Laporan Tugas Kecil 01

IF2211 STRATEGI ALGORITMA

Disusun oleh:

Ardell Aghna Mahendra (13523151)

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

2025

Daftar Isi

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. Dalam permainan ini terdapat beberapa komponen penting, yaitu Board atau Papan yang merupakan tempat yang harus diisi oleh para pemain menggunakan blok-blok yang telah tersedia dan Blok atau Piece yang memiliki bentuk yang unik merupakan komponen untuk mengisi papan kosong menjadi terisi penuh untuk menyelesaikan puzzle. Tantangan utama dari permainan ini adalah mencari solusi yang paling tepat untuk menyelesaikan puzzle tersebut, untuk itu perlu dibuat program menggunakan pendekatan brute force murni yang mampu menghasilkan satu solusi apabila memang puzzle tersebut dapat diselesaikan atau tidak ada solusi jika puzzle tidak dapat diselesaikan, menampilkan waktu eksekusi, dan banyaknya percobaan yang ditinjau.

BAB II

TEORI SINGKAT

2.1. Algoritma Brute Force

Algoritma Brute Force adalah metode sederhana yang digunakan untuk menemukan sebuah solusi atau menyelesaikan suatu masalah. Algoritma Brute Force bekerja dengan cara mengecek satu per satu kemungkinan solusi yang benar. Dengan cara ini, dapat dikatakan bahwa penyelesaian menggunakan Algoritma Brute Force kurang efisien dari segi waktu, namun cara ini dapat menjamin solusi yang di dapat benar.

BAB III

IMPLEMENTASI PROGRAM

Program IQ Puzzle Pro Solver ini menggunakan bahasa pemrograman Java dan menggunakan algoritma brute force murni. Dikarenakan menggunakan algoritma brute force murni, pencarian solusi dilakukan secara menyeluruh yaitu semua kemungkinan penempatan piece atau blok dicoba baik itu dengan cara dirotasi ataupun dicerminkan tanpa menggunakan heuristik seperti contohnya tanpa melihat adanya kemungkinan percobaan tersebut gagal. Program mengawali pencarian dengan cara mengecek apakah seluruh piece yang tersedia sudah digunakan seluruhnya dan apabila sudah, maka program akan melanjutkan untuk mengecek apakah seluruh papan telah terisi untuk menentukan apakah solusi telah ditemukan. Untuk piece dan blok yang belum digunakan, program akan mencoba untuk menempatkan pada setiap posisi yang mungkin di dalam papan dengan tiga looping, yaitu program akan coba setiap piece yang tersedia, lalu dari setiap baris papan, dan kemudian kolom papan. Pada setiap posisi, program akan mencoba semua kemungkinan bentuk piece, yaitu di rotasi sebanyak empat kali (0, 90, 180, 270) dan pencerminan di setiap rotasi, sehingga memberikan total delapan kemungkinan bentuk berbeda untuk setiap piece atau blok pada setiap posisi.

Selanjutnya program akan mengecek apakah piece dapat ditempatkan pada posisi tersebut dan apabila bisa ditempatkan, maka program akan menandakan bahwa piece tersebut telah digunakan. Program kemudian akan melakukan rekursif untuk mencoba meletakkan piece berikutnya dan apabila berhasil akan mengembalikan nilai true dan apabila tidak berhasil piece tersebut akan ditandai dengan belum digunakan dan program akan melanjutkan dengan percobaan atau kemungkinan berikutnya.

Program ini akan selesai dan memberikan satu solusi apabila semua piece berhasil ditempatkan pada papan, namun apabila telah dilakukan semua percobaan dan ternyata seluruh piece yang tersedia tidak dapat memenuhi seluruh papan yang tersedia atau melebihi kapasitas papan, maka program akan menampilkan pesan bahwa tidak ada solusi yang ditemukan. Program akan mencatat dan menampilkan jumlah total dari percobaan atau kemungkinan yang dilakukan program untuk mendapatkan solusi termasuk juga percobaan mengecek apakah piece sudah digunakan di awal dan juga mencatat dan menampilkan waktu program untuk menemukan solusi dimulai saat pengguna mengklik tombol *solve* dan berakhir ketika semua percobaan atau kemungkinan tetah dicoba atau ketika solusi ditemukan. Waktu yang diperoleh mungkin akan makin lama seiring dengan banyaknya percobaan yang dilakukan karena dalam brute force murni semua kemungkinan harus dicoba hingga akhirnya mendapatkan solusi. Berikut ini merupakan source code dari program ini (untuk lebih lengkapnya dapat melihat langsung di repository).

