PENYELESAIAN IQ PUZZLER PRO
DENGAN ALGORITMA BRUTE FORCE

Laporan Tugas Kecil 1

Disusun untuk memenuhi tugas mata kuliah IF2211 Strategi Algoritma

oleh:

Nayla Zahira

13523079

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2025

DAFTAR ISI

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:

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.

1.2. Algoritma Brute Force

Algoritma brute force merupakan suatu pendekatan dalam pemecahan masalah komputasional yang menerapkan metode pencarian secara menyeluruh terhadap seluruh kemungkinan solusi yang ada. Dalam implementasinya, algoritma ini melakukan evaluasi sistematis pada setiap kandidat solusi hingga ditemukan hasil yang sesuai dengan kriteria yang diinginkan, tanpa mempertimbangkan kompleksitas atau karakteristik khusus dari permasalahan yang dihadapi.

Meskipun algoritma brute force dapat menghasilkan solusi yang akurat untuk permasalahan kompleks, terdapat keterbatasan signifikan dalam penggunaannya, terutama terkait dengan aspek efisiensi kinerja. Hal ini terlihat ketika algoritma dihadapkan pada permasalahan dengan ruang solusi yang sangat besar, di mana setiap kemungkinan harus diperiksa secara individual, mengakibatkan peningkatan waktu komputasi secara signifikan seiring dengan bertambahnya ukuran masalah yang harus diselesaikan.

1.3. Algoritma Penyelesaian IQ Puzzler Pro dengan Pendekatan Brute Force

Pada tugas kecil ini, program penyelesaian IQ Puzzler Pro menggunakan pendekatan Brute Force. Implementasi algoritma ini terdapat pada fungsi solve() yang berperan untuk mencari solusi penyusunan potongan puzzle (*pieces*) pada

papan (*board*) dengan mencoba setiap kemungkinan penempatan yang ada. Fungsi ini bekerja secara rekursif dengan memeriksa setiap posisi, rotasi, dan pencerminan yang mungkin untuk setiap potongan puzzle.

Proses pencarian secara brute force ini dimulai dengan mencoba menempatkan suatu potongan puzzle pada papan yang tersedia. Untuk setiap posisi, algoritma akan mencoba 8 orientasi berbeda dari potongan - 4 rotasi (0°, 90°, 180°, 270°) dari bentuk asli dan 4 rotasi dari bentuk yang dicerminkan. Jika suatu penempatan valid ditemukan, algoritma akan melanjutkan ke potongan puzzle berikutnya secara rekursif.

Jika tidak ditemukan posisi yang valid untuk suatu potongan puzzle, algoritma ini akan melakukan *backtracking* dengan kembali ke potongan puzzle sebelumnya dan mencoba posisi atau orientasi yang berbeda. Proses ini akan terus berulang hingga ditemukan kombinasi valid atau semua kemungkinan telah dicoba. Pencarian akan berhenti ketika semua potongan berhasil ditempatkan atau ketika tidak ada solusi yang ditemukan setelah mencoba semua kemungkinan.

