IF2211 Strategi Algoritma

Penyelesaian IQ Puzzler Pro dengan Algoritma *Brute Force*

Laporan Tugas Kecil

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

Oleh

Dita Maheswari 13523125

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2024

DAFTAR ISI

BAB 1 DESKRIPSI MASALAH

1.1 Algoritma *Brute Force*

Algoritma *brute force* adalah salah satu pendekatan dalam dunia komputasi yang sering digunakan secara sistematis. Dalam dunia teknologi informasi, pemahaman mendalam tentang algoritma *brute force* sangat penting, terutama ketika berbicara tentang pemecahan masalah kompleks. Algoritma *brute force* adalah metode yang digunakan untuk memecahkan masalah atau mencari solusi dengan cara mencoba semua kemungkinan secara berurutan hingga menemukan solusi yang benar. Metode ini sering digunakan ketika tidak ada cara yang lebih efisien untuk menyelesaikan masalah tersebut. Algoritma *brute force* adalah metode yang sangat sederhana dan kuat karena tidak mengandalkan pengetahuan sebelumnya atau optimasi khusus.

Tujuan utama dari algoritma *brute force* adalah mencari solusi dari sebuah masalah dengan menguji semua kemungkinan secara sistematis. Beberapa tujuan utama dari penggunaan algoritma ini meliputi:

1. Pencarian solusi
   Pencarian solusi dalam algoritma *brute force* yakni menggunakan pendekatan yang paling sederhana untuk menemukan solusi dari sebuah masalah.
2. Verifikasi
   Verifikasi dalam algoritma *brute force* yakni memastikan bahwa solusi yang ditemukan adalah benar dengan menguji semua kemungkinan.
3. Pemahaman masalah
   Setelah melakukan verifikasi, kita mendapatkan pemahaman yang lebih dalam tentang masalah yang dihadapi.

Langkah-langkah pengerjaan algoritma *brute force* adalah sebagai berikut:

1. Inisialisasi
   Inisiasi merupakan pengaturan semua parameter dan variabel yang diperlukan diatur dengan nilai awal.
2. Iterasi
   Algoritma akan memulai iterasi melalui semua kemungkinan secara berurutan.
3. Pengujian
   Pada setiap iterasi, algoritma akan menguji apakah solusi yang dihasilkan benar atau tidak.
4. Pemutakhiran

Jika solusi yang ditemukan adalah yang benar, maka algoritma akan menghentikan iterasi dan menghasilkan hasil. Jika tidak, algoritma akan melanjutkan ke iterasi berikutnya.

5. Penyelesaian
   Algoritma akan menghasilkan solusi yang benar setelah mengecek semua kemungkinan.

1.2 IQ Puzzler Pro

IQ Puzzler Pro adalah permainan teka-teki logika yang dibuat oleh SmartGames, yaitu sebuah perusahaan yang terkenal dengan berbagai permainan puzzle logika yang menantang. SmartGames sendiri berasal dari Belgia dan telah merancang banyak permainan puzzle edukatif untuk berbagai usia. IQ Puzzler Pro ini menggunakan serangkaian potongan puzzle 2D dan #D dengan berbagai bentuk yang harus disusun dalam sebuah papan sesuai dengan tantangan yang diberikan.

IQ Puzzler Pro merupakan bagian dari seri IQ Games yang dikembangkan oleh SmartGames. Permainan ini dirancang untuk melatih keterampilan berpikir logis, pemecahan masalah, dan spasial. IQ Puzzler Pro terdiri dari papan permainan kecil dan 12 kepingan berwarna dalam bentuk poliamon (kombinasi beberapa unit kubus). Pemain harus menyusun kepingan-kepingan tersebut sesuai dengan tantangan yang diberikan dalam buku panduan. Ada 3 mode permainan dalam IQ Puzzler Pro, yaitu mode 2D yang dimana pemain harus mengisi seluruh papan datar dengan kepingan sesuai tantangan. Kemudian, terdapat mode 3D yaitu pemain harus menyusun kepingan untuk membentuk piramida. Ketiga, mode diagonal, yaitu tantangan tambahan dengan pola permainan yang berbeda.

Sejak dirilis, IQ Puzzler Pro menjadi salah satu permainan puzzle paling populer di SmartGames karena desainnya yang kompak dan tantangan yang bervariasi. Permainan ini juga sering digunakan sebagai alat bantu edukatif untuk meningkatkan kemampuan berpikir logis dan spasial.

1.3 Hubungan Algoritma *Brute Force* dengan Permainan IQ Puzzler Pro

Algoritma *brute force* dalam permainan IQ Puzzler Pro digunakan untuk mencari solusi dengan mencoba semua kemungkinan penempatan masing-masing bagian hingga menemukan konfigurasi yang sesuai dengan aturan permainan. Dengan menggunakan algoritma *brute force*, pemain harus mencoba semua kemungkinan susunan blok secara sistematis hingga menemukan solusi yang valid karena IQ Puzzler Pro merupakan permainan puzzle yang mengharuskan pemain menyusun berbagai blok agar sesuai dengan grid.