PuzzleSolver.java

```
public class PuzzleSolver {
    private final Board board;
    private final Piece[] pieces;
    private final boolean[] used;
    private long attempt;
    private long startTime;
    private long endTime;
    private boolean solved;

    public PuzzleSolver(Board board, Piece[] pieces) {
        this.board = board;
        this.pieces = pieces;
        this.used = new boolean[pieces.length];
        this.attempt = 0;
        this.startTime = 0;
        this.endTime = 0;
        this.solved = false;
    }

    public boolean solve() {
        startTime = System.currentTimeMillis();
        solved = solver();
        endTime = System.currentTimeMillis();
        return solved;
    }

    private boolean solver() {
        attempt++;
        boolean allUsed = true;
        for (boolean u : used) {
            if (!u) {
                allUsed = false;
                break;
            }
        } if (allUsed) {
            for (int i = 0; i < board.rows(); i++) {
                for (int j = 0; j < board.cols(); j++) {
                    if (!board.Occupied()[i][j]) {
                        return false;
                    }
                }
            }
            return true;
        } for (int p = 0; p < pieces.length; p++) {
            if (!used[p]) {
                for (int i = 0; i < board.rows(); i++) {
                    for (int j = 0; j < board.cols(); j++) {
                        Piece piece = pieces[p].clone();
```

```
                        for (int rot = 0; rot < 4; rot++) {
                            if (piece.canPlace(board.Occupied(), i, j)) {
                                piece.place(board.Occupied(), board.Symbols(), i, j);
                                used[p] = true;
                                if (solver()) {
                                    return true;
                                }
                                used[p] = false;
                                piece.remove(board.Occupied(), board.Symbols(), i, j);
                            } piece.rotate();
                        }
                        piece = pieces[p].clone();
                        piece.mirror();
                        for (int rot = 0; rot < 4; rot++) {
                            if (piece.canPlace(board.Occupied(), i, j)) {
                                piece.place(board.Occupied(), board.Symbols(), i, j);
                                used[p] = true;
                                if (solver()) {
                                    return true;
                                }
                                used[p] = false;
                                piece.remove(board.Occupied(), board.Symbols(), i, j);
                            } piece.rotate();
                        }
                    }
                }
            }
        }
        return false;
    } public boolean isSolved() {
        return solved;
    } public long Start() {
        return startTime;
    } public long Time() {
        return endTime - startTime;
    } public long attempts() {
        return attempt;
    } public Board Board() {
        return board;
    }
}
```

Board.java

```
public class Board {
    private final int rows;
    private final int cols;
    private final boolean[][] occupied;
    private final char[][] symbols;

    public Board(int rows, int cols) {
        this.rows = rows;
        this.cols = cols;
```

```
        this.occupied = new boolean[rows][cols];
        this.symbols = new char[rows][cols];

        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                symbols[i][j] = '.';
            }
        }
    }

    public void display() {
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                System.out.print(symbols[i][j] + " ");
            } System.out.println();
        } System.out.println();
    }

    public String toString() {
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                sb.append(symbols[i][j]);
            } sb.append('\n');
        }
        return sb.toString();
    } public boolean[][] Occupied() {
        return occupied;
    } public char[][] Symbols() {
        return symbols;
    } public int rows() {
        return rows;
    } public int cols() {
        return cols;
    }
}
```

