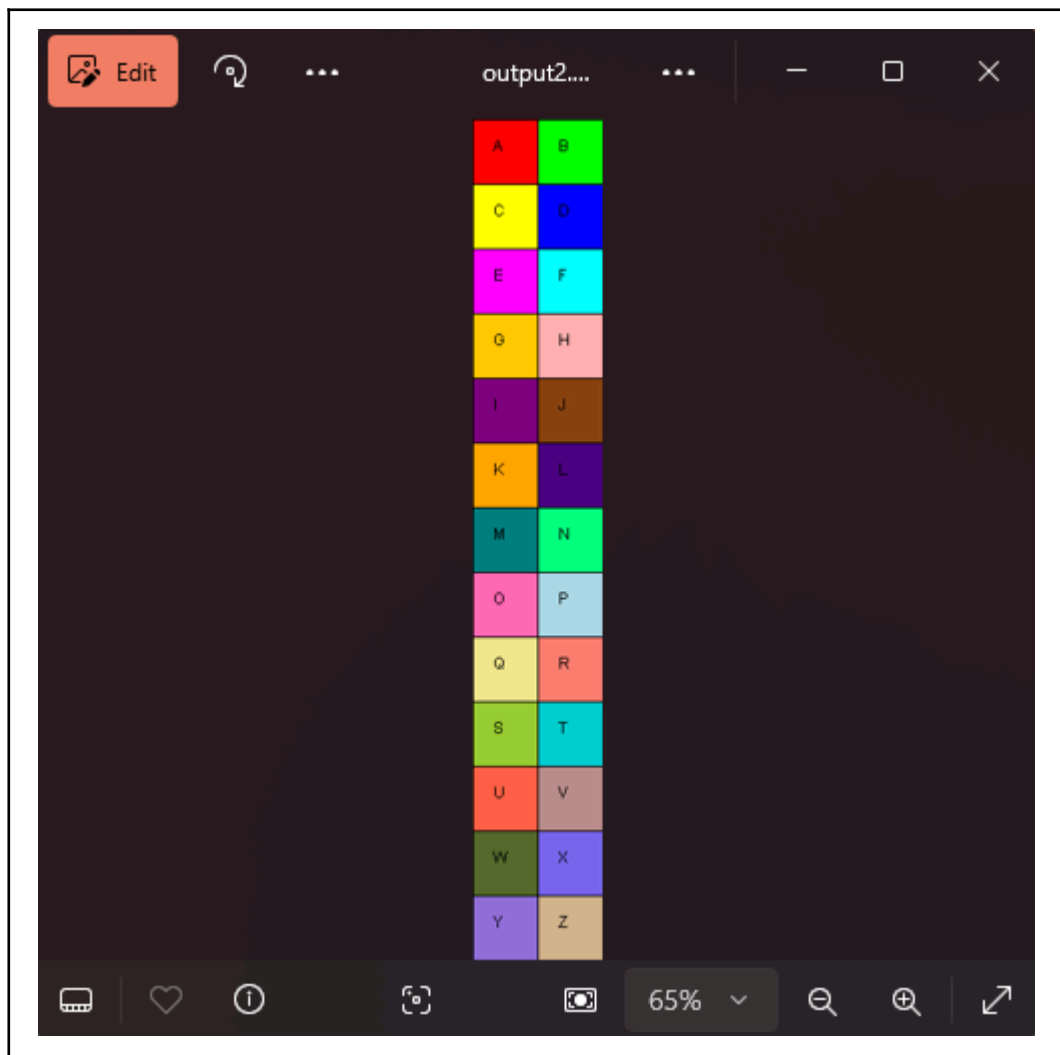
Restart App

File Teks(.txt)

```
test > output >  output2.txt
```

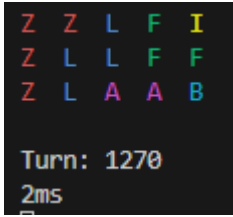
```
1  AB
2  CD
3  EF
4  GH
5  IJ
6  KL
7  MN
8  OP
9  QR
10 ST
11 UV
12 WX
13 YZ
14 Times taken: 7 ms
15 Turns taken: 351 turns
16
```

File Gambar(.png)



4.3 Test Case 3

Masukan
3 5 6 DEFAULT ZZ Z Z

F FF I L LL L A A B
Keluaran
Terminal:

GUI:

IQ Puzzle Pro Solver

Puzzle Result!

