IF2211 Strategi Algoritma

Penyelesaian IQ Puzzler Pro dengan Algoritma Brute Force

Laporan Tugas Kecil

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

Oleh :

Angelina Efrina Prahastaputri 13523060

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

JL. GANESA 10, BANDUNG 40132

2025

DAFTAR ISI

BAB I

DESKRIPSI MASALAH

1.1 Algoritma Brute Force

Algoritma *brute force* adalah algoritma yang menggunakan pendekatan yang lurus atau lempang (*straightforward*) untuk memecahkan suatu persoalan. Algoritma *brute force* memecahkan suatu persoalan dengan strategi pencarian yang sangat sederhana, langsung, mudah dipahami, dan komprehensif untuk mengeksplorasi setiap opsi, kasus, atau kemungkinan hingga suatu jawaban atau solusi dari persoalan tersebut ditemukan.

Algoritma *brute force* pada umumnya bukan merupakan algoritma yang mangkus. Hal ini karena algoritma *brute force* memiliki kompleksitas temporal yang tinggi, sehingga kurang cocok dan tidak efisien untuk persoalan dengan skala besar. Namun, bila ruang lingkup persoalan termasuk kecil dan setiap kemungkinan mudah dieksplorasi, algoritma *brute force* dapat menjadi algoritma yang paling tepat dan efisien. Algoritma *brute force* tidak menggunakan pendekatan optimasi atau heuristik. Algoritma ini bergantung pada pengujian setiap kemungkinan tanpa menggunakan pemangkasan atau heuristik yang cerdas.

1.2 Permainan IQ Puzzler Pro

Permainan IQ Puzzler Pro
Sumber: https://www.smartgames.eu/uk/one-player-games/iq-puzzler-pro-0

IQ Puzzler Pro adalah permainan papan yang diproduksi oleh perusahaan Smart Games. Dalam permainan ini, pemain harus dapat mengisi seluruh papan dengan *piece* (blok puzzle) yang telah tersedia. Terdapat dua komponen utama dari permainan ini. Pertama, *board* (papan) yang akan menjadi tempat untuk pemain mengisi blok-blok puzzle. Kedua, blok-blok puzzle yang akan diletakkan pemain pada papan.

Permainan dimulai dengan papan yang kosong. Pemain dapat meletakkan blok-blok puzzle sedemikian sehingga tidak ada *piece* yang bertumpang tindih kecuali pada kasus permainan 3D. Setiap blok-blok puzzle dapat dirotasikan atau dicerminkan. Puzzle dinyatakan selesai jika dan hanya jika papan dan terisi penuh dan seluruh blok puzzle berhasil diletakkan. Pada tugas kecil ini, dicari sebuah solusi atau penyelesaian atas permainan ini menggunakan algoritma *brute force*.

BAB II

PENYELESAIAN

2.1. Langkah Penyelesaian IQ Puzzler Pro dengan Pendekatan Algoritma Brute Force

Berikut adalah langkah-langkah penyelesaian permainan IQ Puzzler Pro dengan pendekatan algoritma *brute force*:

1. Pengguna diminta untuk memasukkan nama file berekstensi .txt yang harus berisi komponen-komponen dari permainan IQ Puzzler Pro yakni dimensi papan, banyak blok puzzle yang unik, jenis kasus (pada tugas kecil ini hanya diimplementasikan satu jenis kasus yakni DEFAULT), dan bentuk blok puzzle yang dilambangkan oleh konfigurasi *character* berupa huruf A-Z dalam kapital.

2. Program akan membaca file yang dimasukkan pengguna dan menyimpan data yang ada pada file. Program menyimpan data sebagai sebuah objek bernama "Puzzle". Data dimensi papan digunakan untuk inisialisasi papan kosong untuk awal permainan dalam bentuk matriks. Data banyak puzzle yang unik dan blok-blok puzzle akan divalidasi terlebih dahulu untuk memastikan bahwa memang sesuai. Blok-blok puzzle disimpan dalam bentuk matriks dan dimasukkan ke dalam sebuah list.

3. Pendekatan *brute force* dilakukan dengan memanfaatkan *backtracking* untuk mencari seluruh kemungkinan atau kasus penempatan blok-blok puzzle pada papan yang dapat terbentuk. Program akan meninjau setiap kasus dengan menempatkan blok-blok puzzle yang ada secara terurut pada papan, sesuai file masukan pengguna. Pada setiap kasus (status papan), program akan mengecek apakah seluruh blok puzzle sudah diletakkan pada papan dan papan terisi penuh (tidak ada ruang kosong).

4. Implementasi pendekatan *brute force* menggunakan struktur data Stack untuk mencari seluruh kemungkinan peletakkan blok-blok puzzle pada papan. Stack digunakan untuk menyimpan status papan yang berupa objek bernama "PapanPuzzle". Untuk mencari seluruh kemungkinan peletakkan blok-blok puzzle, program akan pertama-tama mencoba meletakkan blok puzzle pertama dengan orientasi semula pada setiap titik pada papan. Program kemudian akan merotasi blok puzzle untuk mencoba orientasi lain karena ada kemungkinan blok puzzle tersebut tidak dapat diletakkan. Kemudian, program akan melakukan hal yang sama untuk blok puzzle yang sudah dicerminkan terlebih dahulu.

