TUGAS KECIL 1
STRATEGI ALGORITMA - IF2211
IQ PUZZLER PRO SOLVER MENGGUNAKAN BRUTE FORCE ALGORITHM

Dibuat Oleh:

David Bakti Lodianto - 13523083

DAFTAR ISI

BAB 1: Pendahuluan

IQ Puzzler Pro adalah suatu permainan puzzle dimana terdapat papan berbentuk persegi panjang dan potongan-potongan puzzle yang dapat memenuhi papan tersebut dengan tepat.

Permainan IQ Puzzler Pro
(Sumber: https://www.smartgamesusa.com)

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. Potongan/Piece – Piece adalah komponen yang digunakan pemain untuk mengisi papan kosong hingga terisi penuh. Setiap potongan memiliki bentuk yang unik dan semua potongan harus digunakan untuk menyelesaikan puzzle.

BAB 2: Algoritma Penyelesaian

Pendekatan penyelesaian puzzle ini diambil dari fakta bahwa jika puzzle dapat diselesaikan, maka sepotong puzzle pasti akan selalu memiliki tempat, dan lainnya dapat menyesuaikan.

Algoritma penyelesaian:

1. Ambil satu potongan puzzle paling atas yang masih belum terpasang.
2. Cocokkan pojok kiri atas potongan dengan pojok kiri atas papan.
3. Coba apakah potongan tersebut muat atau tidak.
    a. Jika iya, letakkan potongan tersebut, dan ulangi langkah 1.
    b. Jika tidak, geser ke kanan 1 petak, dan jika potongan berlebih ke kanan, pindah ke pojok kiri dan geser ke bawah 1 petak. Ulangi langkah 3.
        i. Jika potongan sudah mencapai paling ujung kanan bawah dari papan, maka putar potongan 90 derajat, dan ulangi langkah 3.
        ii. Jika potongan sudah mencapai paling ujung kanan bawah dari papan dan sudah diputar sebanyak 3 kali, maka putar sekali lagi dan cerminkan terhadap sumbu Y, dan ulangi langkah 3.
        iii. Jika potongan sudah mencapai paling ujung kanan bawah dari papan dan sudah dicerminkan, maka putar lagi potongan sebanyak 90 derajat, dan ulangi langkah 3.
        iv. Jika potongan sudah mencapai paling ujung kanan bawah dari papan dan sudah dicerminkan dan diputar lagi sebanyak 3 kali, maka lakukan langkah 3b pada potongan sebelumnya.
        v. Jika potongan pertama sudah mencapai paling ujung kanan bawah dari papan, sudah dicerminkan dan diputar, maka permainan puzzle tidak dapat diselesaikan.

Dengan algoritma ini, semua kasus akan ditinjau, karena setiap piece ditempatkan di setiap tempat pada papan, serta dirotasi dan dicerminkan.

BAB 3: Implementasi

Pemrograman aplikasi solver ini menggunakan bahasa Java, sehingga menggunakan paradigma *Object-Oriented Programming*, dibagi menjadi 6 kelas.

1. **Kelas Piece**

- **Attribute**

Nama	Tipe	Deskripsi
matrix	public boolean [][]	Array boolean 2D yang merepresentasikan piece, dengan 0 berarti tempat kosong, dan sebaliknya untuk 1.
num	public int	Angka yang merepresentasikan piece ke-berapa.
rotation	public int	Status rotasi dari piece
mirror	public boolean	Status apakah piece tercerminkan atau tidak.

```
public boolean[][] matrix;
public int num;
public int rotation;
public boolean mirror;
```

- **Method**

Nama	Tipe	Parameter	Deskripsi
Piece	Public Constructor	ArrayList<String> pieceString	Membuat piece dari list string dari input teks.

```
public Piece(ArrayList<String> pieceString) {
    int row = pieceString.size();
    int col = 0;
    for (String s : pieceString) {
        if (s.length() > col) col = s.length();
    }
    matrix = new boolean[row][col];
    for (int i = 0; i < row; i++) {
        for (int j = 0; j < col; j++) {
            matrix[i][j] = j < pieceString.get(i).length() &&
pieceString.get(i).charAt(j) != ' ';
        }
    }
```

```
    num = lineCheck(pieceString.getFirst());
    rotation = 0;
    mirror = false;
}
```

Piece	Public Constructor	boolean[][] mat, int n, int r, boolean m	Membuat piece dari atribut-atribut pada parameter.

```
public Piece(boolean[][] mat, int n, int r, boolean m){
    matrix = mat;
    num = n;
    rotation = r;
    mirror = m;
}
```

getRow	Public int		Mengambil jumlah baris dari matriks

```
public int getRow() {
    return matrix.length;
}
```

getCol	Public int		Mengambil jumlah kolom dari matriks

```
public int getCol() {
    return matrix[0].length;
}
```

lineCheck	Public static int	String s	Mengembalikan angka urutan piece ke berapa berdasarkan abjad pada piece.

```
public static int lineCheck(String s) {
    for (int i = 0; i < s.length(); i++){
        if(s.charAt(i) >= 65 && s.charAt(i) <= 90){
            return s.charAt(i) - 'A' + 1;
        }
    }
    return -1;
}
```

rotate	Public boolean		Merrotasi piece sebanyak 90 derajat searah

			jarum jam.

```
public Piece rotate () {
    // rotate 90 degrees
    int r = this.getRow();
    int c = this.getCol();
    // Step 1: Transpose the matrix
    boolean[][] res = new boolean[c][r];
    for (int i = 0; i < c; i++) {
        for (int j = 0; j < r; j++) {
            // Swap matrix[i][j] with matrix[j][i]
            res[i][j] = matrix[j][i];
        }
    }

    // Step 2: Reverse each row
    for (int i = 0; i < c; i++) {
        int start = 0;
        int end = r - 1;
        while (start < end) {
            // Swap elements in the row
            boolean temp = res[i][start];
            res[i][start] = res[i][end];
            res[i][end] = temp;
            start++;
            end--;
        }
    }
    return new Piece(res, num, (rotation + 1) % 4, mirror);
}
```

mirror	Public boolean	Piece piece, int r, int c	Mencerminkan piece terhadap sumbu Y

```
public Piece mirror () {
    // mirror through the Y axis
    int r = this.getRow();
    int c = this.getCol();
    boolean[][] res = new boolean[r][c];
    for (int i = 0; i < r; i++) {
        int start = 0;
        int end = c - 1;
        while (start < end) {
            // Swap elements in the row
            res[i][start] = matrix[i][end];
            res[i][end] = matrix[i][start];
            start++;
            end--;
        }
        if (start == end) res[i][start] = matrix[i][start];
    }
    return new Piece(res, num, rotation, !mirror);
}
```

reset	Public void	Piece piece, int r,	Mengembalikan

		int c	rotasi dan pencerminan piece pada piece yang sudah terrotasi 270 derajat dan dicerminkan

```
public Piece reset() { // only to be used on rotated 270 + mirrored!!
    return this.rotate().mirror();
}
```

printPiece	Public void		Mencetak piece pada terminal.

```
public void printPiece() {
    int r = this.getRow();
    int c = this.getCol();
    // Step 1: Transpose the matrix
    for (int i = 0; i < r; i++) {
        for (int j = 0; j < c; j++) {
            if (matrix[i][j]) System.out.print(num);
            else System.out.print(" ");
        }
        System.out.println();
    }
}
```