Turns: 1270Time: 2 ms

Z	Z	L	F	I
Z	L	L	F	F
Z	L	A	A	B

Save as Text

Save as Image

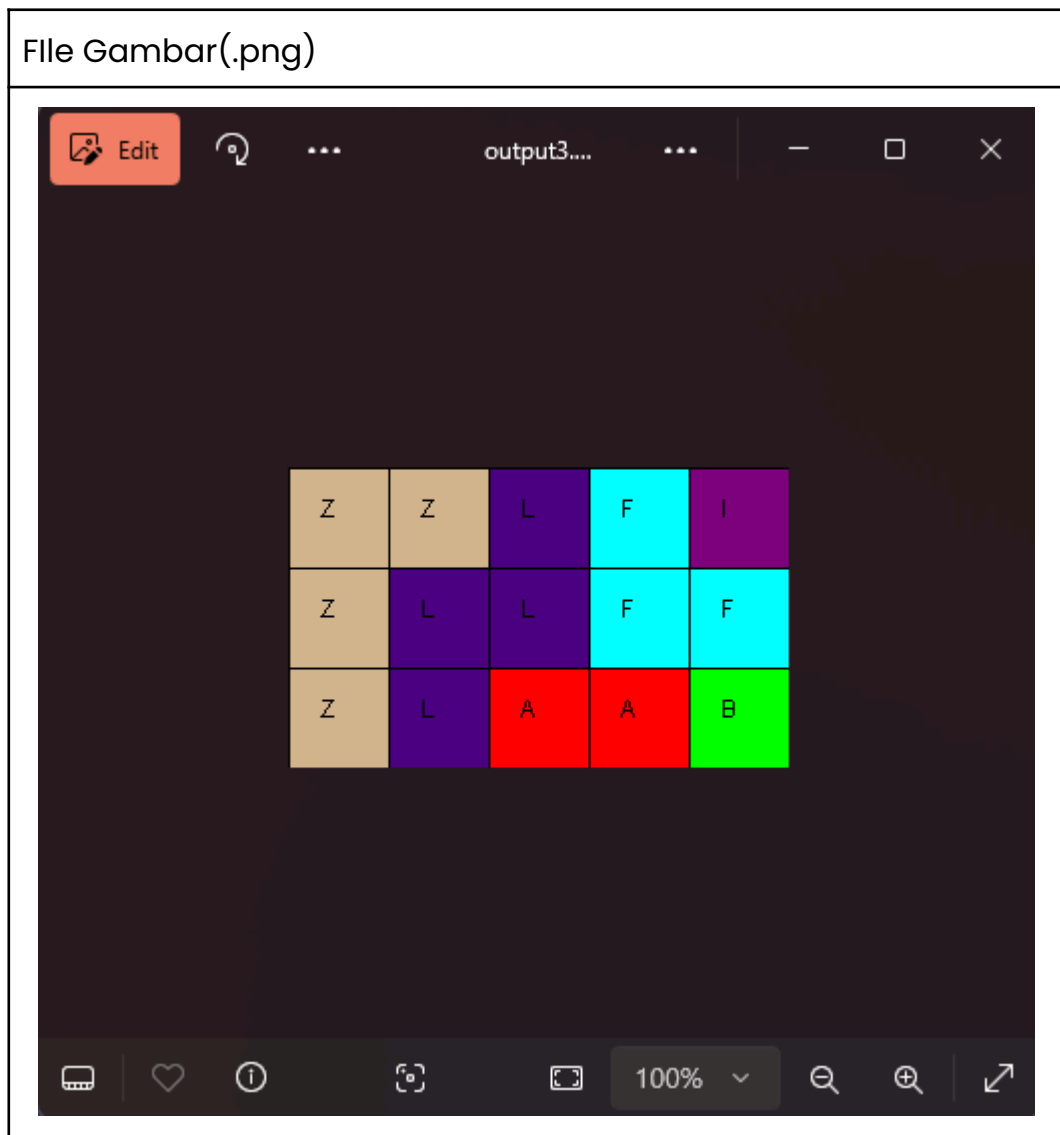
Restart App

File Teks(.txt)

```
test > output > output3.txt
1  ZZLFI
2  ZLLFF
3  ZLAAB
4  Times taken: 2 ms
5  Turns taken: 1270 turns
6
```

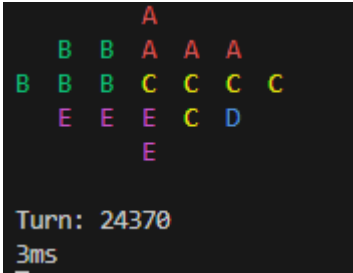
IF2211 Strategi Algoritma

Halaman 31 dari 60 halaman



4.4 Test Case 4

Masukan
3 5 6 DEFAULT ZZ

Z Z F FF I L LL L A A B
Keluaran
Terminal:

GUI:

IQ Puzzle Pro Solver

Puzzle Result!

Turns: 24370Time: 3 ms

.	.	.	A	.	.	.
.	B	B	A	A	A	.
B	B	B	C	C	C	C
.	E	E	E	C	D	.
.	.	.	E	.	.	.

Save as Text

Save as Image

Restart App

File Teks(.txt)

```
test > output >  output4.txt
```

```
1   ...A...
```

```
2   .BBAAA.
```

```
3   BBBCCCC
```

```
4   .EEECD.
```

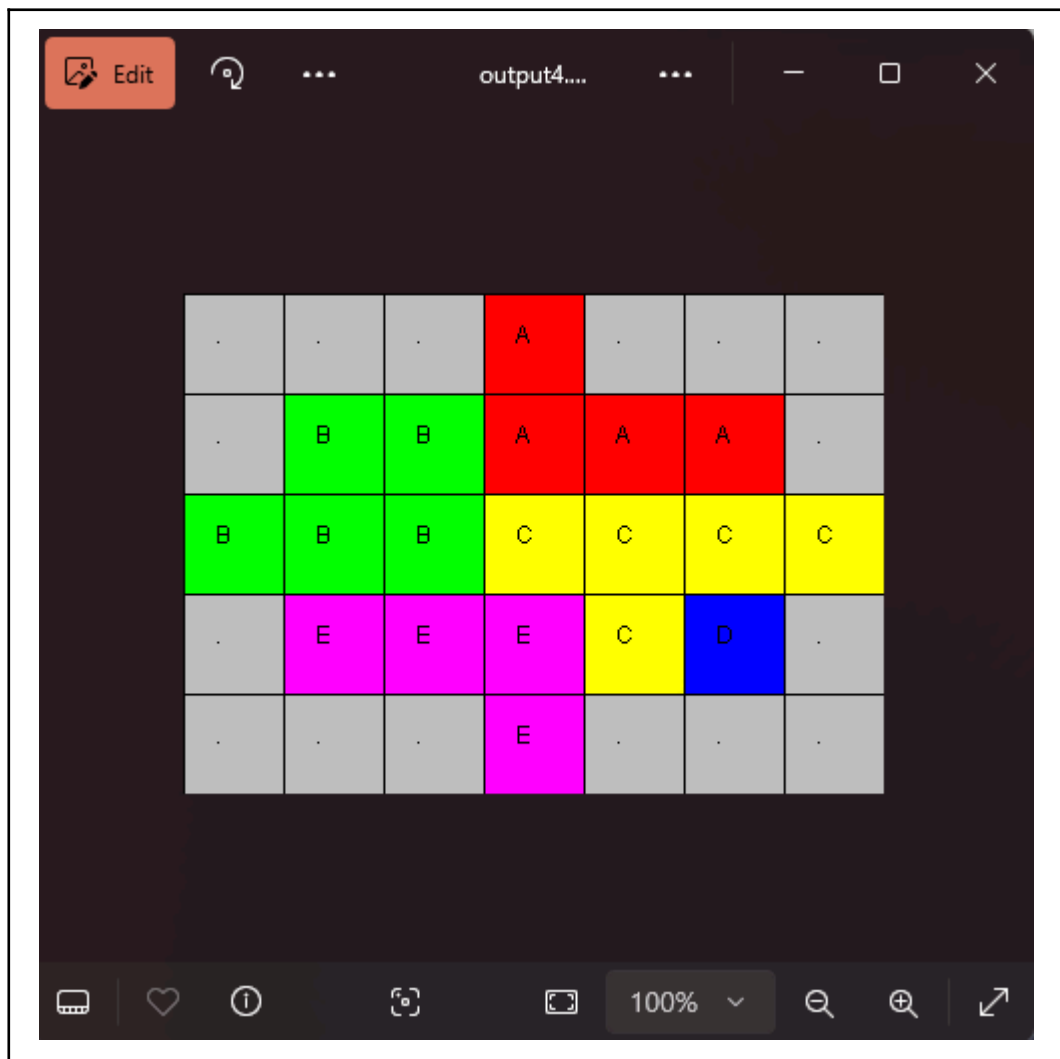
```
5   ...E...
```

```
6   Times taken: 3 ms
```

```
7   Turns taken: 24370 turns
```

```
8   |
```