5. Jika untuk blok puzzle selanjutnya tidak dapat diletakkan pada papan sedangkan

masih ada ruang kosong pada papan, program akan terus melakukan *backtracking* ke status papan sebelumnya dan mencoba ulang peletakkan blok-blok puzzle sebelumnya sedemikian sehingga terdapat ruang kosong untuk meletakkan blok puzzle selanjutnya tersebut pada papan.

6. Jika seluruh blok puzzle sudah diletakkan tetapi masih terdapat ruang kosong pada papan, program akan melakukan *backtracking* kembali seperti pada langkah 5. Jika seluruh blok puzzle sudah diletakkan dan papan terisi penuh, program akan mengembalikan objek bernama "Solusi" yang terdiri atas informasi apakah solusi berhasil ditemukan, hasil akhir papan yang sudah terisi oleh blok-blok puzzle, dan banyak kasus yang ditinjau. Solusi tidak ada jika seluruh kasus telah ditinjau dan tidak dapat ditemukan hasil..

7. Setelah mendapatkan solusi, program akan mengeluarkan *prompt* untuk menyimpan solusi dengan menampilkan hasil akhir papan dengan blok-blok puzzle yang telah diwarnai dan masing-masing memiliki warna yang berbeda. Program kemudian menyimpan solusi dalam file berekstensi .txt. Program juga dapat menyimpan solusi sebagai gambar (bonus).

Dengan demikian, langkah-langkah penyelesaian tersebut membantu penulis dalam menyelesaikan permainan IQ Puzzler Pro dengan menggunakan pendekatan algoritma *brute force*.

BAB III

IMPLEMENTASI

3.1. Spesifikasi Teknis Program

3.1.1. Struktur *Repository*

```
|____ Tucil1_13523060
   |____ bin
      |____ .gitkeep
      |____ InputOutputFile.class
      |____ PapanPuzzle.class
      |____ Puzzle.class
      |____ PuzzleSolver.class
      |____ Solusi.class
   |____ doc
      |____ .gitkeep
      |____ Tucil1_13523060.pdf
   |____ src
      |____ .gitkeep
      |____ InputOutputFile.java
      |____ PapanPuzzle.java
      |____ Puzzle.java
      |____ PuzzleSolver.java
      |____ Solusi.java
   |____ test
      |____ solution
         |____ pic
            |____ solusi1.png
            |____ solusi2.png
            |____ solusi3.png
            |____ solusi4.png
            |____ solusi5.png
            |____ solusi6.png
            |____ solusi7.png
         |____ solusi1.txt
```

```
        |____ solusi2.txt
        |____ solusi3.txt
        |____ solusi4.txt
        |____ solusi5.txt
        |____ solusi6.txt
        |____ solusi7.txt
    |____ testcase1.txt
    |____ testcase2.txt
    |____ testcase3.txt
    |____ testcase4.txt
    |____ testcase5.txt
    |____ testcase6.txt
    |____ testcase7.txt
```

3.1.2. Fungsi dan Prosedur

Fungsi / Prosedur	Tujuan
public class InputOutputFile	
public static String inputFile	Meminta input file .txt
public static Puzzle bacaFilePuzzle	Membaca masukan file .txt dan menyimpannya dalam objek Puzzle
public static char[][] convertBlockToMatrix	Mengubah block puzzle menjadi matriks untuk disimpan dalam sebuah list
public static void outputFile	Menulis output file berupa .txt
public static String getWarna	Mendapatkan warna block puzzle berdasarkan huruf block puzzle
public static void generateGambarSolusi	Menghasilkan gambar dari solusi dan menyimpannya
public class PuzzleSolver	
public static boolean canBlockFit	Mengecek apakah block puzzle dapat ditaruh pada papan
public static char[][] rotateBlock	Merotasi block puzzle sebanyak 90 derajat searah jarum jam

public static char[][] mirrorBlock	Mencerminkan block puzzle
public static char[][] placeBlock	Menaruh block puzzle pada papan
public static char[][] removeBlock	Menghapus block puzzle dari papan
public static boolean isAllBlockPlaced	Mengecek apakah semua block puzzle sudah ditaruh pada papan
public static void printPapan	Menampilkan papan
public static boolean isPapanPenuh	Mengecek apakah papan sudah terisi penuh dan tidak ada ruang kosong
public static Solusi solvePuzzleByBruteForce	Mencari solusi dari puzzle dengan algoritma *brute force*
public class Solusi	
public static String getWarna	Mendapatkan warna block puzzle berdasarkan huruf block puzzle
public static void printSolusiBerwarna	Menampilkan solusi berupa papan dengan block puzzle berwarna

