

IF2211 Strategi Algoritma

Penyelesaian IQ Puzzler Pro dengan Algoritma Brute Force

Laporan Tugas Kecil

Disusun untuk memenuhi tugas mata kuliah IF2211 Strategi Algoritma
pada Semester IV Tahun Akademik 2024/2025



Oleh
Samantha Laqueenna Ginting

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
JATINANGOR
2025

DAFTAR ISI

DAFTAR ISI	2
BAB 1 DESKRIPSI MASALAH	3
A. Algoritma Brute Force	3
B. Permainan IQ Puzzler Pro	3
BAB 2 PENYELESAIAN	5
BAB 3 IMPLEMENTASI PROGRAM	6
A. Struktur Data	6
B. Kelas Algorithm.java	6
1. Algorithm.java	6
2. FileHandler.java	7
3. Main.java	7
C. Source Code Program	8
1. File Main.java	8
2. File FileHandler.java	9
3. File Algorithm.java	12
D. Kekas (Package)	13
1. Java.util (ArrayList, Scanner)	13
2. Java.io (File, FileWriter, IOException, FileNotFoundException)	14
3. Java.awt (Color, Graphics2D, image BufferedImage)	14
4. Javax.imageio	14
BAB 4 ANALISIS DAN PENGUJIAN	15
A. Kasus Uji 1 (Spesifikasi)	15
B. Kasus Uji 2	16
C. Kasus Uji 3	17
D. Kasus Uji 4	18
E. Kasus Uji 5	19
F. Kasus Uji 6	20
G. Kasus Uji 7	21
BAB 5 KESIMPULAN	24
LAMPIRAN	25
A. Github	25
B. Tabel Pemeriksaan	25
DAFTAR PUSTAKA	26

BAB 1 DESKRIPSI MASALAH

A. Algoritma Brute Force

Algoritma Brute Force merupakan sebuah algoritma yang melakukan pendekatan langsung untuk memecahkan sebuah masalah. Algoritma ini didasarkan rumusan masalah dan definisi konsep terlibat dalam persoalan. Algoritma Brute Force dapat memecahkan persoalan dengan sangat sederhana dan langsung, serta langkah penyelesaian yang jelas.

Algoritma ini dapat digunakan untuk memecahkan hampir semua masalah. Persoalan-persoalan komputasi umum seperti *searching* (pencarian), *sorting* (pengurutan), pencocokan string, dan perkalian matriks, dapat diselesaikan dengan Brute Force. Tidak hanya itu, Brute Force telah menjadi standar dalam berbagai persoalan komputasi seperti penjumlahan atau perkalian bilangan, dan menemukan elemen ekstrim dalam sebuah daftar. Kecil kemungkinannya untuk menemukan persoalan yang tidak dapat diselesaikan dengan algoritma ini. Bahkan, terdapat beberapa persoalan yang hanya dapat diselesaikan dengan Brute Force.

Kata *force* (kekuatan) dalam istilah Algoritma Brute Force mengimplikasikan bahwa algoritma ini hanya menggunakan tenaga atau kekuatan dibandingkan kecerdasan. Algoritma ini mengolah semua kombinasi atau permutasi data untuk mencari solusi. Oleh karena itu, Algoritma Brute Force merupakan algoritma yang tidak efisien, terutama untuk persoalan-persoalan yang kompleks dengan data yang banyak. Brute Force membutuhkan waktu yang cukup lama dan ruang yang besar dalam memecahkan masalah. Terkadang, algoritma ini disebut juga dengan *naive algorithm* (algoritma naif).

B. Permainan IQ Puzzler Pro

IQ Puzzler Pro adalah permainan teka-teki yang bertujuan untuk mengasah keterampilan pemecahan masalah, logika dan kemampuan spasial. Permainan ini dibuat oleh Smart Games, sebuah perusahaan asal Belgia yang mempunyai visi untuk tidak hanya merancang permainan yang seru, namun juga mengedukasi. Permainan ini juga dikenal dengan nama Kanoodle yang berasal dari perusahaan yang berbeda.

Permainan IQ Puzzler Pro mempunyai komponen penting yang dinamakan *board* dan *piece*. *Board* merupakan sebuah papan berukuran tertentu di mana *piece* akan ditempatkan. Seluruh area dalam papan harus dapat diisi dengan penuh. *Piece* adalah bola-bola dengan bentuk geometri tertentu. Pemain harus dapat menggunakan seluruh *piece* untuk menyelesaikan permainan. Dalam permainan ini, terdapat berbagai kemungkinan cara untuk menyelesaikan permainan.



Gambar 1. *Board* dan *Piece* Permainan IQ Puzzler Pro
(Sumber: <https://www.smartgamesusa.com/one-player-games/iq-puzzler-pro>)

BAB 2 PENYELESAIAN

Berikut adalah langkah-langkah penyelesaian permainan IQ Puzzler Pro dengan menerapkan Algoritma Brute Force.