2. Kelas Board

- **Attribute**

Nama	Tipe	Deskripsi
matrix	public int [][]	Array Integer 2D yang menyimpan nilai piece yang menempati posisinya.

```
public int[][] matrix;
```

- **Method**

Nama	Tipe	Parameter	Deskripsi
Board	Public Constructor	int row, int col	Membuat board kosong dengan dimensi row x col.

```
public Board(int row, int col) {
   matrix = new int[row][col];
}
```

getRow	Public int		Mengambil jumlah baris dari matriks

```
public int getRow() {
   return matrix.length;
}
```

getCol	Public int		Mengambil jumlah kolom dari matriks

```
public int getCol() {
   return matrix[0].length;
}
```

isSolved	Public boolean		Mengecek apakah board terpenuhi.

```
public boolean isSolved(){
   for (int i = 0; i < this.getRow(); i++) {
       for (int j = 0; j < this.getCol(); j++) {
           if (matrix[i][j] == 0){
               return false;
           }
       }
   }
   return true;
}
```

isFit	Public boolean	Piece piece, int r, int c	Mengecek apakah piece dapat muat di posisi r, c pada board. (pojok kiri atas piece ditempatkan di (r, c))

```
public boolean isFit(Piece piece, int r, int c) {
   // r = row number of the top-left corner of piece
   // c = column number of the top-left corner of piece
```

```
    for (int i = 0; i < piece.getRow(); i++) {
        for (int j = 0; j < piece.getCol(); j++) {
            if (r + i >= this.getRow() || c + j >=
this.getCol()){
                return false;
            }
            if (matrix[r + i][c + j] != 0 &&
piece.matrix[i][j]){
                return false;
            }
        }
    }
    return true;
}
```

addPiece	Public void	Piece piece, int r, int c	Memasukkan piece ke board, dengan pojok kiri atas piece ditempatkan di (r, c).

```
public void addPiece(Piece piece, int r, int c) {
    for (int i = 0; i < piece.getRow(); i++) {
        for (int j = 0; j < piece.getCol(); j++) {
            if (piece.matrix[i][j]) matrix[r + i][c + j] =
piece.num;
        }
    }
}
```

removePieceHelper	Private void	int r, int c	Membantu pelepasan piece

```
private void removePieceHelper(int r, int c) {
    int toRemove = matrix[r][c];
    matrix[r][c] = 0;
    if (r < this.getRow() - 1 && matrix[r + 1][c] == toRemove)
{removePieceHelper(r + 1, c); }
    if (r > 0 && matrix[r - 1][c] == toRemove)
{removePieceHelper(r - 1, c); }
    if (c < this.getCol() - 1 && matrix[r][c + 1] == toRemove)
{removePieceHelper(r, c + 1); }
    if (c > 0 && matrix[r][c - 1] == toRemove)
{removePieceHelper(r, c - 1); }
}
```

removePiece	Public void	Piece piece	Melepas piece dari board

```
public void removePiece(Piece piece) {
    for (int i = 0; i < this.getRow(); i++) {
```

```
        for (int j = 0; j < this.getCol(); j++) {
            if(matrix[i][j] == piece.num){
                removePieceHelper(i, j);
            }
        }
    }
}
```

printBoard	Public void		Mencetak isi board

```
public void printBoard() {
    IO helper = new IO();
    for (int i = 0; i < this.getRow(); i++) {
        for (int j = 0; j < this.getCol(); j++) {
            char toPrint = (char) ('A' + (matrix[i][j] - 1));
            helper.printColor(toPrint, matrix[i][j] - 1);
        }
        System.out.println();
    }
}
```

3. Kelas Game

- **Attribute**

Nama	Tipe	Deskripsi
runtime	public long	Waktu pencarian solusi (dalam milisekon).
iterations	public long	Banyaknya iterasi kasus penyelesaian sebelum solusi ditemukan.
solved	public boolean	Status penanda solusi sudah ditemukan.
board	public Board	Board dari permainan
pieces	public Piece[]	Array pieces dari permainan
invalid	public boolean	Penanda config invalid
deadend	private boolean	Penanda piece sudah tidak dapat dilanjutkan (sudah dicerminkan dan terrotasi 3 kali dan di pojok kanan bawah board).

```
public long runtime;
public long iterations;
public boolean solved;
public Board board;
public Piece[] pieces;
public boolean invalid;
private boolean deadend;
```

- **Method**

Nama	Tipe	Parameter	Deskripsi
Game	Public Constructor	Piece[] p, Board b	Membuat game dengan atribut Piece-piece p dan Board b.

```
public Game(Piece[] p, Board b) {
    board = b;
    pieces = p;
    // Start the timer (in milliseconds)
}
```