3.1.3. *Source Code*

1. InputOutputFile.java

```java
import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.*;
import java.util.*;
import java.util.List;
import javax.imageio.ImageIO;

public class InputOutputFile {

    // Meminta input file .txt
    public static String inputFile(){
        System.out.println("========================================");
        System.out.println("SELAMAT DATANG DI IQ PUZZLER PRO SOLVER!");
        System.out.println("========================================");
        System.out.println();

        Scanner scanner = new Scanner(System.in);
        String filename;
        File file;

        while (true) {
            System.out.println("Masukkan nama file .txt (pastikan file ada pada folder test): ");
            filename = scanner.nextLine().trim();
            if (filename.isEmpty()) {
                System.out.println("Nama file tidak boleh kosong! Coba lagi.");
                continue;
            }
            file = new File("test/" + filename + ".txt");
            if (!file.exists()) {
                System.out.println("File tidak ditemukan pada folder test! Coba lagi.");
                continue;
            } else {
                break;
            }
        }
```

```java
        return ("test/" + filename + ".txt");

    }

    // Membaca masukan file .txt
    public static Puzzle bacaFilePuzzle(String file_name) {
        int N = 0, M = 0, P = 0;
        String S = "";
        HashSet<Character> huruf_blocks = new HashSet<>();
        List<char[][]> puzzle_blocks = new ArrayList<>();


        try {
            BufferedReader reader = new BufferedReader(new FileReader(file_name));
            String line = reader.readLine();
            if (line == null || line.trim().isEmpty()) {
                throw new IOException(message:"File kosong! Coba lagi.");
            }
            // Membaca baris pertama
            String[] first_line = line.trim().split(regex:" ");
            if (first_line.length != 3) {
                throw new IOException(message:"Komponen Puzzle belum sesuai! Coba lagi.");
            }
            N = Integer.parseInt(first_line[0]);
            M = Integer.parseInt(first_line[1]);
            P = Integer.parseInt(first_line[2]);
            // Membaca baris kedua
            line = reader.readLine();
            if (line == null || line.trim().isEmpty()) {
                throw new IOException(message:"Tipe Puzzle belum ada! Coba lagi.");
            }
            S = line.trim();
            if (!(S.equals(anObject:"DEFAULT") || S.equals(anObject:"CUSTOM") || S.equals(anObject:"PYRAMID"))) {
                throw new IOException(message:"Tipe Puzzle belum sesuai! Coba lagi.");
            }
            // Membaca baris-baris selanjutnya/membaca block puzzle dan memvalidasi banyak block puzzle
            List<String> block = new ArrayList<>();
            char current_block = ' ';

            while ((line = reader.readLine()) != null) {
                if (!line.isEmpty()) {
                    char jenis_block = line.trim().charAt(index:0);
                    huruf_blocks.add(jenis_block);
                    if (block.isEmpty() || jenis_block == current_block) {
                        block.add(line);
                        current_block = jenis_block;
                    } else {
                        puzzle_blocks.add(convertBlockToMatrix(block));
                        block.clear();
                        block.add(line);
                        current_block = jenis_block;
                    }
                }
            }
            if (!block.isEmpty()) {
                puzzle_blocks.add(convertBlockToMatrix(block));
            }

            // System.out.println("Huruf Unik: " + huruf_blocks.size());
            // reader.close();

        } catch (Exception e) {
            System.out.println(e.getMessage());
        }

        return new Puzzle(N, M, P, S, puzzle_blocks, huruf_blocks);
    }
```

```java

    // Mengubah block puzzle menjadi matriks untuk disimpan dalam sebuah list
    public static char[][] convertBlockToMatrix(List<String> block) {
        int baris = block.size();
        int kolom = 0;
        for (String line : block) {
            kolom = Math.max(kolom, line.length());
        }
        char[][] matriks = new char[baris][kolom];
        for (int i = 0; i < baris; i++) {
            String line = block.get(i);
            for (int j = 0; j < kolom; j++) {
                matriks[i][j] = (j < line.length()) ? line.charAt(j) : ' ';
            }
        }
        return matriks;
    }
```

```java

    // Menulis output file berupa .txt
    public static void outputFile(char[][] papan, String filenames){
        try (BufferedWriter writer = new BufferedWriter(new FileWriter(filenames))) {
            for (char[] baris : papan) {
                writer.write(baris);
                writer.newLine();
            }
        } catch (IOException e){
            System.out.println(x:"error!");
        }
    }

    public static final String[] WARNA_HEX = {
        "#FFFFFF", // Putih.
        "#A6A6A6", // Abu-Abu.
        "#FF3131", // Merah.
        "#FF914C", // Oranye.
        "#FFDE59", // Kuning.
        "#7ED956", // Hijau Muda.
        "#00BF62", // Hijau.
        "#0CC1E0", // Biru Muda.
        "#004AAD", // Biru Tua.
        "#FF65C3", // Pink.
        "#8C52FF", // Ungu.
        "#FF5757", // Terracotta.
        "#F0FFA2", // Kuning Pucat.
        "#FEBD59", // Kuning Tua.
        "#944912", // Coklat.
        "#C0FF72", // Lime.
        "#00892A", // Hijau Tua.
        "#5CE1E6", // Cyan.
        "#0097B2", // Turqoise.
        "#38B6FF", // Biru Langit.
        "#5271FF", // Indigo.
        "#F539FF", // Magenta.
        "#F0AFFF", // Lavender.
        "#CB6BE6", // Violet.
        "#5D17EB", // Ungu Tua.
        "#800000"  // Maroon.
    };

    public static final String WARNA_DEFAULT_HEX = "#000000";

    // Mendapatkan warna block puzzle berdasarkan huruf block puzzle
    public static String getWarna(char huruf_block) {
        if (huruf_block == ' ') {
            return WARNA_DEFAULT_HEX;
        }
        int indeks = huruf_block - 'A';
        if (indeks >= 0 && indeks < WARNA_HEX.length) {
            return WARNA_HEX[indeks];
        }
        return WARNA_DEFAULT_HEX;
    }

    // Menghasilkan gambar dari solusi dan menyimpannya
    public static void generateGambarSolusi(char[][] papan, String filename) {
        int ukuran = 50;
        int panjang = papan.length * ukuran;
        int lebar = papan[0].length * ukuran;
        BufferedImage image = new BufferedImage(lebar, panjang, BufferedImage.TYPE_INT_RGB);
        Graphics2D g = image.createGraphics();

        for (int i = 0; i < papan.length; i++) {
            for (int j = 0; j < papan[i].length; j++) {
                String hexColor = getWarna(papan[i][j]);
                g.setColor(Color.decode(hexColor));
                g.fillRect(j * ukuran, i * ukuran, ukuran, ukuran);
                g.setColor(Color.BLACK); // warna outline
                g.drawRect(j * ukuran, i * ukuran, ukuran, ukuran);
                g.drawString(String.valueOf(papan[i][j]), j * ukuran + 20, i * ukuran + 30);
            }
        }

        g.dispose();
        try {
            ImageIO.write(image, formatName:"png", new File(filename));
        } catch (Exception e) {
            e.printStackTrace();
        }

    }


}