1. Pemain diminta untuk menautkan *file* berekstensi .txt. *File* ini berisi dimensi *board* yang diinginkan, banyak *piece*, tipe permainan (yaitu DEFAULT, dimana papan berbentuk persegi panjang dan seluruh *piece* harus memenuhi papan tersebut), dan bentuk-bentuk *piece* yang dilambangkan dengan huruf-huruf A-Z dalam kapital.
2. Program akan menyimpan data yang ditaruh pemain dan mengolahnya ke dalam variabel yang sesuai. Seluruh *piece* akan disimpan dalam sebuah *hashmap*, dimana abjad merupakan *key*, dan rangkaian koordinat merupakan *value*. Papan permainan juga akan diinisialisasi dengan dimensi yang telah diberikan.
3. Proses *backtracking* dengan Brute Force pada program ini menggunakan rekursi. Rekursi pada penyelesaian permainan ini hanya diimplementasikan dalam pengulangan fungsi. Basis dari rekursi adalah saat *hashmap* berisikan sisa *piece* yang belum diletakkan sudah kosong.
4. Bagian rekursi dimulai dengan mencari seluruh bentuk kemungkinan *piece* satu per satu. Program akan merotasikan dan merefleksikan *piece*, lalu kembali menyimpannya ke dalam *array of variants*.
5. Kemudian, *piece* menelusuri setiap baris dan kolom papan dari bagian kiri atas, dan mengecek apakah *piece* tersebut dapat diletakkan tanpa keluar dari dimensi papan dan tanpa menimpak bagian papan yang telah terisi oleh *piece* lain. Jika *piece* memenuhi kondisi tersebut, *piece* akan diletakkan. Program akan mengurangi *piece* yang telah diletakkan dari *hashmap* sisa *piece*.
6. Akan tetapi, jika *piece* tidak dapat diletakkan di papan, program akan melakukan *backtracking* dengan menghapus *piece* terakhir di papan dan mencoba penempatan yang lain.
7. Program akan berhenti jika semua *piece* telah diletakkan dengan sempurna di papan, atau jika semua opsi gagal dan program akan menyatakan bahwa kasus tersebut tidak mempunyai solusi.

BAB 3 IMPLEMENTASI PROGRAM

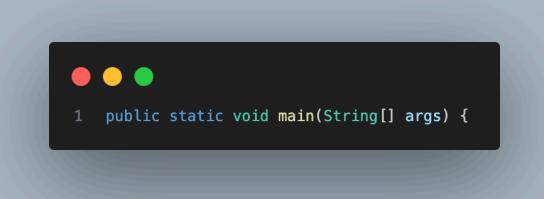
A. Struktur Data

```
Tucil1_13523138
└── bin
    ├── Algorithm.class
    ├── FileHandler.class
    └── Main.class
└── doc
    └── Tucil1_K3_13523138_SamanthaLG.pdf
└── solution
    ├── solution1.png
    ├── solution1.txt
    ├── solution2.png
    ├── solution2.txt
    ├── solution3.png
    ├── solution3.txt
    ├── solution6.png
    ├── solution6.txt
    ├── solution7.png
    └── solution7.txt
└── src
    ├── Algorithm.java
    ├── FileHandler.java
    └── Main.java
└── test
    ├── 1.txt
    ├── 2.txt
    ├── 3.txt
    ├── 4.txt
    ├── 5.txt
    ├── 6.txt
    └── 7.txt
└── README.md
```

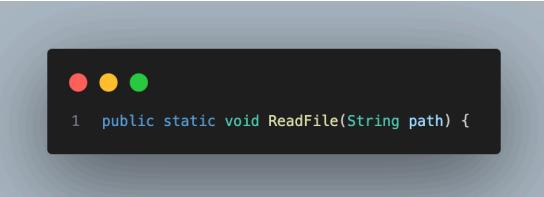
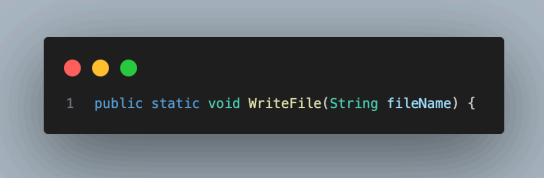
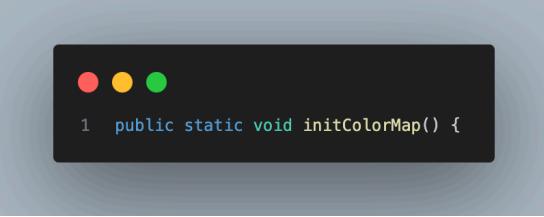
B. Fungsi dan Prosedur

1. Main.java

Fungsi/Prosedur	Tujuan
-----------------	--------

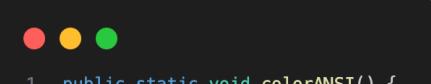
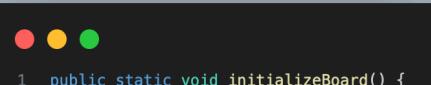
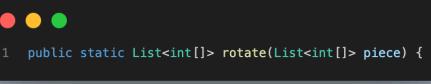
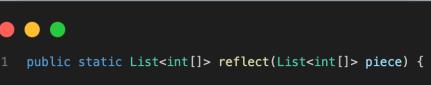
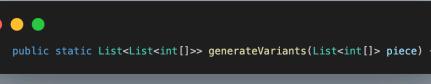
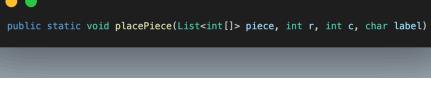
 <pre>1 public static void main(String[] args) {</pre>	<p>Menerima input nama <i>file</i> uji, memanggil fungsi dalam FileHandler.java untuk mengolah <i>file</i> input dan menuliskan hasil ke dalam bentuk txt atau png. Selain itu, fungsi ini memanggil fungsi dalam Algorithm.java untuk menjalankan Algoritma Brute Force dan menghasilkan penyelesaian.</p>
---	---

2. FileHandler.java

Fungsi/Prosedur	Tujuan
 <pre>1 public static void ReadFile(String path) {</pre>	Menerima <i>path directory file</i> uji, membaca isi dari <i>file</i> tersebut dan mengelola data ke variabel-variabel yang sesuai.
 <pre>1 public static void WriteFile(String fileName) {</pre>	Menerima nama <i>file</i> solusi yang ingin disimpan, dan menyimpan hasil <i>board</i> ke dalam bentuk <i>file</i> di folder “solution”.
 <pre>1 public static void initColorMap() {</pre>	Memetakan warna tertentu pada setiap abjad dari A sampai Z, untuk memberikan warna pada <i>file</i> gambar.
 <pre>1 public static void WriteImage(String fileName) {</pre>	Menerima nama <i>file</i> solusi yang ingin disimpan dalam bentuk gambar, dan menyimpan hasil <i>board</i> ke dalam bentuk <i>file</i> di folder “solution”.