Piece.java

```
import java.util.List;

public class Piece {
    private boolean[][] shape;
    private final char symbol;
    private int height;
    private int width;

    public Piece(List<String> pieceStrings, char symbol) {
        this.symbol = symbol;
        this.height = pieceStrings.size();
        this.width = pieceStrings.stream().mapToInt(String::length).max().orElse(0);
        this.shape = new boolean[height][width];
        for (int i = 0; i < height; i++) {
```

```
            String line = pieceStrings.get(i);
            for (int j = 0; j < line.length(); j++) {
                shape[i][j] = (line.charAt(j) == symbol);
            }
        } if (!isValidPiece()) {
            throw new IllegalStateException("Konfigurasi piece tidak valid, mohon cek
kembali");
        }
    }

    public Piece rotate() {
        boolean[][] newShape = new boolean[width][height];
        for (int i = 0; i < height; i++) {
            for (int j = 0; j < width; j++) {
                newShape[j][height - 1 - i] = shape[i][j];
            }
        }
        int temp = width;
        width = height;
        height = temp;
        shape = newShape;
        return this;
    }

    public Piece mirror() {
        boolean[][] newShape = new boolean[height][width];
        for (int i = 0; i < height; i++) {
            for (int j = 0; j < width; j++) {
                newShape[i][width - 1 - j] = shape[i][j];
            }
        }
        shape = newShape;
        return this;
    }

    private boolean isValidPiece() {
        boolean[][] visited = new boolean[height][width];
        boolean found = false;
        int startI = 0, startJ = 0;
        for (int i = 0; i < height && !found; i++) {
            for (int j = 0; j < width && !found; j++) {
                if (shape[i][j]) {
                    startI = i;
                    startJ = j;
                    found = true;
                }
            }
        }
        return found && (dfs(startI, startJ, visited) == countCells());
    }

    private int dfs(int i, int j, boolean[][] visited) {
        if (i < 0 || i >= height || j < 0 || j >= width ||
```

```
            visited[i][j] || !shape[i][j]) {
            return 0;
        }
        visited[i][j] = true;
        int count = 1;
        count += dfs(i+1, j, visited);
        count += dfs(i-1, j, visited);
        count += dfs(i, j+1, visited);
        count += dfs(i, j-1, visited);
        return count;
    }

    private int countCells() {
        int count = 0;
        for (int i = 0; i < height; i++) {
            for (int j = 0; j < width; j++) {
                if (shape[i][j]) count++;
            }
        }
        return count;
    }

    public boolean canPlace(boolean[][] board, int row, int col) {
        if (row + height > board.length || col + width > board[0].length) {
            return false;
        } for (int i = 0; i < height; i++) {
            for (int j = 0; j < width; j++) {
                if (shape[i][j] && board[row + i][col + j]) {
                    return false;
                }
            }
        }
        return true;
    }

    public void place(boolean[][] board, char[][] boardSymbols, int row, int col) {
        for (int i = 0; i < height; i++) {
            for (int j = 0; j < width; j++) {
                if (shape[i][j]) {
                    board[row + i][col + j] = true;
                    boardSymbols[row + i][col + j] = symbol;
                }
            }
        }
    }

    public void remove(boolean[][] board, char[][] boardSymbols, int row, int col) {
        for (int i = 0; i < height; i++) {
            for (int j = 0; j < width; j++) {
                if (shape[i][j]) {
                    board[row + i][col + j] = false;
                    boardSymbols[row + i][col + j] = '.';
                }
```

```
            }
        }
    }

    public Piece clone() {
        Piece clone = new Piece(symbol);
        clone.height = this.height;
        clone.width = this.width;
        clone.shape = new boolean[height][width];
        for (int i = 0; i < height; i++) {
            System.arraycopy(this.shape[i], 0, clone.shape[i], 0, width);
        }
        return clone;
    }

    private Piece(char symbol) {
        this.symbol = symbol;
    }
}
```