```

2. PapanPuzzle.java

```java
public class PapanPuzzle {
    public char[][] papan;
    public int indeks;
    public int count;

    public PapanPuzzle(char[][] papan, int indeks, int count) {
        this.papan = papan;
        this.indeks = indeks;
        this.count = count;
    }
}

```

3. Puzzle.java

```java
import java.util.*;

public class Puzzle {
    public int N, M, P;
    public String S;
    public List<char[][]> puzzle_blocks;
    public HashSet<Character> huruf_blocks;
    public char[][] papan;


    public Puzzle(int N, int M, int P, String S, List<char[][]> puzzle_blocks, HashSet<Character> huruf_blocks) {
        this.N = N;
        this.M = M;
        this.P = P;
        this.S = S;
        this.puzzle_blocks = puzzle_blocks;
        this.huruf_blocks = huruf_blocks;
        this.papan = new char[M][N];

        for (int i = 0; i < M; i++) {
            for (int j = 0; j < N; j++) {
                this.papan[i][j] = ' ';
            }
        }
    }
}

```

4. PuzzleSolver.java

```java
import java.util.*;

public class PuzzleSolver {

    // Mengecek apakah block puzzle dapat ditaruh pada papan
    public static boolean canBlockFit(char[][] papan, char[][] block, int x, int y) {
        // System.out.println("nyoba blok di (" + x + ", " + y + ")");
        int panjang_block = block.length;
        int lebar_block = block[0].length;
        int N = papan.length;
        int M = papan[0].length;

        if (x + panjang_block > N || y + lebar_block > M) {
            return false;
        }

        for (int i = 0; i < panjang_block; i++) {
            for (int j = 0; j < lebar_block; j++) {
                if ((papan[i + x][j + y] != ' ') && (block[i][j] != ' ')) {
                    // System.out.println("hayo");
                    return false;
                }
            }
        }

        return true;
    }

