Tugas Kecil 1 IF2211 Strategi Algoritma
Penyelesaian IQ Puzzler Pro dengan Algoritma Brute Force

Disusun oleh

Bertha Soliany Frandi 13523026

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2024

DAFTAR ISI

BAB I

DESKRIPSI MASALAH

1.1 IQ Puzzler Pro

IQ Puzzler Pro adalah permainan puzzle yang diproduksi oleh Smart Games. Sebagai permainan puzzle, IQ Puzzler Pro membutuhkan satu papan dan sejumlah blok puzzle. Papan permainan yang digunakan mayoritas memiliki dimensi 5x11 sedangkan untuk blok puzzler memiliki dimensi yang berbeda-beda. Blok puzzle dalam yang digunakan memiliki warna yang berbeda-beda untuk memudahkan pemain dalam melihat bentuk unik dari blok puzzle. IQ Puzzler Pro dimainkan dengan cara menyusun semua blok puzzle kedalam papan. Jika masih tersisa blok puzzle namun papan sudah terisi penuh, permainan dianggap tidak terselesaikan.

1.2 Algoritma *Brute Force*

Brute Force memiliki definisi sebagai pendekatan yang terus terang untuk memecahkan suatu persoalan. Terus terang dimaksudkan dengan arti solusi pertama yang terpikirkan untuk menyelesaikan persoalan tanpa adanya optimalisasi atau efisiensi dalam cara menyelesaikan masalah. Algoritma *Brute Force* berarti algoritma yang memecahkan persoalan dengan sangat sederhana, langsung, dan jelas.

BAB II
IMPLEMENTASI ALGORITMA

2.1 Struktur Folder

```
TUCIL1_13523026/
├── bin/
│   ├── lib/
│   ├── Main.class
│   └── Solver.class
│
├── doc/
│   └── Tucil1_K1_13523026_Bertha Soliany Frandi.pdf
│
├── src/
│   ├── lib/
│   ├── Board.java
│   ├── Color.java
│   ├── PuzzlePiece.java
│   ├── Main.java
│   └── Solver.java
│
├── test/
│   ├── 1.txt
│   ├── 2.txt
│   ├── …
│   ├── solution_1.txt
│   ├── solution_2.txt
│   └── …
│
└── README.md
```

2.2 File Main.java

File ini berisi program utama IQ Puzzler Pro.

Nama Kelas	Deskripsi
Main	Kelas Main adalah kelas yang merupakan titik awal program untuk IQ Puzzler Pro solver. Kelas ini berfungsi untuk mengatur alur apakah pengguna ingin memulai pencarian solusi atau ingin keluar dari program.

Nama Method	Deskripsi
public static void main(String[] args)	Method utama untuk titik awal program. Menampilkan pilihan ingin memulai program pencarian atau keluar dari program.

2.3 File Solver.java

File ini berisi *driver* untuk melakukan penerimaan input dan pencarian solusi dari IQ Puzzler Pro.

Nama Kelas	Deskripsi
Solver	Kelas yang digunakan untuk driver. Merupakan program yang menyimpan alur utama.

Nama Method	Deskripsi
public static void driver(Scanner sc)	Method yang dipanggil ketika pengguna memutuskan untuk memulai IQ Puzzler Pro solver. Program masih memiliki alur dan belum sepenuhnya melakukan *Object-Oriented Programming* sehingga pada method ini masih terdapat alur sekuensial mulai dari penerimaan input, pembacaan input, pencarian solusi, hingga pemberitahuan hasil.

2.4 File Board.java

File ini berisi kelas untuk papan permainan yang mempunyai atribut dan method untuk algoritma brute force serta pencetakan solusi. Selektor dan konstruktor tidak dimasukkan pada tabel untuk file ini.

Nama Kelas	Deskripsi
Board	Kelas yang merepresentasikan papan permainan dari IQ Puzzler Pro. Mempunyai atribut dan method yang berguna untuk pencarian solusi dan penulisan solusi pada terminal serta penyimpanan ke file .txt.

