Laporan Tugas Kecil IF2211 Strategi Algoritma

Menggunakan Bahasa Pemrograman Java

Oleh

Muhammad Rizain Firdaus

13523164

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

02/2025

DAFTAR ISI

PENDAHULUAN

IQ Puzzler Pro adalah permainan logika yang menantang pemain untuk menyusun balok-balok dengan berbagai bentuk agar memenuhi papan permainan tanpa ada ruang kosong. Permainan ini menuntut kemampuan berpikir strategis dan pemecahan masalah yang sistematis. Dalam rangka memahami dan mengimplementasikan algoritma pencarian solusi secara optimal, proyek ini dibuat sebagai bagian dari mata kuliah Strategi Algoritma (IF2211) di bawah bimbingan Pak Monterico Adrian S.T., M.T.

Tugas ini bertujuan untuk membangun sebuah solver otomatis untuk IQ Puzzler Pro menggunakan algoritma Brute Force dengan Backtracking. Solver ini dikembangkan dalam bahasa pemrograman Java dan mampu menangani berbagai bentuk permainan dengan fleksibilitas tinggi, termasuk rotasi serta pencerminan balok untuk mencari solusi optimal.

Dalam laporan ini, akan dibahas secara rinci mengenai konsep algoritma yang digunakan, struktur program, serta hasil uji coba dengan berbagai skenario permainan. Program ini tidak hanya berfokus pada efisiensi pencarian solusi, tetapi juga mengakomodasi fitur tambahan seperti penyimpanan hasil dalam format teks dan visualisasi warna pada solusi yang ditemukan.

Diharapkan dengan adanya tugas ini, pemahaman mengenai strategi algoritma pencarian solusi, optimasi, serta implementasi dalam pemrograman dapat semakin diperdalam dan diaplikasikan dalam berbagai permasalahan komputasi yang lebih kompleks.

BAB 1: Program

1. Deskripsi Program

1.1. Penjelasan Singkat

IQ Puzzler Pro adalah permainan papan yang diproduksi oleh perusahaan Smart Games. Tujuan dari permainan ini adalah pemain harus dapat mengisi seluruh papan dengan piece (blok puzzle) yang telah tersedia. Komponen penting dari permainan IQ Puzzler Pro terdiri dari:

1. Board (Papan) – Board merupakan komponen utama yang menjadi tujuan permainan dimana pemain harus mampu mengisi seluruh area papan menggunakan blok-blok yang telah disediakan.

2. Blok/Piece – Blok adalah komponen yang digunakan pemain untuk mengisi papan kosong hingga terisi penuh. Setiap blok memiliki bentuk yang unik dan semua blok harus digunakan untuk menyelesaikan puzzle.

Permainan dimulai dengan papan yang kosong. Pemain dapat meletakkan blok puzzle sedemikian sehingga tidak ada blok yang bertumpang tindih (kecuali dalam kasus 3D). Setiap blok puzzle dapat dirotasikan maupun dicerminkan. Puzzle dinyatakan selesai jika dan hanya jika papan terisi penuh dan seluruh blok puzzle berhasil diletakkan. Tugas anda adalah menemukan cukup satu solusi dari permainan IQ Puzzler Pro dengan menggunakan algoritma Brute Force, atau menampilkan bahwa solusi tidak ditemukan jika tidak ada solusi yang mungkin dari puzzle.

Gambar 1.1. Puzzler Pro

2. Source Code

2.1. Struktur Folder dan File

Mengenai source code yang telah dibuat adapun dibuat dan disusun menggunakan bahasa pemrograman Java dengan menekankan algoritma brute force tanpa bersifat heuristik dan memanfaatkan sifat backtracking untuk mendapatkan satu solusi puzzle yang diminta. Adapun program yang dibuat memiliki struktur folder dan file sebagai berikut:

Adapun tiap kelas yang dibuat telah disusun dalam sebuah file yang berbeda.

2.2. Struktur Kelas

2.2.1. Board

```
CLASS Board

    PRIVATE grid: 2D ARRAY OF CHAR

    PRIVATE rows: INTEGER

    PRIVATE cols: INTEGER

    PRIVATE emptyPoints: LIST OF Point


    CONSTRUCTOR Board(rows, cols, grid)

        SET this.rows = rows

        SET this.cols = cols

        SET this.grid = grid

        SET this.emptyPoints = NEW LIST OF Point


        FOR i FROM 0 TO rows - 1

            FOR j FROM 0 TO cols - 1

                IF grid[i][j] == '.'

                    ADD NEW Point(j, i) TO emptyPoints

                END IF

            END FOR

        END FOR

    END CONSTRUCTOR


    FUNCTION isFull() RETURNS BOOLEAN

        FOR i FROM 0 TO rows - 1

            FOR j FROM 0 TO cols - 1

                IF grid[i][j] == '.'

                    RETURN FALSE

                END IF

            END FOR

        END FOR

        RETURN TRUE

    END FUNCTION


    FUNCTION canPlacePiece(piece, x, y) RETURNS BOOLEAN
```

```
        SET piecePoints = piece.getPointsAt(x, y)

        FOR EACH point IN piecePoints

            IF point.coordX >= cols OR point.coordY >= rows OR
point.coordX < 0 OR point.coordY < 0 OR grid[point.coordY][point.coordX]
!= '.'

                RETURN FALSE

            END IF

        END FOR

        RETURN TRUE

    END FUNCTION


    FUNCTION placePiece(piece, startX, startY)

        SET piecePoints = piece.getPointsAt(startX, startY)

        FOR EACH point IN piecePoints

            SET grid[point.coordY][point.coordX] = piece.getId()

            REMOVE point FROM emptyPoints

        END FOR

    END FUNCTION


    FUNCTION removePiece(piece, startX, startY)

        SET piecePoints = piece.getPointsAt(startX, startY)

        FOR EACH point IN piecePoints

            SET grid[point.coordY][point.coordX] = '.'

            ADD NEW Point(point.coordX, point.coordY) TO emptyPoints

        END FOR

    END FUNCTION


    FUNCTION boardToString() RETURNS STRING

        SET sb = NEW STRING BUILDER

        FOR i FROM 0 TO rows - 1

            FOR j FROM 0 TO cols - 1

                APPEND grid[i][j] TO sb

            END FOR

            APPEND NEWLINE TO sb
```

```
        END FOR
        RETURN sb AS STRING
    END FUNCTION

    FUNCTION boardToColoredString(colorHandler) RETURNS STRING
        SET sb = NEW STRING BUILDER
        FOR i FROM 0 TO rows - 1
            FOR j FROM 0 TO cols - 1
                APPEND colorHandler.colorize(grid[i][j]) TO sb
            END FOR
            APPEND NEWLINE TO sb
        END FOR
        RETURN sb AS STRING
    END FUNCTION

    FUNCTION getRows() RETURNS INTEGER
        RETURN rows
    END FUNCTION

    FUNCTION getCols() RETURNS INTEGER
        RETURN cols
    END FUNCTION

    FUNCTION getEmptyPoints() RETURNS LIST OF Point
        RETURN emptyPoints
    END FUNCTION
END CLASS
```

