LAPORAN TUGAS KECIL 1
IF 2211
STRATEGI ALGORITMA

Disusun Oleh:

Abrar Abhirama Widyadhana
13523038

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
SEMESTER 2 TAHUN 2024/2025

DAFTAR ISI

BAB 1
Deskripsi Masalah

Gambar 1 Permainan IQ Puzzler Pro
(Sumber: https://www.smartgamesusa.com)

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.

Tugas anda adalah menemukan **cukup satu solusi** dari permainan **IQ Puzzler Pro** dengan menggunakan ***algoritma Brute Force***, atau menampilkan bahwa solusi tidak ditemukan jika tidak ada solusi yang mungkin dari puzzle.

Ilustrasi kasus

Diberikan sebuah wadah berukuran 11 x 5 serta 12 buah blok puzzle dan beberapa blok telah ditempatkan dengan bentuk sebagai berikut

Gambar 2 Awal Permainan Game IQ Puzzler Pro
(Sumber: https://www.smartgamesusa.com)

Pemain berusaha untuk mengisi bagian papan yang kosong dengan menggunakan blok yang tersedia

PERHATIAN: Untuk tucil ini permainan diawali dengan papan kosong

Gambar 3 Pemain Mencoba Semua Kemungkinan
(Sumber: https://www.smartgamesusa.com)

Permainan dinyatakan selesai jika pemain mampu mengisi seluruh papan dengan blok (dalam tugas kecil ini ada kemungkinan pemain tidak dapat mengisi seluruh papan)

Gambar 4 Pemain Menyelesaikan Permainan
(Sumber: https://www.smartgamesusa.com)

Agar lebih jelas, amati video cara bermain berikut:
https://youtube.com/shorts/MWiPAS3wfGM?feature=shared

Spesifikasi Wajib

- Buatlah program sederhana dalam bahasa **Java** yang mengimplementasikan ***algoritma Brute Force*** untuk mencari solusi dalam permainan IQ Puzzler Pro.
- Algoritma brute force yang diimplementasikan harus bersifat **"murni"**, tidak boleh memanfaatkan heuristik.

- Papan yang perlu diisi mulanya akan selalu kosong.
- Sebuah blok puzzle bisa saja dirotasi maupun dicerminkan sebelum diletakan pada papan.
- **Input:** program akan memberikan pengguna sebuah prompt untuk memilih file *test case* berekstensi **.txt**, kemudian program membaca file *test case* tersebut yang berisi
  1. **Dimensi Papan** terdiri atas dua buah variabel **N** dan **M** yang membentuk papan berdimensi NxM.
  2. **Banyak blok puzzle** direpresentasikan oleh variabel integer **P**.
  3. **Jenis kasus** sebuah variabel string **S** yang digunakan untuk mengidentifikasi kasus konfigurasi, hanya mungkin bernilai salah satu diantara **DEFAULT/CUSTOM/PYRAMID**.
  4. **Bentuk blok puzzle** yang dilambangkan oleh konfigurasi *Character* berupa huruf. Akan ada **P** buah blok puzzle berbeda yang dibentuk oleh **P buah huruf berbeda**. *Character* yang digunakan adalah huruf **A-Z dalam kapital**.

File .txt yang akan dibaca memiliki format sebagai berikut

```
N M P
S
puzzle_1_shape
puzzle_2_shape
…
puzzle_P_shape
```

Contoh *Test case*

```
5 5 8
DEFAULT
A
AA
B
BB
C
CC
D
DD
EE
EE
E
FF
```

```
FF
F
GGG
```

Contoh *output* berdasarkan contoh *Input* diatas

```
AGGGD
AABDD
CCBBF
CEEFF
EEEFF

Waktu pencarian: 604 ms

Banyak kasus yang ditinjau: 7387

Apakah anda ingin menyimpan solusi? (ya/tidak)
```

Output:

5. Tampilkan **konfigurasi blok puzzle** yang berhasil mengisi papan. **Gunakan print berwarna** untuk menunjukkan blok puzzle dengan jelas. Pastikan setiap blok puzzle berbeda memiliki warna berbeda. Beri tahu pengguna apabila puzzle tidak memiliki solusi.
6. **Waktu eksekusi** program dalam *milisecond* (tidak termasuk waktu membaca masukan dan menyimpan solusi, cukup waktu pencarian oleh algoritma).
7. **Banyak kasus** atau jumlah iterasi yang ditinjau oleh algoritma brute force.
8. **Prompt untuk menyimpan solusi** dalam sebuah berkas berekstensi **.txt** (Struktur untuk file output dibebaskan).