Nama Method	Deskripsi
public Board(int r, int c, int p)	Menginisialisasi papan dengan tinggi, lebar, dan jumlah blok puzzle sesuai dengan parameter. Selain itu juga

	menginisialisasi setiap elemen, waktu eksekusi, total *attempts*, jenis kasus, dan nama soal.
public void printBoard()	Mencetak papan permainan.
public void printSolution()	Mencetak solusi permainan berupa papan permainan, waktu pencarian, dan banyak percobaan.
public void createNewFile(String fileName)	Membuat file dengan nama fileName jika belum ada.
public void writeFile(String fileName, boolean v)	Menulis solusi permainan ke file.
public void saveSolution(Scanner sc, String fileName, boolean v)	Memberikan prompt kepada pengguna untuk menyimpan solusi ke file .txt.
private int factorial(int n)	Mencari faktorial dari n.
public boolean isFull(int[][] b)	Melakukan pengecekan apakah papan permainan sudah penuh atau belum.
public PuzzlePiece[][] setOrderPieces(PuzzlePiece[] pieces)	Menemukan semua urutan kemungkinan sususan blok puzzle dan menyimpannya ke dalam sebuah array.
public void order(PuzzlePiece[] arr, int k, PuzzlePiece[][] res)	Menemukan urutan susunan blok puzzle dengan konsep permutasi.
public void swap(PuzzlePiece[] arr, int i, int j)	Menukar dua elemen dari sebuah array.
public boolean canSetPiece(int[][] b, int[][] shape, int r, int c)	Melakukan pengecekan bisa/tidaknya sebuah blok puzzle diletakkan pada papan permainan.
public void setPiece(int[][] b, int[][] shape, int r, int c, int id)	Menempatkan blok puzzle ke papan permainan.
public boolean solve()	Menghasilkan semua kemungkinan urutan, membuat board baru setiap kali suatu kombinasi urutan gagal, dan memanggil fungsi aBF untuk pengecekan solusi.
public boolean aBF(int[][] b, PuzzlePiece[] order)	Merupakan fungsi untuk Algoritma *Brute Force*. Melakukan pengecekan satu per satu pada setiap blok puzzle dan setiap kemungkinan rotasi/refleksinya.

2.5 File PuzzlePiece.java

File ini berisi kelas untuk blok puzzle yang mempunyai atribut dan method untuk pembacaan input dan rotasi serta refleksi blok puzzle. Selektor dan konstruktor tidak dimasukkan pada tabel untuk file ini.

Nama Kelas	Deskripsi
PuzzlePiece	Kelas yang merepresentasi blok puzzle. Mempunyai atribut berupa id yang menyimpan huruf blok puzzle dalam ASCII, tinggi blok puzzle, lebar blok puzzle, matriks yang merepresentasikan bentuk blok puzzle, serta boolean untuk pengecekan input valid atau tidak.

Nama Method	Deskripsi
public PuzzlePiece()	Menginisialisasi PuzzlePiece sebagai null.
public PuzzlePiece(int Id, int r, int c, int[][] newShape)	Menginisialisasi PuzzlePiece dengan id, tinggi, lebar, bentuk, dan boolean valid yang bernilai true.
private int[][] rotate(int[][] shapes)	Memutar blok puzzle 90°.
private int[][] reflect(int[][] shapes)	Merefleksikan blok puzzle.
public void printPiece()	Mencetak satu blok puzzle.
public void printPieces()	Mencetak semua blok puzzle.
public PuzzlePiece[] getPieces()	Mencari dan menyimpan ke array semua kemungkinan bentuk blok puzzle ketika di rotasi dan refleksi.
public void readFile(String fileName, Board b)	Melakukan pembacaan pada file .txt yang dimasukkan oleh pengguna. Melakukan penyimpanan pada bentuk awal blok puzzle.
private boolean isRowZero(int[] row)	Melakukan pengecekan apakah baris kosong.
private boolean isColumnZero(int[][] shape, int c, int idxAwal, int idxAkhir)	Melakukan pengecekan apakah kolom kosong.

2.6 File Color.java

File ini berisi kelas *Color* yang berguna untuk memberikan warna pada solusi yang ditampilkan di terminal.