3. Algorithm.java

Kelas	Tujuan
-------	--------

 <pre>1 public static void colorANSI() {</pre>	<p>Memetakan warna tertentu pada setiap abjad dari A sampai Z, untuk memberikan warna pada hasil di output terminal.</p>
 <pre>1 public static void initializeBoard() {</pre>	<p>Menginisialisasikan <i>board</i> dalam bentuk <i>array</i> sesuai dimensi dari <i>file</i> input. Isi awal <i>board</i> adalah karakter ‘.’.</p>
 <pre>1 public static List<int[]> rotate(List<int[]> piece) {</pre>	<p>Melakukan rotasi koordinat sejauh 90° searah jarum jam. Fungsi ini mengembalikan <i>List<int[]></i> atau <i>list</i> koordinat-koordinat.</p>
 <pre>1 public static List<int[]> reflect(List<int[]> piece) {</pre>	<p>Melakukan refleksi koordinat terhadap sumbu y atau secara vertikal. Fungsi ini mengembalikan <i>List<int[]></i> atau <i>list</i> koordinat-koordinat.</p>
 <pre>1 public static List<List<int[]>> generateVariants(List<int[]> piece) {</pre>	<p>Menghasilkan semua variasi <i>piece</i>, dan menyimpannya di <i>hashset</i> agar tidak ada duplikat. Di sini, fungsi rotasi dipanggil sebanyak 4 kali untuk merotasi sejauh 90°, 180°, 270°, dan 360°, serta pada tiap rotasi, fungsi refleksi dipanggil sebanyak satu kali. <i>Hashset</i> kemudian diubah menjadi <i>List<List<int[]>></i> dan dikembalikan.</p>
 <pre>1 public static boolean placeable(List<int[]> piece, int r, int c) {</pre>	<p>Mengecek apakah <i>piece</i> dapat diletakkan di <i>board</i> di posisi tertentu, dengan syarat bahwa <i>piece</i> tidak keluar dari dimensi <i>board</i>, dan <i>piece</i> tidak menimpa <i>piece</i> lain yang sudah diletakkan. Mengembalikan nilai <i>true/false</i>.</p>
 <pre>1 public static void placePiece(List<int[]> piece, int r, int c, char label) {</pre>	<p>Meletakkan <i>piece</i> per koordinat di posisi tertentu pada <i>board</i>. <i>Piece</i> mengantikan isi inisialisasi <i>board</i> yaitu ‘.’ dengan abjad yang melambangkan <i>piece</i>.</p>

<pre>1 public static void removePiece(List<int[]> piece, int r, int c) {</pre>	<p>Menghapus <i>piece</i> dari <i>board</i> per koordinat di posisi tertentu. Abjad dari <i>piece</i> dihapus dengan menggantikannya ke karakter awal yaitu ‘.’.</p>
<pre>1 public static boolean solve(List<Map.Entry<String, List<int[]>>> remainingPieces) {</pre>	<p>Melakukan Algoritma Brute Force. Subprogram ini menerima <i>hashmap</i> yang berisi abjad sebagai <i>key</i> dan koordinat-koordinat sebagai <i>value</i>. Pertama, <i>generateVariants(List<int[]> piece)</i> dipanggil. Lalu <i>piece</i> satu per satu ditinjau apakah <i>placeable()</i>. Peninjauan selalu dimulai dari bagian atas dan kiri <i>board</i>. Jika <i>placeable()</i> mengembalikan <i>true</i>, <i>placePiece()</i> akan dipanggil. Kemudian rekursi dilakukan dengan <i>piece</i> yang tersisa. <i>Piece</i> selanjutnya akan diletakkan terus menerus hingga tidak bisa lagi dan <i>backtracking</i> harus dimulai dengan <i>removePiece()</i>. Mengembalikan nilai <i>true</i> jika permainan dapat diselesaikan, dan nilai <i>false</i> jika tidak dapat diselesaikan.</p>
<pre>1 public static void printBoard() {</pre>	<p>Mencetak <i>board</i> yang sudah selesai ke terminal dan memberikan warna pada abjad dengan memanggil fungsi <i>colorANSI()</i>.</p>