Tabel 2.1. Pseudocode kelas Board

Penjelasan dari kelas Board:

- Program menginisialisasikan papan kosong dengan sebuah array 2D berupa atribut `grid` dan mendefinisikan bagian yang kosong dengan “.”. Lalu pada kelas ini terdapat atribut lain seperti `rows`, `cols`, dan `emptyPoints`. Atribut-atribut tersebut mendefinisikan

papan kosong tersebut dimulai dari `rows` yang merepresentasikan besar baris pada sebuah papan dalam atribut grid, lalu `cols` yang merepresentasikan besar kolom dalam atribut `grid`, dan `emptyPoints` sebagai atribut yang membuat sebuah list berupa point untuk mendaftarkan bagian yang kosong dari papan.

- Konstruktor menginisialisasikan papan dengan memanfaatkan atribut kelas dengan menggunakan looping untuk mendaftarkan elemen-elemennya pada sel-sel yang kosong, jika masih ada sel yang tersisa masih kosong maka didaftarkan pada list di `emptyPoints`.
- Kemudian ada beberapa metode pada kelas seperti: isFull (untuk memeriksa apakah papan sudah penuh atau tidak ada sel kosong yang tersisa), `canPlacePiece` (untuk memeriksa apakah piece dapat dipasang pada (x,y) di papan), `placePiece` (memasang piece pada posisi (startX, startY) di papan), `removePiece` (menghapus piece pada posisi (startX, startY) dari papan), `boardToString` (mengkonversikan papan menjadi string untuk ditampilkan), metode `boardToColoredString` (mengkonversi papan menjadi sebuah string yang berwarna menggunakan kelas ColorHandler), dan metode untuk mengambil atribut kelas seperti `getRows`, `getCols`, dan `getEmptyPoints`.

2.2.2. Piece

```
CLASS Piece
    PRIVATE points: LIST OF Point
    PRIVATE id: CHAR

    CONSTRUCTOR Piece(points, id)
        SET this.points = points
        SET this.id = id
    END CONSTRUCTOR

    FUNCTION linesToPiece(lines, id) RETURNS Piece
        SET points = NEW LIST OF Point
        FOR y FROM 0 TO lines.size() - 1
            SET line = lines.get(y)
            FOR x FROM 0 TO line.length() - 1
                IF line.charAt(x) == id
                    ADD NEW Point(x, y) TO points
                END IF
            END FOR
        END FOR
        RETURN NEW Piece(points, id)
    END FUNCTION

    FUNCTION rotate() RETURNS Piece
        SET rotatedPoint = NEW LIST OF Point
        FOR EACH p IN points
            ADD NEW Point(-p.coordY, p.coordX) TO rotatedPoint
        END FOR
        RETURN NEW Piece(rotatedPoint, id)
    END FUNCTION

    FUNCTION mirror() RETURNS Piece
        SET mirroredPoint = NEW LIST OF Point
        FOR EACH p IN points
```

```
            ADD NEW Point(-p.coordX, p.coordY) TO mirroredPoint

        END FOR

        RETURN NEW Piece(mirroredPoint, id)

    END FUNCTION



    FUNCTION getPointsAt(a, b) RETURNS LIST OF Point

        RETURN points.stream().map(p -> p.shift(a,
b)).collect(Collectors.toList())

    END FUNCTION



    FUNCTION getId() RETURNS CHAR

        RETURN id

    END FUNCTION

END CLASS
```

Tabel 2.2. Pseudocode kelas Piece

Penjelasan dari kelas Piece:

- Pada kelas ini terdapat atribut `shape` yang menginisialisasikan titik-titik yang membentuk blok, lalu ada atribut `identifier` yang berupa char untuk mengidentifikasikan blok pada papan.
- Lalu terdapat konstruktor `Piece` untuk menginisialisasikan objek piece yang terdiri atas daftar titik dan ID yang diberikan.
- Terdapat beberapa metode pada kelas ini `linesToPiece` (membuat objek Piece menjadi sebuah susunan daftar string yang merepresentasikan piece), `rotate` (metode untuk memutar piece 90 derajat), `mirror` (metode untuk mencerminkan piece terhadap sumbu Y), `getPointsAt` (Mengembalikan daftar titik-titik blok yang sudah digeser ke posisi (a, b)), dan getId (mengembalikan ID blok).

2.2.3. Point

```
CLASS Point

    PUBLIC coordX: INTEGER

    PUBLIC coordY: INTEGER


    CONSTRUCTOR Point(coordX, coordY)

        SET this.coordX = coordX

        SET this.coordY = coordY

    END CONSTRUCTOR


    FUNCTION shift(deltaX, deltaY) RETURNS Point

        RETURN NEW Point(this.coordX + deltaX, this.coordY + deltaY)

    END FUNCTION


    FUNCTION equals(obj) RETURNS BOOLEAN

        IF this == obj

            RETURN TRUE

        END IF

        IF obj == NULL OR getClass() != obj.getClass()

            RETURN FALSE

        END IF

        SET other = (Point) obj

        RETURN this.coordX == other.coordX AND this.coordY ==
other.coordY

    END FUNCTION


    FUNCTION hashCode() RETURNS INTEGER

        RETURN 31 * this.coordX + this.coordY

    END FUNCTION

END CLASS
```

Tabel 2.3. Pseudocode kelas Point

Penjelasan dari kelas Point:

- Terdapat atribut koordinat berupa x dan y.
- Inisialisasi objek Point dengan koordinat (x, y) yang diberikan.