BAB 2
Teori Singkat

Brute Force adalah metode pencarian solusi. Brute force adalah metode untuk mencari solusi sederhana tetapi memakan waktu. Definisi masalah dengan mengerucutkan ke solusi yang memungkinkan dengan mencoba semua kemungkinan atau kombinasi elemen penyusun sampai jawaban benar ditemukan. Dalam konteks komputer dan keamanan siber, definisi brute force juga berlaku, metode brute force digunakan untuk memecahkan masalah enkripsi atau kata sandi. Ini adalah teknik brute force karena semua kemungkinan harus dicoba secara berurutan sampai yang cocok ditemukan.

Metode pencarian ini memiliki keuntungan kesederhanaan tetapi kelemahan adalah bahwa waktu pencarian bisa sangat lama jika ruang pencarian ini sangat besar. Masalah yang dapat dipecahkan dengan metode algoritma brute force adalah kebanyakan masalah getah masalah, berarti mereka memiliki jumlah solusi eksponensial yang harus diuji satu per satu.

BAB 3
Implementasi

Algoritma Brute Force

Pendekatan brute force digunakan untuk mencoba meletakan setiap blok pada papan hingga semua papan terisi dengan benar dan mencoba semua kemungkinan sampai dapat jika ada solusi.

Langkah – Langkah Algoritma Brute Force

1. **Inisialisasi**
    a. Mendefinisikan papan
    b. Membaca blok kemudian merotasi dan mencerminkan setiap blok
2. **Rekursif**
    a. Mulai Rekursif dengan basis jika papan sudah penuh maka cetak solusi dan berhenti
    b. Jika semua blok telah di tempatkan, periksa apakah papannya sudah penuh
3. **Iterasi**
    a. Ambil 1 blok dari list
    b. Iterasi semua kemungkinan bentuk (ada 8 kemungkinan) dari blok
    c. Untuk setiap kemunngkinan blok, coba semua kemungkinan posisi di dalam papan dengan nested loop
4. **Validasi**
    a. Sebelum meletakan blok, periksa apakah blok dapat di tempatkan dngan memeriksa terdapat blok lain atau di batas luas papan.
    b. Jika bias di tempatkan maka panggil fungsi itu kembali dengan index + 1 secara rekursif

5. **Backtracking**
    a. Jika tidak ada penempaatan yang menghasilkan solusi, kembalikan keadaan papan ke kondisi sebelumnya dengan menghapus blok yang terakhir di tempatkan
    b. Coba bentuk lain
6. **Cetak solusi**
    a. Jika semua blok dapat di tempatkan maka cetak solusi dengan papannya
    b. Lalu jika setelah mencoba semua kemungkinan tidak ada solusi, cetak pesan bahwa solusi tidak di temukan

Kompleksitas Algoritma

Karena algoritma ini mencoba semua kemungkinan, maka kompleksitasnya sangat tinggi, Dalam kasus terburuk adalah O(S ^ B * P), dengan

- S : jumlah kemungkinan rotasi setiap blok
- B : jumlah blok yang harus di tempatkan
- P : jumlah posisi yang dapat di tempati papan.

Notasi Algortima

```
function solvePuzzle (Board board, List of Block blocklist, integer index)-> boolean

    if (index = blocklist.size) then
        if (board.isComplete) then
            Output(printBoard)
            -> true
        -> false

    block <- blocklist[index]
    shapes <- block.getAllShape

    shape traversal in shapes:
        x traversal[0..board.rows]  do
            for y in range(board.cols)     do
                if (board.isPlaceAble)     then
                    board.placeBlock
                    if (solvePuzzle)       then
                        -> true
                    board.removeBlock

    -> false
```