Sebenarnya, *brute force* buka cara yang efisien untuk menyelesaikan sebuah permainan puzzle besar karena pertumbuhan eksponensial jumlah kemungkinan. Namun,

algoritma ini dapat diterapkan pada IQ Puzzler Pro karena permainan ini merupakan permainan puzzle kecil dengan jumlah blok yang terbatas, terutama jika dikombinasikan dengan teknik optimasi seperti pemangkasan pencarian yang mungkin tidak berhasil agar lebih cepat.

BAB 2 PENYELESAIAN

2.1 Langkah-Langkah Penyelesaian Permainan IQ Puzzler Pro Dengan Pendekatan Algoritma *Brute Force*

1. Pengguna diminta untuk memasukkan file .txt yang sudah disediakan di folder "test" atau pengguna dapat membuat file .txt sendiri di dalam folder "test" kemudian dimasukkan ke bagian yang telah disediakan. File .txt tersebut harus berisi dimensi papan N x M dengan jumlah blok puzzle P (termasuk bentuk tiap blok puzzle) serta terdapat jenis kasus (DEFAULT)
2. Program akan menyimpan data-data dari file input pengguna ke dalam variabel atau objek yang sesuai
3. Pendekatan *brute force* dimulai dengan mencoba semua kemungkinan peletakan blok pada papan hingga menemukan solusi yang valid atau menyimpulkan bahwa tidak ada solusi. Langkah pertama dalam algoritma ini adalah membaca input dari file test case yang diberikan dalam format .txt. File ini berisi informasi mengenai dimensi papan (N x M), jumlah blok puzzle (P), jenis konfigurasi puzzle (DEFAULT), serta bentuk masing-masing blok yang diwakili oleh huruf kapital.
4. Program akan merepresentasikan papan permainan dan blok-blok puzzle dalam struktur data yang sesuai. Papan permainan dapat direpresentasikan sebagai array 2D (matriks) dengan ukuran N x M, sedangkan blok-blok puzzle dapat disimpan dalam bentuk daftar yang memungkinkan operasi rotasi dan pencerminan. Representasi ini penting karena setiap blok dapat ditempatkan dengan berbagai orientasi dan perlu dicocokkan dengan ruang kosong yang tersedia pada papan.
5. Proses pencarian solusi dilakukan dengan menggunakan metode backtracking berbasis *Brute Force*. Dalam pendekatan ini, program akan mencoba meletakkan setiap blok puzzle pada setiap posisi kosong di papan permainan. Untuk setiap blok yang akan ditempatkan, dilakukan pemeriksaan apakah blok dapat diletakkan tanpa bertabrakan dengan blok lain yang sudah ada. Jika peletakan blok valid, maka blok tersebut ditempatkan sementara di papan, dan algoritma melanjutkan pencarian untuk menempatkan blok berikutnya secara rekursif. Jika pada tahap tertentu tidak ditemukan solusi yang memungkinkan, maka program akan melakukan backtracking, yaitu mencabut blok terakhir yang ditempatkan dan mencoba kemungkinan lain.

6. Setelah solusi ditemukan, program akan mencetak hasil dalam bentuk tampilan berwarna untuk membedakan blok yang berbeda. Selain itu, program juga harus mencatat waktu eksekusi dalam milidetik untuk mengukur efisiensi pencarian solusi serta mencatat jumlah iterasi yang telah diperiksa selama proses brute force berlangsung. Setelah hasil ditampilkan, pengguna akan diberikan opsi untuk menyimpan solusi ke dalam file .txt

BAB 3 IMPLEMENTASI PROGRAM

3.1 Struktur Data

```
└── Tucil1_13523125
    └── bin
        ├── BoardPiece.class
        ├── InputOutput.class
        ├── Puzzle.class
        ├── PuzzleImage.class
        └── PuzzleSolver.class
    └── doc
        └── Tucil1_K3_13523125_Dita Maheswari.pdf
    └── solusi
        ├── solusi_1.txt
        ├── solusi_2.txt
        ├── solusi_3.txt
        ├── solusi_4.txt
        ├── solusi_5.txt
        ├── solusi_6.txt
        ├── solusi_7.txt
        └── solusi_gambar6.png
    └── src
        ├── BoardPiece.java
        ├── InputOutput.java
        ├── Puzzle.java
        ├── PuzzleImage.java
        └── PuzzleSolver.java
    └── test
        ├── 1.txt
        ├── 2.txt
        ├── 3.txt
        ├── 4.txt
        ├── 5.txt
        ├── 6.txt
        ├── 7.txt
        ├── 8.txt
        └── impossible_case.txt
```

3.2 Fungsi dan Kelas

3.2.1 Puzzle.java

`class Puzzle {`	Program ini berfungsi untuk merepresentasikan permainan IQ Puzzler

	Pro yang terdiri dari papan yang berukuran N x M, blok puzzle dalam list karakter, dan HashSet untuk menyimpan karakter dalam blok.
`public Puzzle(int N, int M, int P, String S, List<char[][]> puzzle_blocks, HashSet<Character> huruf_blocks) {`	Konstruktor yang menerima parameter N, M, P, S, puzzle_blocks, dan huruf_blocks untuk inisiasi objek
`public char[][] getPapan() {`	Mengembalikan salinan papan agar tidak terjadi perubahan langsung pada variabel papan