```

```java
    // Merotasi block puzzle sebanyak 90 derajat searah jarum jam
    public static char[][] rotateBlock(char[][] block) {
        int panjang_block = block.length;
        int lebar_block = block[0].length;
        char[][] rotated = new char[lebar_block][panjang_block]; // Swap dimensions

        for (int i = 0; i < lebar_block; i++) {
            for (int j = 0; j < panjang_block; j++) {
                rotated[i][j] = ' ';
            }
        }
        for (int i = 0; i < panjang_block; i++) {
            for (int j = 0; j < lebar_block; j++) {
                rotated[j][panjang_block - 1 - i] = block[i][j];
            }
        }

        return rotated;
    }

    // Mencerminkan block puzzle
    public static char[][] mirrorBlock(char[][] block) {
        int panjang_block = block.length;
        int lebar_block = block[0] .length;
        char[][] mirrored = new char[panjang_block][lebar_block];

        for (int i = 0; i < panjang_block; i++) {
            for (int j = 0; j < lebar_block; j++) {
                mirrored[i][j] = block[i][lebar_block - 1 - j];
            }
        }
        return mirrored;
    }

    // Menaruh block puzzle pada papan
    public static char[][] placeBlock(char[][] papan, char[][] block, int x, int y) {
        // System.out.println("naruh blok di (" + x + ", " + y + ")");
        int panjang_block = block.length;
        int lebar_block = block[0].length;
        int N = papan.length;
        int M = papan[0].length;
        char [][] new_papan = new char[N][M];

        for (int i = 0; i < N; i++) {
            for (int j = 0; j < M; j++) {
                new_papan[i][j] = papan[i][j];
            }
        }

        for (int i = 0; i < panjang_block; i++) {
            for (int j = 0; j < lebar_block; j++) {
                if (block[i][j] != ' ') {
                    new_papan[x + i][y + j] = block[i][j];
                }
            }
        }

        return new_papan;
    }
```

```java
    // Menghapus block puzzle dari papan
    public static char[][] removeBlock(char[][] papan, char[][] block, int x, int y) {
        int panjang_block = block.length;
        int lebar_block = block[0].length;
        int N = papan.length;
        int M = papan[0].length;
        char [][] new_papan = new char[N][M];

        for (int i = 0; i < N; i++) {
            for (int j = 0; j < M; j++) {
                new_papan[i][j] = papan[i][j];
            }
        }

        for (int i = 0; i < panjang_block; i++) {
            for (int j = 0; j < lebar_block; j++) {
                if (block[i][j] != '.') {
                    new_papan[x + i][y + j] = ' ';
                }
            }
        }

        return new_papan;
    }
```

```java
    // Mengecek apakah semua block puzzle sudah ditaruh pada papan
    public static boolean isAllBlockPlaced(HashSet<Character> huruf_blocks, char[][] papan) {
        for (char huruf : huruf_blocks) {
            boolean found = false;
            for (char[] baris : papan) {
                for (char c : baris) {
                    if (c == huruf) {
                        found = true;
                        break;
                    }
                }
                if (found) {
                    break;
                }
            }
            if (!found) {
                return false;
            }
        }
        return true;
    }

    // Menampilkan papan
    public static void printPapan(char[][] papan) {
        for (char[] baris : papan) {
            for (char c : baris) {
                System.out.print(c);
            }
            System.out.println();
        }
    }

    // Mengecek apakah papan sudah terisi penuh
    public static boolean isPapanPenuh(char[][] papan) {
        for (char[] baris : papan) {
            for (char c : baris) {
                if (c == ' ') {
                    return false;
                }
            }
        }
        return true;
    }

    // Mencari solusi dari puzzle dengan brute force
    public static Solusi solvePuzzleByBruteForce(char[][] papan, List<char[][]> puzzle_blocks, HashSet<Character> huruf_blocks
        Stack<PapanPuzzle> stack = new Stack<>();
        stack.push(new PapanPuzzle(papan, indeks:0, count:0));
        int cases = 0;

        while (!stack.isEmpty()) {
            PapanPuzzle current = stack.pop();
            char[][] current_papan = current.papan;
            int indeks = current.indeks;
            int count = current.count;
            cases++;

            if (indeks >= puzzle_blocks.size()) {
                if (isPapanPenuh(current_papan)) {
                    //printPapan(current_papan);
                    return new Solusi(current_papan, is_solved:true, cases);
                }
                continue;
            }

            char[][] original_block = puzzle_blocks.get(indeks);
            char[][] mirrored_block = mirrorBlock(original_block);
            int N = current_papan.length;
            int M = current_papan[0].length;
```

```
            for (int i = 0; i < N; i++) {
                for (int j= 0; j < M; j++) {
                    char[][] current_block = original_block;
                    for (int r = 0; r < 4; r++) {
                        if (canBlockFit(current_papan, current_block, i, j)) {
                            char[][] new_papan = placeBlock(current_papan, current_block, i, j);
                            // printPapan(new_papan);
                            // System.out.println();
                            stack.push(new PapanPuzzle(new_papan, indeks + 1, cases));
                        }
                        current_block = rotateBlock(current_block);
                        cases++;
                    }

                    current_block = mirrored_block;
                    for (int r = 0; r < 4; r++) {
                        if (canBlockFit(current_papan, current_block, i, j)) {
                            char[][] new_papan = placeBlock(current_papan, current_block, i, j);
                            stack.push(new PapanPuzzle(new_papan, indeks + 1, cases));
                        }
                        current_block = rotateBlock(current_block);
                        cases++;
                    }
                }
            }
        }
        return new Solusi(papan, is_solved:false, cases);

    }



    Run | Debug
    public static void main(String[] args) {

        // Meminta input file
        String input_filename = InputOutputFile.inputFile();
        Puzzle puzzle = InputOutputFile.bacaFilePuzzle(input_filename);

        // Mencari solusi puzzle dengan brute force
        long start_time = System.currentTimeMillis();

        Solusi solusi = solvePuzzleByBruteForce(puzzle.papan, puzzle.puzzle_blocks, puzzle.huruf_blocks);

        long end_time = System.currentTimeMillis();

        // Menampilkan waktu pencarian
        System.out.println();
        System.out.println("Waktu pencarian: " + (end_time - start_time) + " ms");
        System.out.println();

        // Menampilkan hasil
        if (solusi.is_solved) {
            System.out.println(x:"Puzzle solved! Yay :D");
            // printPapan(solusi.papan);
            System.out.println();
            Solusi.printSolusiBerwarna(solusi.papan, puzzle.huruf_blocks);
            System.out.println();

            System.out.println("Banyak kasus yang ditinjau: " + solusi.count);
            System.out.println();

            // Prompt menyimpan hasil
            System.out.println(x:"Apakah anda ingin menyimpan hasilnya? (ya/tidak)");
            Scanner scanner = new Scanner(System.in);
            String save_solusi = scanner.nextLine();
            if (save_solusi.equals(anObject:"ya")) {
                System.out.println(x:"Masukkan nama file untuk menyimpan hasil: ");
                String output_filename = scanner.nextLine();
                InputOutputFile.outputFile(solusi.papan, "test/solution/" + output_filename + ".txt");
                System.out.println(x:"Solusi berhasil disimpan pada test/solution!");
            }
            System.out.println();
```

```
            // Prompt menyimpan hasil
            System.out.println(x:"Apakah anda ingin menyimpan hasilnya? (ya/tidak)");
            Scanner scanner = new Scanner(System.in);
            String save_solusi = scanner.nextLine();
            if (save_solusi.equals(anObject:"ya")) {
                System.out.println(x:"Masukkan nama file untuk menyimpan hasil: ");
                String output_filename = scanner.nextLine();
                InputOutputFile.outputFile(solusi.papan, "test/solution/" + output_filename + ".txt");
                System.out.println(x:"Solusi berhasil disimpan pada test/solution!");
            }
            System.out.println();

            // Prompt menyimpan gambar
            System.out.println(x:"Apakah anda ingin menyimpan hasilnya sebagai gambar? (ya/tidak)");
            String save_gambar = scanner.nextLine();
            if (save_gambar.equals(anObject:"ya")) {
                System.out.println(x:"Masukkan nama file untuk menyimpan gambar: ");
                String output_filename = scanner.nextLine();
                InputOutputFile.generateGambarSolusi(solusi.papan, "test/solution/pic/" + output_filename + ".png");
                System.out.println(x:"Gambar berhasil disimpan pada test/solution/pic!");
            }

        } else {
            System.out.println(x:"Puzzle cannot be solved! Good luck next time.");
        }

    }

}
```