BAB 4
Source Program

1. **Main.java**

```
package src;
import java.io.*;
import java.util.*;
import java.nio.file.*;
import src.*;

public class Main {
    public static void main(String[] args){
        try{
            // Baca File
            Scanner scanner = new Scanner(System.in);
            System.out.println("Masukkan file .txt,  ex (input1): ");
            String fileIn = scanner.nextLine().trim().toLowerCase();
            String fileInput = fileIn + ".txt";
            Path path = Paths.get("src", "testcase", fileInput);

            BufferedReader reader = new BufferedReader(new
FileReader(path.toFile()));
            List<String[][]> matrix = new ArrayList<>();
            String words;
            String mode;
            int currMax = -1;
            int rowBoard = 0, colBoard = 0, blocks = 0;

            // Simpan Semua Blocknya
            List<Block> blocklist = new ArrayList<>();
            List<String[][]> allShape = new ArrayList<>();

            // Baca Row, Cols, Jumlah Blok
            while ((words = reader.readLine()) != null &&
words.trim().isEmpty()) {
                continue;
            }

            if (words != null) {
```

```
                String[] word = words.split(" ");
                if (word.length >= 3){
                    rowBoard = Integer.parseInt(word[0]);
                    colBoard = Integer.parseInt(word[1]);
                    blocks = Integer.parseInt(word[2]);

                    if (rowBoard <= 0 || colBoard <= 0){
                        reader.close();
                        throw new IllegalArgumentException("Dimensi
papan tidak valid");
                    }
                    // System.out.println("row = " + rowBoard);
                    // System.out.println("col = " + colBoard);
                    // System.out.println("blocks = " + blocks);
                }
            }


            while ((mode = reader.readLine()) != null &&
mode.trim().isEmpty()) {
                continue; // Lewati baris kosong
            }

            Board board = null;

            if (mode.equals("CUSTOM")) {
                char[][] customGrid = new char[rowBoard][colBoard];

                for (int i = 0; i < rowBoard; i++) {
                    String line = reader.readLine().trim();
                    for (int j = 0; j < colBoard; j++) {
                        customGrid[i][j] = line.charAt(j);
                    }
                }

                board = new Board(rowBoard, colBoard, customGrid);
            }

            else if(mode.equals("DEFAULT")){
                board = new Board(rowBoard, colBoard);
            }

            else{
                System.out.println("Jenis Kasus Tidak Valid
(DEAFULT/CUSTOM)");
```

```
                System.exit(1);
            }

            List<String[]> currentCharMatrix= new ArrayList<>();
            String prevchar = "";

            while((words = reader.readLine()) != null){
                if (words.trim().isEmpty()) {
                    continue; // Skip baris kosong
                }
                String dots = words.replace(" ", ".");
                // Ubah 1 line jadi array
                String[] row = dots.split("");
                if (row.length > currMax){
                    currMax = row.length;
                }

                int i = 0;
                //cari sampai bukan "." , harus char A-Z
                while (row[i].equals(".") && i < row.length){
                    i++;
                }

                String currentChar = row[i];

                // System.out.println(currentChar);

                i = 0;

                if (!currentChar.equals(prevchar) &&
!currentCharMatrix.isEmpty()){
                    // biar N x M, yang spasi di isi "."
                    for (int k = 0; k< currentCharMatrix.size(); k++){
                        String[] rowMatrix = currentCharMatrix.get(k);
                        if (rowMatrix.length < currMax){
                            String[] newRowMatrix = new String[currMax];
                            for (int z = 0; z < rowMatrix.length; z++){
                                newRowMatrix[z] = rowMatrix[z];
                            }

                            for(int x = rowMatrix.length; x < currMax
;x++){
                                newRowMatrix[x] = ".";
                            }
```

```
                            currentCharMatrix.set(k, newRowMatrix);
                        }
                    }
                    matrix.add(currentCharMatrix.toArray(new
String[0][]));

                    // Ubah list ke matrix (biar gampang)

                    String[][] matrixblock =
Block.convertToMatrix(currentCharMatrix);
                    String[][] block90 =
Block.rotateBlock90(matrixblock);
                    String[][] block180 =
Block.rotateBlock180(matrixblock);
                    String[][] block270 =
Block.rotateBlock270(matrixblock);
                    String[][] blockMirror =
Block.mirrorVertical(matrixblock);
                    String[][] blockMirror90 =
Block.mirrorVertical(block90);
                    String[][] blockMirror180 =
Block.mirrorVertical(block180);
                    String[][] blockMirror270 =
Block.mirrorVertical(block270);
                    allShape.add(matrixblock);
                    allShape.add(block90);
                    allShape.add(block180);
                    allShape.add(block270);
                    allShape.add(blockMirror);
                    allShape.add(blockMirror90);
                    allShape.add(blockMirror180);
                    allShape.add(blockMirror270);

                    List<String[][]> newAllShape = new
ArrayList<>(allShape);
                    Block block = new Block(prevchar.charAt(0),
matrixblock, new ArrayList<>(newAllShape));
                    blocklist.add(block);
                    currentCharMatrix.clear();
                    allShape.clear();
                    currMax = -1;
                }

                currentCharMatrix.add(row);
                prevchar = currentChar;
```

```
            }

            if (!currentCharMatrix.isEmpty()){
                matrix.add(currentCharMatrix.toArray(new String[0][]));
                String[][] matrixblock =
Block.convertToMatrix(currentCharMatrix);
                String[][] block90 = Block.rotateBlock90(matrixblock);
                String[][] block180 = Block.rotateBlock180(matrixblock);
                String[][] block270 = Block.rotateBlock270(matrixblock);
                String[][] blockMirror =
Block.mirrorVertical(matrixblock);
                String[][] blockMirror90 =
Block.mirrorVertical(block90);
                String[][] blockMirror180 =
Block.mirrorVertical(block180);
                String[][] blockMirror270 =
Block.mirrorVertical(block270);
                allShape.add(matrixblock);
                allShape.add(block90);
                allShape.add(block180);
                allShape.add(block270);
                allShape.add(blockMirror);
                allShape.add(blockMirror90);
                allShape.add(blockMirror180);
                allShape.add(blockMirror270);

                List<String[][]> newAllShape = new
ArrayList<>(allShape);
                Block block = new Block(prevchar.charAt(0), matrixblock,
new ArrayList<>(newAllShape));
                blocklist.add(block);
            }


            reader.close();

            if (blocklist.size() != blocks) {
                System.out.println("Jumlah block tidak sama dengan block
yang diinput.");
                System.exit(1);
            }

            // Debug
            // System.out.println("\nDaftar Blok:");
            // for (Block b : blocklist) {
```

```
            //     System.out.println("Block ID: " + b.getId());
            //     Block.printMatrix(Block.convertCharToString(b.getShap
e()));

            // }

            int totalCells = rowBoard * colBoard;
            int totalBlockCells = 0;

            if (mode.equals("DEFAULT")){
                for (Block block : blocklist) {
                    char[][] shape = block.getShape();
                    for (char[] row : shape) {
                        for (char cell : row) {
                            if (cell != '.') {
                                totalBlockCells++;
                            }
                        }
                    }
                }

                if (totalBlockCells < totalCells) {
                    System.out.println("Blok yang diberikan kurang untuk
mengisi papan.\nTidak ada solusi.");
                    return;
                } else if (totalBlockCells > totalCells) {
                    System.out.println("Blok yang diberikan lebih banyak
dari yang dibutuhkan.\nTidak ada solusi.");
                    return;
                }
            }

            long startTime = System.nanoTime();
            BruteForce bruteforce = new BruteForce(board, blocklist);



            if (!bruteforce.solve()) {
                System.out.println("Solusi Tidak Ada");
                long endTime = System.nanoTime();

                long durationNano = (endTime - startTime);
                long durationMillis = durationNano / 1_000_000;
                System.out.println("Waktu pencarian: " + "("+
durationMillis + " ms)");
```

```
            }
            else{
                long endTime = System.nanoTime();
                long durationNano = (endTime - startTime);
                long durationMillis = durationNano / 1_000_000;
                System.out.println("Waktu pencarian: " + "("+
durationMillis + " ms)");
                System.out.println();
                System.out.println("Apakah anda ingin menyimpan solusi?
(ya/tidak)");
                Scanner scanners = new Scanner(System.in);
                String save = scanners.nextLine().trim().toLowerCase();
                if (save.equals("ya")){
                    System.out.print("Silahkan pilih jenis file
output\n1. Image\n2. Txt\n( 1 / 2 )? : ");
                    String output = scanners.nextLine().trim();
                    System.out.print("\nKetikkan Nama file yang Akan
Dibuat: ");
                    String File = scanners.nextLine().trim();
                    if (output.equals("1")){
                        board.saveImage(File + ".png");
                    }

                    else if (output.equals("2")){
                        try(BufferedWriter writer = new
BufferedWriter(new FileWriter(File))){
                            char[][] grid = board.getGrid();
                            for (int j= 0; j < grid.length; j++){
                                for (int k =0; k < grid[j].length;k++){
                                    writer.write(grid[j][k] + " ");
                                }
                                writer.newLine();
                            }
                        }
                        catch (IOException error){
                            System.out.println("Ada Error dalam
menyimpan file");
                        }
                    }


                }

                else if (save.equals("tidak")){
                    System.out.println("Tidak di Simpan");
```

```
                }

                else{
                    System.out.println("Input Tidak Valid");
                }
                scanners.close();

            }

        }
        catch(FileNotFoundException e ){
            System.out.println("File Tidak di Temukan");
        }

        catch(IOException e){
            e.printStackTrace();
        }

    }

}
```