- Terdapat metode translate (menggeser titik sejauh (dx, dy) dan mengembalikan titik baru yang sudah digeser), `equals` (membandingkan dua objek Point untuk menentukan apakah mereka sama), dan `hashCode` (menghasilkan kode hash untuk objek Point. Kode hash ini digunakan dalam struktur data seperti HashMap atau HashSet).

2.2.4. ColorPack

```
CLASS ColorPack

    PRIVATE CONSTANT COLOR_PACK: ARRAY OF STRING = [

        "\u001B[31m",       // Red

        "\u001B[32m",       // Green

        "\u001B[33m",       // Yellow

        "\u001B[34m",       // Blue

        "\u001B[35m",       // Magenta

        "\u001B[36m",       // Cyan

        "\u001B[91m",       // Bright Red

        "\u001B[92m",       // Bright Green

        "\u001B[93m",       // Bright Yellow

        "\u001B[94m",       // Bright Blue

        "\u001B[95m",       // Bright Magenta

        "\u001B[96m",       // Bright Cyan

        "\u001B[31;1m",     // Bold Red

        "\u001B[32;1m",     // Bold Green

        "\u001B[33;1m",     // Bold Yellow

        "\u001B[34;1m",     // Bold Blue

        "\u001B[35;1m",     // Bold Magenta

        "\u001B[36;1m",     // Bold Cyan

        "\u001B[31;2m",     // Dim Red

        "\u001B[32;2m",     // Dim Green

        "\u001B[33;2m",     // Dim Yellow

        "\u001B[34;2m",     // Dim Blue

        "\u001B[35;2m",     // Dim Magenta

        "\u001B[36;2m",     // Dim Cyan

        "\u001B[37m",      // White

        "\u001B[90m"        // Gray

    ]

    PRIVATE CONSTANT RESET: STRING = "\u001B[0m"


    FUNCTION colorize(pieceId: CHAR) RETURNS STRING

        IF pieceId == '.'
```

```
            RETURN STRING VALUE OF pieceId
        IF pieceId == ' '
            RETURN " "
        RETURN COLOR_PACK[pieceId - 'A'] + pieceId + RESET
    END FUNCTION
END CLASS
```

Tabel 2.4. Pseudocode kelas ColorPack

Penjelasan dari kelas ColorPack:

- Atribut dari kelas ini yakni berupa `COLOR_PACK` berupa sebuah array yang berisikan kode warna. Berdasarkan namanya kode warna yang digunakan menggunakan ANSI.
- Metode `colorize` (sebuah metode untuk memberikan warna pada ID piece yang diberikan) dengan memanfaatkan pieceId untuk menggunakan basis dari list warna yang sudah dibuat.

2.2.5. Input

```
CLASS Input
    PUBLIC rows: INTEGER
    PUBLIC cols: INTEGER
    PUBLIC numPieces: INTEGER
    PUBLIC mode: STRING
    PUBLIC pieces: LIST OF Piece
    PUBLIC grid: 2D ARRAY OF CHAR

    CONSTRUCTOR Input(rows, cols, numPieces, mode, pieces, grid)
        SET this.rows = rows
        SET this.cols = cols
        SET this.numPieces = numPieces
        SET this.mode = mode
        SET this.pieces = pieces
        SET this.grid = grid
    END CONSTRUCTOR
END CLASS
```

Tabel 2.5. Pseudocode kelas Input

Penjelasan dari kelas Input:

- Menginisialisasikan atribut papan untuk `rows` (jumlah baris), `cols` (jumlah kolom), `numPieces` (jumlah balok), mode (mode permainan), `pieces` (daftar pieces yang akan digunakan untuk menyelesaikan puzzle), dan `grid` (pada kondisi awal papan kosong berisikan “.”).
- Konstruktor Input berfungsi untuk menginisialisasikan objek Input dengan atribut kelas.

2.2.6. Solver

```
CLASS Solver

    PRIVATE board: Board

    PRIVATE pieces: LIST OF Piece

    PRIVATE cases: LONG

    PRIVATE startTime: LONG

    PRIVATE colorHandler: ColorPack


    CONSTRUCTOR Solver(rows, cols, pieces, initGrid)

        SET this.board = NEW Board(rows, cols, initGrid)

        SET this.pieces = pieces

        SET this.cases = 0

        SET this.colorHandler = NEW ColorPack()

    END CONSTRUCTOR


    FUNCTION solve() RETURNS BOOLEAN

        SET startTime = CURRENT TIME IN MILLISECONDS

        RETURN solvePuzzle(0)

    END FUNCTION


    FUNCTION solvePuzzle(pieceIndex: INTEGER) RETURNS BOOLEAN

        IF pieceIndex >= pieces.size()

            RETURN board.isFull()

        END IF


        SET piece = pieces.get(pieceIndex)

        SET currentPiece = piece

        SET positions = NEW LIST OF Point(board.getEmptyPoints())


        FOR i FROM 0 TO 1

            FOR r FROM 0 TO 3

                FOR EACH p IN positions

                    INCREMENT cases BY 1
```

```
                    IF board.canPlacePiece(currentPiece, p.coordX,
p.coordY)
                        board.placePiece(currentPiece, p.coordX,
p.coordY)
                        IF solvePuzzle(pieceIndex + 1)
                            RETURN TRUE
                        END IF
                        board.removePiece(currentPiece, p.coordX,
p.coordY)
                    END IF
                END FOR
                SET currentPiece = currentPiece.rotate()
            END FOR
            SET currentPiece = piece.mirror()
        END FOR

        RETURN FALSE
    END FUNCTION

    FUNCTION getExecutionTime() RETURNS LONG
        RETURN CURRENT TIME IN MILLISECONDS - startTime
    END FUNCTION

    FUNCTION getCases() RETURNS LONG
        RETURN cases
    END FUNCTION

    FUNCTION getBoard() RETURNS STRING
        RETURN board.boardToString()
    END FUNCTION

    FUNCTION getColoredBoard() RETURNS STRING
        RETURN board.boardToColoredString(colorHandler)
    END FUNCTION
```

```
END CLASS
```