Game	Public Constructor		Membuat game invalid.

```
public Game(Piece[] p, Board b) {
    board = b;
    pieces = p;
    // Start the timer (in milliseconds)
}
```

solve	Public void		Mencari solusi, dan merekam banyak iterations dan runtime program.

```
public void solve() {
    iterations = 0;
    long startTime = System.currentTimeMillis();
    solveRecurse(0, 0, 0);
    long endTime = System.currentTimeMillis();
    runtime = endTime - startTime;
}
```

solveRecurse	Private void	int idx, int r, int c	Method pembantu solve, untuk piece

			ke-idx dan posisi r, c secara rekursif.

```
private void solveRecurse(int idx, int r, int c) {
    if (solved || deadend) {
        return;
    }
    if (r + pieces[idx].getRow() - 1 < board.getRow()) {
        if (c + pieces[idx].getCol() - 1 < board.getCol()) {
            if (board.isFit(pieces[idx], r, c)) {
                board.addPiece(pieces[idx], r, c);
                if (idx != pieces.length - 1) {
                    solveRecurse(idx + 1, 0, 0);
                }
                if (board.isSolved()) {
                    solved = true;
                    return;
                }
                board.removePiece(pieces[idx]);
                deadend = false;
            }
            solveRecurse(idx, r, c + 1);
        }
        solveRecurse(idx, r + 1, 0);
    }
    else {
        if (pieces[idx].rotation < 3) {
            pieces[idx] = pieces[idx].rotate();
        } else if (!pieces[idx].mirror) {
            pieces[idx] = pieces[idx].rotate();
            pieces[idx] = pieces[idx].mirror();
        }
        else {
            deadend = true;
            iterations++;
            pieces[idx] = pieces[idx].reset();
            return;
        }
        solveRecurse(idx, 0, 0);
    }
}
```

4. Kelas GUI

- **Attribute**
  Tidak ada atribut untuk kelas ini.

- **Method**

Nama	Tipe	Parameter	Deskripsi
show	Public static void		Meluncurkan aplikasi

start	Public void	Stage primaryStage	Membuat stage dan scene dari aplikasi utama

5. Kelas IO

- **Atribut**

Nama	Tipe	Deskripsi
escCodes	private final String[]	Array string berisikan ANSI escape code untuk mencetak teks berwarna pada terminal
resetCode	private final String	ANSI reset code untuk mengembalikan formatting print teks terminal.
hexCodes	private final int[]	Array integer berisikan nilai warna hex untuk mewarnai gambar pada board.

- **Method**

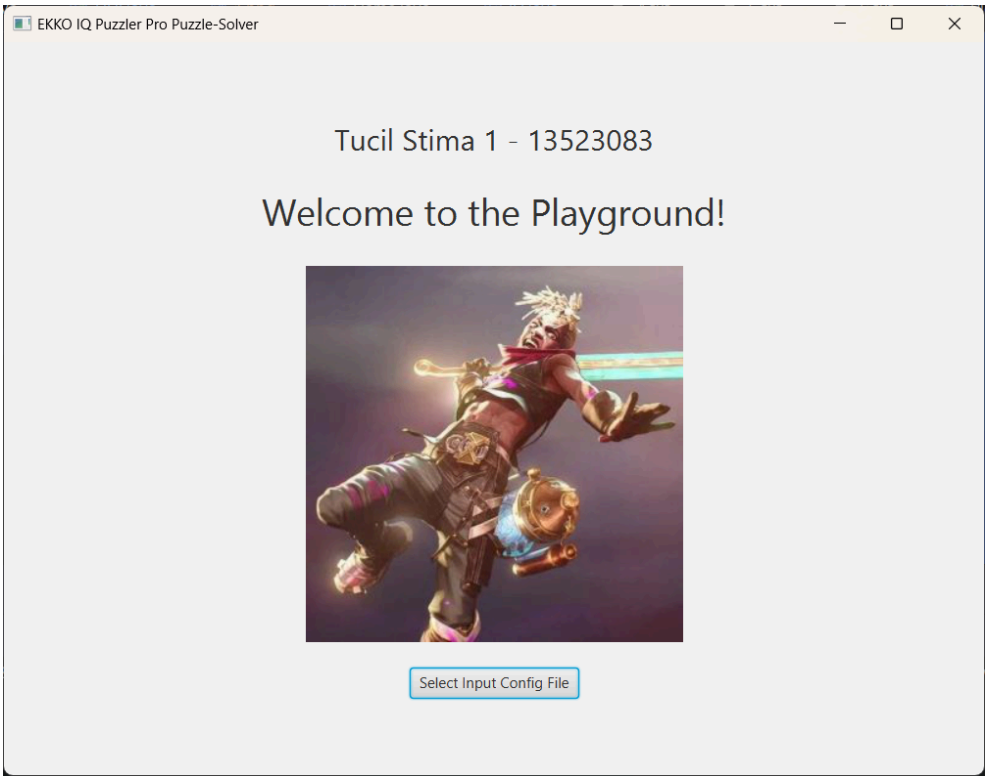
Nama	Tipe	Parameter	Deskripsi
readConfigFile	Public static Game	File file	Membaca file input .txt dan mengembalikan objek game
printColor	Public void	char a, int colorCode	Mencetak karakter a pada terminal dengan index warna colorCode
saveText	Public void	Board board, String fileName	Menuliskan hasil penyelesaian board pada file dengan nama fileName.
generateImage	Public BufferedImage	Board board	Menghasilkan gambar dari board, menggunakan

			warna hexCodes.
imageSave	Public void	BufferedImage img, String fileName	Menyimpan gambar img ke file dengan nama fileName

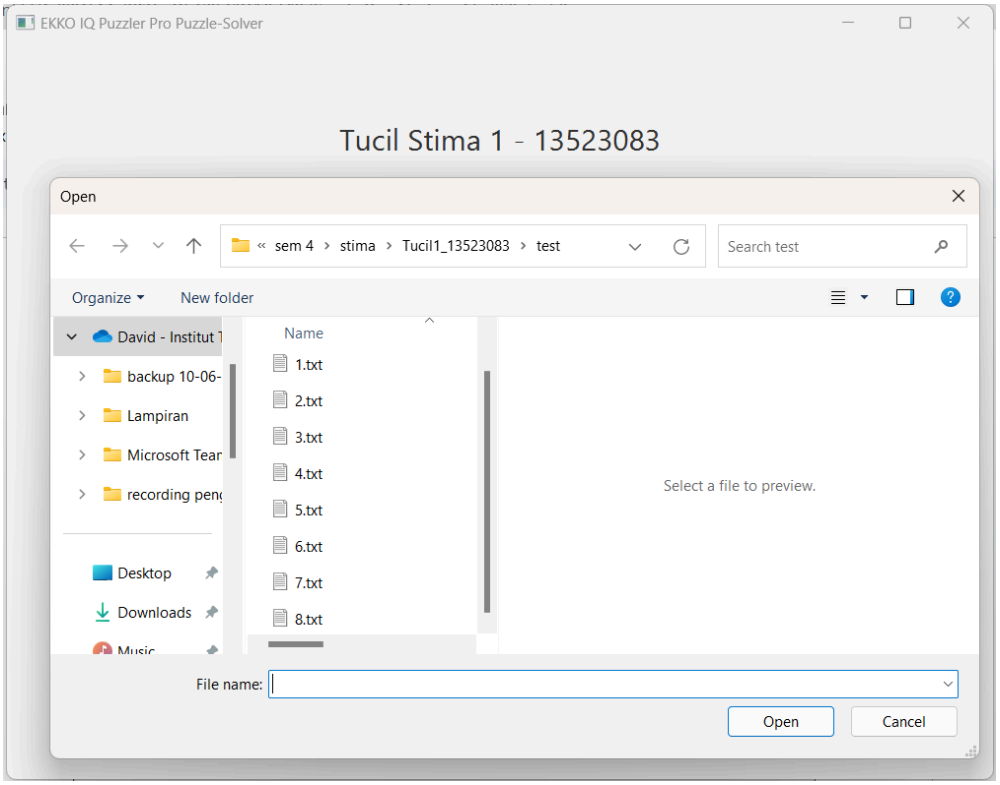
6. **Kelas Main**
   Kelas ini memiliki 1 method – main, untuk dijalankan sebagai kelas utama.