2. **BruteForce.java**

```
package src;
import java.util.*;

public class BruteForce {
    private Board board;
    private List<Block> blocklist;
    private boolean success;
    private int count;

    public BruteForce(Board board, List<Block> blocklist) {
        this.board = board;
        this.blocklist = blocklist;
        this.success = false;
        this.count = 0;
    }

    public boolean solve() {
        boolean result = putBlock(0);
```

```
        System.out.println("Banyak kasus yang ditinjau : " + count );
        System.out.println();
        return result;
    }

    private boolean putBlock(int index) {
        if (index == blocklist.size()) {
            if (board.isComplete()) {
                printSolution();
                success = true;
                return true;
            }
            return false;
        }

        Block block = blocklist.get(index);
        List<String[][]> shapes = block.getAllShape();

        for (int shapeIndex = 0; shapeIndex < shapes.size();
shapeIndex++) {
            for (int x = 0; x < board.getGrid().length; x++) {
                for (int y = 0; y < board.getGrid()[0].length; y++) {
                    count += 1;
                    if (board.isPlaceAble(block, shapeIndex, x, y)) {
                        board.placeBlock(block, shapeIndex, x, y);
                        if (putBlock(index + 1)) {
                            return true;
                        }
                        board.removeBlock(block, shapeIndex, x, y);
                    }
                }
            }
        }

        return false;
    }

    private void printSolution() {
        System.out.println("Solusi ditemukan :\n");
        board.printBoard();

    }
}
```