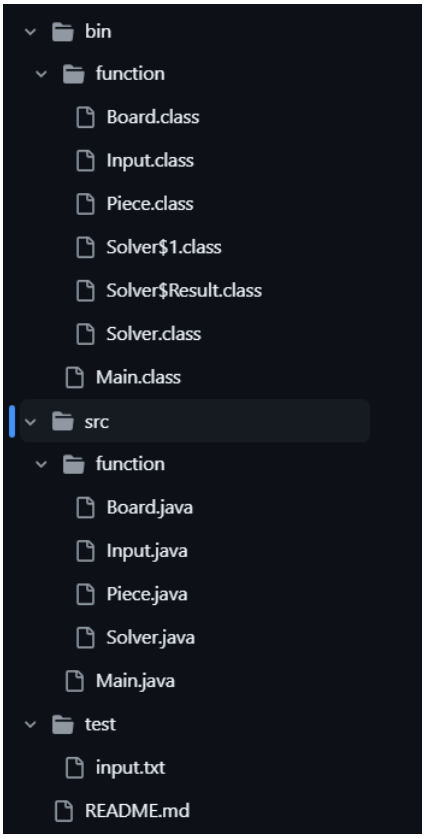
1.4. Ilustrasi Pseudocode

```
procedure displayBoard(char[][] board)
function placePiece(int r, int c, char[][] board, char[][] piece)
-> char[][]
function canPlacePiece(int r, int c, char[][] board, char[][]
piece) -> boolean
function solve(input char[][] board, input List<char[][]> pieces,
input integer index) -> output Result(char[][], integer)

KAMUS
count, N, M, i, j, r : integer
piece, currentPiece, mirrored: char[][]
newBoard : char[][]

ALGORITMA
if(index == 0)then
      count = 0
```

```
if (index >= pieces.size()) then
    if (isBoardFull(board)) then
        displayBoard(board)
        -> new Result(board, count)
    } else{
        -> new Result(null, count)
    }
}
piece <- pieces.get(index)
mirrored <- Piece.flipHorizontally(piece)
N <- board.length
M <- board[0].length

i traversal(0..N-1)
    j traversal (0..M-1) {
        currentPiece = piece
        r traversal (0..3)
            count <- count + 1
            if (canPlacePiece(i, j, board, currentPiece)) then
                newBoard = placePiece(i, j, board, currentPiece)
                Result result <- solve(newBoard, pieces,index+1)
                if (result != null) then
                    -> result
            currentPiece <- Piece.rotateClockwise(currentPiece)
        currentPiece <- mirrored
        r traversal (0..3)
            count <- count + 1
            if (canPlacePiece(i, j, board, currentPiece)) then
                newBoard = placePiece(i, j, board,currentPiece)
                Result result <- solve(newBoard,pieces,index+1);
                if (result != null)
                    -> result
            currentPiece <- Piece.rotateClockwise(currentPiece)
-> null
```

BAB II
IMPLEMENTASI PROGRAM

2.1. Struktur Kode Program



Pada program ini, struktur folder terdiri dari beberapa bagian utama:

1. Folder Utama:
    a. bin: berisi *executable file*
    b. doc: berisi laporan dalam bentuk pdf
    c. src: berisi source code program
    d. test: berisi input file untuk testing dan output file solusi
2. Komponen Program:
    a. Board.java: menangani representasi papan permainan IQ Puzzler Pro
    b. Input.java: berfungsi untuk membaca dan menyimpan hasil input
    c. Piece.java: berisi fungsi operasi pada piece
    d. Solver.java: berisi algoritma penyelesaian puzzle
    e. Main.java: berisi program utama
3. File Pendukung:
    a. input.txt: berisi data input untuk testing

b. README: berisi penjelasan singkat program

2.2. Source Code Program

a. Input.java

```
package function;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;

public class Input {
    public static Board readInput(String filename) throws
IOException {
        List<char[][]> pieces = new ArrayList<>();
        HashSet<Character> pieceChar = new HashSet<>();