5. Solusi.java

```
import java.util.*;

public class Solusi {
    public char[][] papan;
    public boolean is_solved;
    public int count;

    public Solusi(char[][] papan, boolean is_solved, int count) {
        this.papan = papan;
        this.is_solved = is_solved;
        this.count = count;
    }

    public static final String[] WARNA = {
        "\u001B[38;2;255;255;255m", // Putih.
        "\u001B[38;2;166;166;166m", // Abu-Abu.
        "\u001B[38;2;255;49;49m", // Merah.
        "\u001B[38;2;255;145;76m", // Oranye.
        "\u001B[38;2;255;222;89m", // Kuning.
        "\u001B[38;2;126;217;86m", // Hijau Muda.
        "\u001B[38;2;0;191;98m", // Hijau.
        "\u001B[38;2;12;193;224m", // Biru Muda.
        "\u001B[38;2;0;74;173m", // Biru Tua.
        "\u001B[38;2;255;101;195m", // Pink.
        "\u001B[38;2;140;82;255m", // Ungu.
        "\u001B[38;2;255;87;87m", // Terracota.
        "\u001B[38;2;240;255;162m", // Kuning Pucat.
        "\u001B[38;2;254;189;89m", // Kuning Tua.
        "\u001B[38;2;148;73;18m", // Coklat.
        "\u001B[38;2;192;255;114m", // Lime.
        "\u001B[38;2;0;137;42m", // Hijau Tua.
        "\u001B[38;2;92;225;230m", // Cyan.
        "\u001B[38;2;0;151;178m", // Turqoise.
        "\u001B[38;2;56;182;255m", // Biru Langit.
        "\u001B[38;2;82;113;255m", // Indigo.
        "\u001B[38;2;245;57;255m", // Magenta.
        "\u001B[38;2;240;175;255m", // Lavender.
        "\u001B[38;2;203;107;230m", // Violet.
        "\u001B[38;2;93;23;235m", // Ungu Tua.
        "\u001B[38;2;128;0;0m" // Maroon.
    };
```

```java
    public static final String WARNA_DEFAULT = "\u001B[0m";

    // Mendapatkan warna block puzzle berdasarkan huruf block puzzle
    public static String getWarna(char huruf_block) {
        if (huruf_block == ' ') {
            return WARNA_DEFAULT;
        }
        int indeks = huruf_block - 'A';
        return WARNA[indeks];
    }

    // Menampilkan solusi berupa papan dengan block puzzle berwarna
    public static void printSolusiBerwarna(char[][] papan, HashSet<Character> huruf_blocks) {
        for (char[] baris : papan) {
            for (char huruf_block : baris) {
                System.out.print(getWarna(huruf_block) + huruf_block + WARNA_DEFAULT);
            }
            System.out.println();
        }
    }

}
```

BAB IV

ANALISIS DAN PENGUJIAN

4.1. Kasus Uji

4.1.1. Kasus Uji Berdasarkan Contoh pada Spesifikasi Tugas Kecil

1. File input testcase1.txt

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

Output:

```
==========================================
SELAMAT DATANG DI IQ PUZZLER PRO SOLVER!
==========================================

Masukkan nama file .txt (pastikan file ada pada folder test):
testcase1

Waktu pencarian: 265 ms

Puzzle solved! Yay :D

FDDEE
FFDEE
FFGGG
FCBAA
CCBBA

Banyak kasus yang ditinjau: 1582273

Apakah anda ingin menyimpan hasilnya? (ya/tidak)
ya
Masukkan nama file untuk menyimpan hasil:
solusi1
Solusi berhasil disimpan pada test/solution!

Apakah anda ingin menyimpan hasilnya sebagai gambar? (ya/tidak)
ya
Masukkan nama file untuk menyimpan gambar:
solusi1
Gambar berhasil disimpan pada test/solution/pic!
```

Solusi:

```
1   FDDEE
2   FFDEE
3   FFGGG
4   FCBAA
5   CCBBA
6
```

Analisis:

Program berhasil menampilkan solusi dari puzzle, seluruh block puzzle berhasil diletakkan pada papan dan tidak ada ruang kosong pada papan. Namun, program menghasilkan susunan peletakkan block puzzle yang berbeda dengan yang ada pada spesifikasi. Hal ini karena program mencoba seluruh titik pada papan setiap kali percobaan peletakkan block puzzle dan terdapat beberapa kali *backtracking* yang dilakukan.

4.1.2. Kasus Uji Berdasarkan Contoh pada Spesifikasi Tugas Kecil

1. File input testcase2.txt

```
3 3 3
DEFAULT
A
BB
CCC
```

Output:

```
========================================
SELAMAT DATANG DI IQ PUZZLER PRO SOLVER!
========================================

Masukkan nama file .txt (pastikan file ada pada folder test):
testcase2

Waktu pencarian: 53 ms

Puzzle cannot be solved! Good luck next time.
```

Analisis:

Program tidak menemukan hasil atau solusi dari permainan. Dapat terlihat pada file input bahwa akan selalu ada ruang kosong pada papan karena bentuk dan banyak blok puzzle yang ada tidak dapat mengisi seluruh titik pada papan.