3. **Board.java**

```java
package src;
import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import javax.imageio.ImageIO;

public class Board {
    private int row;
    private int col;
    private char[][] grid;
    private char[][] customGrid;

    public Board(int row, int col) {
        this.row = row;
        this.col = col;
        this.grid = new char[row][col];
        this.customGrid = new char[row][col];

        for (int i = 0; i < row; i++) {
            for (int j = 0; j < col; j++) {
                this.grid[i][j] = '.';
                this.customGrid[i][j] = 'X';
            }
        }
    }

    public Board(int row, int col, char[][] customGrid) {
        this.row = row;
        this.col = col;
        this.grid = new char[row][col];
        this.customGrid = customGrid;

        for (int i = 0; i < row; i++) {
            for (int j = 0; j < col; j++) {
                if (customGrid[i][j] == 'X') {
                    this.grid[i][j] = '.';
                } else {
                    this.grid[i][j] = ' ';
                }
            }
```

```
        }
    }


    public char[][] getGrid(){
        return grid;
    }

    public boolean isPlaceAble(Block block, int shapeIndex, int X, int
Y){
        char[][] shape =
convertToCharMatrix(block.getAllShape().get(shapeIndex));
        int rowBlock = shape.length;
        int colBlock = shape[0]. length;

        if (X + rowBlock > row || Y + colBlock > col){
            return false;
        }

        for (int i = 0; i < rowBlock; i++) {
            for (int j = 0; j < colBlock; j++) {
                if (shape[i][j] != '.') {
                    if (grid[X + i][Y + j] != '.' || customGrid[X + i][Y
+ j] != 'X') {
                        return false;
                    }
                }
            }
        }

        return true;
    }

    public void placeBlock(Block block, int shapeIndex, int X, int Y){
        char[][] shape =
convertToCharMatrix(block.getAllShape().get(shapeIndex));
        int rowBlock = shape.length;
        int colBlock = shape[0].length;

        for (int i = 0; i < rowBlock; i++){
            for (int j = 0; j < colBlock ; j++){
                if (shape[i][j] != '.'){
                    grid[X + i][Y + j] = shape[i][j];
                }
            }
```

```
        }
    }

    public void removeBlock(Block block, int shapeIndex, int startX, int
startY) {
        char[][] shape =
convertToCharMatrix(block.getAllShape().get(shapeIndex));
        int blockRows = shape.length;
        int blockCols = shape[0].length;

        for (int i = 0; i < blockRows; i++) {
            for (int j = 0; j < blockCols; j++) {
                if (shape[i][j] != '.') {
                    grid[startX + i][startY + j] = '.';
                }
            }
        }
    }

    public boolean isComplete(){
        int rowBoards = grid.length;
        int colBoards = grid[0].length;

        for (int i =0 ;i < rowBoards ; i++){
            for (int j = 0; j < colBoards; j++){
                if (customGrid[i][j] == 'X' && grid[i][j] == '.') {
                    return false;
                }
            }
        }

        return true;
    }

    public void printBoard(){
        String[] color = {
            "\u001B[38;5;196m",
            "\u001B[38;5;202m",
            "\u001B[38;5;208m",
            "\u001B[38;5;214m",
            "\u001B[38;5;220m",
            "\u001B[38;5;226m",
            "\u001B[38;5;190m",
            "\u001B[38;5;46m",
            "\u001B[38;5;47m",
```

```
            "\u001B[38;5;51m",
            "\u001B[38;5;27m",
            "\u001B[38;5;21m",
            "\u001B[38;5;57m",
            "\u001B[38;5;93m",
            "\u001B[38;5;129m",
            "\u001B[38;5;201m",
            "\u001B[38;5;165m",
            "\u001B[38;5;198m",
            "\u001B[38;5;160m",
            "\u001B[38;5;124m",
            "\u001B[38;5;166m",
            "\u001B[38;5;202m",
            "\u001B[38;5;214m",
            "\u001B[38;5;118m",
            "\u001B[38;5;50m",
            "\u001B[38;5;21m"
        };

        String reset = "\u001B[0m";
        for (int i = 0; i < this.row; i++) {
            for (int j = 0; j < this.col; j++) {
                char cell = grid[i][j];
                if (cell == '.') {
                    System.out.print(".  ");
                }
                else if (cell < 'A' || cell > 'Z') {
                    System.out.print(cell + "  ");
                }
                else {
                    int colorsIdx = (cell - 'A') % color.length;
                    System.out.print(color[colorsIdx] + cell + "  " +
reset);
                }
            }
            System.out.println();
        }
        System.out.println();
    }

    public void saveImage(String file) {
        int cellSize = 50;
        int width = col * cellSize;
        int height = row * cellSize;
```

```
        BufferedImage image = new BufferedImage(width, height,
BufferedImage.TYPE_INT_RGB);
        Graphics2D g = image.createGraphics();

        Color[] colors = {
            new Color(255, 0, 0), new Color(255, 95, 0), new Color(255,
135, 0),
            new Color(255, 175, 0), new Color(255, 215, 0), new
Color(255, 255, 0),
            new Color(190, 255, 0), new Color(0, 255, 0), new Color(0,
255, 95),
            new Color(0, 255, 255), new Color(0, 175, 255), new Color(0,
0, 255),
            new Color(95, 0, 255), new Color(175, 0, 255), new
Color(215, 0, 255),
            new Color(255, 0, 215), new Color(255, 0, 135), new
Color(255, 0, 175),
            new Color(255, 0, 95), new Color(215, 0, 0), new Color(175,
95, 0),
            new Color(255, 95, 0), new Color(255, 175, 0), new Color(0,
255, 135),
            new Color(0, 175, 95), new Color(0, 0, 175)
        };

        g.setColor(Color.BLACK);
        g.fillRect(0, 0, width, height);

        for (int i = 0; i < row; i++) {
            for (int j = 0; j < col; j++) {
                char block = grid[i][j];
                int x = j * cellSize;
                int y = i * cellSize;

                g.setColor(Color.WHITE);
                g.drawRect(x, y, cellSize, cellSize);

                if (block == '.' || block == ' ') {
                   continue;
                } else if (block >= 'A' && block <= 'Z') {
                    int colorIndex = (block - 'A') % colors.length;
                    g.setColor(colors[colorIndex]);
                }

                g.fillOval(x + 1, y + 1, cellSize - 4, cellSize - 4);
            }
```

```
        }

        g.dispose();

        try {
            ImageIO.write(image, "png", new File(file));
            System.out.println("Solusi telah disimpan di: " + file);
        } catch (IOException e) {
            System.out.println("Gagal menyimpan gambar: " +
e.getMessage());
        }
    }


    private char[][] convertToCharMatrix(String[][] shape) {
        int rows = shape.length;
        int cols = shape[0].length;
        char[][] result = new char[rows][cols];

        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                result[i][j] = shape[i][j].charAt(0);
            }
        }
        return result;
    }
}
```