        try (BufferedReader br = new BufferedReader(new
FileReader(filename))) {
            String line = br.readLine();
            if (line == null || line.trim().isEmpty()) throw
new IllegalArgumentException("File is empty.");
            String[] firstLine = line.trim().split("\\s+");
            if (firstLine.length != 3) throw new
IllegalArgumentException("First line must contain N, M, and
P.");

            int N = Integer.parseInt(firstLine[0]);
            int M = Integer.parseInt(firstLine[1]);
            int P = Integer.parseInt(firstLine[2]);
            if (N <= 0 || M <= 0) throw new
IllegalArgumentException("Board size (N x M) must be
positive.");
            if (P < 1 || P > 26) throw new
IllegalArgumentException("P must be between 1 and 26.");

            String S = readNextNonEmptyLine(br);
            if (S == null || (!S.equals("DEFAULT"))) {
                throw new IllegalArgumentException("Board type
```

```
must be 'DEFAULT'");
            }

            List<String> piece = new ArrayList<>();
            char currentPiece = ' ';
            while ((line = br.readLine()) != null) {
                if(!line.isEmpty()) {
                    char firstChar = line.trim().charAt(0);
                    pieceChar.add(firstChar);
                    if (piece.isEmpty() || currentPiece ==
firstChar) {
                        piece.add(line);
                        currentPiece = firstChar;
                    } else {
                        pieces.add(convertToCharMatrix(piece));
                        piece.clear();
                        piece.add(line);
                        currentPiece = firstChar;
                    }
                }
            }
            if (!piece.isEmpty()) {
                pieces.add(convertToCharMatrix(piece));
            }
            return new Board(N, M, pieces, pieceChar);
        } catch (IOException e) {
            System.out.println(e.getMessage());
            return null;
        }
    }

    private static String readNextNonEmptyLine(BufferedReader
br) throws IOException {
        String line;
        while ((line = br.readLine()) != null) {
            line = line.trim();
            if (!line.isEmpty()) return line;
        }
        return null;
    }
    public static char[][] convertToCharMatrix(List<String>
piece) {
      int row = piece.size();
        int col = 0;
```

```
        for(String line : piece) {
            col = Math.max(col, line.length());
        }
        char[][] result = new char[row][col];
        for(int i = 0; i < row; i++) {
            String line = piece.get(i);
            for(int j = 0; j < col; j++) {
                result[i][j] = (j < line.length()) ?
line.charAt(j) : ' ';
            }
        }
        return result;
    }
}
```