InputFile.java

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.*;

public class InputFile {
    private final int N;
    private final int M;
    private final int P;
    private final String puzzleType;
    private final List<List<String>> puzzlePieces;
    private final List<Character> pieceSymbols;

    public InputFile(String fileName) throws Exception {
        puzzlePieces = new ArrayList<>();
        pieceSymbols = new ArrayList<>();
        Map<Character, List<String>> pieceMap = new HashMap<>();
        try (Scanner scanner = new Scanner(new File(fileName))) {
            String[] dimensions = scanner.nextLine().trim().split("\\s+");
            if (dimensions.length != 3) {
                throw new Exception("Input tidak valid, format seharusnya N M P");
            } try {
                this.N = Integer.parseInt(dimensions[0]);
                this.M = Integer.parseInt(dimensions[1]);
                this.P = Integer.parseInt(dimensions[2]);
                if (this.P > 26) {
                    throw new Exception("P harus huruf (A-Z) yaa");
                }
            } catch (NumberFormatException e) {
                throw new Exception("Angka tidak valid");
            }
            this.puzzleType = scanner.nextLine().trim();
```

```
            if (!this.puzzleType.equals("DEFAULT")) {
                throw new Exception("Maaf untuk saat ini tipe puzzle hanya bisa DEFAULT");
            }
            List<String> currentPiece = new ArrayList<>();
            char currentChar = ' ';
            Set<Character> usedSymbols = new HashSet<>();
            while (scanner.hasNextLine()) {
                String line = scanner.nextLine();
                if (line.trim().isEmpty()) {
                    if (!currentPiece.isEmpty()) {
                        validateAndAddPiece(currentPiece, currentChar, usedSymbols, pieceMap);
                        pieceSymbols.add(currentChar);
                        currentPiece = new ArrayList<>();
                        currentChar = ' ';
                    } continue;
                } char lineChar = ' ';
                for (char c : line.toCharArray()) {
                    if (Character.isUpperCase(c)) {
                        lineChar = c;
                        break;
                    }
                } if (lineChar == ' ') continue;
                if (currentChar != ' ' && lineChar != currentChar) {
                    if (!currentPiece.isEmpty()) {
                        validateAndAddPiece(currentPiece, currentChar, usedSymbols, pieceMap);
                        pieceSymbols.add(currentChar);
                        currentPiece = new ArrayList<>();
                    }
                }
                currentChar = lineChar;
                currentPiece.add(line);
            } if (!currentPiece.isEmpty()) {
                validateAndAddPiece(currentPiece, currentChar, usedSymbols, pieceMap);
                pieceSymbols.add(currentChar);
            } if (pieceMap.size() != this.P) {
                throw new Exception("Ketemu " + pieceMap.size() + " pieces nih, Seharusnya " +
this.P + " pieces");
            } for (char symbol : pieceSymbols) {
                puzzlePieces.add(pieceMap.get(symbol));
            }

        } catch (FileNotFoundException e) {
            throw new Exception("File tidak ditemukan huhu: " + fileName);
        }
    }

    private void validateAndAddPiece(List<String> piece, char symbol, Set<Character>
usedSymbols,
                                     Map<Character, List<String>> pieceMap) throws Exception {
        if (usedSymbols.contains(symbol)) {
            throw new Exception("OMG Duplicate symbol found!! : " + symbol);
        } for (String line : piece) {
            for (char c : line.toCharArray()) {
```

```
                if (c != ' ' && c != symbol) {
                    throw new Exception("Yah invalid karena ada " + c + " di piece: '" + symbol
+ "'");
                }
            }
        }
        usedSymbols.add(symbol);
        pieceMap.put(symbol, new ArrayList<>(piece));
    } public int n() {
        return N;
    } public int m() {
        return M;
    } public int p() {
        return P;
    } public String puzzleType() {
        return puzzleType;
    } public List<List<String>> pieces() {
        return puzzlePieces;
    } public List<Character> pieceSymbols() {
        return pieceSymbols;
    }
}
```

GUI.java (sekilas)