4. **Block.java**

```
package src;
import java.util.*;
public class Block {
    private char id;
    private char[][] shape;
    private List<String[][]> allShape;

    public Block(char id, String[][] shape, List<String[][]> allShape) {
        this.id = id;
        // ubah String[][] jadi char[][]
        this.shape = new char[shape.length][];
        for (int i = 0; i < shape.length; i++) {
```

```
            this.shape[i] = new char[shape[i].length];
            for (int j = 0; j < shape[i].length; j++) {
                this.shape[i][j] = shape[i][j].charAt(0);
            }
        }
        this.allShape = allShape;
    }

    public char getId() {
        return id;
    }

    public char[][] getShape(){
        return shape;
    }

    public List<String[][]> getAllShape(){
        return allShape;
    }

    public void printShape() {
        for (int i = 0; i < shape.length; i++) {
            for (int j = 0; j < shape[i].length; j++) {
                System.out.print(shape[i][j] + " ");
            }
            System.out.println();
        }
    }

    public static String[][] rotateBlock90(String[][] shape){
        int row = shape.length;
        int cols = shape[0].length;
        String[][] result = new String[cols][row];

        for (int i = 0; i < row ;i++){
            for (int j = 0; j < cols; j++){
                result[j][row-1-i] = shape[i][j];
            }
        }
        return result;
    }

    public static String[][] rotateBlock180(String[][] shape){
        return(rotateBlock90(rotateBlock90(shape)));
    }
```

```
    public static String[][] rotateBlock270(String[][] shape){
        return(rotateBlock90(rotateBlock90(rotateBlock90(shape))));
    }

    public static String[][] mirrorVertical(String[][] shape) {
        int rows = shape.length;
        int cols = shape[0].length;
        String[][] result = new String[rows][cols];

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

    public static void printMatrix(String[][] matrix) {
        for (int i = 0; i < matrix.length; i++) {
            for (int j = 0; j < matrix[0].length; j++) {
                System.out.print(matrix[i][j] + " ");
            }
            System.out.println();
        }
        System.out.println();
    }

    public static String[][] convertCharToString(char[][] charMatrix) {
        String[][] stringMatrix = new
String[charMatrix.length][charMatrix[0].length];
        for (int i = 0; i < charMatrix.length; i++) {
            for (int j = 0; j < charMatrix[i].length; j++) {
                stringMatrix[i][j] = String.valueOf(charMatrix[i][j]);
            }
        }
        return stringMatrix;
    }

    public static String[][] convertToMatrix(List<String[]> list) {
        int rows = list.size();
        int maxColumns = list.stream().mapToInt(arr ->
arr.length).max().orElse(0);

        String[][] matrix = new String[rows][maxColumns];
```

```
        for (int i = 0; i < rows; i++) {
            String[] rowArray = list.get(i);
            for (int j = 0; j < maxColumns; j++) {
                if (j < rowArray.length && rowArray[j] != null) {
                    matrix[i][j] = rowArray[j];
                } else {
                    matrix[i][j] = ".";
                }
            }
        }


        return matrix;
    }
```