Nama Kelas	Deskripsi
Color	Kelas yang menyimpan atribut dan method untuk melakukan pencetakan solusi dengan warna yang berbeda untuk setiap blok puzzle.

Nama Atribut/Method	Deskripsi
private static String[] colors	Atribut untuk mendefinisikan 26 warna dengan menggunakan *ANSI escape codes*.
private static String RESET	Atribut untuk menyimpan warna *default* berguna untuk reset warna.
public static String getColor(char piece)	Menentukan warna berdasarkan huruf/karakter.
public static String colorPrint(char piece)	Mencetak teks dengan warna yang telah ditentukan.

BAB III
SOURCE CODE PROGRAM

3.1 *Repository* Program

Berikut adalah pranala ke *repository* yang berisi kode program.
https://github.com/BerthaSoliany/Tucil1_13523026.

3.2 Source Code

Source code yang ditampilkan hanyalah kode utama pada program. Fungsi selektor dan konstruktor sederhana tidak ditampilkan.

3.2.1 Main.java

```
import java.util.Scanner;

// javac -d bin -sourcepath src src/Main.java
// java -cp bin Main
public class Main {
    public static void main(String[] args) {
        int pilihan = 0;
        try (Scanner sc = new Scanner(System.in)) {
            while (pilihan != 2) {
                System.out.println("Menu:");
                System.out.println("1. Mulai IQ Puzzler Pro
Solver");
                System.out.println("2. Keluar");
                System.out.print("Pilihan: ");
                pilihan = Integer.parseInt(sc.nextLine());
                // pilihan = sc.nextInt();
                // sc.nextLine();

                while (pilihan<1 || pilihan>2) {
                    System.out.println("Pilihan tidak
valid");
                    System.out.print("Pilihan: ");
                    pilihan =
Integer.parseInt(sc.nextLine());
                    // pilihan = sc.nextInt();
                    // sc.nextLine();
                }
```

```
                switch (pilihan) {
                    case 1 -> Solver.driver(sc);
                    case 2 -> System.out.println("Bye
byee~");
                }
            }
        }
    }
}
```

3.2.2 Solver.java

```
public class Solver {
    public static void driver(Scanner sc) {
        System.out.println("Masukkan nama file test case
(dengan .txt): ");
        String fileName = sc.nextLine();

        while (fileName.isEmpty() ||
!fileName.contains(".txt")) {
            System.out.println("Masukkan nama file test case
(dengan .txt): ");
            fileName = sc.nextLine();
        }

        File file = new File("test\\"+fileName);

        while(!file.exists()) {
            System.out.println("Tidak ada file test case pada
folder test\nMasukkan nama file test case (dengan .txt): ");
            fileName = sc.nextLine();
            file = new File("test\\"+fileName);
        }

        PuzzlePiece p = new PuzzlePiece();
        Board b = new Board(0, 0, 0);
        p.setValid(true);
        b.setFileName(fileName);
```

```
        p.readFile(fileName,b);
        b.setPieces(p.pieces);
        // p.printPieces();

        System.out.println("\n====================\n");

        boolean solved = false;
        if (p.getValid() && b.getKasus().equals("DEFAULT")) {
            long startTime = System.nanoTime();
            solved = b.solve();
            long endTime = System.nanoTime();
            long duration = ((endTime-startTime)/1_000_000);
            b.setTime(duration);
        }

        if (solved) {
            System.out.println("Solusi berhasil ditemukan!");
        } else {
            System.out.println("Tidak ada solusi yang
ditemukan");
            if (!b.getKasus().equals("DEFAULT")) {
                System.out.println("Kasus bukan DEFAULT");
            }
        }
        p.printPieces();
        b.printSolution();
        b.saveSolution(sc, fileName);
    }
}
```