User Interface

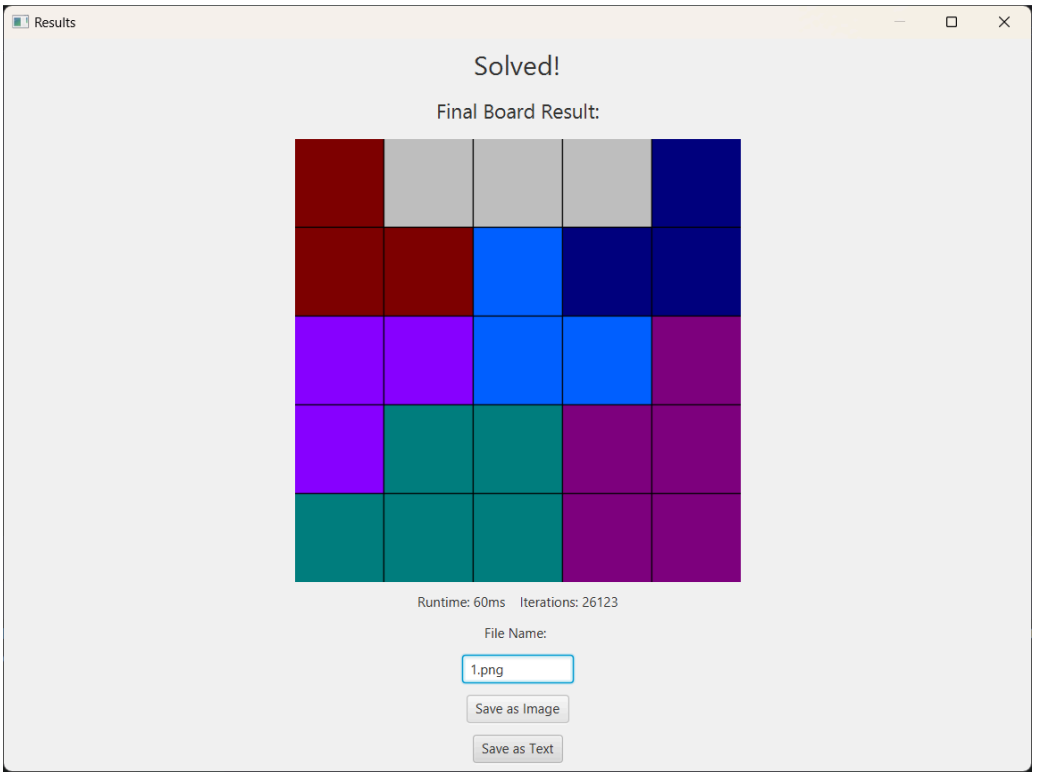
1. Tampilan Utama



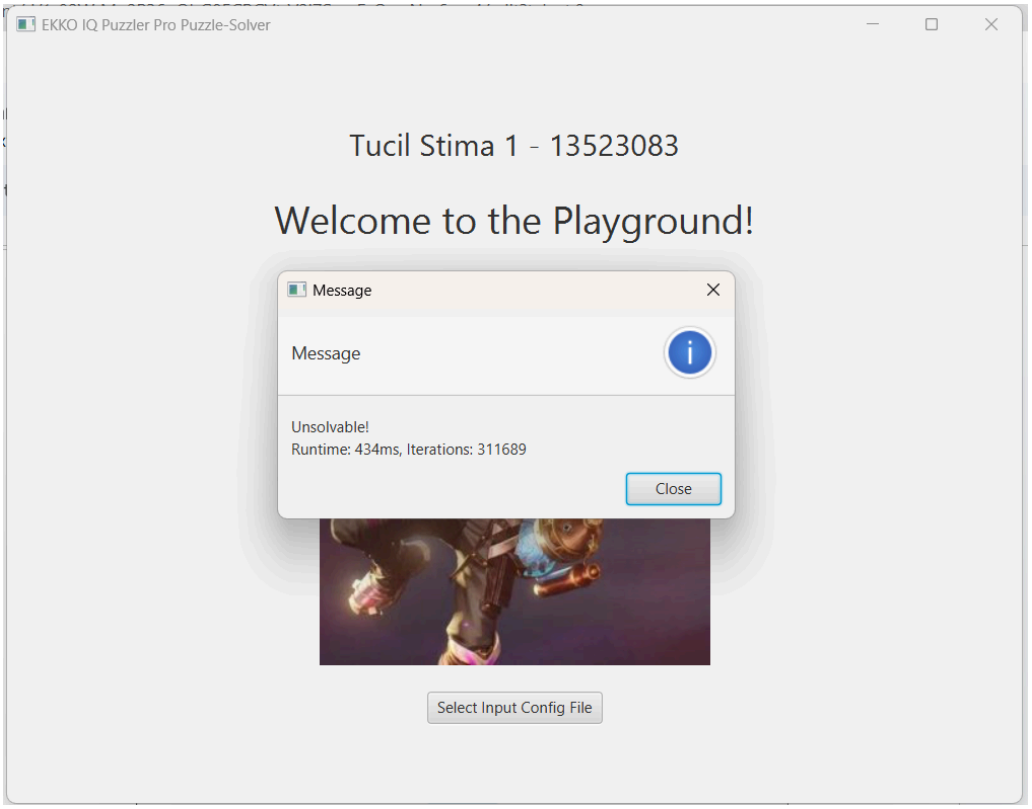
2. Tampilan pemilihan file



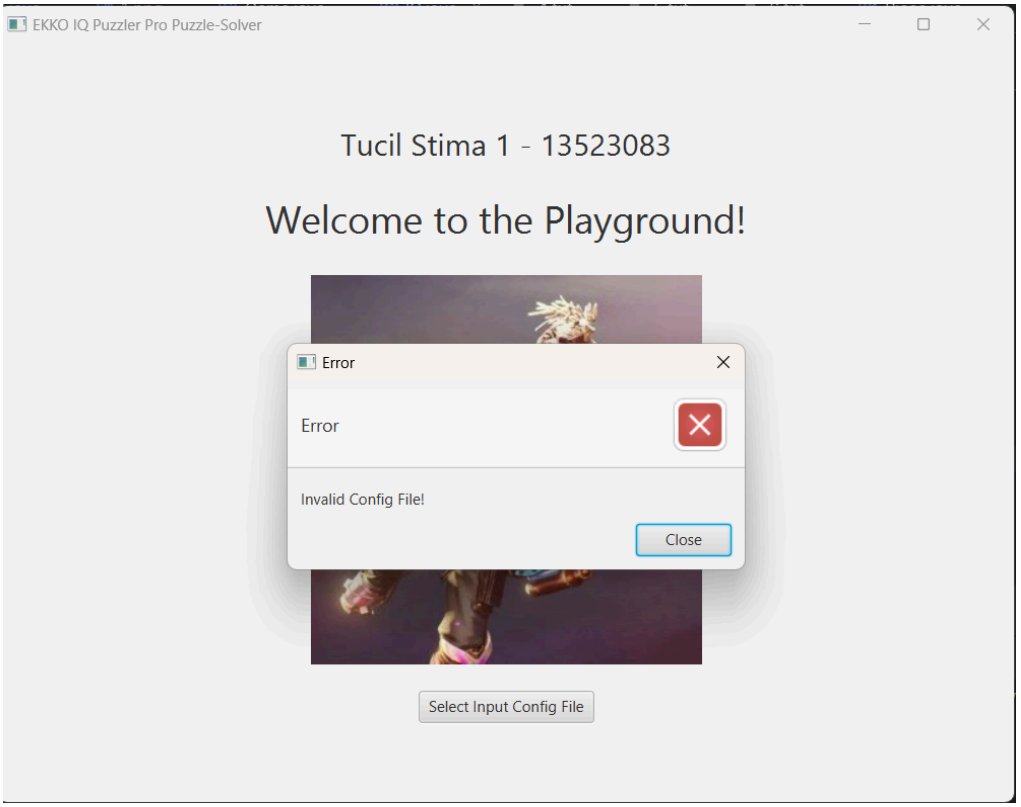
