LAPORAN TUGAS KECIL 1
IF2211 STRATEGI ALGORITMA

Penyelesaian IQ Puzzler Pro dengan Algoritma Brute Force

Disusun Oleh :

13523005 Muhammad Alfansya

PROGRAM STUDI TEKNIK INFORMATIKA SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA INSTITUT TEKNOLOGI BANDUNG JL. GANESHA 10, BANDUNG 40132

2025

DAFTAR ISI

BAB 1
Deskripsi Masalah

1.1. IQ Puzzler Pro

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**.

1.2. Brute Force

Brute force adalah sebuah pendekatan yang lempang (straightforward) untuk memecahkan suatu masalah, biasanya didasarkan pada pernyataan masalah (problem statement) dan definisi konsep yang dilibatkan. Algoritma brute force memecahkan persoalan secara sangat sederhana, langsung, jelas caranya dan mudah dipahami.

Algoritma brute force yang digunakan untuk menyelesaikan permainan IQ Puzzler Pro ini yaitu mencoba segala kemungkian susunan dari balok balok ke papan permainan. Algoritma brute force memiliki kemungkinan iterasi yang sangat tinggi dibanding dengan algoritma intuitif lainnya namum dipastikan akan mendapat solusi jika memang persoalan memiliki Solusi

Penerapan program dibuat berdasarkan konsep penyelesaian permainan dengan brute force secara manual. Program akan mencoba semua kemungkinan penempatan blok satu persatu, jika ada ketidak sesuaian saat menempatkan blok program akan melakukan pencarian solusi ulang dengan backtracking rekursif. Balok akan dicoba dengan segala konfigurasi nya, balok bisa dirotasi dan dicerminkan sehingga bisa didapatkan penyelesaian akhir.

BAB 2

Implementasi

Program diimplementasikan dengan bahasa Java. Program menggunakan paradigma OOP dengan menganggap Blok dan Board sebagai kelas utama.

Class Board

```
public class Board {
    private String[][] board;
    private int rows, cols;
```

Memiliki atribut rows dan cols bertipe integer yang merepresentasikan baris dan kolom papan dan atribut board bertipe Matriks string yang merepresentasikan papan.

Class ini memilii methode :

- Board(), menginisiasi pembentukan objek Board
- canPlace(), memeriksa apakah bisa meletakan blok di papan pada titik tertentu
- placePiece(), meletakkan blok di papan
- removePiece(), menghapus blok dari papan
- printBoard(), mencetak kondisi papan
- getCols(), mengembalikan cols
- getRows(), mengembalikan rows
- getBoard(), mengembalikan kondisi papan dalam bentuk String

Class Piece

```
public class Piece {
    private String[] shape;
```

Memiliki atribut shape bertipe array of String yang merepresentasikan bentuk blok ke String.

Class ini memiliki methode :

- Piece(), menginisiasi pembentukan objek Piece
- getHeight(), mengembalikan tinggi piece
- getWidth(), mengembalikan lebar piece
- getSpotAt(), mengembalikan bagian piece spesifik di titik tertentu
- getShape(), mengembalikan shape piece
- toMatrix(), mengubah shape ke matrix, merupakan private method yang digunakan dalam rotate()
- rotateMatrix(), merotasi matrix shape
- rotate(), merotasi piece, menggunakan bantuan method toMatrix dan rotateMatrix dalam implementasi nya
- mirror(), mencerminkan piece

- printPiece(), mencetak piece, digunakan untuk debugging
- printMatrix(), mencetak matriks, digunakan untk debugging

Class FileManager

Kelas controller yang digunakan untuk memproses file, baik untuk membaca file masukan maupun menulis file keluaran.

Memiliki method:

- FileManager(), untuk membaca file masukan
- saveToFile(), untuk menulis file keluaran

Class inputFile

Merupakan kelas untuk menyimpan hasil masukan dari membaca file masukan. Memiliki atribut :

```
public class inputFile {
    public final int N, M, P;
    public final String mode;
    public final List<Piece> pieces;
```

- Int N M P sesuai spesifikasi, masing masing merepresentasikan baris, kolom, dan jumlah piece
- Mode , merepresentasikan mode penyusunan
- Pieces, List berisikan Piece untuk menampung pieces sesuai masukan

Class Solver

Merupakan kelas controller untuk menyelesaikan puzzle

```
public class Solver {
    private Board board;
    private List<Piece> pieces;
    private boolean solved;
    private long attempts;
    private long startTime;
```

Memiliki atribut board yang merepresentasikan papan permainan, pieces sebagai blok, Boolean solved untuk mengetahui status penyelesaian, attempts dan startTime untuk menyimpan jumlah percobaan dan waktu dimulai proses penyelesaian.