b. Board.java

```
package function;

import java.util.*;

public class Board{
    public int N, M;
    public List<char[][]> pieces;
    public HashSet<Character> pieceChar;
    public char[][] board;

    public Board(int N, int M, List<char[][]> pieces,
HashSet<Character> pieceChar) {
        this.N = N;
        this.M = M;
        this.pieces = pieces;
        this.pieceChar = pieceChar;
        this.board = new char[N][M];
        for (int i = 0; i < N; i++){
            for (int j = 0; j < M; j++){
                this.board[i][j] = ' ';
            }
        }
    }
}
```

c. Piece.java

```
package function;

public class Piece {

    public static char[][] rotateClockwise(char[][] piece) {
        int rows = piece.length, cols = piece[0].length;
        char[][] rotated = new char[cols][rows];

        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                rotated[j][rows - 1 - i] = piece[i][j];
            }
        }
        return rotated;
    }

    public static char[][] flipHorizontally(char[][] piece) {
        int rows = piece.length, cols = piece[0].length;
        char[][] flipped = new char[rows][cols];

        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                flipped[i][cols - 1 - j] = piece[i][j];
            }
        }
        return flipped;
    }
}
```

d. Solver.java

```
package function;

import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class Solver {
    private static int count;

    public static class Result{
        public char[][] board;
        public int count;

        public Result(char[][] board, int count){
            this.board = board;
            this.count = count;
        }
    }
    public static boolean canPlacePiece(int r, int c, char[][]
board, char[][] piece) {
        int pieceRow = piece.length;
        int pieceCol = piece[0].length;

        if (r + pieceRow > board.length || c + pieceCol >
board[0].length) {
            return false;
        }

        for (int i = 0; i < pieceRow; i++) {
            for (int j = 0; j < pieceCol; j++) {
                if (piece[i][j] != ' ' && board[r + i][c + j]
!= ' ' ) {
                    return false;
                }
            }
        }
        return true;
    }
```

```
    public static char[][] placePiece(int r, int c, char[][]
board, char[][] piece) {
        char[][] newBoard = new
char[board.length][board[0].length];
        for (int i = 0; i < board.length; i++) {
            for (int j = 0; j < board[0].length; j++) {
                newBoard[i][j] = board[i][j];
            }
        }
        for (int i = 0; i < piece.length; i++) {
            for (int j = 0; j < piece[0].length; j++) {
                if (piece[i][j] != ' ') {
                    newBoard[i+r][j+c] = piece[i][j];
                }
            }
        }
        return newBoard;
    }

    public static Result solve(char[][] board, List<char[][]>
pieces, int index) {
        if(index == 0) count = 0;

        if (index >= pieces.size()) {
            if (isBoardFull(board)){
                System.out.println("\n=======================
========================\n");
                displayBoard(board);
                return new Result(board, count);
            } else{
                return new Result(null, count);
            }
        }

        char[][] piece = pieces.get(index);
        char[][] mirrored = Piece.flipHorizontally(piece);
        int N = board.length, M = board[0].length;

        for (int i = 0; i < N; i++) {
            for (int j = 0; j < M; j++) {
                char[][] currentPiece = piece;

                for (int r = 0; r < 4; r++) {
```

```
                    count++;
                    if (canPlacePiece(i, j, board,
currentPiece)) {
                        char[][] newBoard = placePiece(i, j,
board, currentPiece);
                        Result result = solve(newBoard, pieces,
index + 1);
                        if (result != null) {
                            return result;
                        }
                    }
                    currentPiece =
Piece.rotateClockwise(currentPiece); // Rotate 90°
                }

                currentPiece = mirrored;
                for (int r = 0; r < 4; r++) {
                    count++;
                    if (canPlacePiece(i, j, board,
currentPiece)) {
                        char[][] newBoard = placePiece(i, j,
board, currentPiece);
                        Result result = solve(newBoard, pieces,
index + 1);
                        if (result != null) {
                            return result;
                        }
                    }
                    currentPiece =
Piece.rotateClockwise(currentPiece);
                }
            }
        }
        return null;
    }

    private static final Map<Character, String> colorMap = new
HashMap<>() {{
        put('A', "\033[38;5;34m");   // Green
        put('B', "\033[38;5;196m");  // Bright Red
        put('C', "\033[38;5;208m");  // Dark Orange
        put('D', "\033[38;5;135m");  // Purple
        put('E', "\033[38;5;220m");  // Golden Yellow
```

```
        put('F', "\033[38;5;45m");   // Cyan
        put('G', "\033[38;5;160m");  // Dark Red
        put('H', "\033[38;5;100m");  // Muted Blue
        put('I', "\033[38;5;93m");   // Magenta
        put('J', "\033[38;5;118m");  // Bright Green
        put('K', "\033[38;5;201m");  // Pink
        put('L', "\033[38;5;27m");   // Blue
        put('M', "\033[38;5;214m");  // Light Orange
        put('N', "\033[38;5;136m");  // Tan
        put('O', "\033[38;5;51m");   // Light Cyan
        put('P', "\033[38;5;165m");  // Lavender
        put('Q', "\033[38;5;88m");   // Dark Maroon
        put('R', "\033[38;5;46m");   // Bright Teal
        put('S', "\033[38;5;142m");  // Olive
        put('T', "\033[38;5;207m");  // Bright Pink
        put('U', "\033[38;5;255m");  // White
        put('V', "\033[38;5;226m");  // Yellow
        put('W', "\033[38;5;129m");  // Purple
        put('X', "\033[38;5;130m");  // Brown
        put('Y', "\033[38;5;37m");   // Teal
        put('Z', "\033[38;5;90m");   // Grayish Blue
    }};
    public static void displayBoard(char[][] board) {
        String reset = "\033[0m"; // Reset color

        for (char[] row : board) {
            for (char cell : row) {
                if (cell == ' ') {
                    System.out.print("  ");
                } else if (colorMap.containsKey(cell)) {
                    System.out.print(colorMap.get(cell) + cell
+ reset + " ");
                } else {
                    System.out.print(cell + " ");
                }
            }
            System.out.println();
        }
    }

    public static boolean isBoardFull(char[][] board) {
        for (char[] row : board) {
            for (char cell : row) {
```

```
                if (cell == ' ') {
                    return false;
                }
            }
        }
        return true;
    }
}
```