```
import java.awt.event.*;
import javax.swing.border.*;
import java.io.File;
import java.awt.image.BufferedImage;
import javax.imageio.ImageIO;

public class GUI extends JFrame {
    private static final String TITLE = "IQ PUZZLER PRO SOLVER";
    private static final int MIN_WIDTH = 500;
    private static final int MIN_HEIGHT = 400;
    private static final int CELL_SIZE = 50;
    private static final int SPACING = 2;

    private JPanel boardPanel;
    private JLabel statusLabel;
    private JLabel timeLabel;
    private JLabel attemptLabel;
    private JButton[][] cells;
    private final Color[] pieceColors;
    private final int rows;
    private final int cols;
    private JButton solveButton;
    private JButton saveButton;
    private JButton saveImageButton;
    private JButton exitButton;
    private PuzzleSolver solver;
    private boolean isSolving = false;
    private Timer progressTimer;
```

```
    public GUI(int rows, int cols, PuzzleSolver solver) {
        this.rows = rows;
        this.cols = cols;
        this.solver = solver;
        this.pieceColors = initializePieceColors();

        setupMainFrame();
        setupComponents();
        setupKeyboardShortcuts();
        initializeProgressTimer();
    }
    private void exitApplication() {
        if (isSolving) {
            int confirm = JOptionPane.showConfirmDialog(this,
                "Proses pencarian solusi sedang berjalan. Anda yakin ingin keluar?",
                "Konfirmasi",
                JOptionPane.YES_NO_OPTION);
            if (confirm == JOptionPane.YES_OPTION) {
                System.exit(0);
            }
        } else {
            System.exit(0);
        }
     }
```

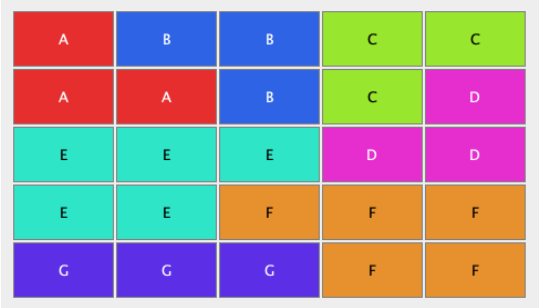
Main.java

```
public class Main {
    public static void main(String[] args) {
        GUI.main(args);
    }
}
```

BAB IV

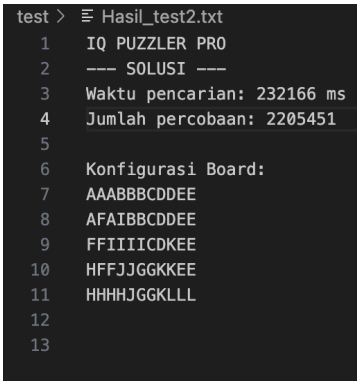
EKSPERIMEN

Gambar 4.1 Tampilan awal program



Gambar 4.2 Contoh solusi yang disimpan dalam bentuk gambar

Gambar 4.3 Tampilan setelah berhasil menginput file (.txt)

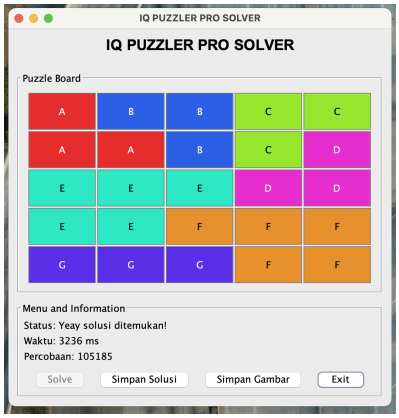


Gambar 4.4 Contoh solusi yang disimpan dalam bentuk file .txt

test > test1.txt

```
5 5 7
DEFAULT
A
AA
B
BB
C
CC
D
DD
EE
EE
E
FF
FF
F
GGG
```

Gambar 4.5 Data uji 1

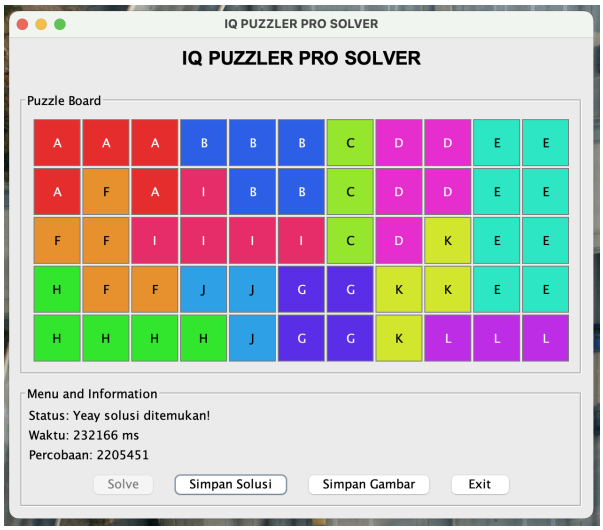


Gambar 4.6 Hasil data uji 1

test > test2.txt