BAB 5
Hasil dan Pembahasan

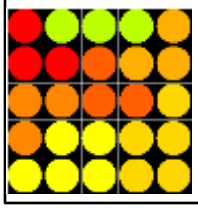
1. Jenis Kasus DEFAULT

a. Test Case 1

Input1.txt :

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

```
Masukkan file .txt,  ex (input1):
input1
Solusi ditemukan :

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

Banyak kasus yang ditinjau : 5372776

Waktu pencarian: (1671 ms)

Apakah anda ingin menyimpan solusi? (ya/tidak)
ya
Silahkan pilih jenis file output
1. Image
2. Txt
( 1 / 2 )? : 1

Ketikkan Nama file yang Akan Dibuat: input1
Solusi telah disimpan di: input1.png
```



b. Test Case 2

Input2.txt :

```
2 2 1
DEFAULT
AA
AA
```

```
Masukkan file .txt,  ex (input1):
input2
Solusi ditemukan :

A  A
A  A

Banyak kasus yang ditinjau : 1

Waktu pencarian: (35 ms)

Apakah anda ingin menyimpan solusi? (ya/tidak)
ya
Silahkan pilih jenis file output
1. Image
2. Txt
( 1 / 2 )? : 2

Ketikkan Nama file yang Akan Dibuat: input.txt


Input.txt
A A
A A
```

c. Test Case 3
   Input3.txt:
   Input3.txt :

```
3 3 2


DEFAULT

AA
AA
BB
BBB
```

```
Masukkan file .txt,  ex (input1):
input3
Banyak kasus yang ditinjau : 2376

Solusi Tidak Ada
Waktu pencarian: (15 ms)
```

d. Test Case 4
   Input4.txt:

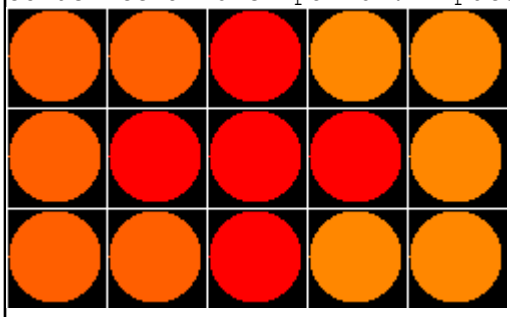
```
11 5 12
DEFAULT
A
AAA
BB
BBB
CC
 CC
D
DD
 DD
EEE
  EE
F
FFFF
GGGG
  G
 H
HH
 HH
II
I
J
J
JJJ
K
KK
K
LL
L
LLL
```

```
Masukkan file .txt,  ex (input1):
input4
Blok yang diberikan lebih banyak dari yang dibutuhkan.
Tidak ada solusi.
```