C. Source Code Program

1. File Main.java

2. File FileHandler.java

```
import java.awt.Color;
import java.awt.Graphics2D;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.io.IOException;
import java.util.*;
import javax.imageio.ImageIO;

public class FileHandler {
    public static int N;
    public static int M;
    public static int P;
    public static HashMap<String, List<int[]>> Pieces;
    public static double err;

    public static void ReadFile(String path) {
        err = 0;
        Pieces = new HashMap<>();
        try {
            File file = new File(path);
            Scanner scanner = new Scanner(file);

            // Read Dimensions
            String dimensions = scanner.nextLine();
            String[] dimension = dimensions.split(" ");

            // Check N, M, and P
            if (dimension.length == 3) {
                N = Integer.parseInt(dimension[0]);
                M = Integer.parseInt(dimension[1]);
                P = Integer.parseInt(dimension[2]);

                // Read Type
                String Type = scanner.nextLine();

                // Check Type and Read Pieces
                if (Type.equals("DEFAULT")) {
                    String prevPiece = "zzz";
                    int row = 0;
                    int col = 0;

                    while (scanner.hasNextLine()) {
                        String line = scanner.nextLine();
                        row = 0;
                        for (int i=0; i<line.length(); i++) {
                            if (line.charAt(i) >= 'A' && line.charAt(i) <= 'Z') {
                                String currPiece = String.valueOf(line.charAt(i));
                                if (!currPiece.equals(prevPiece)) {
                                    prevPiece = currPiece;
                                    col = 0;
                                }
                                Pieces.putIfAbsent(currPiece, new ArrayList<>());
                                Pieces.get(currPiece).add(new int[]{col, row});
                            }
                            row++;
                        }
                        col++;
                    }
                }
            }
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

```

        // Check Jumlah Piece
        if (Pieces.size() != P) {
            System.out.println("\nInvalid Number of Pieces.");
            err = -405;
        }
    } else {
        System.out.println("\nInvalid Type.");
        err = -405;
    }
}
} else {
    System.out.println("\nInvalid Dimensions.");
    err = -405;
}
}

catch (FileNotFoundException e) {
    System.out.println("\nFile not found.");
    err = -405;
}
}

public static void WriteFile(String fileName) {
try {
    int rows = Algorithm.Board.length;
    int cols = Algorithm.Board[0].length;
    FileWriter writer = new FileWriter("../solution/" + fileName);

    for (int c = 0; c < cols; c++) {
        for (int r = 0; r < rows; r++) {
            writer.write(Algorithm.Board[r][c]);
        }
        writer.write("\n");
    }
    writer.close();

    System.out.println("\nSuccessfully Wrote To File.");
} catch (IOException e) {
    System.out.println("\nFailed To Write To File.");
}
}

public static Map<Character, Color> colorImage = new HashMap<>();

public static void initColorMap() {
    colorImage.put('A', new Color(255, 0, 0));      // Merah
    colorImage.put('B', new Color(0, 255, 0));      // Hijau
    colorImage.put('C', new Color(0, 0, 255));      // Biru
    colorImage.put('D', new Color(255, 255, 0));    // Kuning
    colorImage.put('E', new Color(255, 0, 255));    // Magenta
    colorImage.put('F', new Color(0, 255, 255));    // Cyan
    colorImage.put('G', new Color(255, 165, 0));    // Oranye
    colorImage.put('H', new Color(128, 0, 128));    // Ungu
    colorImage.put('I', new Color(255, 105, 180)); // Pink
    colorImage.put('J', new Color(50, 205, 50));    // Lime
    colorImage.put('K', new Color(64, 224, 208));   // Turquoise
    colorImage.put('L', new Color(255, 215, 0));    // Emas
    colorImage.put('M', new Color(139, 69, 19));    // Coklat
    colorImage.put('N', new Color(107, 142, 35));   // Olive
    colorImage.put('O', new Color(230, 230, 250)); // Lavender
    colorImage.put('P', new Color(0, 128, 128));    // Teal
}

```

```

        colorImage.put('Q', new Color(211, 211, 211)); // Abu-abu terang
        colorImage.put('R', new Color(169, 169, 169)); // Abu-abu gelap
        colorImage.put('S', new Color(128, 0, 0)); // Merah marun
        colorImage.put('T', new Color(0, 0, 139)); // Biru laut
        colorImage.put('U', new Color(0, 100, 0)); // Hijau gelap
        colorImage.put('V', new Color(255, 20, 147)); // Rose
        colorImage.put('W', new Color(192, 192, 192)); // Perak
        colorImage.put('X', new Color(30, 144, 255)); // Neon biru
        colorImage.put('Y', new Color(57, 255, 20)); // Neon hijau
        colorImage.put('Z', new Color(255, 20, 147)); // Neon pink
    }

    public static void WriteImage(String fileName) {
        initColorMap();

        int piece_size = 60;
        int margin = 0;

        int rows = Algorithm.Board.length;
        int cols = Algorithm.Board[0].length;
        int width = rows * (piece_size + margin);
        int height = cols * (piece_size + margin);

        BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
        Graphics2D g2d = image.createGraphics();

        g2d.setColor(Color.BLACK);
        g2d.fillRect(0, 0, width, height);

        // for (int r = 0; r < rows; r++) {
        //     for (int c = 0; c < cols; c++) {
        //         char piece = Algorithm.Board[r][c];
        //         g2d.setColor(colorImage.getOrDefault(piece, Color.WHITE));
        //         int x = c * (piece_size + margin);
        //         int y = r * (piece_size + margin);
        //         g2d.fillOval(x, y, piece_size, piece_size);
        //     }
        // }

        // Print (kebalik)
        for (int c = 0; c < cols; c++) {
            for (int r = 0; r < rows; r++) {
                char piece = Algorithm.Board[r][c];
                g2d.setColor(colorImage.getOrDefault(piece, Color.WHITE));
                int x = c * (piece_size + margin);
                int y = r * (piece_size + margin);
                g2d.fillOval(y, x, piece_size, piece_size);
            }
        }

        g2d.dispose();
        try {
            File output = new File("../solution/" + fileName);
            ImageIO.write(image, "png", output);
            System.out.println("\nSuccessfully Wrote To Image.");
        } catch (IOException e) {
            System.out.println("\nFailed To Write To Image.");
        }
    }
}

```

3. File Algorithm.java