3.2.3 Board.java

Hanya yang berhubungan dengan algoritma *Brute Force*.

```
public class Board {
    int rows, cols, pieceCount, attempts;
    long time;
    int[][] elmnts;
    String kasus, fileName;
```

```
    PuzzlePiece[] pieces;
    private int Idx = 0;

    public Board(int r, int c, int p) {
        rows = r;
        cols = c;
        pieceCount = p;
        elmnts = new int[rows][cols];
        time = 0;
        attempts = 0;
        kasus = "";
        fileName = "";
    }

    private boolean isFull(int[][] board) {
        for (int i=0;i<rows;i++) {
            for (int j=0;j<cols;j++) {
                if (board[i][j]==0) {
                    return false;
                }
            }
        }
        return true;
    }

    // mencari urutan piece akan ditempatkan
    private int factorial(int n) {
        int res = 1;
        for (int i=2;i<=n;i++) {
            res*=i;
        }
        return res;
    }

    private PuzzlePiece[][] setOrderPieces(PuzzlePiece[]
pieces) {
        int total = factorial(pieces.length);
        PuzzlePiece[][] newPieces = new
PuzzlePiece[total][pieces.length];
```

```
        Idx = 0;
        order(pieces, 0, newPieces);

        return newPieces;
    }

    private void order(PuzzlePiece[] arr, int k,
PuzzlePiece[][] res) {
        if (k == arr.length) {
            res[Idx] = arr.clone();
            Idx++;
        } else {
            for (int i=k;i<arr.length;i++) {
                swap(arr, i, k);
                order(arr, k + 1, res);
                swap(arr, i, k);
            }
        }
    }

    private void swap(PuzzlePiece[] arr, int i, int j) {
        PuzzlePiece temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
    }

    public boolean canSetPiece(int[][] board, int[][] shape,
int r, int c) {
        for (int i=0;i<shape.length;i++) {
            for (int j=0;j<shape[i].length;j++) {
                if (shape[i][j]!=0 && board[r+i][c+j]!=0) {
                    return false;
                }
            }
        }
        return true;
    }
```

```
    public void setPiece(int[][] board, int[][] shape, int r,
int c, int id) {
        for (int i=0;i<shape.length;i++) {
            for (int j=0;j<shape[i].length;j++) {
                if (shape[i][j]!=0) {
                    board[r+i][c+j]=id;
                }
            }
        }
    }

    public boolean solve() {
        PuzzlePiece[][] orders = setOrderPieces(pieces);

        for (int i=0;i<orders.length;i++) {
            PuzzlePiece[] order = orders[i];
            int[][] copy = new int[rows][cols];

            if (aBF(copy, order)) {
                elmnts = copy;
                return true;
            }
        }
        return false;
    }

    // inti algoritma brute force
    private boolean aBF(int[][] board, PuzzlePiece[] order) {
        for (int n=0;n<order.length;n++) {
            PuzzlePiece piece = order[n];
            boolean placed = false;
            PuzzlePiece[] rotations = piece.getPieces();

            for (int i = 0; i < rotations.length; i++) {
                PuzzlePiece rotated = rotations[i];
                for (int j = 0; j <= rows -
rotated.shape.length; j++) {
                    for (int k = 0; k <= cols -
rotated.shape[0].length; k++) {
```

```
                        attempts++;
                        if (canSetPiece(board, rotated.shape,
j, k)) {
                            setPiece(board, rotated.shape, j,
k, piece.id);
                            placed = true;
                            break;
                        }
                    }
                    if (placed) break;
                }
                if (placed) break;
            }

            if (!placed) return false;
        }
        return isFull(board);
    }

}
```

Algoritma *Brute Force* terdapat pada fungsi solve() dan fungsi aBF(int[][] board, PuzzlePiece[] order). Fungsi solve() adalah fungsi yang dipanggil pada driver Solver. Fungsi tersebut dimulai dari mencari semua urutan penempatan blok puzzle dengan memanggil fungsi setOrderPieces(pieces). Hal ini dilakukan agar semua kemungkinan penempatan bisa terjadi. Blok puzzle yang sudah memiliki urutan tersebut dimasukkan ke perulangan (*loop*) dan mencari solusi dengan memanggil fungsi aBF. Pada perulangan ini akan dibuat dibuat board kosong. Solusi ditemukan apabila salah satu kombinasi yang di cek dengan fungsi aBF bernilai true.