e. Main.java

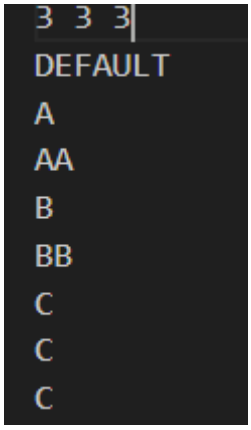
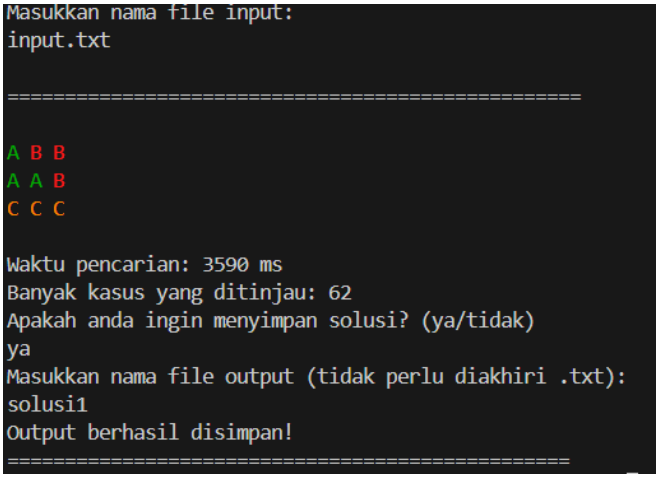
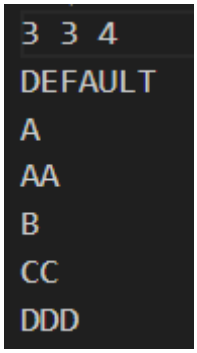
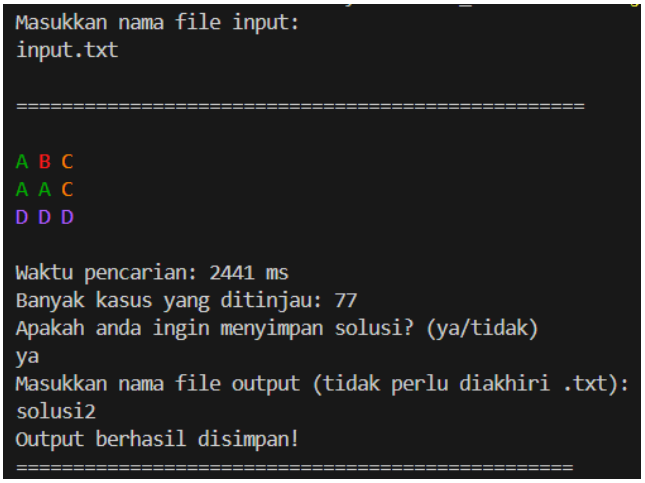
```
import java.util.Scanner;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import function.*;
// import java.io.FileNotFoundException;

public class Main {
    public static void main(String[] args) {
        long startTime = System.currentTimeMillis();

        System.out.println("Masukkan nama file input: ");
        Scanner scanner = new Scanner(System.in);
        String fileName;
        boolean fileExists = false;

        while (!fileExists) {
            fileName = scanner.nextLine();
            try {
                Board board = Input.readInput("../test/" +
fileName);
                if (board == null){
                    break;
                }
                Solver.Result result =
Solver.solve(board.board, board.pieces, 0);
                long endTime = System.currentTimeMillis();
                System.out.println("\nWaktu pencarian: " +
(endTime - startTime) + " ms");
```