```
import java.util.*;

public class Algorithm {
    public static char[][] Board = new char[FileHandler.M][FileHandler.N];
    public static long counter = 0; // Backtrack count
    public static HashMap<Character, String> colorMap = new HashMap<>();
    public static String ANSI_RESET = "\u001B[0m";

    public static void colorANSI() {
        colorMap.put('A', "\u033[38;5;196m"); // Merah
        colorMap.put('B', "\u033[38;5;46m"); // Hijau
        colorMap.put('C', "\u033[38;5;21m"); // Biru
        colorMap.put('D', "\u033[38;5;226m"); // Kuning
        colorMap.put('E', "\u033[38;5;201m"); // Magenta
        colorMap.put('F', "\u033[38;5;51m"); // Cyan
        colorMap.put('G', "\u033[38;5;208m"); // Oranye
        colorMap.put('H', "\u033[38;5;93m"); // Ungu
        colorMap.put('I', "\u033[38;5;205m"); // Pink
        colorMap.put('J', "\u033[38;5;118m"); // Lime
        colorMap.put('K', "\u033[38;5;80m"); // Turquoise
        colorMap.put('L', "\u033[38;5;214m"); // Emas
        colorMap.put('M', "\u033[38;5;130m"); // Coklat
        colorMap.put('N', "\u033[38;5;100m"); // Olive
        colorMap.put('O', "\u033[38;5;147m"); // Lavender
        colorMap.put('P', "\u033[38;5;30m"); // Teal
        colorMap.put('Q', "\u033[38;5;250m"); // Abu-abu terang
        colorMap.put('R', "\u033[38;5;240m"); // Abu-abu gelap
        colorMap.put('S', "\u033[38;5;88m"); // Merah marun
        colorMap.put('T', "\u033[38;5;25m"); // Biru laut
        colorMap.put('U', "\u033[38;5;22m"); // Hijau gelap
        colorMap.put('V', "\u033[38;5;169m"); // Rose
        colorMap.put('W', "\u033[38;5;7m"); // Perak
        colorMap.put('X', "\u033[38;5;45m"); // Neon biru
        colorMap.put('Y', "\u033[38;5;82m"); // Neon hijau
        colorMap.put('Z', "\u033[38;5;198m"); // Neon pink
    }

    public static void initializeBoard() {
        for (int i = 0; i < FileHandler.M; i++) {
            Arrays.fill(Board[i], '.');
        }
    }

    // Rotate 90°
    public static List<int[]> rotate(List<int[]> piece) {
        List<int[]> rotated = new ArrayList<>();
        for (int[] p : piece) {
            rotated.add(new int[]{p[1], -p[0]});
        }
        return rotated;
    }

    // Refleksi terhadap sumbu y
    public static List<int[]> reflect(List<int[]> piece) {
        List<int[]> reflected = new ArrayList<>();
        for (int[] p : piece) {
            reflected.add(new int[]{-p[0], p[1]}));
        }
        return reflected;
    }
}
```

```

// Generate semua kombinasi piece di HashSet (agar tidak ada duplikat)
public static List<List<int[]>> generateVariants(List<int[]> piece) {
    Set<List<int[]>> variants = new HashSet<>();
    List<int[]> currentPiece = piece;
    for (int i = 0; i < 4; i++) { // 4 kali rotasi (0°, 90°, 180°, 270°)
        currentPiece = rotate(currentPiece);
        variants.add(new ArrayList<>(currentPiece));
        variants.add(reflect(currentPiece));
    }
    return new ArrayList<>(variants);
}

// Check piece di board
public static boolean placeable(List<int[]> piece, int r, int c) {
    for (int[] p : piece) {
        int nr = r + p[0], nc = c + p[1];
        if (nr < 0 || nr >= FileHandler.M || nc < 0 || nc >= FileHandler.N || Board[nr][nc] != '.') {
            // Syarat: tidak boleh keluar dimensi board, dan tidak boleh menimpa piece lain
            return false;
        }
    }
    return true;
}

// Place per koordinat
public static void placePiece(List<int[]> piece, int r, int c, char label) {
    for (int[] p : piece) {
        Board[r + p[0]][c + p[1]] = label;
    }
}

// Remove per koordinat
public static void removePiece(List<int[]> piece, int r, int c) {
    for (int[] p : piece) {
        Board[r + p[0]][c + p[1]] = '.';
    }
}

public static boolean solve(List<Map.Entry<String, List<int[]>>> remainingPieces) {
    if (remainingPieces.isEmpty()) return true; // Basis

    Map.Entry<String, List<int[]>> entry = remainingPieces.get(0);
    String label = entry.getKey();
    List<List<int[]>> pieceVariants = generateVariants(entry.getValue());

    for (List<int[]> piece : pieceVariants) {
        for (int r = 0; r < FileHandler.M; r++) {
            for (int c = 0; c < FileHandler.N; c++) {
                if (placeable(piece, r, c)) {
                    placePiece(piece, r, c, label.charAt(0));