3.2.2 InputOutput.java

`public static Puzzle bacaFilePuzzle(String fileName) {`	Membaca isi dari input file .txt dan mengembalikan objek Puzzle
`public static char[][] convertBlockToMatrix(List<String> block) {`	Mengonversi daftar string dari file menjadi matriks karakter (char[][])
`public static void outputFile(char[][] papan, String fileName) {`	Menyimpan solusi permainan ke dalam file berbentuk .txt dan disimpan ke dalam folder solusi

3.2.3 BoardPiece.java

`static final String[] COLORS` `static final String RESET`	Menyimpan 26 warna berbeda berdasarkan ANSI color codes dalam array dan akan dicetak pada terminal. Kemudian akan mereset warna setelah mencetak warna warni
`public static void printPapan(char[][] board)`	Menampilkan papan permainan dalam bentuk matriks. Fungsi ini melakukan looping pada tiap baris dan kolom dari “board”. Kemudian setiap bagian dari

	papan akan ditampilkan sesuai warna yang ditentukan pada fungsi COLORS.
`static boolean isPapanFull(char [][] papan)`	Memeriksa apakah papan sudah penuh atau belum. Fungsi ini melakukan looping ke seluruh elemen. Jika masih ada yang kosong, maka akan mengembalikan nilai *false*. Sedangkan, jika semua elemen papan sudah berisi bagian puzzle (huruf), maka akan mengembalikan nilai *true*.
`static List<char[][]> generateVariasi(char[][] piece)`	Melakukan berbagai kombinasi metode untuk menyelesaikan permainan ini, seperti melakukan *rotate* dan *flip*
`public static char[][] rotateBlock(char[][] piece)`	Memutar bagian puzzle sebesar 90 derajat searah jarum jam. Jika bagian puzzle berukuran m x n, maka akan menjadi n x m. Kemudian, fungsi ini melakukan iterasi setiap elemen dan ditempatkan di posisi yang benar, seperti baris lama menjadi kolom baru dan kolom lama dibalik menjadi baris baru.
`static char[][] flip(char[][] piece)`	Membalik bagian puzzle secara vertikal. Fungsi ini melakukan iterasi dari bawah ke atas kemudian disalin ke array yang baru.
`static boolean isFit(char[][] papan, char[][] piece, int a, int b)`	Memeriksa apakah bagian puzzle dapat dimasukkan ke papan. Fungsi ini melakukan looping untuk semua sel dama piece. Jika bagian puzzle bukan kosong (.) dan papan di posisi tersebut tidak kosong, maka akan mengembalikan nilai *false* (tidak dapat dipasang). Jika semua bagian papan dan puzzle sudah sesuai, maka akan mengembalikan nilai *true*.
`static void placePiece(char[][] papan, char[][] piece, int a, int b, char label)`	Memasukkan potongan puzzle ke papan di posisi (a,b), dan mengganti posisi yang kosong menjadi huruf (label). Fungsi ini melakukan semua sel dalam piece. Jika bagian puzzle bukan kosong, letakkan label pada papan.
`static void removePiece(char[][]`	Menghapus bagian puzzle dari papan jika

`papan, char[][] piece, int a, int b)`	tidak sesuai. Fungsi ini akan melakukan looping kepada semua sel dalam piece. Jika bagian puzzle bukan kosong, maka akan mengembalikan posisi kosong (.) pada papan.

3.2.4 PuzzleSolver.java

`public class PuzzleSolver`	Mengidentifikasi variabel apa saja yang digunakan dalam program ini
`public static void main(String[] args)`	Merupakan fungsi utama sebagai driver pada program ini. Pengguna akan diminta memasukkan nama file yang berisi konfigurasi dari puzzle. Setelah itu, program akan membaca apa yang sudah dimasukkan oleh pengguna dan mulai mencari solusi yang tepat. Jika sudah ada solusi, program akan menampilkan solusi tersebut dan menawarkan kepada pengguna apakah mau menyimpan hasil solusi tersebut atau tidak.
`static boolean solve(int idx)`	Merupakan fungsi paling penting karena fungsi ini menerapkan algoritma *brute force* dengan rekursif dan *backtracking* untuk menyelesaikan permainan ini. Fungsi ini menerima parameter idx yang menunjukkan indeks bagian puzzle yang diproses. Jika idx mencapai jumlah total bagian yang tersedia, maka akan memanggil fungsi isPapanFull sampai solusi ditemukan. Iterasi akan bertambah satu selama pencarian solusi. Bagian puzzle akan melakukan semua variasi rotasi dan flip sampai menemukan solusi penempatan pada bagian puzzle. Dengan melakukan iterasi program akan memeriksa apakah bagian puzzle dapat dipasang (melalui isFit). Jika dapat dipasang, maka akan diletakkan di papan menggunakan fungsi placePiece. Kemudian, metode solve(idx + 1) dipanggil secara rekursif untuk menangani bagian puzzle berikutnya

	sampai mengembalikan nilai *true*. Jika pada titik tertentu ada bagian puzzle yang tidak dapat dipasang, mapa program akan melakukan *backtracking* dengan menghapus bagian puzzle dengan removePiece kemudian mencari variasi lainnya hingga mendapatkan solusi yang tepat.