Pada fungsi aBF, fungsi diawali dengan mencari/mengambil semua kemungkinan bentuk blok puzzle (rotasi dan refleksi dari blok puzzle input). Setelah itu, dilakukan pengecekan untuk semua kemungkinan untuk satu blok puzzle tersebut. Fungsi akan mencoba meletakkan blok puzzle di setiap posisi yang memungkinan pada papan permainan. Pengecekan peletakkan ini dilakukan dengan memanggil fungsi canSetPiece. Fungsi tersebut berguna untuk melihat apakah terjadi penimpaan dalam penempatan blok puzzle. Tentunya apabila blok puzzle tidak bisa ditempatkan menurut fungsi canSetPiece, rotasi blok puzzle lain akan dicoba. Fungsi aBF akan menghasilkan true apabila semua blok puzzle telah

dicoba dan papan permainan penuh. Akan menghasilkan false apabila ketika mencapai kombinasi urutan terakhir yang didapatkan dari solve namun papan permainan belum terisi penuh atau masih ada blok puzzle yang terpisah.

3.2.4 PuzzlePiece.java

```
public class PuzzlePiece {
    int id, row, col;
    boolean valid;
    int[][] shape;
    public PuzzlePiece[] pieces;

    public PuzzlePiece() {
        this.pieces = null;
    }

    public PuzzlePiece(int Id, int r, int c, int[][]
newShape) {
        id = Id;
        row = r;
        col = c;
        shape = new int[row][col];
        for (int i=0;i<row;i++) {
            System.arraycopy(newShape[i], 0, shape[i], 0,
newShape[i].length);
        }
        valid = true;
    }

    private int[][] rotate(int[][] shapes) {
        int r = shapes.length;
        int c = shapes[0].length;
        int[][] rotated = new int[c][r];
        for (int i=0;i<r;i++) {
            for (int j=0;j<c;j++) {
                rotated[j][r-1-i] = shapes[i][j];
            }
        }
        return rotated;
    }
```

```
    private int[][] reflect(int[][] shapes) {
        int r = shapes.length;
        int c = shapes[0].length;
        int[][] flipped = new int[r][c];
        for (int i=0;i<r;i++) {
            for (int j=0;j<c;j++) {
                flipped[i][c-1-j] = shapes[i][j];
            }
        }
        return flipped;
    }

    public PuzzlePiece[] getPieces() {
        PuzzlePiece[] newPieces = new PuzzlePiece[8];
        int count = 0;
        int[][] shape1 = shape;
        int[][] shape2 = reflect(shape);

        for (int i=0;i<4;i++) {
            newPieces[count++] = new PuzzlePiece(id,
shape1.length, shape1[0].length, shape1);
            shape1 = rotate(shape1);
        }
        for (int i=0;i<4;i++) {
            newPieces[count++] = new PuzzlePiece(id,
shape2.length, shape2[0].length, shape2);
            shape2 = rotate(shape2);
        }
        return newPieces;
    }

    public void readFile(String fileName, Board b) {
        fileName = "test\\" + fileName;
        try {
            int N,M,P;
            BufferedReader br = new BufferedReader(new
FileReader(fileName));
            String firsString = br.readLine();
```

```
            StringTokenizer st = new
StringTokenizer(firsString);

            if
(firsString==null||firsString.trim().isEmpty()||st.countToken
s()<3||st.countTokens()>3) {
                br.close();
                setValid(false);
                throw new IOException("Format file N M P
tidak valid");
            }

            try {
                N = Integer.parseInt(st.nextToken());
                M = Integer.parseInt(st.nextToken());
                P = Integer.parseInt(st.nextToken());
            } catch (NumberFormatException e) {
                br.close();
                setValid(false);
                throw new IOException("Format salah: N, M,
dan P harus berupa angka.");
            }
            System.out.println("N:"+N+" M:"+M+" P:"+P);

            if (N<=0||M<=0){
                br.close();
                setValid(false);
                throw new IOException("Dimensi papan tidak
valid.");
            }

            String kasus = br.readLine();
            System.out.println("Kasus:"+kasus);

            // set board
            b.setHeights(N);
            b.setWidths(M);
            b.setPieceCount(P);