2. File input testcase3.txt

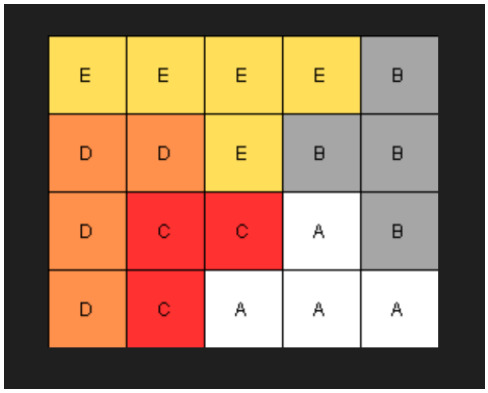
```
5 4 5
DEFAULT
AAA
 A
B
BB
B
 C
CC
 D
 D
DD
 E
EEEE
```

Output:

```
=======================================
SELAMAT DATANG DI IQ PUZZLER PRO SOLVER!
=======================================

Masukkan nama file .txt (pastikan file ada pada folder test):
testcase3

Waktu pencarian: 55 ms

Puzzle solved! Yay :D

EEEEB
DDEBB
DCCAB
DCAAA

Banyak kasus yang ditinjau: 186761

Apakah anda ingin menyimpan hasilnya? (ya/tidak)
ya
Masukkan nama file untuk menyimpan hasil:
solusi3
Solusi berhasil disimpan pada test/solution!

Apakah anda ingin menyimpan hasilnya sebagai gambar? (ya/tidak)
ya
Masukkan nama file untuk menyimpan gambar:
solusi3
Gambar berhasil disimpan pada test/solution/pic!
```

Solusi:

```
EEEEB
DDEBB
DCCAB
DCAAA
```



Analisis:
Program berhasil menampilkan solusi dari puzzle, seluruh block puzzle berhasil diletakkan pada papan dan tidak ada ruang kosong pada papan. Program meninjau kasus atau kemungkinan lebih banyak daripada kasus uji file input testcase1.txt karena bentuk blok puzzle yang semakin rumit.

3. File input testcase4.txt

```
5 4 5
DEFAULT
 AA
AAA
 A
 BBB
BBB
 CC
CC
 DD
DDD
 EEE
 E
EEE
```

Output:

```
========================================
SELAMAT DATANG DI IQ PUZZLER PRO SOLVER!
========================================

Masukkan nama file .txt (pastikan file ada pada folder test):
testcase4

Waktu pencarian: 16 ms

Puzzle cannot be solved! Good luck next time.
```

Analisis:
Program tidak menemukan hasil atau solusi dari permainan. Tidak semua blok puzzle berhasil diletakkan pada papan karena bentuknya yang cukup besar dan jumlahnya yang banyak.

4. File input testcase5.txt

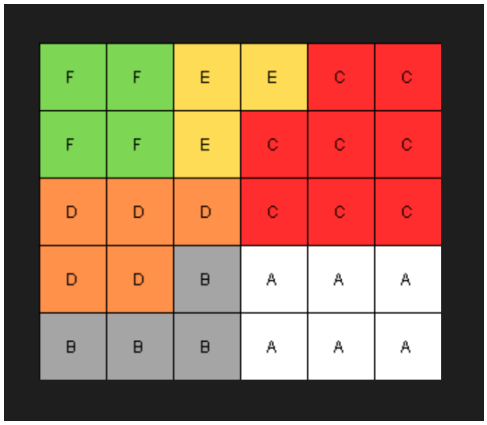
```
6 5 6
DEFAULT
AAA
AAA
BB
B
B
 CC
CCC
CCC
DDD
DD
 E
EE
FF
FF
```

Output:

```
========================================
SELAMAT DATANG DI IQ PUZZLER PRO SOLVER!
========================================

Masukkan nama file .txt (pastikan file ada pada folder test):
testcase5

Waktu pencarian: 31 ms

Puzzle solved! Yay :D

FFEECC
FFECCC
DDDCCC
DDBAAA
BBBAAA

Banyak kasus yang ditinjau: 212804

Apakah anda ingin menyimpan hasilnya? (ya/tidak)
ya
Masukkan nama file untuk menyimpan hasil:
solusi5
Solusi berhasil disimpan pada test/solution!

Apakah anda ingin menyimpan hasilnya sebagai gambar? (ya/tidak)
ya
Masukkan nama file untuk menyimpan gambar:
solusi5
Gambar berhasil disimpan pada test/solution/pic!
```

Solusi:

```
1   FFEECC
2   FFECCC
3   DDDCCC
4   DDBAAA
5   BBBAAA
6
```



Analisis:

Program berhasil menampilkan solusi dari puzzle, seluruh block puzzle berhasil diletakkan pada papan dan tidak ada ruang kosong pada papan. Program meninjau kasus atau kemungkinan lebih banyak dan membutuhkan waktu pencarian yang lebih lama daripada kasus uji lainnya karena dimensi papan yang lebih besar.