Memiliki method:

- solve(), method yang dipanggil untuk menjalankan proses penyelesaian puzzle, menggunakan algoritma Brute Force dan memanfaatkan backtracking rekursif.
- getStartTime(), mengembalikan startTime
- getAttempts(), mengembalikan attempts

BAB 3

Source Code

Struktur Program

Sesuai spesifikasi tugas, program memiliki struktur sebagai berikut :

- Folder **src** berisi *source code* program.
- Folder **bin** berisi *executable file* (Sesuaikan dengan bahasa pemrograman yang digunakan).
- Folder **test** berisi solusi jawaban dari data uji yang digunakan dalam laporan.
- Folder **doc** berisi laporan tugas kecil dalam bentuk PDF.

Source Code

Board.java

```java
package src;

public class Board {
    private String[][] board;
    private int rows, cols;

    public Board(int rows,int cols) {
        this.rows = rows;
        this.cols = cols;
        this.board = new String[rows][cols] ;
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                board[i][j] = ".";
            }
        }
    }

    public boolean canPlace(Piece piece, int row, int col) {
        for (int i = 0; i < piece.getHeight(); i++) {
            for (int j = 0; j < piece.getWidth(i); j++) {
                if (piece.getSpotAt(i, j) != ' ') {
                    int newRow = row + i;
                    int newCol = col + j;
                    if (newRow >= rows || newCol >= cols || !board[newRow][newCol].equals(anObject:".")) {
                        return false;
                    }
                }
            }
        }
        return true;
    }

    public void placePiece(Piece piece, int row, int col, char ch) {
        for (int i = 0; i < piece.getHeight(); i++) {
            for (int j = 0; j < piece.getWidth(i); j++) {
                if (piece.getSpotAt(i, j) != ' ') {
                    board[row + i][col + j] = String.valueOf(ch);
                }
            }
        }
    }

    public void removePiece(Piece piece, int row, int col) {
        for (int i = 0; i < piece.getHeight(); i++) {
            for (int j = 0; j < piece.getWidth(i); j++) {
                if (piece.getSpotAt(i, j) != ' ') {
                    board[row + i][col + j] = ".";
                }
            }
        }
    }
```

```java
public void printBoard() {
    final String[] colors ={
        "\u001B[31m",
        "\u001B[33m",
        "\u001B[32m",
        "\u001B[35m",
        "\u001B[34m",
        "\u001B[36m",
        "\u001B[37m",
        "\u001B[90m",
        "\u001B[91m",
        "\u001B[92m",
        "\u001B[93m",
        "\u001B[94m",
        "\u001B[95m",
        "\u001B[96m",
        "\u001B[90m",
        "\u001B[41m",
        "\u001B[42m",
        "\u001B[43m",
        "\u001B[44m",
        "\u001B[45m",
        "\u001B[46m",
        "\u001B[100m",
        "\u001B[101m",
        "\u001B[102m",
        "\u001B[103m",
        "\u001B[104m"
    };
    final String RESET = "\u001B[0m";

    System.out.println(x:"Solusi ditemukan:");

    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            String ch = board[i][j];
            if (ch.trim().charAt(index:0) >= 'A' && ch.trim().charAt(index:0) - 'A' <='Z'){
                System.out.print(colors[ch.trim().charAt(index:0) - 'A'] + board[i][j] + RESET);
            }
            else{
                System.out.print(board[i][j]);
            }
        }
        System.out.println();
    }
}
```

```java
    public int getCols(){
        return cols;
    }

    public int getRows(){
        return rows;
    }

    public String getBoard() {
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                sb.append(board[i][j]);
            }
            sb.append(c:'\n');
        }
        return sb.toString();
    }
}
```