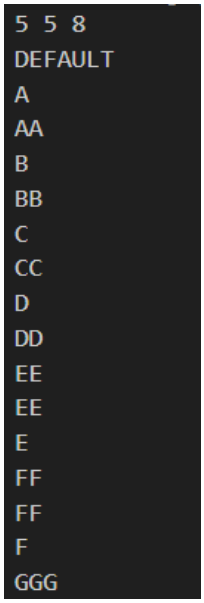
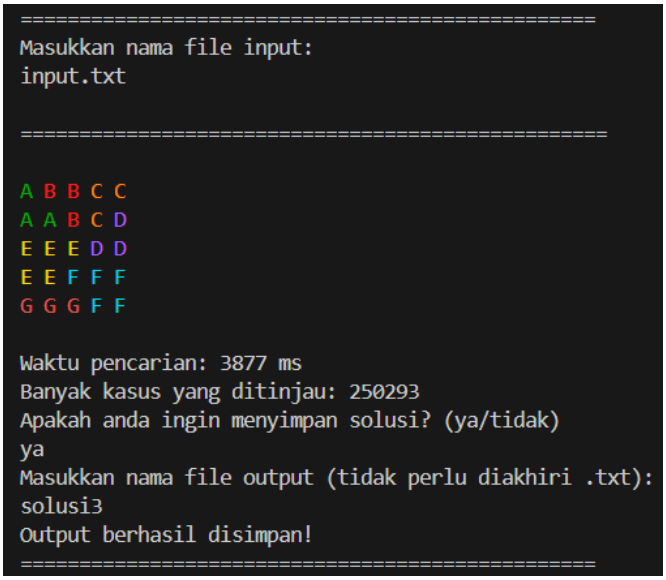
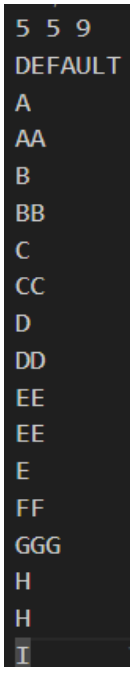
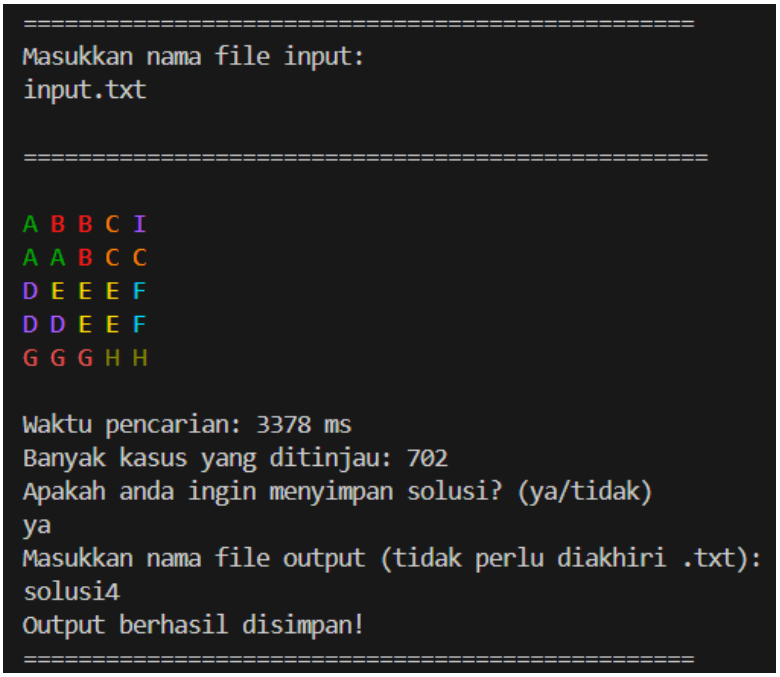
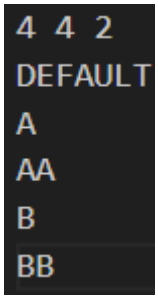
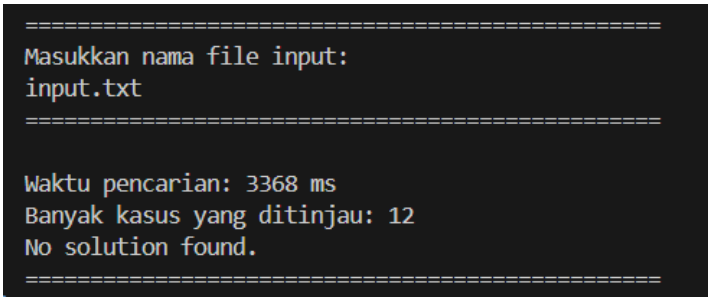
```
                if (result != null) {
                    System.out.println("Banyak kasus yang
ditinjau: " + result.count);
                    Scanner save_scanner = new
Scanner(System.in);
                    String save;
                    String namaFile;
                    while (true) {
                        System.out.println("Apakah anda ingin
menyimpan solusi? (ya/tidak)");
                        save = save_scanner.nextLine();
                        if (save.equals("ya")) {
                            System.out.println("Masukkan nama
file output (tidak perlu diakhiri .txt): ");
                            Scanner scan = new
Scanner(System.in);
                            namaFile = scan.nextLine();
                            scan.close();
                            outputFile(result.board, "../test/"
+ namaFile + ".txt");
                            break;
                        } else if (save.equals("tidak")) {
                            break;
                        } else {
                            System.out.println("\nInput tidak
valid, silakan coba lagi.");
                        }
                    }
                    save_scanner.close();
                } else {
                    System.out.println("No solution found.");
                }
                fileExists = true;
            }
            catch (IOException e) {
                System.out.println("\nFile tidak ditemukan atau
terjadi kesalahan, masukkan nama file input kembali: ");
            }
        }
        scanner.close();

    }
```