3. Tampilan solusi ditemukan



4. Tampilan solusi tidak ditemukan



5. Tampilan file konfigurasi .txt invalid

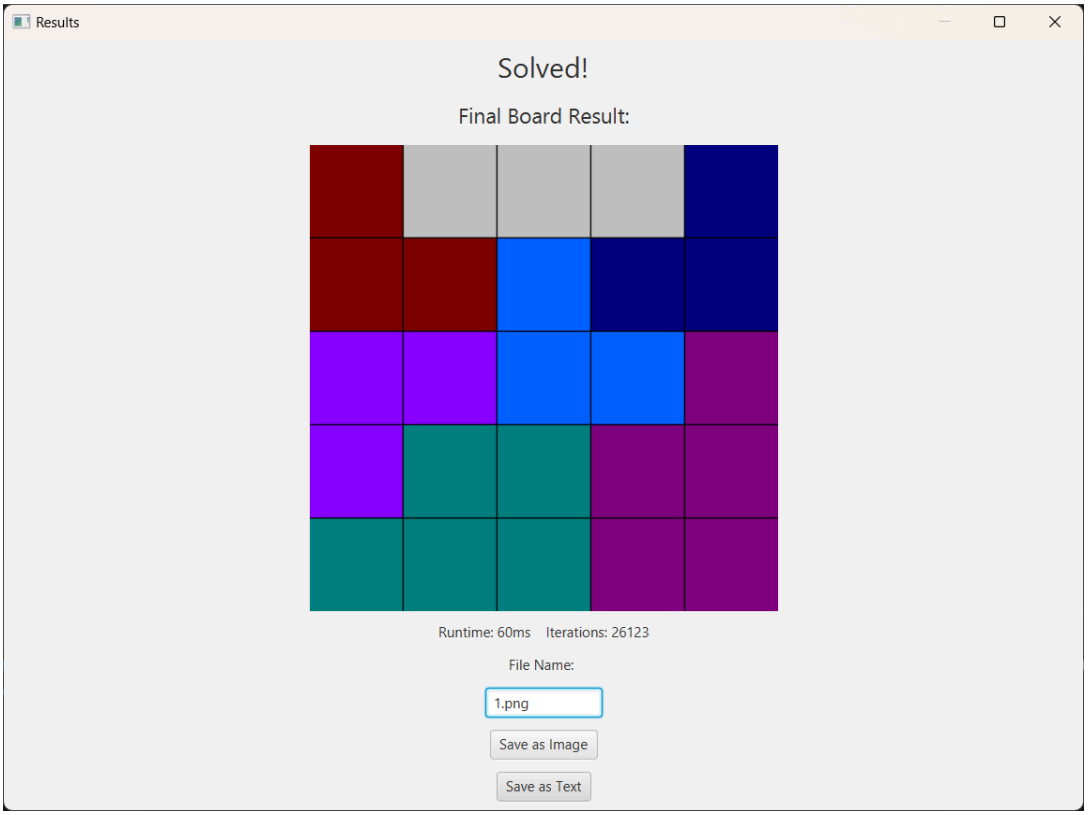


BAB 4: Testing