3.2.5 PuzzleImage.java (bonus : Output dalam bentuk gambar)

`public static void savePuzzleImage(char[][] papan, String filename)`	Program ini bertanggung jawab untuk menyimpan output dalam bentuk gambar. Fungsi savePuzzleImage menerima papan permainan dan nama file untuk menyimpan gambar hasil output dalam bentuk PNG. Di fungsi ini juga sudah diatur agar warna sesuai yang ada di class BoardPiece.

3.3 Source Code Program

1. Puzzle.java
   File ini berfungsi untuk merepresentasikan variabel dalam permainan IQ Puzzler Pro

```
import java.util.*;

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

        for (char[] row : papan) {
            Arrays.fill(row, '.');
        }
    }

    public char[][] getPapan() {
        char[][] papanCopy = new char[N][M];
        for (int i = 0; i < N; i++) {
            System.arraycopy(papan[i], 0, papanCopy[i], 0, M);
        }
        return papanCopy;
    }
}
```

2. InputOutput.java
   File ini berfungsi untuk membaca bagian bagian puzzle dari file .txt, kemudian diolah dan menyimpan solusinya setelah diproses.

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

public class InputOutput {

    // Membaca masukan file .txt
    public static Puzzle bacaFilePuzzle(String fileName) {
        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(fileName));
            String line = reader.readLine();
            if (line == null || line.trim().isEmpty()) {
                throw new IOException("Isi file kosong! Coba lagi.");
            }
            // Membaca baris pertama
            String[] first_line = line.trim().split(" ");
            if (first_line.length != 3) {
                throw new IOException("Block 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("Tipe Puzzle belum ada! Coba lagi.");
            }
            S = line.trim();
            if (!(S.equals("DEFAULT") || S.equals("CUSTOM"))) {
                throw new IOException("Tipe Puzzle belum sesuai! Coba
lagi.");
            }

            // Membaca baris-baris selanjutnya/membaca block puzzle
            List<String> block = new ArrayList<>();
            char current_block = ' ';

            while ((line = reader.readLine()) != null) {
                if (!line.isEmpty()) {
                    char jenis_block = line.trim().charAt(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));
            }

            reader.close();

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

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

    // membaca txt menjadi matriks
```

```
    public static char[][] convertBlockToMatrix(List<String> block) {
        int baris = block.size();
        int kolom = block.stream().mapToInt(String::length).max().orElse(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;
    }

    public static void outputFile(char[][] papan, String fileName) {
        String filePath = "solusi/" + fileName;
        try (BufferedWriter writer = new BufferedWriter(new
FileWriter(filePath))) {
            for (char[] baris : papan) {
                for (char c : baris) {
                    writer.write(c);
                }
                writer.newLine();
            }
            System.out.println("Solusi berhasil disimpan di: " + filePath);
        } catch (IOException e) {
            System.out.println("Error : " + e.getMessage());
        }
    }
}
```

3. BoardPiece.java
   File ini berfungsi untuk mencari solusi dari potongan puzzle terhadap papan yang disesuaikan dari file .txt, seperti mencari variasi bentuk puzzle (rotate dan flip), kemudian memeriksa apakah puzzle bisa ditempatkan di papan, serta dapat menambahkan atau menghapus bagian puzzle dari papan jika tidak sesuai.

```
import java.util.*;

public class BoardPiece {
    static char[][] papan;
    static final String[] COLORS = {
```

```
        "\u001B[30m", "\u001B[31m", "\u001B[32m", "\u001B[33m", "\u001B[34m",
        "\u001B[35m", "\u001B[36m", "\u001B[37m", "\u001B[90m", "\u001B[91m",
        "\u001B[92m", "\u001B[93m", "\u001B[94m", "\u001B[95m", "\u001B[96m",
        "\u001B[97m", "\u001B[38;5;21m", "\u001B[38;5;27m",
"\u001B[38;5;46m",
        "\u001B[38;5;82m", "\u001B[38;5;124m", "\u001B[38;5;130m",
"\u001B[38;5;172m",
        "\u001B[38;5;184m", "\u001B[38;5;201m", "\u001B[38;5;208m"
    };
    static final String RESET = "\u001B[0m";

    // memunculkan board atau papan
    public static void printPapan(char[][] board) {
        for (char[] baris : board) {
            for (char cell : baris) {
                if (cell == '.') {
                    System.out.print(cell + " ");
                } else {
                    int colorIndex = (cell - 'A') % COLORS.length;
                    System.out.print(COLORS[colorIndex] + cell + RESET + "
");
                }
            }
            System.out.println();
        }
    }

    // mengecek apakah papan sudah penuh
    static boolean isPapanFull(char [][] papan) {
        for (char[] baris : papan) {
            for (char cell : baris) {
                if (cell == '.') {
                    return false;
                }
            }
        }
        return true;
    }