e. Test Case 5
Input5.txt :

```
3 5 3
DEFAULT
 A
AAA
 A
BB
B
BB
CC
C
CC
```

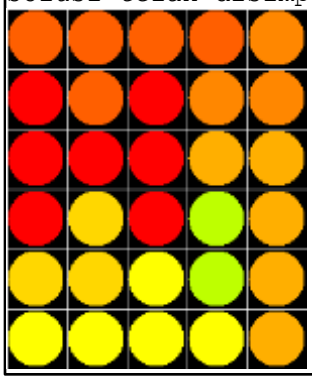
```
Masukkan file .txt,  ex (input1):
input5
Solusi ditemukan :

B  B  A  C  C
B  A  A  A  C
B  B  A  C  C

Banyak kasus yang ditinjau : 877

Waktu pencarian: (107 ms)

Apakah anda ingin menyimpan solusi? (ya/tidak)
ya
Silahkan pilih jenis file output
1. Image
2. Txt
( 1 / 2 )? : 1

Ketikkan Nama file yang Akan Dibuat: input5
Solusi telah disimpan di: input5.png
```



f. Test Case 6
Input6.txt

```
6 5 8
DEFAULT
A A
AAA
A A
BBBB
 B
CC
C
DDDD
D
EE
E
FFFF
  F
GG
```

```
Masukkan file .txt,  ex (input1):
input6
Solusi ditemukan :

B  B  B  B  C
A  B  A  C  C
A  A  A  D  D
A  E  A  G  D
E  E  F  G  D
F  F  F  F  D

Banyak kasus yang ditinjau : 4037859

Waktu pencarian: (1406 ms)

Apakah anda ingin menyimpan solusi? (ya/tidak)
ya
Silahkan pilih jenis file output
1. Image
2. Txt
( 1 / 2 )? : 1

Ketikkan Nama file yang Akan Dibuat: input6
Solusi telah disimpan di: input6.png
```



:

g. Test Case 7
Input7.txt:

```
7 6 6
DEFAULT
AA
AA
AA
BBB
B
BBB
 C
CCC
CCC
DD
DDD
DD
EEE
EE
```

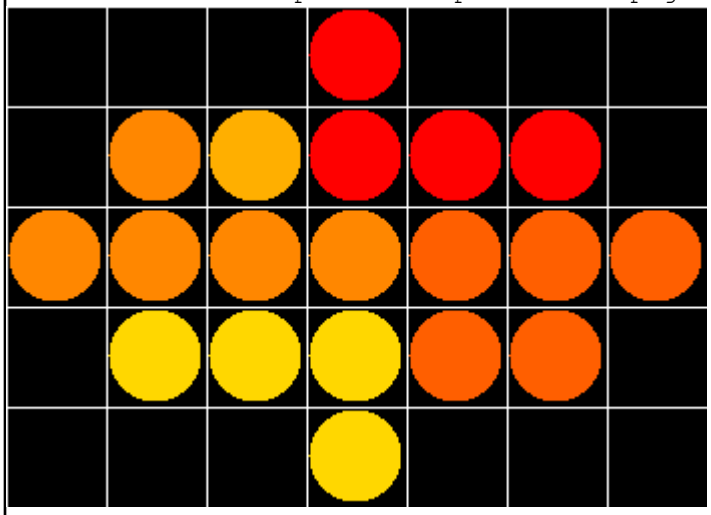
```
Masukkan file .txt,  ex (input1):
input7
Banyak kasus yang ditinjau : 834667344

Solusi Tidak Ada
Waktu pencarian: (171444 ms)
```

2. Jenis Kasus Custom

a. inputCustom1

```
5 7 5
CUSTOM
...X...
.XXXXX.
XXXXXXX
.XXXXX.
...X...
A
AAA
BBB
 BB
 C
CCCC
D
E
EEE
```

```
Masukkan file .txt,  ex (input1):
inputCustom1
Solusi ditemukan :

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

Banyak kasus yang ditinjau : 164914

Waktu pencarian: (93 ms)

Apakah anda ingin menyimpan solusi? (ya/tidak)
ya
Silahkan pilih jenis file output
1. Image
2. Txt
( 1 / 2 )? : 1

Ketikkan Nama file yang Akan Dibuat: inputCustom1
Solusi telah disimpan di: inputCustom1.png
```



b. inputCustom2

```
2 3 2
CUSTOM
X.X
.XX
AA
A
B
```

```
Masukkan file .txt,  ex (input1):
inputCustom2
Solusi ditemukan :

B     A
   A  A

Banyak kasus yang ditinjau : 15

Waktu pencarian: (37 ms)

Apakah anda ingin menyimpan solusi? (ya/tidak)
ya
Silahkan pilih jenis file output
1. Image
2. Txt
( 1 / 2 )? : 2

Ketikkan Nama file yang Akan Dibuat: inputCustom2


inputCustom2.txt:
B   A
  A A
```

Daftar Pustaka

- Aho, A.V., Hopcroft, J.E., & Ullman, J.D. (2003). *The Design and Analysis of Computer Algorithms*. Addison-Wesley.
- Stinson, D.R. (2006). *Cryptography: Theory and Practice*. CRC Press.

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	✓	