Tabel 2.6. Pseudocode kelas Solver

Penjelasan dari kelas Solver:

- Memiliki atribut yang sudah dibuat dalam sebuah kelas seperti board, pieces, lalu ada atribut yang digunakan untuk keperluan pengeluaran program seperti `cases`, `startTime`, dan `colors`.
- Konstruktor solver yang digunakan untuk menginisialisasikan objek solver dengan sebuah ukuran papan (`rows`, `cols`) dan daftar piece (`pieces`) serta grid awal (`grid`).
- Beberapa metode yang sifatnya melakukan penyelesaian pada puzzle seperti, `solve` (memulai penyelesaian), `solvePuzzle` (mencoba semua kemungkinan dengan menekankan backtracking), `getExecutionTime` (mengembalikan waktu eksekusi dalam milidetik), `getCases` (mengembalikan jumlah kasus yang diperiksa), `getBoard` (mengembalikan representasi papan dalam bentuk string tanpa warna), dan `getColoredBoard` (mengembalikan representasi papan dalam bentuk string menggunakan warna).

2.2.7. FileHandler

```
CLASS FileHandler

    PRIVATE CONSTANT CELL_SIZE: INTEGER = 50

    PUBLIC CONSTANT COLORS: ARRAY OF Color = [

        NEW Color(255, 0, 0),       // Red

        NEW Color(0, 0, 255),       // Blue

        NEW Color(0, 255, 0),       // Green

        NEW Color(255, 165, 0),     // Orange

        NEW Color(255, 0, 255),     // Magenta

        NEW Color(0, 255, 255),     // Cyan

        NEW Color(255, 192, 203),   // Pink

        NEW Color(255, 255, 0),     // Yellow

        NEW Color(128, 0, 0),       // Maroon

        NEW Color(0, 128, 0),       // Dark Green

        NEW Color(0, 0, 128),       // Navy

        NEW Color(128, 0, 128),     // Purple

        NEW Color(128, 128, 0),     // Olive

        NEW Color(0, 128, 128),     // Teal

        NEW Color(240, 128, 128),   // Light Coral

        NEW Color(255, 140, 0),     // Dark Orange

        NEW Color(218, 112, 214),   // Orchid

        NEW Color(176, 224, 230),   // Powder Blue

        NEW Color(255, 235, 205),   // Blanched Almond

        NEW Color(173, 216, 230),   // Light Blue

        NEW Color(221, 160, 221),   // Plum

        NEW Color(144, 238, 144),   // Light Green

        NEW Color(255, 160, 122),   // Light Salmon

        NEW Color(102, 205, 170),   // Medium Aquamarine

        NEW Color(135, 206, 235),   // Sky Blue

        NEW Color(219, 112, 147)    // Pale Violet Red

    ]


    FUNCTION readInputFile(filename: STRING) RETURNS Input

        TRY
```

```
            OPEN FILE "../test/input/" + filename FOR READING

            READ firstLine FROM FILE

            IF firstLine IS EMPTY

                THROW EXCEPTION "Format error: Please input rows, cols,
and total pieces"

            END IF



            SPLIT firstLine INTO dims BY WHITESPACE

            IF dims LENGTH != 3

                THROW EXCEPTION "Format error: rows, cols, and total
pieces line must contain exactly 3 numbers"

            END IF



            SET N = INTEGER PARSED FROM dims[0]

            SET M = INTEGER PARSED FROM dims[1]

            SET P = INTEGER PARSED FROM dims[2]



            IF N <= 0 OR M <= 0 OR P <= 0

                THROW EXCEPTION "Invalid input: rows, cols, and total
pieces cannot be a negative number"

            END IF



            READ mode FROM FILE

            SET grid = NEW 2D ARRAY OF CHAR WITH DIMENSIONS N x M



            IF mode == "DEFAULT"

                FOR i FROM 0 TO N - 1

                    FOR j FROM 0 TO M - 1

                        SET grid[i][j] = '.'

                    END FOR

                END FOR

            ELSE IF mode == "CUSTOM"

                FOR i FROM 0 TO N - 1

                    READ line FROM FILE
```

```
                    IF line IS NULL OR line LENGTH != M
                        THROW EXCEPTION "Invalid custom configuration
format"
                    END IF
                    FOR j FROM 0 TO M - 1
                        SET grid[i][j] = (line[j] == 'X') ? '.' : ' '
                    END FOR
                END FOR
            ELSE
                THROW EXCEPTION "Invalid mode"
            END IF

            SET pieces = NEW LIST OF Piece
            SET currentLines = NEW LIST OF STRING
            SET currentId = '\0'
            SET maxWidth = 0

            WHILE READ line FROM FILE
                TRIM line
                IF line IS NOT EMPTY
                    SET pieceId = line[0]
                    SET currentLineWidth = COUNT OF pieceId IN line
                    SET maxWidth = MAX(maxWidth, currentLineWidth)

                    IF currentId == '\0'
                        SET currentId = pieceId
                    ELSE IF pieceId != currentId