                    // Rekursi
                    List<Map.Entry<String, List<int[]>>> nextPieces = new ArrayList<>(remainingPieces.subList(1, remainingPieces.size()));
                    if (solve(nextPieces)) {
                        return true;
                    }
                }

                removePiece(piece, r, c);
            }
        }
    }
    counter++;
}

return false;
}

// Print board (kebalik)
public static void printBoard() {
    int rows = Board.length;
    int cols = Board[0].length;

    colorANSI();

    for (int c = 0; c < cols; c++) {
        for (int r = 0; r < rows; r++) {
            System.out.print(colorMap.get(Board[r][c]) + Board[r][c] + ANSI_RESET);
        }
        System.out.println();
    }
}
}

```

D. Kakas (*Package*)

1. Java.util (ArrayList, Scanner)

Kakas java.util menyediakan berbagai kelas yang menangani struktur data, input pengguna, dan fungsi utilitas lainnya.

2. Java.io (File, FileWriter, IOException, FileNotFoundException)

Kakas `java.io` menyediakan kelas dan metode untuk melakukan operasi input dan output (I/O) seperti pembacaan dan pembuatan *file*. Selain itu, kakas ini juga memiliki penanganan error dalam I/O.

3. Java.awt (Color, Graphics2D, image BufferedImage)

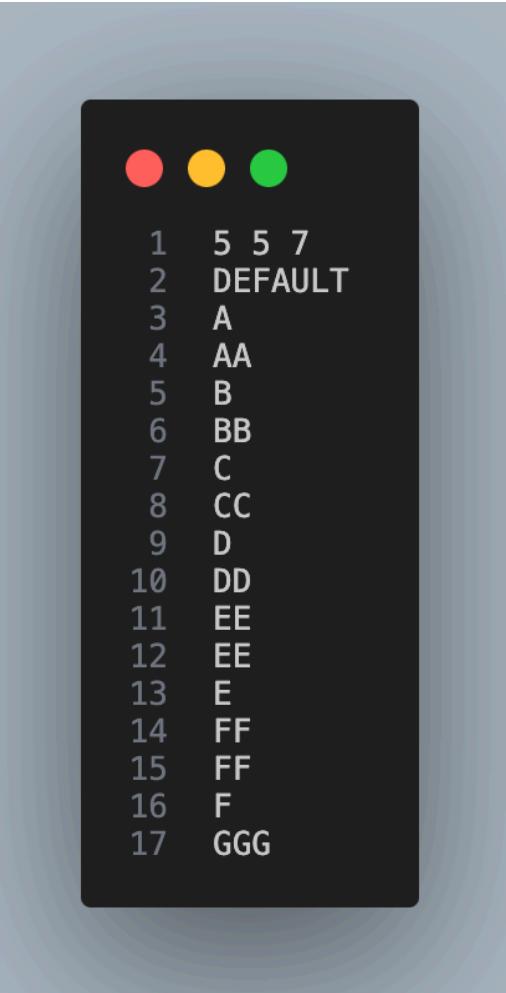
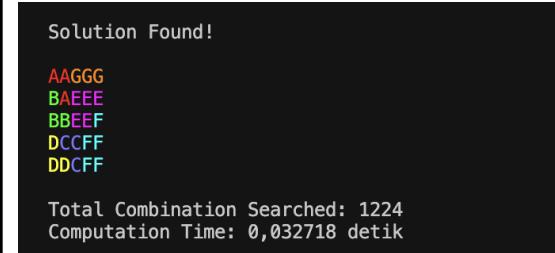
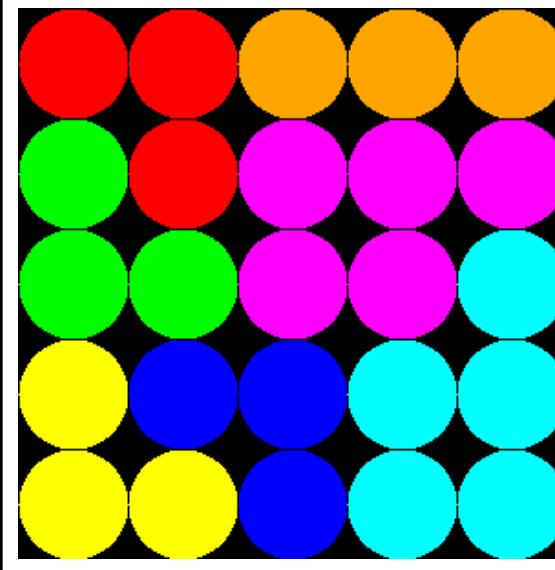
Kakas `java.awt` (Abstract Window Toolkit) berfungsi untuk membuat GUI (Graphical User Interface) program. Contohnya adalah pemberian warna dan pembuatan grafis atau komponen.

4. Javax.imageio

Kakas `javax.imageio` adalah kakas API (Application Programming Interface) yang digunakan untuk membaca dan menulis (I/O) *file* gambar dalam format seperti PNG, JPEG, BMP, dan GIF.

BAB 4 ANALISIS DAN PENGUJIAN

A. Kasus Uji 1 (Spesifikasi)

Isi File (1.txt)	Solusi (Terminal, File Txt, dan File Png)
 <pre>1 5 5 7 2 DEFAULT 3 A 4 AA 5 B 6 BB 7 C 8 CC 9 D 10 DD 11 EE 12 EE 13 E 14 FF 15 FF 16 F 17 GGG</pre>	 <p>Solution Found!</p> <pre>AAGGG BAEEE BBEEF DCCFF DDCFF</pre> <p>Total Combination Searched: 1224 Computation Time: 0,032718 detik</p>  

Analisa

Program menulis bahwa papan berukuran 5x5 dan terdapat 7 buah *piece* (A, B, C, D, E, F, dan G). Terdapat solusi pada kasus uji ini. Dengan total kombinasi yang dicoba (*backtracking*) sejumlah 1224 kombinasi selama 0,03 detik.

B. Kasus Uji 2

Isi File (2.txt)

Solusi (Terminal, File Txt, dan File Png)

● ● ●

```

1 10 12 8
2 DEFAULT
3 VV
4 VV
5 VV
6 II
7 II
8 II
9 NNN
10 NNN
11 G
12 G
13 G
14 G
15 G
16 HHH
17 H
18 M
19 M
20 BBB
21 BBBB
22 BBBBB
23 AAAAAAAA
24 AAAAAAAA
25 AAAA
26 AAAAAA
27 AAAAAA
28 AAAAAAAA
29 AAAAAAAA
30 AAAAAAAA
31 AAA AAA
32 AAA AAA

```

Solution Found!