    // memberitahu apa saja yang bisa dilakukan oleh puzzle, seperti rotate,
flip, dan lain-lain
    static List<char[][]> generateVariasi(char[][] piece) {
        Set<String> set = new HashSet<>();
        List<char[][]> variasi = new ArrayList<>();
        char[][] curr = piece;
        for (int i = 0; i < 4; i++) {
            if (set.add(Arrays.deepToString(curr))) {
                variasi.add(curr);
```

```
            }
            curr = rotateBlock(curr);
        }
        piece = flip(piece);
        curr = piece;
        for (int i = 0; i < 4; i++) {
            if (set.add(Arrays.deepToString(curr))) {
                variasi.add(curr);
            }
            curr = rotateBlock(curr);
        }
        return variasi;
    }

    // memutar searah 90 derajat
    public static char[][] rotateBlock(char[][] piece) {
        int baris = piece.length;
        int kolom = piece[0].length;
        char[][] putar = new char[kolom][baris];
        for (int a = 0; a < baris; a++) {
            for (int b = 0; b < kolom; b++) {
                putar[b][baris - 1 - a] = piece[a][b];
            }
        }
        return putar;
    }

    // memutar searah 180 derajat (membalik bagian puzzle nya)
    static char[][] flip(char[][] piece) {
        int baris = piece.length;
        char[][] balik = new char[baris][];
        for (int i = 0; i < baris; i++) {
            balik[i] = piece[baris - i - 1].clone();
        }
        return balik;
    }

    // mengecek apakah bagian puzzle bisa dimasukkan ke papan
    static boolean isFit(char[][] papan, char[][] piece, int a, int b) {
        for (int i = 0; i < piece.length; i++) {
            for (int j = 0; j < piece[0].length; j++) {
                if (piece[i][j] != '.' && piece[i][j] != ' ' && papan[a +
i][b + j] != '.') {
                    return false;
                }
            }
        }
        return true;
```

```
    }

    // memasukkan bagian puzzle ke papan
    static void placePiece(char[][] papan, char[][] piece, int a, int b, char
label) {
        for (int i = 0; i < piece.length; i++) {
            for (int j = 0; j < piece[0].length; j++) {
                if (piece[i][j] != '.' && piece[i][j] != ' ') {
                    papan[a + i][b + j] = label;
                }
            }
        }
    }


    // menghapus bagian puzzle dari papan
    static void removePiece(char[][] papan, char[][] piece, int a, int b) {
        for (int i = 0; i < piece.length; i++) {
            for (int j = 0; j < piece[0].length; j++) {
                if (piece[i][j] != '.' && piece[i][j] != ' ') {
                    papan[a + i][b + j] = '.';
                }
            }
        }
    }
}
```

4. PuzzleSolver.java
   Merupakan driver utama untuk menjalankan program ini

```
// import java.io.*;
import java.nio.file.Paths;
import java.nio.file.Path;
import java.util.*;


public class PuzzleSolver {
    static char[][] papan;
    static List<char[][]> bagian;
    static long iterasi = 0;
    static long waktuMulai;


    public static void main(String[] args) {
        System.out.println("\r\n" + //
```

```
                    "                                                   \r\n" + //
                    "                                                   \r\n" + //
                    "                                                   \r\n" + //
                    "                                                   \r\n" + //
                    "                                                   \r\n" + //
                    "                                                   \r\n" + //
                    "
\r\n" + //
                    "
\r\n" + //
                    "                                                   \r\n" + //
                    "                                                   \r\n" + //
                    "                                                   \r\n" + //
                    "                                                   \r\n" + //
                    "                                                   \r\n" + //
                    "
\r\n" + //
                    "
\r\n" + //
                    ""); // ini iseng aja sih kak, mau nyoba nyoba aja

        Scanner scanner = new Scanner(System.in);
        System.out.print("Masukkan nama file: ");
        String fileName = scanner.nextLine().trim();

        Path filePath = Paths.get("test", fileName); // folder txt nya ada di
test ya kak

        try {
            Puzzle puzzle = InputOutput.bacaFilePuzzle(filePath.toString());
            papan = new char[puzzle.N][puzzle.M];
            bagian = puzzle.puzzle_blocks;


            for (char[] baris : papan) Arrays.fill(baris, '.');

            waktuMulai = System.currentTimeMillis();
            BoardPiece.papan = papan;
            boolean solusi = solve(0);
```

```
            long waktuSelesai = System.currentTimeMillis();

            if (solusi) {
                BoardPiece.printPapan(papan);
            } else {
                System.out.println("Tidak ada solusi");
            }

            System.out.println("Waktu pencarian: " + (waktuSelesai -
waktuMulai) + "ms");
            System.out.println("Banyak kasus yang ditinjau: " + iterasi);

            System.out.print("Apakah ingin menyimpan solusi? (ya/tidak): ");
            String response = scanner.nextLine().trim().toLowerCase();
            if (response.equals("ya")) {
                System.out.print("Masukkan nama file solusi: ");
                String outputFile = scanner.nextLine();
                InputOutput.outputFile(papan, outputFile);
            }
        } catch (Exception e) {
            System.out.println("Terjadi kesalahan: " + e.getMessage());
        }