                        IF currentLines SIZE > N OR maxWidth > M
                            THROW EXCEPTION "Piece " + pieceId + " size
exceeds board dimensions"
                        END IF

                        ADD Piece.linesToPiece(currentLines, currentId)
TO pieces
```

```
                        SET currentLines = NEW LIST OF STRING
                        SET currentId = pieceId
                        SET maxWidth = currentLineWidth
                    END IF

                    FOR EACH c IN line
                        IF c != ' ' AND c != currentId
                            THROW EXCEPTION "Invalid piece format: mixed
Letters in same piece"
                        END IF
                    END FOR

                    ADD line TO currentLines
                END IF
            END WHILE

            IF currentLines IS NOT EMPTY
                SET height = currentLines SIZE
                SET maxDimension = MAX(height, maxWidth)
                IF maxDimension > MAX(N, M)
                    THROW EXCEPTION "Piece " + currentId + " size
exceeds board dimensions"
                END IF

                ADD Piece.linesToPiece(currentLines, currentId) TO
pieces
            END IF

            IF pieces SIZE != P
                THROW EXCEPTION "Number of pieces (" + pieces SIZE + ")
does not match piece (" + P + ")"
            END IF

            SET uniqueIds = NEW SET OF CHAR
            FOR EACH piece IN pieces
```

```
                IF NOT uniqueIds ADD piece.getId())

                    THROW EXCEPTION "Duplicate piece ID found: " +
piece.getId()

                END IF

            END FOR



            RETURN NEW Input(N, M, P, mode, pieces, grid)

        CATCH EXCEPTIONS

            THROW EXCEPTION WITH MESSAGE

        END TRY

    END FUNCTION



    FUNCTION saveText(filename: STRING, boardState: STRING,
executionTime: LONG, iterations: LONG)

        CREATE FOLDER "../test/output" IF NOT EXISTS

        OPEN FILE "../test/output/" + filename FOR WRITING

        WRITE "Solution:" TO FILE

        WRITE boardState TO FILE

        WRITE "Execution time: " + executionTime + " ms" TO FILE

        WRITE "Iterations: " + iterations TO FILE

        CLOSE FILE

    END FUNCTION



    FUNCTION saveImage(outputFilename: STRING, boardState: STRING, N:
INTEGER, M: INTEGER, executionTime: LONG, cases: LONG)

        CREATE FOLDER "../test/saved" IF NOT EXISTS

        SET padding = 20

        SET infoWidth = 200

        SET imageWidth = M * CELL_SIZE + infoWidth + 3 * padding

        SET imageHeight = N * CELL_SIZE + 2 * padding



        SET image = NEW BufferedImage(imageWidth, imageHeight,
BufferedImage.TYPE_INT_RGB)

        SET g2d = image.createGraphics()
```

```
        SET g2d COLOR TO WHITE
        FILL RECTANGLE (0, 0, imageWidth, imageHeight) WITH WHITE

        SPLIT boardState INTO lines BY NEWLINE
        SET boardX = padding
        SET boardY = padding

        FOR i FROM 0 TO N - 1
            FOR j FROM 0 TO M - 1
                SET c = lines[i][j]
                SET cellX = boardX + j * CELL_SIZE
                SET cellY = boardY + i * CELL_SIZE

                IF c != ' ' AND c != '.'
                    SET g2d COLOR TO COLORS[c - 'A']
                    FILL RECTANGLE (cellX, cellY, CELL_SIZE, CELL_SIZE)
WITH COLOR
                END IF

                SET g2d COLOR TO BLACK
                SET g2d STROKE TO NEW BasicStroke(2)
                DRAW RECTANGLE (cellX, cellY, CELL_SIZE, CELL_SIZE) WITH
BLACK

                IF c != ' ' AND c != '.'
                    SET g2d COLOR TO BLACK
                    SET g2d FONT TO NEW Font("Arial", Font.BOLD, 20)
                    SET textX = cellX + CELL_SIZE / 3
                    SET textY = cellY + 2 * CELL_SIZE / 3
                    DRAW STRING c AT (textX, textY)
                END IF
            END FOR
        END FOR
```

```
        SET infoX = boardX + M * CELL_SIZE + padding
        SET infoY = boardY

        SET g2d COLOR TO NEW Color(240, 240, 240)
        FILL RECTANGLE (infoX, infoY, infoWidth, N * CELL_SIZE) WITH
COLOR

        SET g2d COLOR TO BLACK
        SET g2d FONT TO NEW Font("Arial", Font.PLAIN, 14)

        SET executionInfo = "Execution time: " + executionTime + " ms"
        SET casesInfo = "Total cases: " + cases

        SET textOffsetY = 20
        DRAW STRING executionInfo AT (infoX + 10, infoY + textOffsetY)
        DRAW STRING casesInfo AT (infoX + 10, infoY + textOffsetY + 20)

        DISPOSE g2d
        SAVE image TO "../test/saved/" + outputFilename AS PNG
    END FUNCTION
END CLASS
```