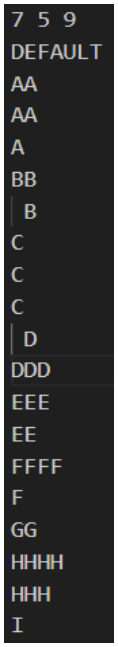
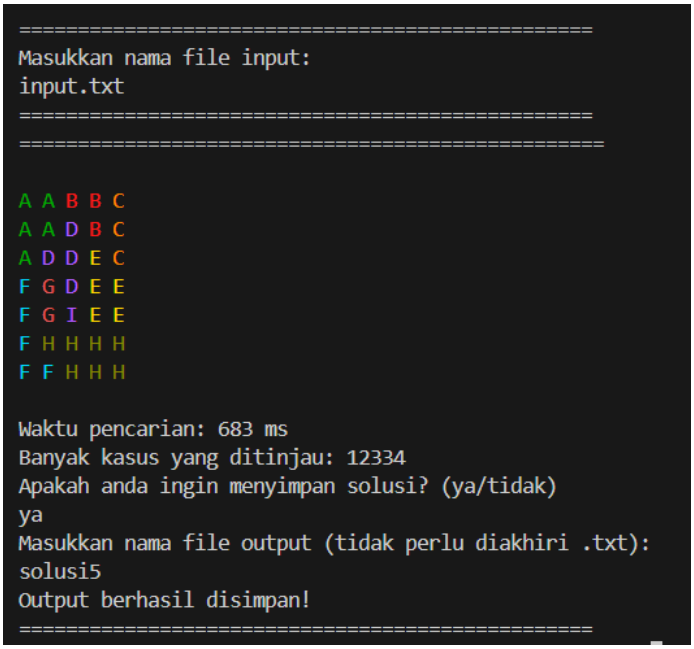
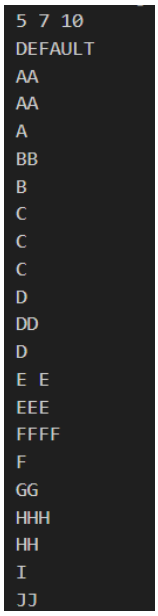
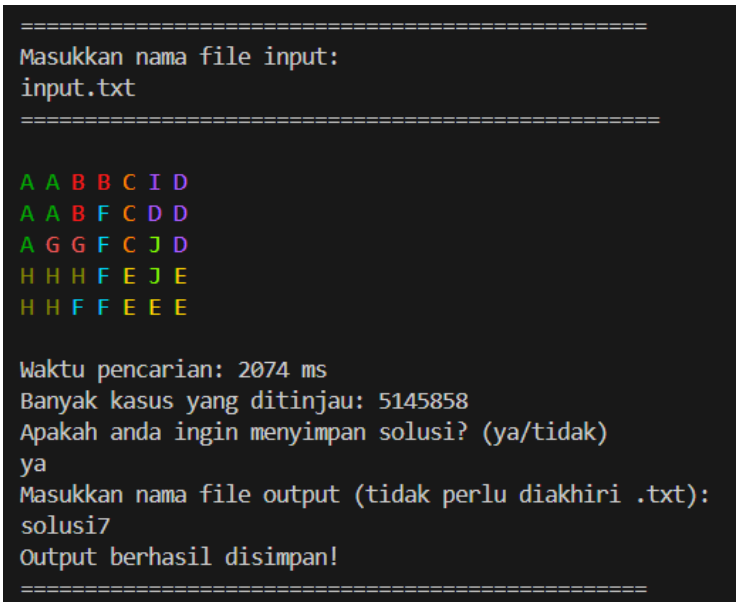
```
    public static void outputFile(char[][] board, String
filenames){
        try (BufferedWriter writer = new BufferedWriter(new
FileWriter(filenames))) {
            for (char[] line : board) {
                writer.write(line);
                writer.newLine();
            }
        } catch (IOException e){
            System.out.println("error writing to file!");
        }
    }
}
```

BAB III

EKSPERIMEN

No	Input.txt	Output
1		
2		

3		
4		
5	 *Contoh input salah*	

6	 7 5 9 DEFAULT AA AA A BB B C C C D DDD EEE EE FFFF F GG HHHH HHH I	 Masukkan nama file input: input.txt A A B B C A A D B C A D D E C F G D E E F G I E E F H H H H F F H H H Waktu pencarian: 683 ms Banyak kasus yang ditinjau: 12334 Apakah anda ingin menyimpan solusi? (ya/tidak) ya Masukkan nama file output (tidak perlu diakhiri .txt): solusi5 Output berhasil disimpan!
7	 5 7 10 DEFAULT AA AA A BB B C C C D DD D E E EEE FFFF F GG HHH HH I JJ	 Masukkan nama file input: input.txt A A B B C I D A A B F C D D A G G F C J D H H H F E J E H H F F E E E Waktu pencarian: 2074 ms Banyak kasus yang ditinjau: 5145858 Apakah anda ingin menyimpan solusi? (ya/tidak) ya Masukkan nama file output (tidak perlu diakhiri .txt): solusi7 Output berhasil disimpan!

BAB IV
PRANALA & LAMPIRAN

5.1. Pranala

https://github.com/naylzhra/Tucil1_13523079

5.2. 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 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	✓	