        System.out.print("Apakah ingin menyimpan solusi sebagai gambar?
(ya/tidak): ");
        String response = scanner.nextLine().trim().toLowerCase();
        if (response.equals("ya")) {
            System.out.print("Masukkan nama file gambar (tanpa ekstensi): ");
            String outputFile = scanner.nextLine();
            PuzzleImage.savePuzzleImage(papan, outputFile + ".png");
        }
    }

    static boolean solve(int idx) {
        if (idx == bagian.size()) return BoardPiece.isPapanFull(papan);
        iterasi++;
        char[][] piece = bagian.get(idx);
        List<char[][]> variasi = BoardPiece.generateVariasi(piece);

        for (char[][] macam : variasi) {
            for (int a = 0; a <= papan.length - macam.length; a++) {
                for (int b = 0; b <= papan[0].length - macam[0].length; b++)
{
                    if (BoardPiece.isFit(papan, macam, a, b)) {
                        BoardPiece.placePiece(papan, macam, a, b, (char) ('A'
+ idx));
                        if (solve(idx + 1)) return true;
                        BoardPiece.removePiece(papan, macam, a, b);
```

```
                    }
                }
            }
        }
        return false;
    }
}
```

5. PuzzleImage.java
   File ini merupakan file bonus yaitu menyimpan solusi ke dalam bentuk gambar dan sesuai dengan warna yang telah ditentukan (dapat dilihat di file BoardPiece.java)

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

public class PuzzleImage {
    public static void savePuzzleImage(char[][] papan, String filename) {
        int cellSize = 50; // ukuran 50x50 px
        int lebar = papan[0].length * cellSize;
        int tinggi = papan.length * cellSize;

        BufferedImage image = new BufferedImage(lebar, tinggi,
BufferedImage.TYPE_INT_ARGB);
        Graphics2D g2d = image.createGraphics();

        // 26 warna berbeda untuk setiap blok puzzle (A sampai Z), sesuai RGB
        Color[] colors = {
            new Color(169, 169, 169), // Abu Abu Terang
            new Color(255, 0, 0),     // Merah
            new Color(0, 255, 0),     // Hijau
            new Color(255, 255, 0),   // Kuning
            new Color(0, 0, 255),     // Biru
            new Color(255, 0, 255),   // Magenta
            new Color(0, 255, 255),   // Cyan
            new Color(255, 255, 255), // Putih
            new Color(128, 128, 128), // Abu-abu gelap
            new Color(255, 99, 71),   // Tomato
            new Color(144, 238, 144), // Light Green
            new Color(255, 215, 0),   // Gold
            new Color(65, 105, 225),  // Royal Blue
            new Color(186, 85, 211),  // Medium Orchid
            new Color(32, 178, 170),  // Light Sea Green
            new Color(240, 248, 255), // Alice Blue
```

```
            new Color(25, 25, 112),   // Midnight Blue
            new Color(70, 130, 180),  // Steel Blue
            new Color(50, 205, 50),   // Lime Green
            new Color(0, 255, 127),   // Spring Green
            new Color(139, 0, 0),     // Dark Red
            new Color(210, 105, 30),  // Chocolate
            new Color(218, 165, 32),  // Goldenrod
            new Color(255, 228, 181), // Moccasin
            new Color(255, 105, 180), // Hot Pink
            new Color(255, 140, 0)    // Dark Orange
            };

        // Background putih
        g2d.setColor(Color.WHITE);
        g2d.fillRect(0, 0, lebar, tinggi);

        // Gambar lingkaran untuk setiap blok puzzle
        for (int baris = 0; baris < papan.length; baris++) {
            for (int kolom = 0; kolom < papan[0].length; kolom++) {
                char c = papan[baris][kolom];
                if (c != '.') {
                    g2d.setColor(colors[(c - 'A') % colors.length]); // Ambil
warna berdasarkan huruf blok
                    g2d.fillOval(kolom * cellSize, baris * cellSize,
cellSize, cellSize); // Gambar lingkaran

                    // Tambahkan label huruf di tengah lingkaran
                    g2d.setColor(Color.BLACK);
                    g2d.setFont(new Font("Comic Sans MS", Font.BOLD, 20));
                    FontMetrics metrics = g2d.getFontMetrics();
                    int x = kolom * cellSize + (cellSize -
metrics.stringWidth(String.valueOf(c))) / 2;
                    int y = baris * cellSize + ((cellSize -
metrics.getHeight()) / 2) + metrics.getAscent();
                    g2d.drawString(String.valueOf(c), x, y);
                }
            }
        }

        g2d.dispose();

        try {
            File outputFile = new File("solusi/" + filename);   // di save ke
folder solusi
            ImageIO.write(image, "PNG", outputFile);
            System.out.println("Solusi disimpan dalam file: " + filename);
        } catch (Exception e) {
            System.out.println("Gagal menyimpan gambar: " + e.getMessage());
```

```
            }
        }
}
```