            b.setKasus(kasus);
```

```
            // cek baris file
            int max = (P+N+2)*10;
            String[] lines = new String[max];
            int count = 0;
            String line;

            while ((line = br.readLine()) != null) {
                if (line.trim().isEmpty()) {
                    lines[count++] = " ";
                } else {
                    lines[count++] = line;
                }
            }

            String[] fixLines = new String[count];
            System.arraycopy(lines, 0, fixLines, 0, count);
            lines = fixLines;

            int r = lines.length;
            int c = 0;
            for (int i = 0; i < r; i++) {
                if (lines[i].length() > c) {
                    c = lines[i].length();
                }
            }

            // menemukan banyak pieces
            boolean[] foundChars = new boolean[256];
            int idCount = 0;
            for (int i=0; i<r;i++) {
                for (int j=0;j<lines[i].length();j++) {
                    char huruf = lines[i].charAt(j);
                    if (huruf>='A' && huruf<='Z' &&
!foundChars[huruf]) {
                        foundChars[huruf] = true;
                        idCount++;
                    }
                }
```

```
            }

            char[] id = new char[idCount];
            int index = 0;
            for (int i=0;i<256;i++) {
                if (foundChars[i]) {
                    id[index++] = (char) i;
                }
            }

            // inisialiasi untuk semua pieces
            pieces = new PuzzlePiece[idCount];

            for (int i=0;i<idCount;i++) {
                char huruf = id[i];
                int[][] shape = new int[r][c];

                for (int j=0;j<r;j++) {
                    char[] chars = lines[j].toCharArray();
                    for (int k=0;k<chars.length;k++) {
                        if (chars[k] == huruf) {
                            shape[j][k] = huruf;
                        }
                    }
                }

                // hapus row/col yg 0 semua
                int idxAwal = 0, idxAkhir = r-1;
                int idxKiri = 0, idxKanan = c-1;

                while (idxAwal<r&& isRowZero(shape[idxAwal]))
idxAwal++;
                while (idxAkhir>=0 &&
isRowZero(shape[idxAkhir])) idxAkhir--;
                while (idxKiri<c && isColumnZero(shape,
idxKiri, idxAwal, idxAkhir)) idxKiri++;
                while (idxKanan>=0 && isColumnZero(shape,
idxKanan, idxAwal, idxAkhir)) idxKanan--;
```

```
                if (idxAwal>idxAkhir||idxKiri>idxKanan) {
                    pieces[i] = new PuzzlePiece(huruf, 0, 0,
new int[0][0]);
                    continue;
                }

                int[][] newShape = new
int[idxAkhir-idxAwal+1][idxKanan-idxKiri+1];
                for (int j = idxAwal; j <= idxAkhir; j++) {
                    System.arraycopy(shape[j], idxKiri,
newShape[j-idxAwal], 0, newShape[0].length);
                }
                // simpan shape ke public pieces
                pieces[i] = new PuzzlePiece(huruf,
newShape.length, newShape[0].length, newShape);
            }

            br.close();

            if (pieces.length != P) {
                setValid(false);
                throw new IOException("Jumlah blok puzzle
tidak sesuai dengan P.");
            }
        } catch (IOException e) {
            System.out.println("Terjadi kesalahan dalam
membaca file.");
            setValid(false);
            e.printStackTrace();
        }
    }

    private boolean isRowZero(int[] row) {
        for (int i=0;i<row.length;i++) {
            int num = row[i];
            if (num!=0) {
                return false;
            }
        }
```

```
        return true;
    }

    private boolean isColumnZero(int[][] shape, int c, int
idxAwal, int idxAkhir) {
        for (int i=idxAwal;i<=idxAkhir;i++) {
            if (shape[i][c]!=0) {
                return false;
            }
        }
        return true;
    }
}
```