```

VVAAANNNAAAG
VVAAANNNAAAG
VVAAAAAAAAG
AAAAAAAAG
AAAAAAAAG
AAAAABBBHHH
AAAAABBBBBH
IIAAAABBBBBB
IIAAAAAAAAM
IIAAAAAAAAM

```

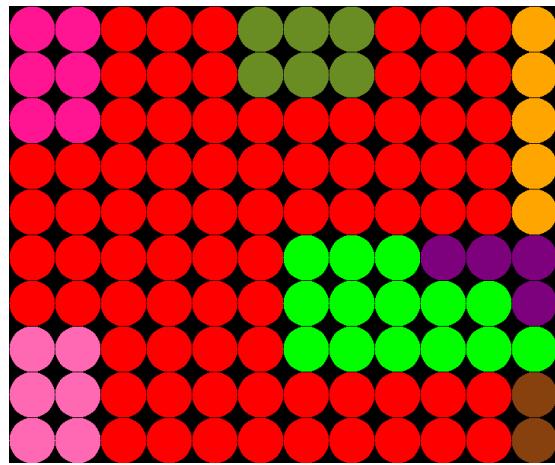
Total Combination Searched: 598
Computation Time: 0,031592 detik

● ● ●

```

1 VVAAANNNAAAG
2 VVAAANNNAAAG
3 VVAAAAAAAAG
4 AAAAAAAAAG
5 AAAAAAAAAG
6 AAAAAABBBHHH
7 AAAAAABBBBBH
8 IIAAAABBBBBB
9 IIAAAAAAAAM
10 IIAAAAAAAAM
11

```



Analisa

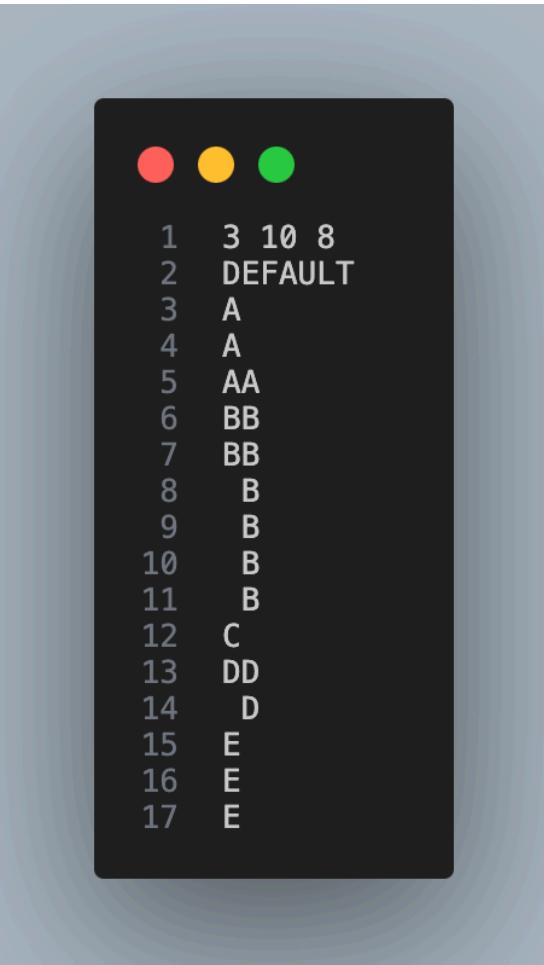
Program menulis bahwa papan berukuran 10x12 dan terdapat 8 buah *piece* (V, I, N, G, H, M, B, dan A). Terdapat solusi pada kasus uji ini. Dengan total kombinasi yang dicoba (*backtracking*) sejumlah 598 kombinasi selama 0,03 detik.

C. Kasus Uji 3

Isi File (3.txt)	Solusi (Terminal, File Txt, and File Png)
<pre> ● ○ ● 1 3 10 8 2 DEFAULT 3 A 4 A 5 AA 6 BB 7 BB 8 B 9 B 10 B 11 B 12 C 13 DD 14 D 15 E 16 E 17 E 18 FF 19 GGG 20 G G 21 GGG 22 H </pre>	<pre> Solution Found! ABBBBBBGGG ABDEEEGHG AACDDFFGGG Total Combination Searched: 27556861 Computation Time: 162,004086 detik </pre> <div style="text-align: center;"> </div>
Analisa	

Program menulis bahwa papan berukuran 3x10 dan terdapat 8 buah *piece* (A, B, C, D, E, F, G, dan H). Terdapat solusi pada kasus uji ini. Dengan total kombinasi yang dicoba (*backtracking*) sejumlah 27556861 kombinasi selama 162 detik.

D. Kasus Uji 4

Isi File (1.txt)	Solusi (Terminal, File Txt, dan File Png)
	Invalid Number of Pieces.
Analisa	
Program menulis bahwa papan berukuran 3x10 dan terdapat 8 buah <i>piece</i> . Namun, kasus uji ini hanya memuat 5 buah <i>piece</i> . Program membacanya sebagai tidak valid, dan mengeluarkan pemberitahuan bahwa jumlah <i>piece</i> tidak sesuai.	

E. Kasus Uji 5

Isi File (1.txt)	Solusi (Terminal, File Txt, dan File Png)
 A screenshot of a terminal window. Inside, there is a 2x2 grid representing a board. The top-left square contains a red circle. The top-right square contains a yellow circle. The bottom-left square contains a green circle. The bottom-right square is empty. Below the grid, there is text: 1 2 2 2 2 DEFAULT 3 A 4 A 5 AA 6 BB	No Solution Found.
Analisa	
<p>Program menulis bahwa papan berukuran 2x2 dan terdapat 2 buah <i>piece</i> (A dan B). Walaupun jumlah <i>piece</i> sudah sesuai, kedua <i>piece</i> yang diberikan tidak valid karena melebihi dimensi papan. Program tidak mendapat solusi dan mengeluarkan pemberitahuan.</p>	

F. Kasus Uji 6

Isi File (1.txt)	Solusi (Terminal, File Txt, dan File Png)
------------------	---