3.4 Library

1. java.io.*
   Library ini digunakan untuk membaca dan menulis file
2. java.util.*
   Library ini digunakan untuk struktur data seperti HashShet dan ArrayList. Bisa juga sebagai scanner untuk membaca file
3. java.nio.file.Paths dan java.nio.file.Path
   Library ini digunakan untuk mengelola path file yang akan dibaca
4. java.awt.*
   Library ini digunakan untuk menggambar hasil solusi
5. java.awt.image.BufferedImage
   Library ini digunakan untuk menangani gambar sebelum disimpan ke dalam file
6. javax.imageio.ImageIO
   Library ini digunakan untuk menyimpan hasil gambar dalam format PNG

BAB 4 PENGUJIAN

1. Kasus 1

Tucil1_13523125 > test > 1.txt

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

```
Masukkan nama file: 1.txt
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
Waktu pencarian: 245ms
Banyak kasus yang ditinjau: 7393
Apakah ingin menyimpan solusi? (ya/tidak): ya
Masukkan nama file solusi: solusi_1.txt
Solusi berhasil disimpan di: solusi/solusi_1.txt
Apakah ingin menyimpan solusi sebagai gambar? (ya/tidak): tidak
```

Tucil1_13523125 > solusi > solusi_1.txt

```
AGGGD
AABDD
CCBBE
CFFEE
FFFEE
```

2. Kasus 2

Tucil1_13523125 > test > 2.txt

```
3 10 8
DEFAULT
A
A
AA
BB
BB
 B
 B
 B
 B
C
DD
 D
E
E
E
FF
GGG
G G
GGG
H
```

```
Masukkan nama file: 2.txt
A B B B B B B G G G
A B B C D D F G H G
A A E E E D F G G G
Waktu pencarian: 1577ms
Banyak kasus yang ditinjau: 130699
Apakah ingin menyimpan solusi? (ya/tidak): ya
Masukkan nama file solusi: solusi_2.txt
Solusi berhasil disimpan di: solusi/solusi_2.txt
Apakah ingin menyimpan solusi sebagai gambar? (ya/tidak): tidak
```

Tucil1_13523125 > solusi > solusi_2.txt

```
ABBBBBBGGG
ABBCDDFGHG
AAEEEDFGGG
```

3. Kasus 3

Tucil1_13523125 > test > 3.txt

```
4 4 6
DEFAULT
F
FF
AAA
A
BB
C
DDD
E
EE
```

```
Masukkan nama file: 3.txt
A C C D
A A F F
B B B F
B E E E
Waktu pencarian: 9ms
Banyak kasus yang ditinjau: 6
Apakah ingin menyimpan solusi? (ya/tidak): ya
Masukkan nama file solusi: solusi_3.txt
Solusi berhasil disimpan di: solusi/solusi_3.txt
Apakah ingin menyimpan solusi sebagai gambar? (ya/tidak): tidak
```

Tucil1_13523125 > solusi > solusi_3.txt

```
ACCD
AAFF
BBBF
BEEE
```

4. Kasus 4

Tucil1_13523125 > test > 4.txt

```
5 5 7
DEFAULT
A
AA
B
BB
C
CCC
D
DD
E
EEE
F
FFFF
G
GG
```

```
Masukkan nama file: 4.txt
A E E E F
A A B E F
G G B B F
D G C F F
D D C C C
Waktu pencarian: 265ms
Banyak kasus yang ditinjau: 9662
Apakah ingin menyimpan solusi? (ya/tidak): ya
Masukkan nama file solusi: solusi_4.txt
Solusi berhasil disimpan di: solusi/solusi_4.txt
Apakah ingin menyimpan solusi sebagai gambar? (ya/tidak): tidak
```

Tucil1_13523125 > solusi > solusi_4.txt

```
AEEEF
AABEF
GGBBF
DGCFF
DDCCC
```

5. **Kasus 5**

Tucil1_13523125 > test > 5.txt

```
3 3 4
DEFAULT
A
B
B
B
C
C
D
D
D
```

```
Masukkan nama file: 5.txt
A B D
C B D
C B D
Waktu pencarian: 16ms
Banyak kasus yang ditinjau: 5
Apakah ingin menyimpan solusi? (ya/tidak): ya
Masukkan nama file solusi: solusi_5.txt
Solusi berhasil disimpan di: solusi/solusi_5.txt
Apakah ingin menyimpan solusi sebagai gambar? (ya/tidak): tidak
```

Tucil1_13523125 > solusi > solusi_5.txt

```
ABD
CBD
CBD
```

6. **Kasus 6**

Tucil1_13523125 > test > 7.txt

```
4 6 6
DEFAULT
AAA
B
BB
CC
C
DDD
 D
DD
 E
EE
EE
FFF
F
```

```
Masukkan nama file: 7.txt
A A A E E E
F B D D E E
F B B D C C
F F D D D C
Waktu pencarian: 80ms
Banyak kasus yang ditinjau: 992
Apakah ingin menyimpan solusi? (ya/tidak): ya
Masukkan nama file solusi: solusi_7.txt
Solusi berhasil disimpan di: solusi/solusi_7.txt
Apakah ingin menyimpan solusi sebagai gambar? (ya/tidak): tidak
```