5. File input testcase6.txt

```
6 5 6
DEFAULT
ZZZZ
 Z
Y
YYY
 Y
X X
XXX
WW
WWW
VVV
 V
UU
 U
UU
 U
```

Output:

```
========================================
SELAMAT DATANG DI IQ PUZZLER PRO SOLVER!
========================================

Masukkan nama file .txt (pastikan file ada pada folder test):
testcase6

Waktu pencarian: 581 ms

Puzzle solved! Yay :D

VVVXXX
UVYXWX
UUYYWW
UYYZWW
UUZZZZ

Banyak kasus yang ditinjau: 3454736

Apakah anda ingin menyimpan hasilnya? (ya/tidak)
ya
Masukkan nama file untuk menyimpan hasil:
solusi6
Solusi berhasil disimpan pada test/solution!

Apakah anda ingin menyimpan hasilnya sebagai gambar? (ya/tidak)
ya
Masukkan nama file untuk menyimpan gambar:
solusi6
Gambar berhasil disimpan pada test/solution/pic!
```

Solusi:

```
1   VVVXXX
2   UVYXWX
3   UUYYWW
4   UYYZWW
5   UUZZZZ
6
```

Analisis:

Program berhasil menampilkan solusi dari puzzle, seluruh block puzzle berhasil diletakkan pada papan dan tidak ada ruang kosong pada papan.

Program meninjau kasus atau kemungkinan lebih banyak dan membutuhkan waktu pencarian yang lebih lama jika dibandingkan dengan kasus uji file input testcase5.txt karena meskipun dimensi papan sama besar, bentuk blok puzzle yang ada lebih rumit.

6. File input testcase7.txt

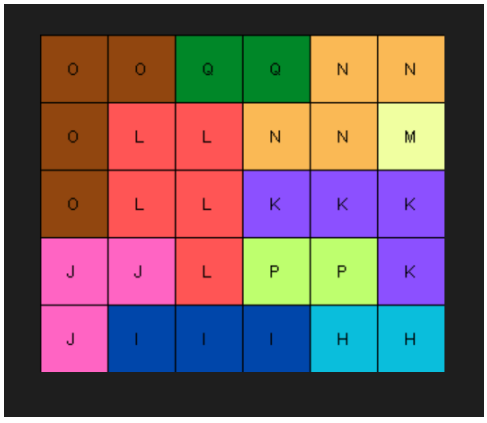
```
6 5 10
DEFAULT
H
H
III
J
JJ
 K
KKK
LL
LL
L
M
 N
NN
N
 O
 O
OO
PP
QQ
```

Output:

```
========================================
SELAMAT DATANG DI IQ PUZZLER PRO SOLVER!
========================================

Masukkan nama file .txt (pastikan file ada pada folder test):
testcase7

Waktu pencarian: 70 ms

Puzzle solved! Yay :D

OOQQNN
OLLNNM
OLLKKK
JJLPPK
JIIIHH

Banyak kasus yang ditinjau: 497425

Apakah anda ingin menyimpan hasilnya? (ya/tidak)
ya
Masukkan nama file untuk menyimpan hasil:
solusi7
Solusi berhasil disimpan pada test/solution!

Apakah anda ingin menyimpan hasilnya sebagai gambar? (ya/tidak)
ya
Masukkan nama file untuk menyimpan gambar:
solusi7
Gambar berhasil disimpan pada test/solution/pic!
```

Solusi:

```
1   OOQQNN
2   OLLNNM
3   OLLKKK
4   JJLPPK
5   JIIIHH
6
```



Analisis:

Program berhasil menampilkan solusi dari puzzle, seluruh block puzzle berhasil diletakkan pada papan dan tidak ada ruang kosong pada papan. Meskipun waktu pencarian lebih sedikit daripada kasus uji file input testcase 6.txt, program meninjau kasus atau kemungkinan lebih banyak dan membutuhkan waktu pencarian yang lebih lama karena jumlah blok puzzle yang lebih banyak yakni sebanyak 10.

BAB V

KESIMPULAN

Berdasarkan analisis yang telah dilakukan terhadap implementasi algoritma *brute force* pada penyelesaian permainan IQ Puzzler Pro, dapat disimpulkan bahwa algoritma ini berhasil menemukan penyelesaian atau solusi atas permainan (baik ada maupun tidak ada). Meskipun demikian, susunan peletakkan blok-blok puzzle pada solusi bisa saja berbeda dengan susunan peletakkan blok-blok puzzle bila disusun oleh manusia.Waktu eksekusi algoritma brute force menunjukkan performa yang baik untuk dimensi papan yang berukuran kecil. Semakin besar dimensi papan, semakin lama waktu pencarian yang dibutuhkan untuk menemukan solusi. Selain itu, bentuk blok-blok puzzle yang semakin rumit juga memperlama waktu pencarian. Meskipun algoritma *brute force* cenderung kurang mangkus untuk masukan berukuran besar, algoritma ini tetap menjadi pilihan yang valid untuk masukan dengan ukuran yang kecil dan untuk masalah-masalah sederhana yang memerlukan solusi yang langsung dan jelas. Namun, untuk masalah yang lebih kompleks dan memerlukan efisiensi waktu yang tinggi, diperlukan pendekatan algoritma yang lebih canggih dan pintar. Dengan demikian, pemahaman tentang kekuatan dan batasan algoritma *brute force*, dapat membantu dalam pemilihan dan penggunaan algoritma *brute force* dalam menyelesaikan berbagai masalah permasalahan.

LAMPIRAN

Repository Github:

https://github.com/angelinaefrina/Tucil1_13523060

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 *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	✔	

REFERENSI

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/02-Algoritma-Brute-Force-(2025)-Bag1.pdf

https://www.geeksforgeeks.org/brute-force-approach-and-its-pros-and-cons/

https://www.smartgames.eu/uk/one-player-games/iq-puzzler-pro-0