```
5 11 12
DEFAULT
AAA
A A
BBB
 BB
C
C
C
DD
DD
D
EE
EE
EE
EE
 F
FF
 FF
GG
GG
H
HHHH
IIII
 I
 J
JJ
 K
KK
K
LLL
```

Gambar 4.7 Data uji 2

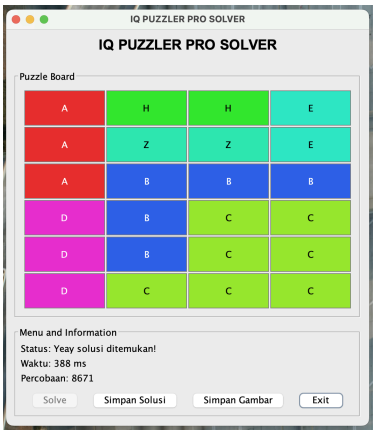


Gambar 4.8 Hasil Data uji 2

test > test3.txt

```
1   6 4 7
2   DEFAULT
3   A
4   A
5   A
6   HH
7   EE
8   B
9   B
10  BBB
11  C
12  CCC
13  CCC
14  ZZ
15  D
16  D
17  D
```

Gambar 4.9 Data uji 3

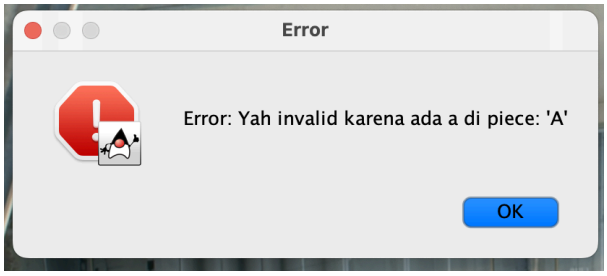


Gambar 4.10 Hasil data uji 3

test > test4.txt

```
1   5 5 2
2   DEFAULT
3   AAAAa
4   B
```

Gambar 4.11 Data uji 4



Gambar 4.12 Hasil Data uji 4

test > test5.txt

```
1   5 6 4
2   DEFAULT
3   A
4   AA
5   AA
6   B
7   BBB
8   BBB
9   C
10  CC
11  CCC
12  D
13  DDD
14  DD
```

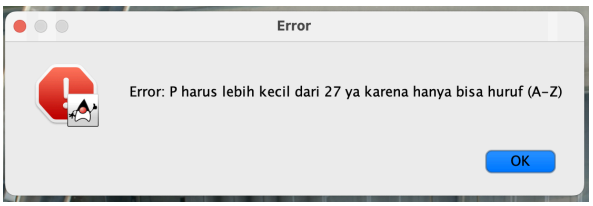
Gambar 4.13 Data uji 5

Gambar 4.14 Hasil data uji 5

test > test6.txt

```
5 5 30
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
```

Gambar 4.15 Data uji 6

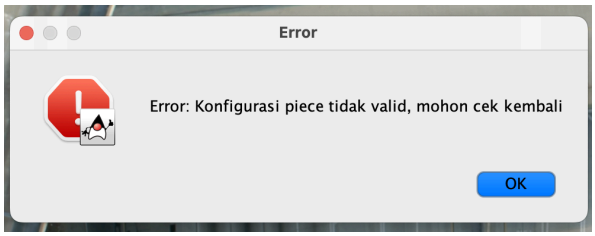


Gambar 4.16 Hasil data uji 6

test > test7.txt