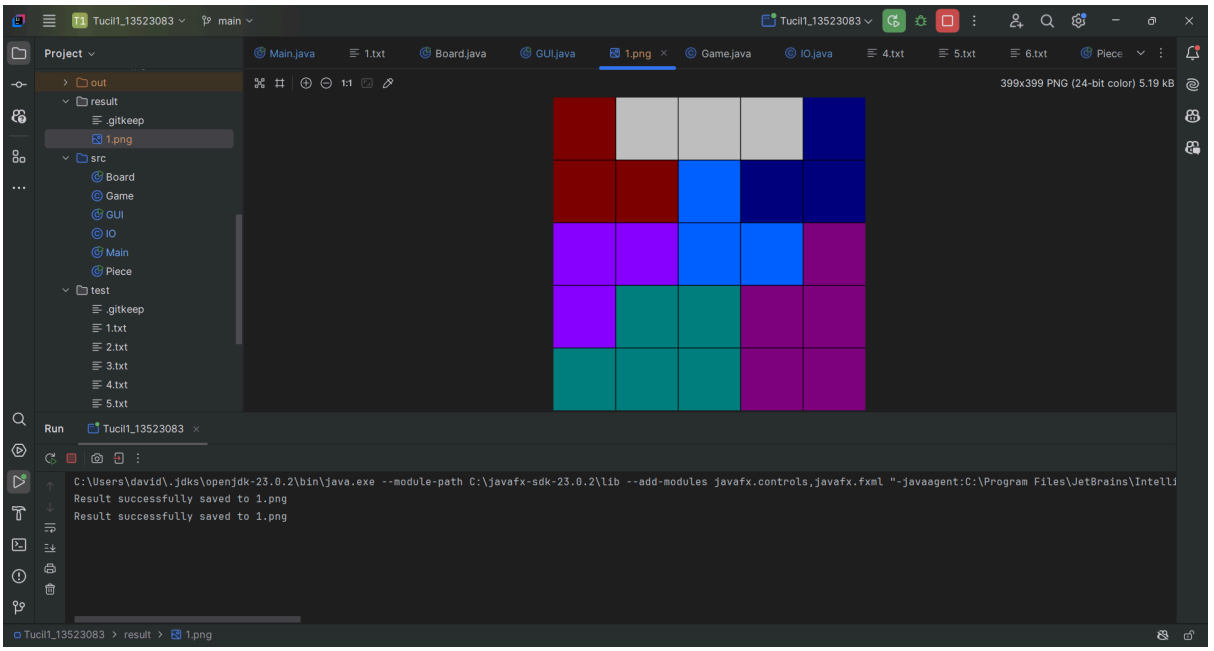
1. File 1.txt

```
5 5 7
DEFAULT
A
AA
B
BB
CC
C
D
DD
EE
EEE
FF
FFF
GGG
```

Hasil:



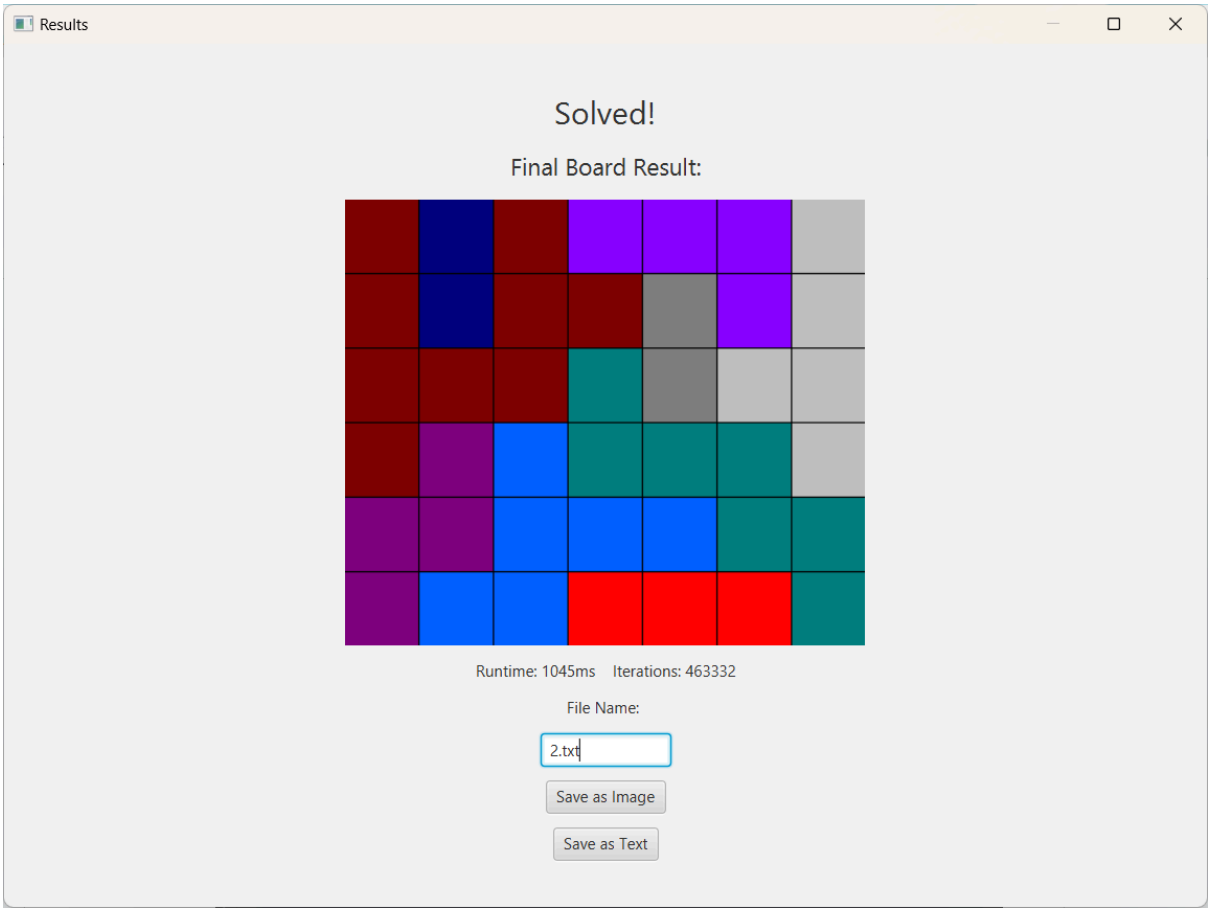
Hasil Penyimpanan Gambar:



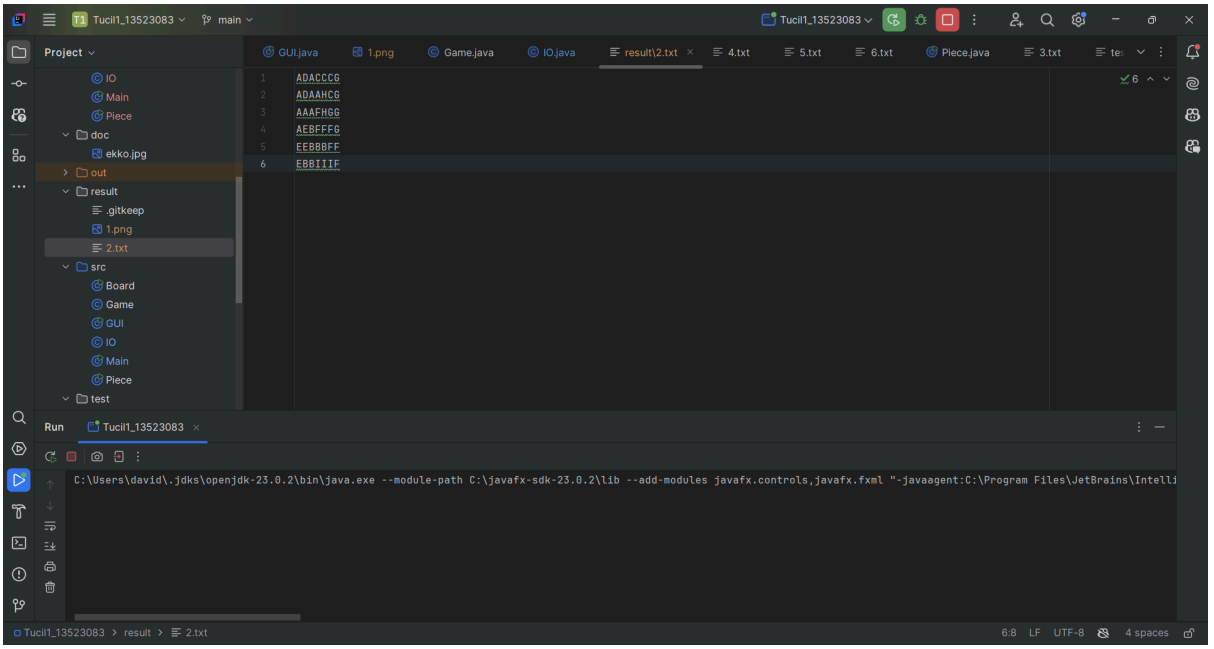
2. File 2.txt

```
6 7 9
DEFAULT
A A
A AA
AAA
A
B
BBB
BB
CC
C
C
DD
EE
EE
 FF
FF
F
FF
 G
GGGG
HH
III
```

Hasil:



Hasil penyimpanan teks:



3. File 3.txt

```
5 7 5
DEFAULT
A
```

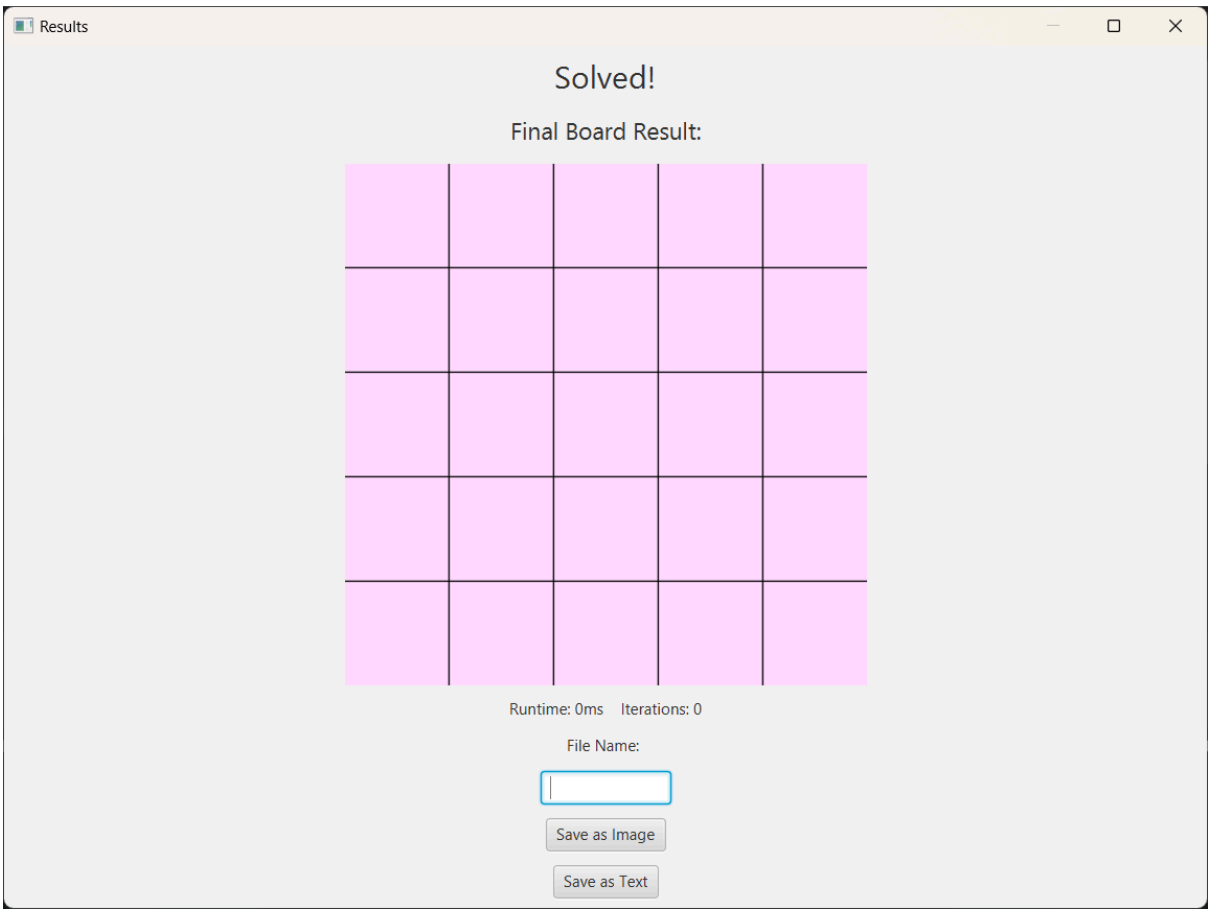
```
AAA
BBB
 BBBB
C
CCC
DDDDDDD
D     DD
DDDDDD
EEEE
```

Hasil:

4. File 4.txt

```
5 5 1
DEFAULT
XXXXX
XXXXX
XXXXX
XXXXX
XXXXX
```

Hasil:

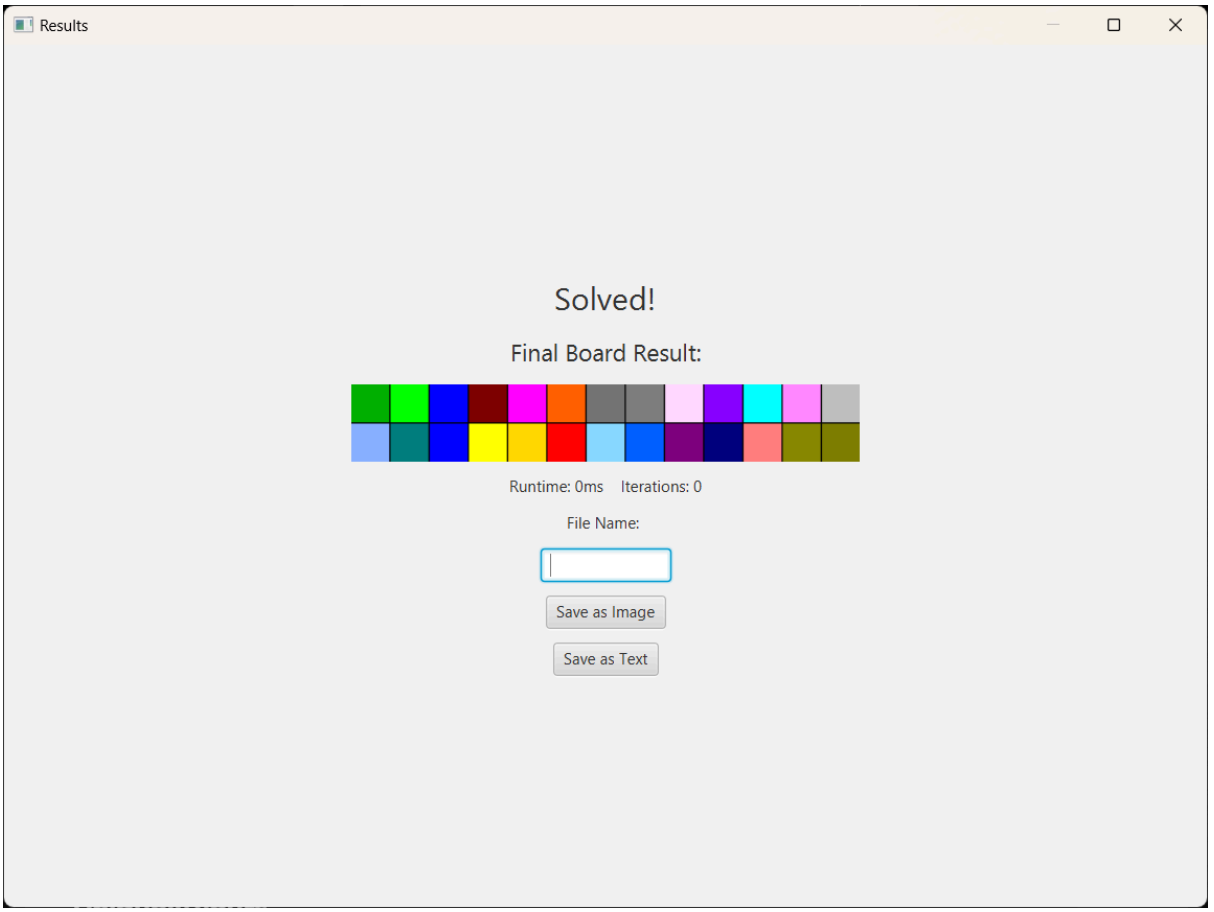


5. File 5.txt

```
2 13 26
DEFAULT
W
J
O
A
M
T
Z
H
X
C
N
Q
G
Y
F
L
K
S
I
U
```

```
B
E
D
P
V
R
```

Hasil:



6. File 6.txt

```
6 8 9
DEFAULT
AAA
AA
BBB
C
CC
C
D
DDD
E
EEE
F
FF
```

```
F
FF
F
FF
F
G G G G
GGGGGGG
HHH
H
III
```

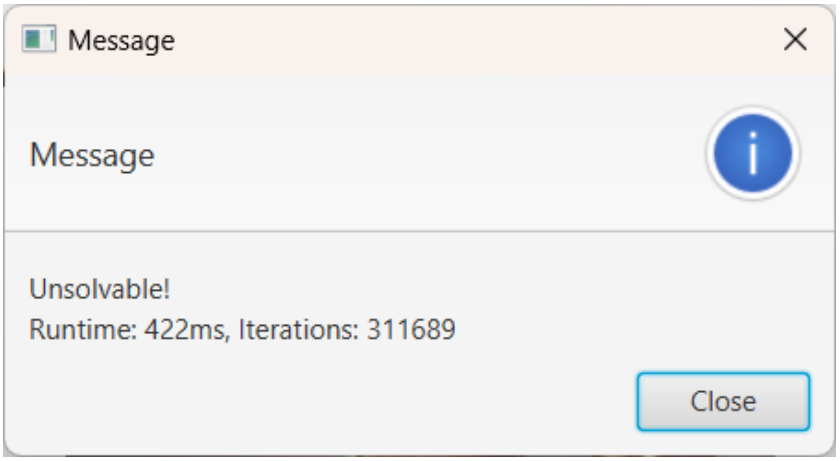
Hasil:

7. File 7.txt

```
4 4 5
DEFAULT
A
AA
B
BB
CC
C
D
DD
```

```
EE
E
```

Hasil:



8. File 8.txt

```
4 5 6
DEFAULT
A
AA
B
BB
CC
C
D
DD
EE
EEE
FF
FFF
GGG
```

Hasil:

Bab 5: Kesimpulan

1. Kesimpulan

Berdasarkan testing, penggunaan strategi algoritma brute force pada pencarian solusi permainan puzzle terbukti efektif, meskipun bukan yang paling efisien. Dengan algoritma penyelesaian yang digunakan, semua kasus akan ditinjau. Sehingga, pasti akan ada salah satu kasus yang ditinjau yang merupakan solusi jika permainan memang memiliki solusi.

Melalui tugas kecil ini, saya mempelajari penggunaan strategi algoritma ***brute force*** dapat diaplikasikan ke hampir setiap permasalahan komputasi seperti permainan ini. Selain itu, saya juga mempelajari konsep-konsep di luar pembelajaran kelas, seperti penggunaan paradigma pemrograman berorientasi objek, dan penggunaan bahasa Java.

LAMPIRAN

Tabel Kelengkapan

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

Link Repositori Github: https://github.com/koinen/Tucil1_13523083