File Gambar(.png)



4.5 Test Case 5


Masukan
<pre> 10 12 8 DEFAULT AAAAAAAAAA AAAAAAAAAA AAAA </pre>

AAAAAA AAAAAA AAAAAAAAAAAA AAAAAAAAAAAA AAAAAAAAAA AAA AAA AAA AAA HHH H BBB BBBBB BBBBBB RR KKKKK VVV VVV III III NNN NNN
Keluaran
Terminal:

```
R A A A A A A A A A I I
R A A A A A A A A A I I
H H H B B B A A A A A A
H B B B B B A A A A A A
B B B B B B A A A A A A
K A A A A A A A A A A A
K A A A A A A A A A A A
K A A A A A A A A A A A
K A A A V V V A A A N N
K A A A V V V A A A N N

Turn: 872481
49ms
```

GUI:


IQ Puzzle Pro Solver
—
□
×

Puzzle Result!

Turns: 872481

Time: 49 ms

R	A	A	A	A	A	A	A	A	A	I	I
R	A	A	A	A	A	A	A	A	A	I	I
H	H	H	B	B	B	A	A	A	A	I	I
H	B	B	B	B	B	A	A	A	A	A	A
B	B	B	B	B	B	A	A	A	A	A	A
K	A	A	A	A	A	A	A	A	A	A	A
K	A	A	A	A	A	A	A	A	A	A	A
K	A	A	A	A	A	A	A	A	A	N	N
K	A	A	A	V	V	V	A	A	A	N	N
K	A	A	A	V	V	V	A	A	A	N	N

Save as Text

Save as Image

Restart App

File Teks(.txt)

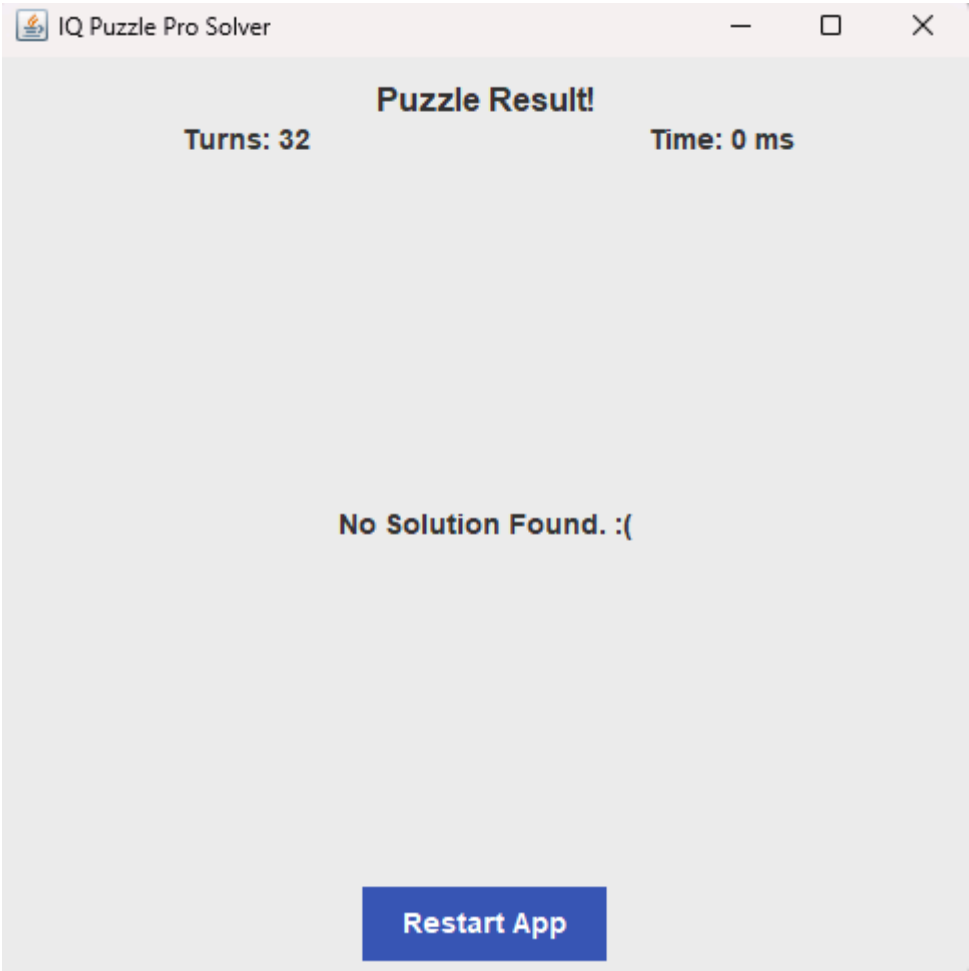
```
test > output > output5.txt
1  RAAAAAAAAAII
2  RAAAAAAAAAII
3  HHHBBBAAAAII
4  HBBBBBAAAAAA
5  BBBBBBAAAAAA
6  KAAAAAAAAAAAA
7  KAAAAAAAAAAAA
8  KAAAAAAAAAANN
9  KAAAVVVAAANN
10 KAAAVVVAAANN
11 Times taken: 49 ms
12 Turns taken: 872481 turns
13
```

File Gambar(.png)

Edit												
R	A	A	A	A	A	A	A	A	A	I	I	
R	A	A	A	A	A	A	A	A	A	I	I	
H	H	H	B	B	B	A	A	A	A	I	I	
H	B	B	B	B	B	A	A	A	A	A	A	
B	B	B	B	B	B	A	A	A	A	A	A	
K	A	A	A	A	A	A	A	A	A	A	A	
K	A	A	A	A	A	A	A	A	A	A	A	
K	A	A	A	A	A	A	A	A	A	N	N	
K	A	A	A	V	V	V	A	A	A	N	N	
K	A	A	A	V	V	V	A	A	A	N	N	