3.2.5 Color.java

```
package lib;

public class Color {
    private static String[] colors = {
        "\u001B[30m",
        "\u001B[31m",
        "\u001B[32m",
        "\u001B[33m",
        "\u001B[34m",
        "\u001B[35m",
        "\u001B[36m",
        "\u001B[91m",
        "\u001B[92m",
        "\u001B[93m",
        "\u001B[94m",
        "\u001B[95m",
        "\u001B[96m",
        "\u001B[97m",
        "\u001B[90m",
        "\u001B[41m",
        "\u001B[42m",
        "\u001B[43m",
        "\u001B[44m",
```

```
        "\u001B[45m",
        "\u001B[46m",
        "\u001B[101m",
        "\u001B[102m",
        "\u001B[103m",
        "\u001B[104m",
        "\u001B[105m"
    };

    private static String RESET = "\u001B[0m";

    public static String getColor(char piece) {
        int idx = (piece-'A') % colors.length;
        return colors[idx];
    }

    public static String colorPrint(char piece) {
        return getColor(piece) + piece + RESET;
    }

}
```

BAB IV

TEST CASE

4.1 Test Case 1

File Input	test > 1.txt 2 2 2 DEFAULT A AA B
Terminal	Masukkan nama file test case (dengan .txt): 1.txt N:2 M:2 P:2 Kasus:DEFAULT ========== Solusi berhasil ditemukan! Piece A A . A A Piece B B Dimensi papan: 2x2 Jumlah blok puzzle: 2 A B A A Waktu pencarian: 0ms Banyak kasus yang ditinjau: 3 Apakah anda ingin menyimpan solusi? (ya/tidak)
File Output	test > solution_1.txt Solusi 1.txt A B A A Waktu pencarian: 0ms Banyak kasus yang ditinjau: 3

4.2 Test Case 2: Input selain A-Z

File Input

test > 2.txt

```
4 5 5
DEFAULT
AA
BBB
BBB
C
C
CC
D
DDD1
DD
EE
#
3
o
```

Terminal

```
Masukkan nama file test case (dengan .txt):
2.txt
N:4 M:5 P:5
Kasus:DEFAULT

====================

Solusi berhasil ditemukan!
Piece A
A A
Piece B
B B B
B B B
Piece C
C .
C .
C C
Piece D
D . .
D D D
D D .
Piece E
E E
Dimensi papan: 4x5
Jumlah blok puzzle: 5
A A D E E
B B D D D
B B D D C
B B C C C
Waktu pencarian: 0ms

Banyak kasus yang ditinjau: 1253
```

File Output

test > solution_2.txt

```
Solusi 2.txt
A A D E E
B B D D D
B B D D C
B B C C C
Waktu pencarian: 0ms
Banyak kasus yang ditinjau: 1253
```

4.3 Test Case 3: Jenis kasus yang diberikan bukan DEFAULT

File Input	test > 3.txt 1 2 2 2 2 CUSTOM 3 AA 4 BB
Terminal	Masukkan nama file test case (dengan .txt): 3.txt N:2 M:2 P:2 Kasus:CUSTOM ========== Tidak ada solusi yang ditemukan Kasus bukan DEFAULT Piece A A A Piece B B B Dimensi papan: 2x2 Jumlah blok puzzle: 2 0 0 0 0 Waktu pencarian: 0ms Banyak kasus yang ditinjau: 0 Apakah anda ingin menyimpan solusi? (ya/tidak)
File Output	test > solution_3.txt 1 Solusi 3.txt 2 0 0 3 0 0 4 Waktu pencarian: 0ms 5 Banyak kasus yang ditinjau: 0

4.4 Test Case 4: Blok puzzle terpisah 1

File Input

test > 4.txt

```
3 3 2
DEFAULT
A
A
A
BB
BB

BB
```

Terminal

```
Masukkan nama file test case (dengan .txt):
4.txt
N:3 M:3 P:2
Kasus:DEFAULT

==================

Tidak ada solusi yang ditemukan
Piece A
A
A
A
Piece B
B B
B B
. .
B B
Dimensi papan: 3x3
Jumlah blok puzzle: 2
0 0 0
0 0 0
0 0 0
Waktu pencarian: 0ms

Banyak kasus yang ditinjau: 1

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

File Output

test > solution_4.txt

```
solusi 4.txt
0 0 0
0 0 0
0 0 0
Waktu pencarian: 0ms
Banyak kasus yang ditinjau: 1
```

4.5 Test Case 5: Blok puzzle terpisah 2

File Input

test > 7.txt