Tabel 2.7. Pseudocode kelas FileHandler

Penjelasan dari kelas FileHandler:

- Memiliki atribut `CELLS_SIZE` yakni ukuran sel yang ditetapkan dan `COLORS` yakni penyimpanan array berisikan daftar warna yang digunakan untuk menghasilkan pengembalian berupa papan yang memiliki daftar piece berwarna.
- Beberapa metode yang mengatasi input file, pembacaan file, penyimpanan teks ke file, dan menyimpan solusi dalam bentuk gambar.

2.2.8. AppLauncher

```
CLASS AppLauncher

    FUNCTION main(args: ARRAY OF STRING)

        SET scanner = NEW Scanner(System.in)


        PRINT "Choose your mode:"

        PRINT "1. GUI Mode"

        PRINT "2. CLI Mode"

        PRINT "Enter choice (1/2): "


        SET choice = scanner.nextInt()

        scanner.nextLine() // Consume newline


        IF choice == 1

            CALL MainGUI.main(args) // Launch GUI Mode

        ELSE IF choice == 2

            CALL runCLI() // Launch CLI Mode

        ELSE

            PRINT "Choice is invalid."

        END IF


        CLOSE scanner

    END FUNCTION


    FUNCTION runCLI()

        SET scanner = NEW Scanner(System.in)


        PRINT "Input filename: "

        SET inputFilename = scanner.nextLine()


        TRY

            SET input = FileHandler.readInputFile(inputFilename)

            SET solver = NEW Solver(input.rows, input.cols,
input.pieces, input.grid)
```

```
            PRINT "\nSolving... Please wait."
            SET solved = solver.solve()

            SET executionTime = solver.getExecutionTime()
            SET cases = solver.getCases()

            IF solved
                PRINT "\nSolution found:"
                PRINT solver.getColoredBoard()
            ELSE
                PRINT "\nNo solution found."
            END IF

            PRINT "Time searching: " + executionTime + " ms"
            PRINT "Total cases: " + cases

            IF solved
                PRINT "\nSave solution? (y/n): "
                SET saveChoice = scanner.nextLine().trim().toLowerCase()

                IF saveChoice == "y"
                    PRINT "\nChoose output format:"
                    PRINT "1. Image (.png)"
                    PRINT "2. Text (.txt)"
                    PRINT "Enter choice (1/2): "

                    SET formatChoice = scanner.nextLine().trim()

                    IF formatChoice == "1"
                        SET outputFilename = "output_image_" +
inputFilename.replace(".txt", ".png")
                        CALL FileHandler.saveImage(outputFilename,
solver.getBoard(), input.rows, input.cols, executionTime, cases)
```

```
                        PRINT "Image saved to: test/saved/" +
outputFilename

                    ELSE IF formatChoice == "2"

                        SET outputFilename = "output_text_" +
inputFilename

                        CALL FileHandler.saveText(outputFilename,
solver.getBoard(), executionTime, cases)

                        PRINT "Solution saved to: test/saved/" +
outputFilename

                    ELSE

                        PRINT "Invalid choice. Solution not saved."

                    END IF

                END IF

            END IF



        CATCH IOException AS e

            PRINT "Error: " + e.getMessage()

        FINALLY

            CLOSE scanner

        END TRY

    END FUNCTION

END CLASS
```

Tabel 2.8. Pseudocode kelas AppLauncher

Penjelasan dari kelas AppLauncher:

- Berisikan handling CLI seperti pemilihan mode solving (menggunakan GUI atau tidak).
- Memiliki akses dalam handling input nama file yang ingin dicari solusinya, melakukan solusi, dan mengeluarkan hasil solusi.
- Melakukan proses penyimpanan solusi dalam bentuk file ketika pengguna meminta.

2.2.9. MainGUI

```
CLASS MainGUI EXTENDS JFrame

    PRIVATE fileInputField: JTextField

    PRIVATE solveButton: JButton

    PRIVATE boardPanel: JPanel

    PRIVATE infoLabel: JLabel

    PRIVATE saveTextButton: JButton

    PRIVATE saveImageButton: JButton


    PRIVATE solver: Solver

    PRIVATE input: Input


    CONSTRUCTOR MainGUI()

        SET TITLE "IQ Puzzler Pro Solver"

        SET SIZE 800x600

        SET DEFAULT CLOSE OPERATION TO EXIT ON CLOSE

        SET LAYOUT TO BorderLayout()


        // Input panel

        SET inputPanel = NEW JPanel()

        SET fileInputField = NEW JTextField(20)

        SET solveButton = NEW JButton("Solve")

        ADD NEW JLabel("Enter test filename: ") TO inputPanel

        ADD fileInputField TO inputPanel

        ADD solveButton TO inputPanel

        ADD inputPanel TO NORTH OF FRAME


        // Board panel

        SET boardPanel = NEW JPanel()

        SET boardPanel BACKGROUND TO WHITE

        ADD boardPanel TO CENTER OF FRAME


        // Info panel

        SET infoPanel = NEW JPanel()
```

```
        SET infoLabel = NEW JLabel("Execution time: 0 ms | Iterations:
0")

        ADD infoLabel TO infoPanel

        ADD infoPanel TO SOUTH OF FRAME


        // Save buttons panel

        SET savePanel = NEW JPanel()

        SET saveTextButton = NEW JButton("Save as Text")

        SET saveImageButton = NEW JButton("Save as Image")

        DISABLE saveTextButton

        DISABLE saveImageButton

        ADD saveTextButton TO savePanel

        ADD saveImageButton TO savePanel

        ADD savePanel TO EAST OF FRAME


        // Solve button action

        ADD ACTION LISTENER TO solveButton

            WHEN solveButton IS CLICKED

                SET inputFilename = fileInputField.getText()

                TRY

                    SET input = FileHandler.readInputFile(inputFilename)

                    SET solver = NEW Solver(input.rows, input.cols,
input.pieces, input.grid)


                    // Start solving in a separate thread

                    START NEW THREAD

                        DISABLE solveButton

                        SET solved = solver.solve()

                        SET executionTime = solver.getExecutionTime()

                        SET iterations = solver.getCases()


                        RUN ON UI THREAD

                            IF solved
```

```
                                SET infoLabel TEXT TO "Execution time: "
+ executionTime + " ms | Iterations: " + iterations

                                CALL drawBoard(solver.getBoard())

                                ENABLE saveTextButton

                                ENABLE saveImageButton

                            ELSE

                                SET infoLabel TEXT TO "No solution
exists!"

                            END IF

                            ENABLE solveButton

                    END THREAD

                CATCH IOException AS ex

                    SHOW ERROR MESSAGE "Error: " + ex.getMessage()

                END TRY

            END ACTION LISTENER


        // Save as Text button action

        ADD ACTION LISTENER TO saveTextButton

            WHEN saveTextButton IS CLICKED

                SET outputFilename = "solution_text_" +
fileInputField.getText()

                TRY

                    CALL FileHandler.saveText(outputFilename,
solver.getBoard(), solver.getExecutionTime(), solver.getCases())

                    SHOW SUCCESS MESSAGE "Solution saved to:
test/output/" + outputFilename

                CATCH IOException AS ex

                    SHOW ERROR MESSAGE "Error: " + ex.getMessage()

                END TRY

            END ACTION LISTENER


        // Save as Image button action

        ADD ACTION LISTENER TO saveImageButton

            WHEN saveImageButton IS CLICKED

                SET outputFilename = "solution_image_" +
fileInputField.getText().replace(".txt", ".png")
```