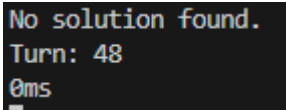
4.6 Test Case 6

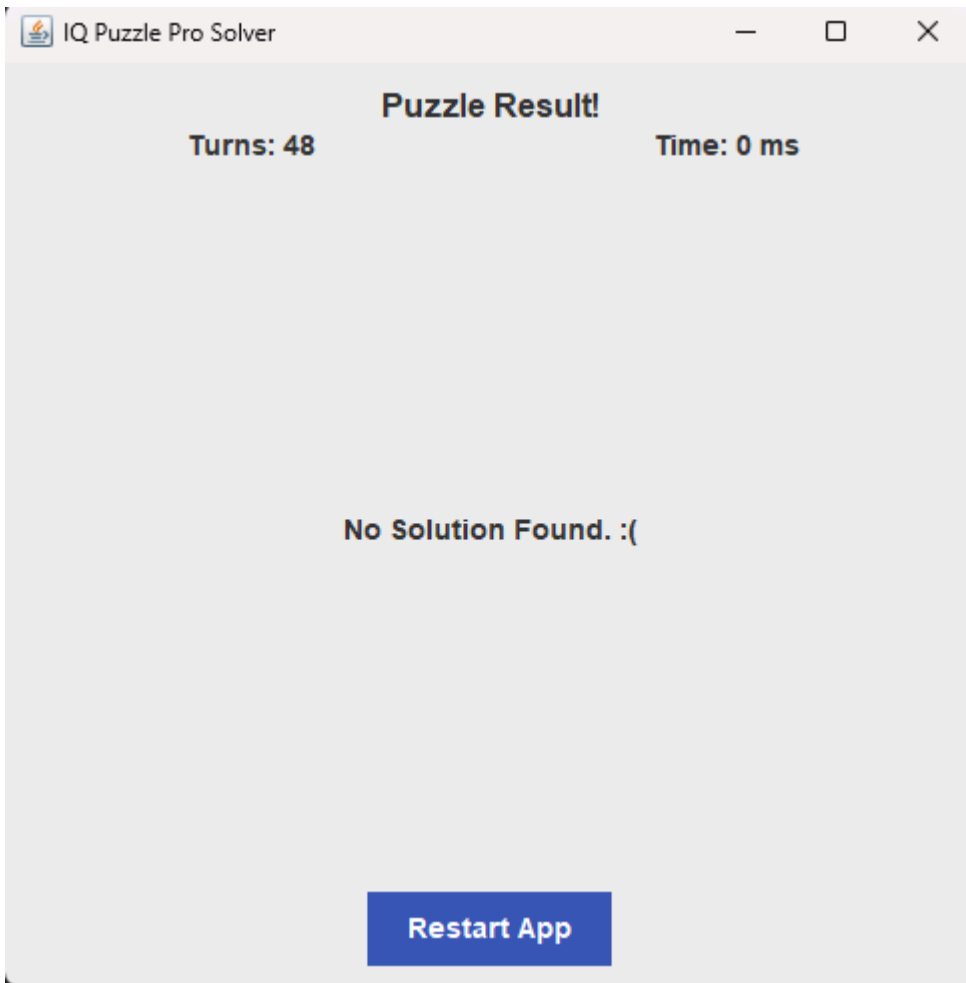
Masukan
2 2 2
DEFAULT
A
A
AA

BB
Keluaran
Terminal:
<pre>No solution found. Turn: 32 0ms □</pre>
GUI:
 <p>The screenshot shows a window titled 'IQ Puzzle Pro Solver'. Inside, the text 'Puzzle Result!' is centered at the top. Below it, 'Turns: 32' is on the left and 'Time: 0 ms' is on the right. In the center, it says 'No Solution Found. :('. At the bottom center, there is a blue button with the text 'Restart App'.</p>
File Teks(.txt)

-
File Gambar(.png)
-

4.7 Test Case 7

Masukan
2 2 2 DEFAULT AA B
Keluaran
Terminal:
 <pre>No solution found. Turn: 48 0ms</pre>
GUI:

	
File Teks(.txt)	
	-
File Gambar(.png)	
	-

4.8 Test Case 8

Masukan

5 11 12
DEFAULT
CC
C
CC
W
WW
WW
L
LLL
Z
ZZ
Z
Z
A
A
AAA
TTT
T
M
MMMM
G
GG
GG
QQ
QQ
BB
B
PP
PPP
N
NNN

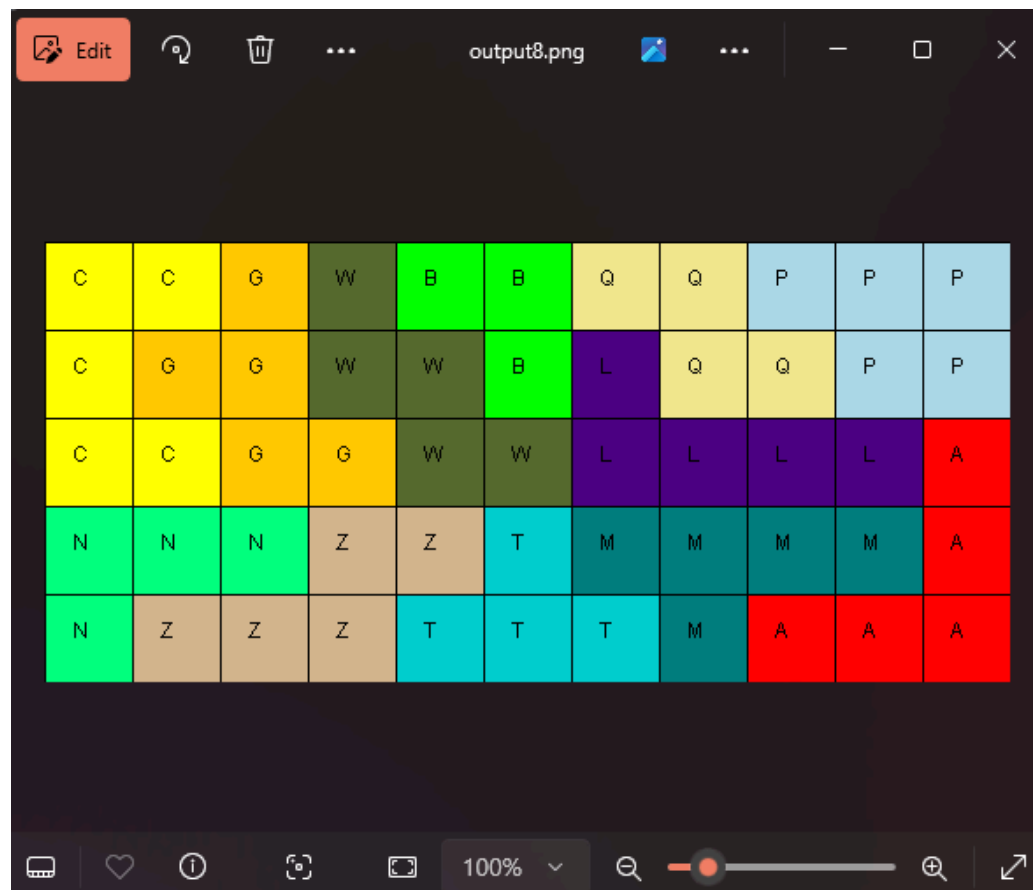
Keluaran

Terminal:

```
C C G W B B Q Q P P P
C G G W W B L Q Q P P
C C G G W W L L L L A
N N N Z Z T M M M M A
N Z Z Z T T T M A A A

Turn: 503305998364
2112865ms
```

GUI:



File Teks(.txt)

```
test > output > output8.txt
```

```
1 CCGWBBQQPPP
```

```
2 CGGWBLQQPP
```

```
3 CCGWWLLLLA
```

```
4 NNNZZTMMMMMA
```

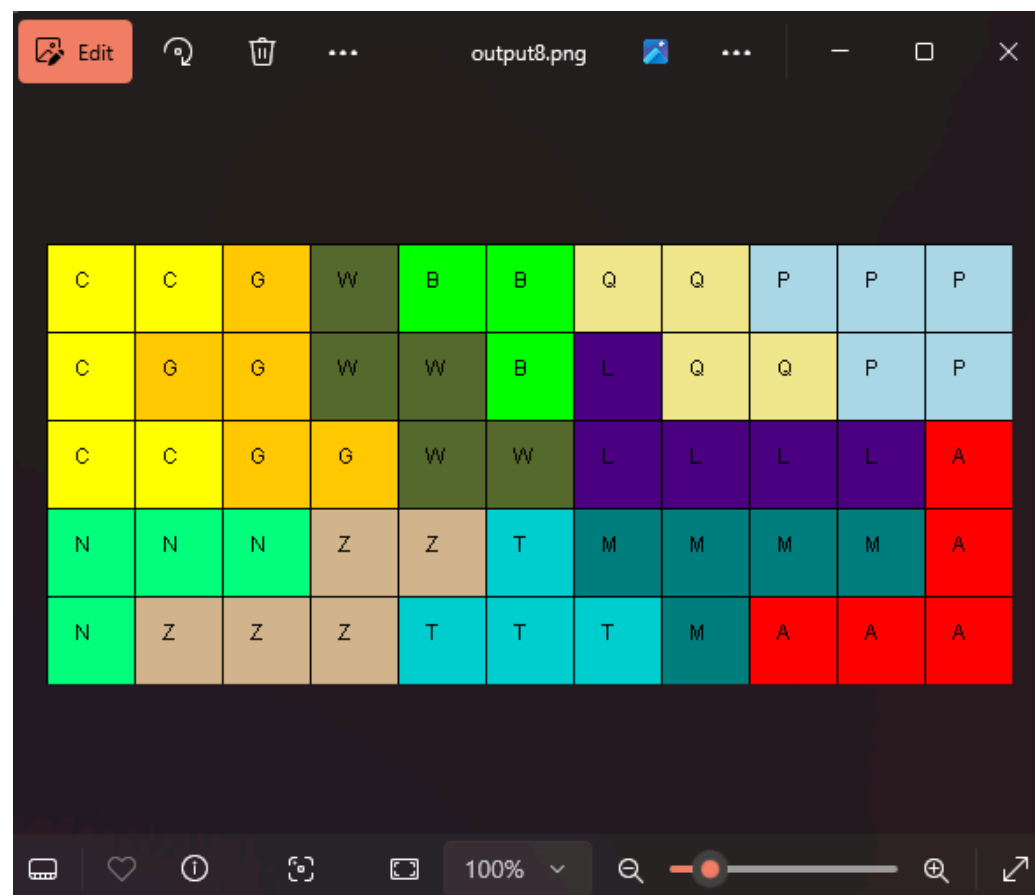
```
5 NZZZTTTMAAA
```

```
6 Times taken: 2112865 ms
```

```
7 Turns taken: 503305998364 turns
```