Tucil1_13523125 > solusi > solusi_7.txt

```
AAAEEE
FBDDEE
FBBDCC
FFDDDC
```

7. **Kasus 7**

Tucil1_13523125 > test > 8.txt

```
3 4 3
DEFAULT
AAAAA
BB
BB
CCC
```

```
Masukkan nama file: 8.txt
Tidak ada solusi
Waktu pencarian: 10ms
Banyak kasus yang ditinjau: 1
Apakah ingin menyimpan solusi? (ya/tidak): tidak
Apakah ingin menyimpan solusi sebagai gambar? (ya/tidak): tidak
```

Hal ini disebabkan karena papan hanya berukuran 3x4, sedangkan ada bagian yang berbentuk 1x5, jadi tidak akan ada solusi untuk menyelesaikan permainan ini

8. Kasus 8

(menyimpan output berbentuk gambar)

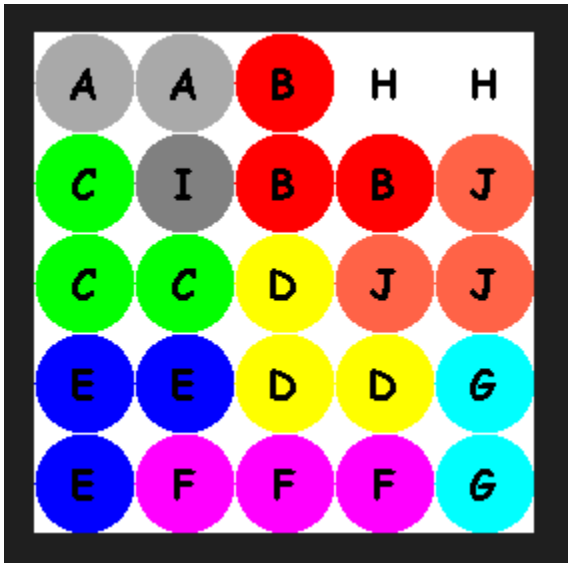
Tucil1_13523125 > test > 6.txt

```
5 5 10
DEFAULT
AA
B
BB
C
CC
D
DD
E
EE
FFF
G
G
HH
I
J
JJ
```

```
Masukkan nama file: 6.txt
A A B H H
C I B B J
C C D J J
E E D D G
E F F F G
Waktu pencarian: 46ms
Banyak kasus yang ditinjau: 250
Apakah ingin menyimpan solusi? (ya/tidak): ya
Masukkan nama file solusi: solusi_6.txt
Solusi berhasil disimpan di: solusi/solusi_6.txt
Apakah ingin menyimpan solusi sebagai gambar? (ya/tidak): ya
Masukkan nama file gambar (tanpa ekstensi): solusi_gambar6
Solusi disimpan dalam file: solusi_gambar6.png
```

Tucil1_13523125 > solusi > solusi_6.txt

```
AABHH
CIBBJ
CCDJJ
EEDDG
EFFFG
```



BAB 5 KESIMPULAN

IQ Puzzler Pro merupakan permainan yang memerlukan pencarian kombinasi peletakan blok puzzle pada papan hingga semua bagian terisi dengan benar. Dalam tugas ini, algoritma *brute force* digunakan untuk mencari solusi dengan mencoba semua kemungkinan konfigurasi blok hingga menemukan satu solusi atau menyimpulkan bahwa tidak ada solusi yang memungkinkan. Waktu eksekusi dalam milidetik diukur sejak proses pencarian solusi dimulai hingga solusi ditemukan. Banyak kasus yang ditinjau merupakan iterasi yang diperiksa oleh algoritma *brute force* sebelum menemukan solusi. Kompleksitas algoritma pada program kali ini terbilang cukup besar dikarenakan banyak metode yang harus dilakukan dalam penyelesaian permainan IQ Puzzler Pro kali ini, seperti *rotate, flip, remove,* dan sebagainya. Selain itu, *brute force* harus mencoba semua kemungkinan penempatan blok pada papan sehingga menyebabkan kompleksitas algoritma yang cukup besar.

Walaupun memiliki kompleksitas algoritma yang cukup tinggi, algoritma *brute force* ini sederhana untuk digunakan dan dapat memastikan semua kemungkinan diuji sehingga solusi yang ditemukan pasti benar. Akan tetapi, algoritma ini memakan waktu eksekusi sangat lama jika permainan ini memiliki banyak bagian dan papan yang lebih besar. Algoritma *brute force* ini kurang efisien dibandingkan metode lain seperti backtracking dengan heuristik.

BAB 6 LAMPIRAN

Link github : https://github.com/DitaMaheswari05/Tucil1_13523125.git

DAFTAR PUSTAKA

https://www.cloudeka.id/id/berita/web-sec/cara-kerja-algoritma-brute-force/
(diakses pada 21 Februari 2025)

https://id.wikipedia.org/wiki/Daftar_warna
(diakses pada 23 Februari 2025)