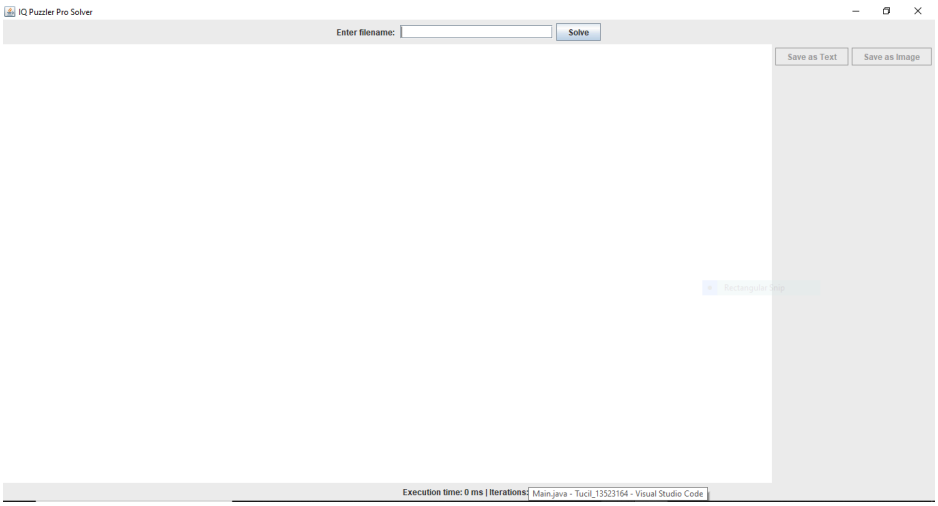
```
                TRY

                    CALL FileHandler.saveImage(outputFilename,
solver.getBoard(), input.rows, input.cols, solver.getExecutionTime(),
solver.getCases())

                    SHOW SUCCESS MESSAGE "Image saved to: test/output/"
+ outputFilename

                CATCH IOException AS ex

                    SHOW ERROR MESSAGE "Error: " + ex.getMessage()

                END TRY

            END ACTION LISTENER

    END CONSTRUCTOR


    FUNCTION drawBoard(boardState: STRING)

        REMOVE ALL COMPONENTS FROM boardPanel

        SPLIT boardState INTO lines BY NEWLINE

        SET rows = lines LENGTH

        SET cols = lines[0] LENGTH


        SET boardPanel LAYOUT TO GridLayout(rows, cols)


        FOR i FROM 0 TO rows - 1

            FOR j FROM 0 TO cols - 1

                SET c = lines[i][j]

                SET cell = NEW JLabel(String.valueOf(c), CENTER)

                SET cell OPAQUE TO TRUE

                SET cell BACKGROUND TO getColorForPiece(c)

                SET cell BORDER TO LineBorder(Color.BLACK)

                ADD cell TO boardPanel

            END FOR

        END FOR


        REVALIDATE boardPanel

        REPAINT boardPanel

    END FUNCTION
```

```
    FUNCTION getColorForPiece(pieceId: CHAR) RETURNS Color
        IF pieceId == '.' OR pieceId == ' '
            RETURN Color.WHITE
        END IF
        SET index = pieceId - 'A'
        IF index >= 0 AND index < FileHandler.COLORS LENGTH
            RETURN FileHandler.COLORS[index]
        END IF
        RETURN Color.LIGHT_GRAY
    END FUNCTION

    FUNCTION main(args: ARRAY OF STRING)
        RUN ON UI THREAD
            SET gui = NEW MainGUI()
            SET gui VISIBLE TO TRUE
        END RUN
    END FUNCTION
END CLASS
```

Tabel 2.9. Pseudocode kelas MainGUI

Penjelasan dari kelas FileHandler:

- GUI untuk proses input hingga penyimpanan solusi.
- Terdiri atas input text field, button, cells, dan tabel/tempat penyelesaian.



Gambar 2.1. Tampilan GUI

2.2.10. Main

```
CLASS Main

    FUNCTION main(args: ARRAY OF STRING)

        CALL AppLauncher.main(args)

    END FUNCTION

END CLASS
```

Tabel 2.10. Pseudocode kelas Main

Penjelasan dari kelas Main:

- Memanggil kelas AppLauncher.

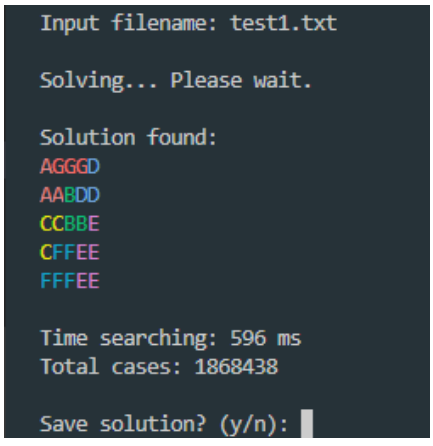
BAB 2: UJI KASUS

1. Uji kasus 1 (test1.txt)

Teks masukan:

```
5 5 7

DEFAULT

A

AA

B

BB

C

CC

D

DD

EE

EE

E

FF

FF

F

GGG
```

Keluaran:



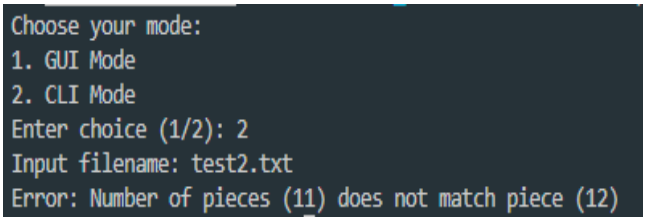
2. Uji kasus 2 (test2.txt)

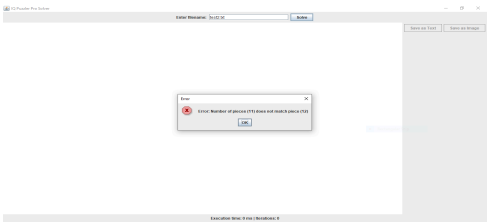
Teks masukan:

```
8 8 12
DEFAULT
  A
 AAA
AAAAA
  A
  A
B
BB
BBB
C
CC
CCC
DD
DDD
DDDD
EE
EEE
EEEE
FF
FFF
FFFF
GG
GGG
GGGG
HH
HHH
HHHH
II
III
IIII
JJ
JJJ
```

```
JJJJ

KK

KKK

KKKK
```

Keluaran:





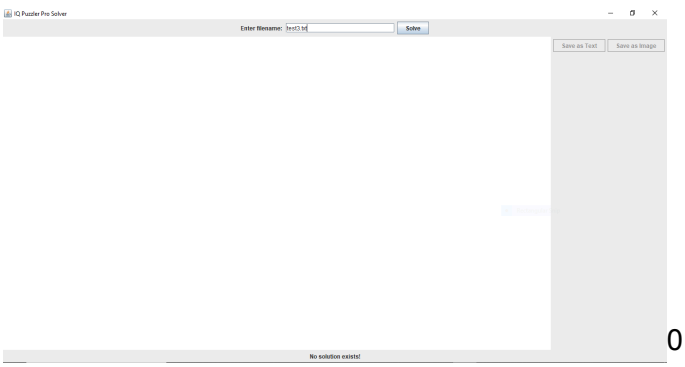
3. Uji kasus 3 (test3.txt)

Teks masukan:

```
6 6 6

DEFAULT

A

AA

AAA

AAAA

B

BB

BBB

BBBB

C

CC

CCC

CCCC

D

DD

DDD

DDDD

E

EE

EEE

EEEE

F

FF

FFF

FFFF
```

Keluaran:

```
Choose your mode:
1. GUI Mode
2. CLI Mode
Enter choice (1/2): 2
Input filename: test3.txt

Solving... Please wait.

No solution found.
Time searching: 144 ms
Total cases: 91040
```



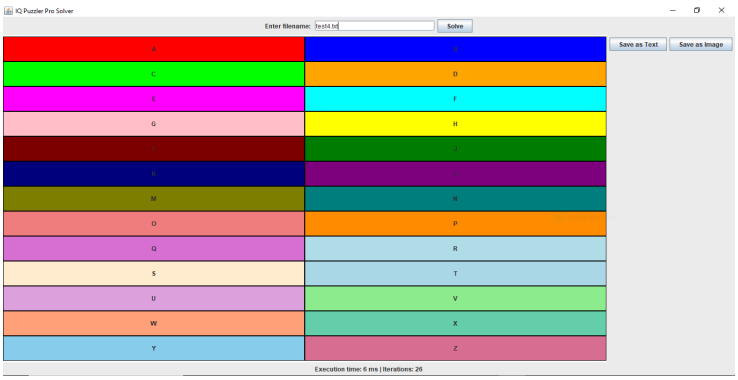
4. Uji Kasus 4 (test4.txt)

Teks masukan:

```
13 2 26
DEFAULT
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
```

Keluaran:

```
Solution found:
AB
CD
EF
GH
IJ
KL
MN
OP
QR
ST
UV
WX
YZ

Time searching: 14 ms
Total cases: 26

Save solution? (y/n):
```

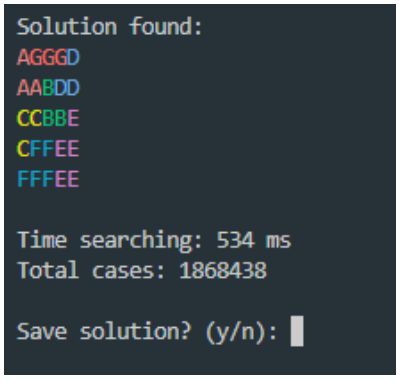


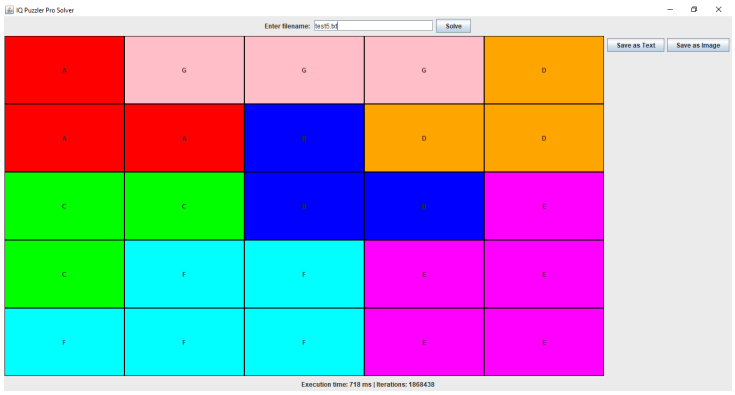
5. Uji Kasus 5 (test5.txt)

Teks masukan:

```
5 5 7

DEFAULT

A

AA

B

BB

C

CC

D

DD

EE

EE

E

FF

FF

F

GGG
```

Keluaran:





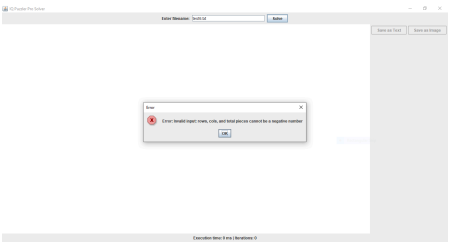
6. Uji Kasus 6 (test6.txt)

Teks masukan:

```
0 0 0

DEFAULT

A

AA

B

BB

C

CC

D

DD

EE

EE

E

FF

FF

F

GGG
```

Keluaran:

```
Choose your mode:
1. GUI Mode
2. CLI Mode
Enter choice (1/2): 2
Input filename: test6.txt
Error: Invalid input: rows, cols, and total pieces cannot be a negative number
```

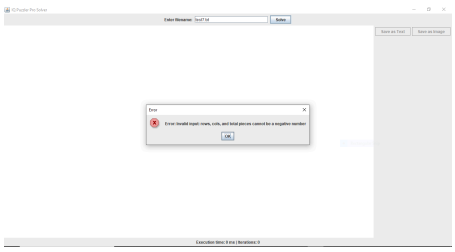


7. Uji Kasus 7 (test7.txt)

Teks masukan:

Keluaran:

```
Choose your mode:
1. GUI Mode
2. CLI Mode
Enter choice (1/2): 2
Input filename: test7.txt
Error: Invalid input: rows, cols, and total pieces cannot be a negative number
```



BAB 3: Pemenuhan dan Pranala

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	☐	

Link repository: https://github.com/inRiza/Tucil_13523164