```
5 6 4
DEFAULT
A
AA
AA
B
    BBB
BBB
C
  CC
CCC
D
DDD
DD
```

Gambar 4.17 Data uji 7

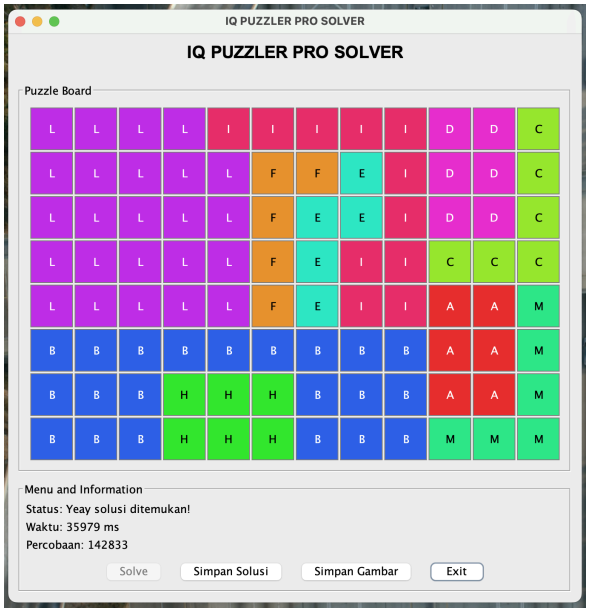


Gambar 4.18 Hasil data uji 7

test > test8.txt

```
8 12 10
DEFAULT
LLLL
LLLLL
LLLLL
LLLLL
LLLLL
BBBBBBBBB
BBB   BBB
BBB   BBB
I
I
I
I  II
IIIII
DD
DD
DD
AAA
AAA
FF
F
F
F
 E
EE
E
E
HHH
HHH
CCCC
C
C
M
M
MMMM
```

Gambar 4.19 Data uji 8



Gambar 4.20 Hasil data uji 8

test > test9.txt

```
8 12 9
DEFAULT
    BBBB
   BBBBB
   BBBBB
   BBBBB
   BBBBB
BBBBBBBBB
BBB   BBB
BBB   BBB
I
I
I
I  II
IIIII
AAA
AAA
 D
DD
DD
DD
EEE
E
E
GG
GG
  H
  H
HHHH
HH
CCC
 C
KK
KK
KK
```

Gambar 4.21 Data uji 9

Gambar 4.22 Hasil data uji 9

BAB V

PENUTUP

5.1. Kesimpulan

Pada Tugas kecil mata kuliah IF2211 Strategi Algoritma ini, saya berhasil mengembangkan program solver untuk permainan IQ Puzzle Pro menggunakan algoritma brute force murni. Meskipun algoritma ini memakan banyak sekali waktu, namun dengan mencoba semua kemungkinan yang mungkin memberikan solusi yang tepat atau benar. Waktu yang diperoleh juga dapat dipengaruhi oleh perangkat yang digunakan. Pengerjaan tugas ini menambah pengalaman serta ilmu saya dalam menggunakan bahasa java dan juga menambah pemahaman saya tentang algoritma brute force, terutama pemahaman tentang heuristik.

5.2. Saran

Untuk pengembangan selanjutnya, sebaiknya dicoba untuk menggunakan algoritma yang lebih efisien dibandingkan brute force agar saat bertemu dengan *case* yang kompleksitasnya tinggi, program dapat menyelesaikan *case* tersebut dalam waktu singkat dan juga untuk input agar pengguna bisa input secara manual tidak hanya dari file.

5.3. Komentar

Akhirnya tugas kecil ini selesai juga. Sebuah pemanasan untuk menyambut banyaknya tugas yang akan datang di semester 4 ini.

LAMPIRAN

Link Repository

Link Repository Tugas Kecil 1 : https://github.com/ArdellAghna/Tucil1_13523151

Tabel Checkpoint

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	✔	

DAFTAR REFERENSI

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/2024-2025/03-Algoritma-Brute-Force-(2025)-Bag2.pdf

https://www.guru99.com/java-swing-gui.html