```
6 6 8
DEFAULT
AAA
A AA
AAA
BB
BB
B
C
DD
D
FFFF
   FF
GGGG
GG
HHH
H


II
```

Terminal

```
Masukkan nama file test case (dengan .txt):
7.txt
N:6 M:6 P:8
Kasus:DEFAULT

==================

Solusi berhasil ditemukan!
Piece A
A A A .
A . A A
A A A .
Piece B
B B
B B
B .
Piece C
C
Piece D
D D
D .
Piece F
F F F F .
. . . F F
Piece G
G G G G
G G . .
Piece H
H H H
H . .
Piece I
I I
Dimensi papan: 6x6
Jumlah blok puzzle: 8
F F F F D D
A A A F F D
A C A A I I
A A A H H H
B B B H G G
B B G G G G
Waktu pencarian: 427ms

Banyak kasus yang ditinjau: 4216057
```

File Output

test > solution_7.txt

```
Solusi 7.txt
F F F F D D
A A A F F D
A C A A I I
A A A H H H
B B B H G G
B B G G G G
Waktu pencarian: 465ms
Banyak kasus yang ditinjau: 4216057
```

4.6 Test Case 6: Jumlah blok puzzle tidak sesuai dengan nilai P

File Input

test > 5.txt

```
5 4 2
DEFAULT
AAAA
AAAA
A AA
AAAA
AAAA
B
B
C
```

Terminal

```
Masukkan nama file test case (dengan .txt):
5.txt
N:5 M:4 P:2
Kasus:DEFAULT
Terjadi kesalahan dalam membaca file.
java.io.IOException: Jumlah blok puzzle tidak sesuai dengan P.
        at lib.PuzzlePiece.readFile(PuzzlePiece.java:272)
        at Solver.driver(Solver.java:28)
        at Main.main(Main.java:27)

==================

Tidak ada solusi yang ditemukan
Piece A
A A A A
A A A A
A . A A
A A A A
A A A A
Piece B
B
B
Piece C
C
Dimensi papan: 5x4
Jumlah blok puzzle: 2
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
Waktu pencarian: 0ms

Banyak kasus yang ditinjau: 0

Apakah anda ingin menyimpan solusi? (ya/tidak)

File test\solution_5.txt sudah ada. Overwrite file
Solusi berhasil di simpan pada test\solution_5.txt
```

File Output

test > solution_5.txt

```
Solusi 5.txt
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
Waktu pencarian: 0ms
Banyak kasus yang ditinjau: 0
```

4.7 Test Case 7: Blok puzzle sisa

File Input

test > 6.txt

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

Terminal

```
Masukkan nama file test case (dengan .txt)
6.txt
N:5 M:5 P:8
Kasus:DEFAULT

====================

Tidak ada solusi yang ditemukan
Piece A
A .
A A
Piece B
B .
B B
Piece C
C .
C C
Piece D
D .
D D
Piece E
E E
E E
E .
Piece F
F F
F F
F .
Piece G
G G G
Piece H
H
Dimensi papan: 5x5
Jumlah blok puzzle: 8
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
Waktu pencarian: 686ms

Banyak kasus yang ditinjau: 6843072
```

File Output

test > solution_6.txt

```
Solusi 6.txt
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
Waktu pencarian: 686ms
Banyak kasus yang ditinjau: 6843072
```

LAMPIRAN

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

References

- https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/Tucil1-Stima-2025.pdf
- https://github.com/antonpinchuk/iq-puzzler-pro-solution
- https://github.com/marhoy/iq-puzzler-pro-solver/tree/master
- https://howtodoinjava.com/java/date-time/execution-elapsed-time/
- https://stackoverflow.com/questions/5762491/how-to-print-color-in-console-using-system-out-println
- https://codehs.com/tutorial/ryan/add-color-with-ansi-in-javascript

DAFTAR PUSTAKA

SmartGames, “IQ Puzzler Pro,” [Online]. Available: https://www.smartgames.eu/uk/one-player-games/iq-puzzler-pro-0. [Accessed: 24-Feb-2025].

R. Munir, “Algoritma Brute Force,” Institut Teknologi Bandung, 2016. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Algoritma-Brute-Force-(2016).pdf. [Accessed: 24-Feb-2025].