```

1 3 5 6
2 DEFAULT
3 ZZ
4 Z
5 Z
6 F
7 FF
8 I
9 L
10 LL
11 L
12 A
13 A
14 B

```

Solution Found!

AIZZZ
AFLLZ
BFFLL

Total Combination Searched: 436
Computation Time: 0,021434 detik

```

1 AIZZZ
2 AFLLZ
3 BFFLL
4

```

Analisa

Program menulis bahwa papan berukuran 3x5 dan terdapat 6 buah *piece* (Z, F, I, L, A, dan B). Terdapat solusi pada kasus uji ini. Dengan total kombinasi yang dicoba (*backtracking*) sejumlah 436 kombinasi selama 0,02 detik.

G. Kasus Uji 7

Isi File (1.txt)	Solusi (Terminal, File Txt, dan File Png)
------------------	---

● ○ ●

1	5	11	12
2	DEFAULT		
3	A		
4	A		
5	AA		
6	A		
7	B	B	
8	BBB		
9	CCCC		
10	C		
11	D		
12	DD		
13	DD		
14	E		
15	EEE		
16	E		
17	F		
18	FFFF		
19	GGG		
20	G		
21	HHH		
22	HH		
23	I		
24	II		
25	I		
26	JJ		
27	J		
28	J		
29	KK		
30	K		
31	L		
32	L		
33	LLL		

Solution Found!

AKBLLLEHHC
AKKBLEEEHHC
AABBLGEDDHHC
JAIIGGDDFCC
JJJIIGDFFFF

Total Combination Searched: 18171487
Computation Time: 92,934045 detik

● ○ ●

1	AKBLLLEHHC		
2	AKKBLEEEHHC		
3	AABBLGEDDHHC		
4	JAIIGGDDFCC		
5	JJJIIGDFFFF		
6			

The grid consists of 36 colored circles arranged in a 6x6 pattern. The colors are: Row 1: Red, Cyan, Green, Yellow, Magenta, Blue. Row 2: Red, Cyan, Green, Yellow, Magenta, Blue. Row 3: Red, Green, Yellow, Orange, Magenta, Blue. Row 4: Green, Red, Magenta, Orange, Yellow, Cyan. Row 5: Green, Magenta, Yellow, Orange, Cyan, Blue. Row 6: Magenta, Yellow, Orange, Cyan, Blue, Cyan.

Analisa

Program menulis bahwa papan berukuran 5x11 dan terdapat 12 buah *piece* (A, B, C, D, E, F, G, H, I, J, K, dan L). Terdapat solusi pada kasus uji ini. Dengan total kombinasi yang dicoba (*backtracking*) sejumlah 18171487 kombinasi selama 92,93 detik.

BAB 5 KESIMPULAN

Berdasarkan analisis yang telah dilakukan terhadap implementasi Algoritma Brute Force pada penyelesaian permainan IQ Puzzler Pro, dapat disimpulkan bahwa algoritma ini berhasil menyelesaikan permainan dengan optimal, seperti yang telah diharapkan. Hasil permainan dalam sebuah kasus dapat beragam. Waktu *backtracking* yang dilakukan juga menunjukkan performa yang baik untuk papan yang berdimensi lebih kecil, dan membutuhkan waktu yang sedikit lebih lama untuk papan yang berdimensi besar dengan *piece* yang cenderung banyak. Hal ini membuktikan bahwa Algoritma Brute Force tidaklah efisien untuk persoalan yang kompleks dengan data yang besar.

Meskipun Algoritma Brute Force tidak selalu efisien, algoritma ini tetap menjadi pilihan yang valid untuk masalah-masalah yang lebih sederhana dan membutuhkan solusi yang cepat. Maka dari itu, pemahaman tentang kelebihan dan kekurangan Algoritma Brute Force membantu dalam pemilihan penyelesaian masalah pemrograman.

LAMPIRAN

A. Github

https://github.com/sammmine/Tucil1_13523138

B. Tabel Pemeriksaan

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

DAFTAR PUSTAKA

Levitin, Anany. 2003. Introduction to The Design and Analysis of Algorithms. https://homel.vsb.cz/~fai0013/Kniha_Algoritmy.pdf. Diakses pada 17 Februari 2025.

Munir, Rinaldi. 2025. Algoritma Brute Force (Bagian 1). [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/2024-2025/02-Algoritma-Brute-Force-(2025)-Bag1.pdf). Diakses pada 17 Februari 2025.

Smart Games. 2021. IQ Puzzler Pro. <https://www.smartgamesusa.com/one-player-games/iq-puzzler-pro>. Diakses pada 17 Februari 2025.