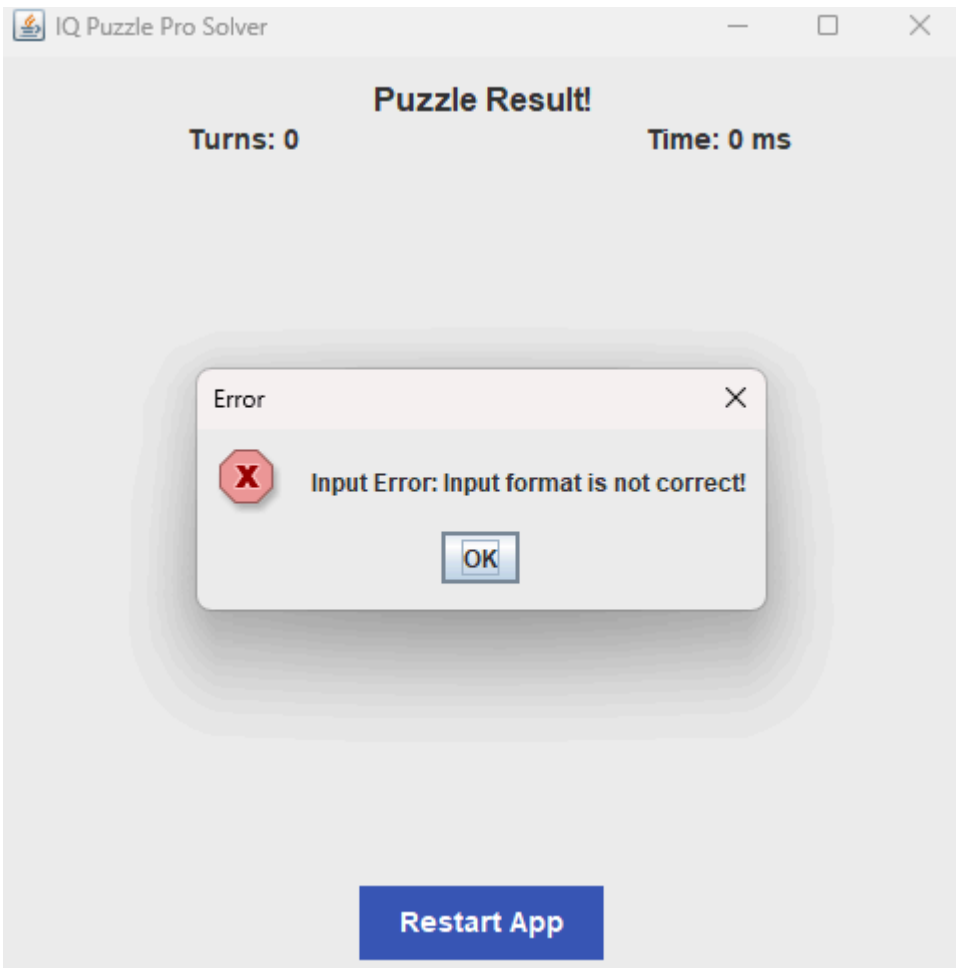
```
8
```

File Gambar(.png)



4.9 Test Case 9

Masukan
5 5 DEFAULT A AA B BB C CC D DD EE EE E FF FF F GGG
Keluaran
Terminal:
-
GUI:

	
File Teks(.txt)	
	-
File Gambar(.png)	
	-

4.10 Test Case 10

Masukan

2 2 2

AA

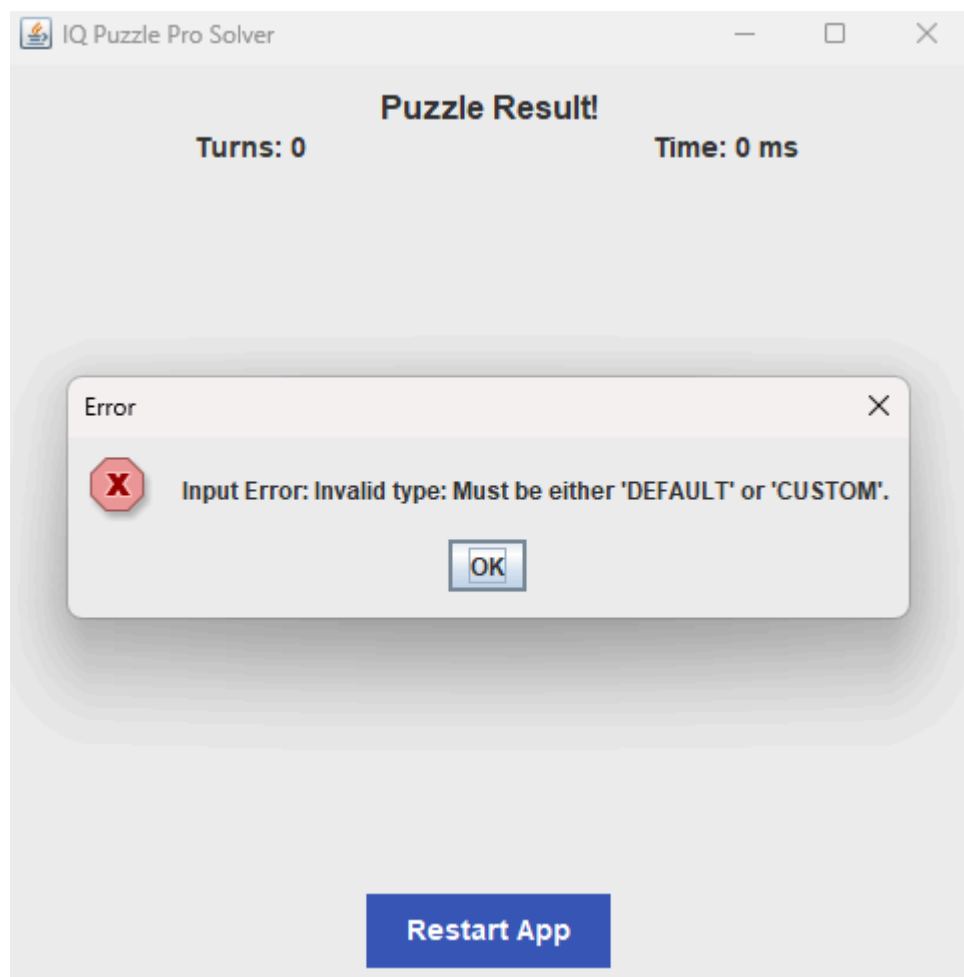
BB

Keluaran

Terminal:

-

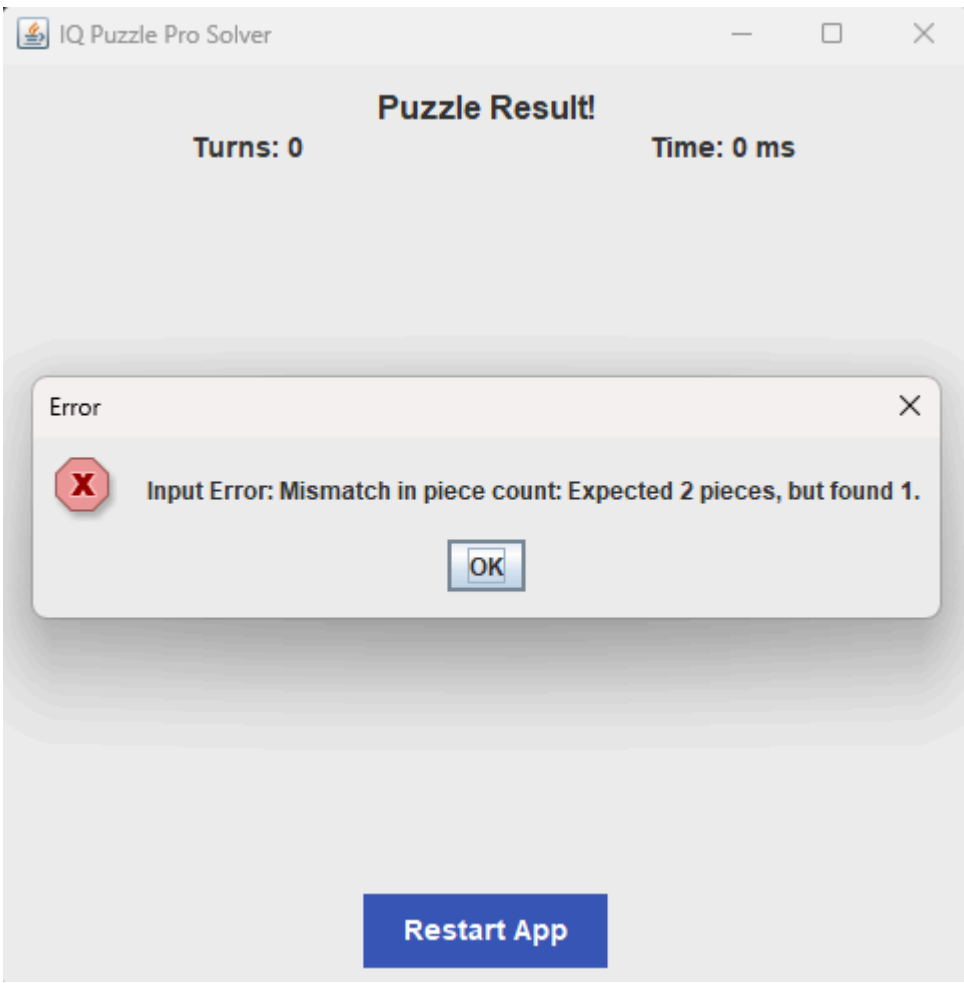
GUI:



File Teks(.txt)
-
File Gambar(.png)
-

4.11 Test Case 11

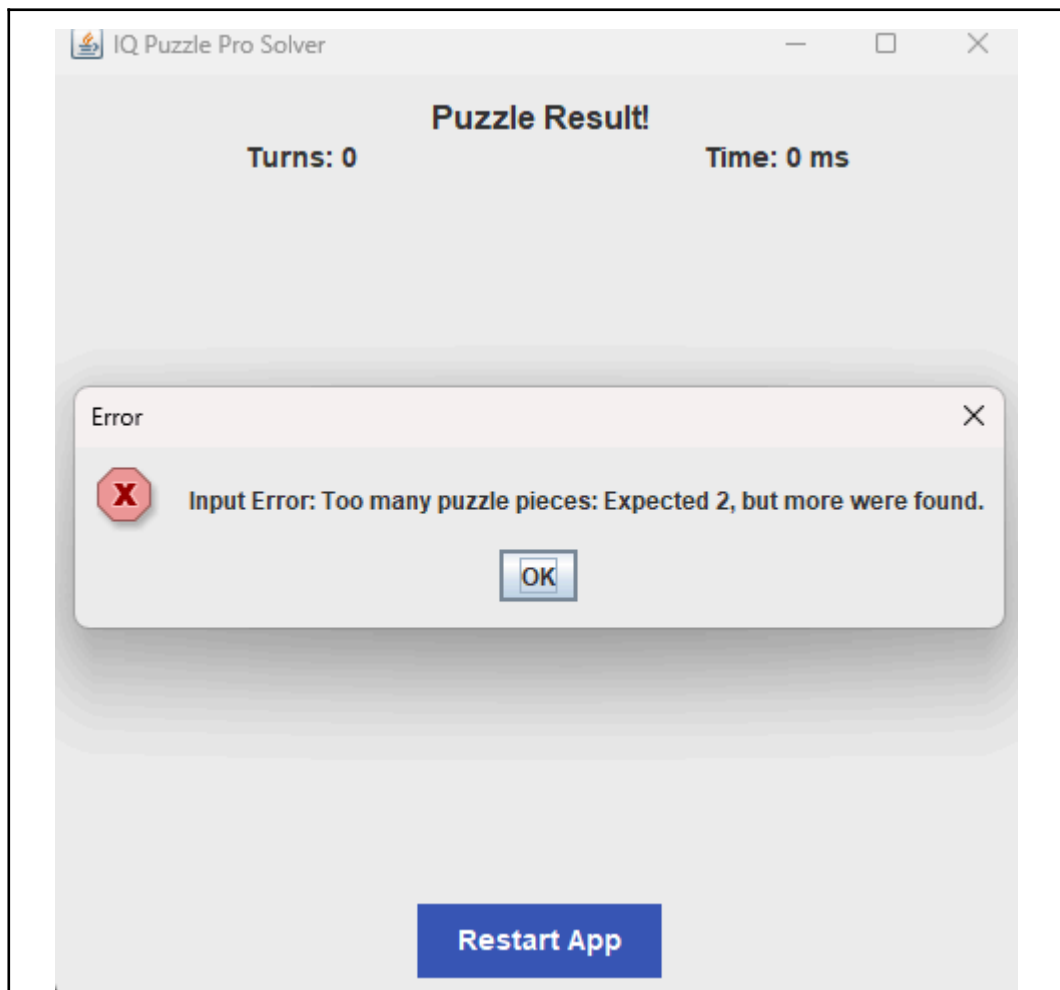
Masukan
2 2 2 DEFAULT AA
Keluaran
Terminal:
-
GUI:

	
File Teks(.txt)	
	-
File Gambar(.png)	
	-

4.12 Test Case 12

Masukan

2 2 2 DEFAULT AA B C
Keluaran
Terminal:
-
GUI:

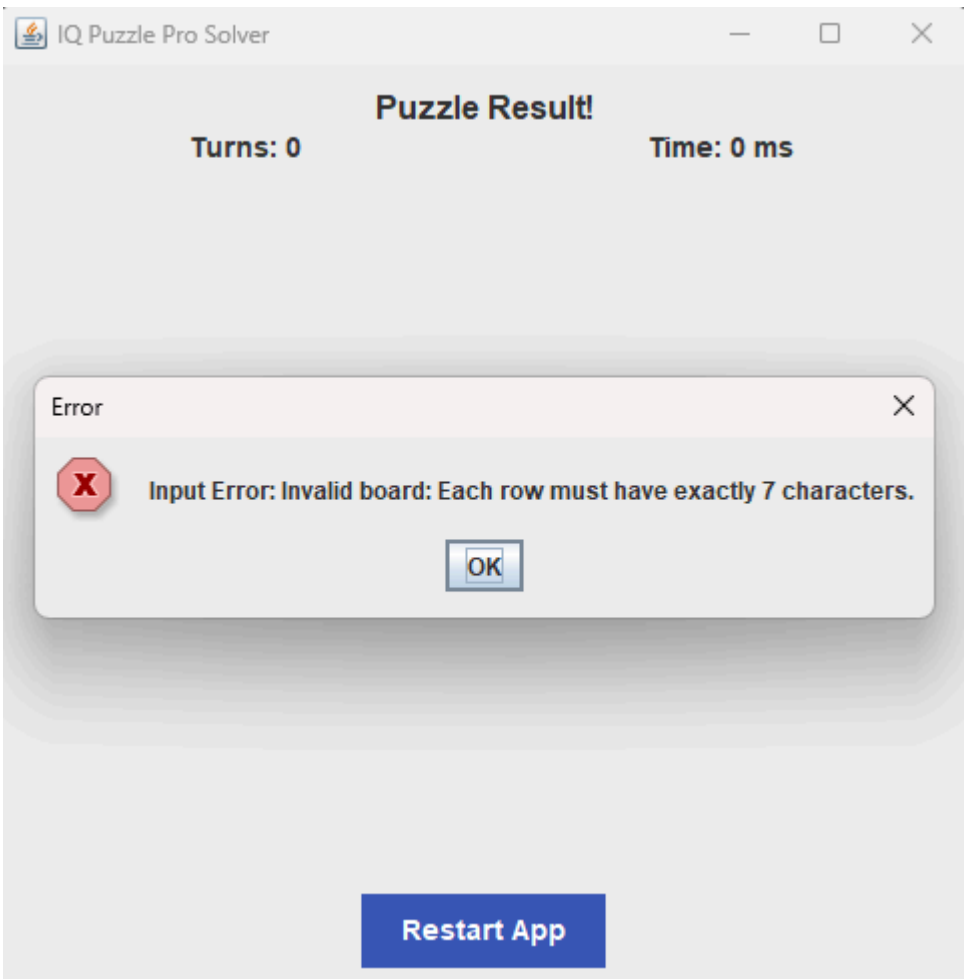


File Teks(.txt)
-
File Gambar(.png)
-

4.13 Test Case 13

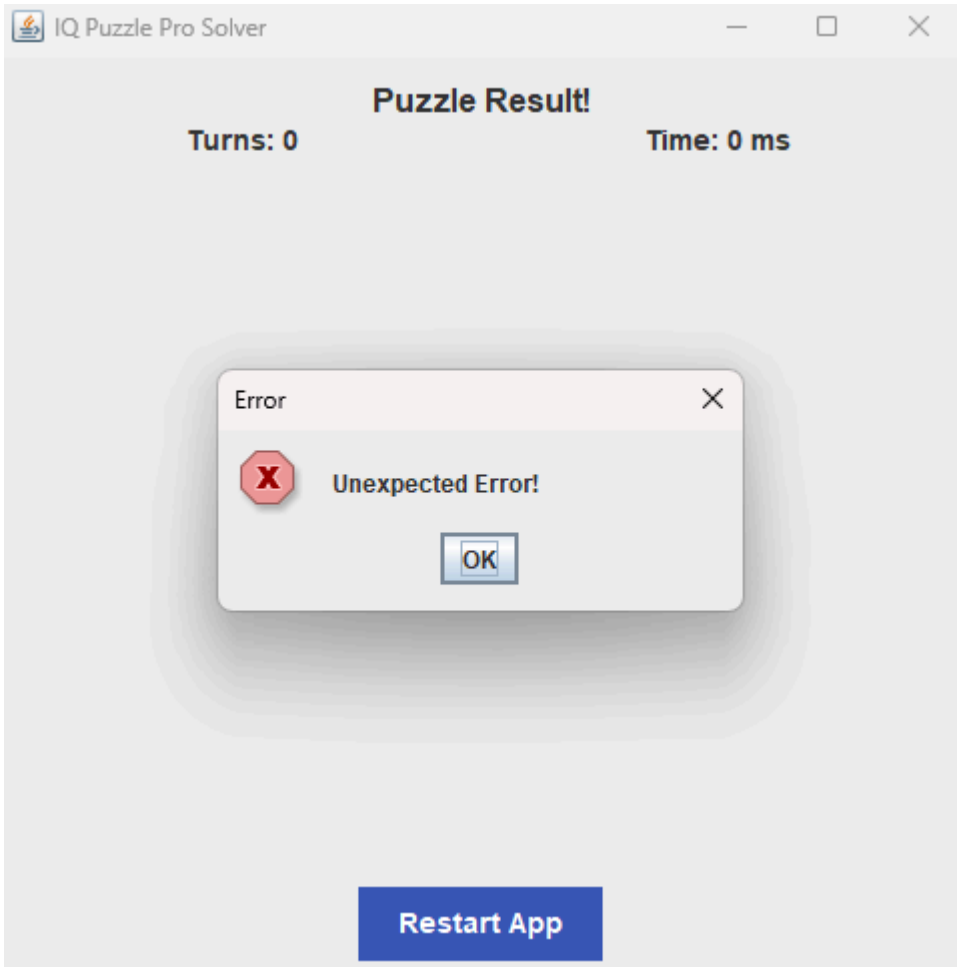
Masukan

5 7 5 CUSTOM ...X.... .XXXXX.. XXXXXXXXX .XXXXX.. ...X.... A AAA BB BBB CCCC C D EEE E
Keluaran
Terminal:
-
GUI:

	
File Teks(.txt)	
	-
File Gambar(.png)	
	-

4.14 Test Case 14

Masukan

Keluaran
Terminal:
-
GUI:

File Teks(.txt)
-

File Gambar(.png)
-

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	✓	

Repository Program: https://github.com/kin-ark/Tucill_13523152

Pustaka

Munir, R. (2025). *Algoritma brute force bagian 1*. Retrieved from [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)