Piece.java

```java
package src;

public class Piece {
    private String[] shape;

    public Piece(String[] shape) {
        this.shape = shape;
    }

    public int getHeight() {
        return shape.length;
    }

    public int getWidth(int row) {
        return shape[row].length();
    }

    public char getSpotAt(int row, int col) {
        return shape[row].charAt(col);
    }

    public String[] getShape() {
        return shape;
    }

    public char[][] toMatrix() {
        // Hitung jumlah baris dan kolom maksimum
        int rows = shape.length;
        int cols = 0;
        for (String row : shape) {
            if (row.length() > cols) {
                cols = row.length();
            }
        }

        // Buat matriks dengan ukuran yang sesuai
        char[][] matrix = new char[rows][cols];

        // Isi matriks dengan karakter dari shape
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < shape[i].length(); j++) {
                matrix[i][j] = shape[i].charAt(j);
            }
            // Isi sisa kolom dengan spasi jika panjang baris tidak seragam
            for (int j = shape[i].length(); j < cols; j++) {
                matrix[i][j] = ' ';
            }
        }

        return matrix;
    }
```

```java
private static char[][] rotateMatrix(char[][] block){
    int rows = block.length;
    int cols = block[0].length;
    char[][] rotated = new char[cols][rows];

    for (int i = 0; i < rows; ++i){
        for (int j = 0; j < cols; ++j){
            rotated[j][rows - 1 - i] = block[i][j];
        }
    }

    return rotated;
}

public Piece rotate() {
    char[][] matrix = toMatrix();
    char[][] rotatedMatrix = rotateMatrix(matrix);

    String[] newShape = new String[rotatedMatrix.length];

    for (int i = 0; i < rotatedMatrix.length; i++) {
        StringBuilder sb = new StringBuilder();
        for (int j = 0; j < rotatedMatrix[i].length; j++) {
            sb.append(rotatedMatrix[i][j]);
        }

        newShape[i] = sb.toString();
    }

    return new Piece(newShape);
}


public Piece mirror() {
    String[] newShape = new String[shape.length];
    for (int i = 0; i < shape.length; i++) {
        newShape[i] = new StringBuilder(shape[i]).reverse().toString();
    }
    return new Piece(newShape);
}

// Cetak bentuk potongan (untuk debugging)
public void printPiece() {
    for (String row : shape) {
        System.out.println(row);
    }
}
```

```java
    public static void printMatrix(char[][] matrix) {
        for (char[] row : matrix) {
            for (char cell : row) {
                System.out.print(cell + " ");
            }
            System.out.println();
                }
    }
}
```

FileManager.java

```java
package src;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class FileManager {


    public static inputFile FileManager(String filename) throws IOException{

        File file = new File("../test/input/" + filename);
        Scanner scanner = new Scanner(file);

        String firstLine = scanner.nextLine();

        if (firstLine == null || firstLine.trim().isEmpty()) {
            throw new IOException(message:"Pastikan inputan N M P benar");
        }

        String[] first = firstLine.trim().split(regex:"\\s+");
        if (first.length != 3) {
            throw new IOException(message:"Pastikan inputan N M P benar");
        }
        try {
            int N = Integer.parseInt(first[0]);
            int M = Integer.parseInt(first[1]);
            int P = Integer.parseInt(first[2]);

            if (N <= 0 || M <= 0 || P <= 0) {
                throw new IOException(message:"Invalid input: N, M, P must be positive");
            }

            String mode = scanner.nextLine().trim();
            if (!mode.equals(anObject:"DEFAULT")) {
                System.out.println(x:"Tidak Membuat Bonus Custom.\n");
                System.out.println(x:"Solusi DEFAULT ");
            }

            List<Piece> pieces = new ArrayList<>();
            List<String> currentPiece = new ArrayList<>();
            char currentChar = '\0';
```

```
            while (scanner.hasNextLine()) {
                String line = scanner.nextLine();

                if (line.trim().isEmpty()) {
                    continue;
                }

                char firstChar = line.trim().charAt(index:0);

                if (currentChar != '\0' && firstChar != currentChar) {
                    // Simpan potongan saat ini ke pieces
                    pieces.add(new Piece(currentPiece.toArray(new String[0])));
                    // Mulai potongan baru
                    currentPiece = new ArrayList<>();
                }

                currentPiece.add(line);
                currentChar = firstChar;
            }

            if (!currentPiece.isEmpty()) {
                pieces.add(new Piece(currentPiece.toArray(new String[0])));
            }
            scanner.close();
            return new inputFile(N, M, P, mode, pieces);
        }
        catch  (NumberFormatException e) {
            throw new IOException(message:"Invalid number format in N M P line");
        }
    }

    public static void saveToFile(String filename, String boardState, long executionTime, long attempts) throws IOException {
        File testFolder = new File(pathname:"../test/output");
        if (!testFolder.exists()) {
            testFolder.mkdir();
        }

        try (PrintWriter writer = new PrintWriter(new FileWriter("../test/output/" + filename))) {
            writer.println(x:"Solution:");
            writer.println(boardState);
            writer.println("Execution time: " + executionTime + " ms");
            writer.println("Iterations: " + attempts);
        }
    }
}
```

BAB 4

Eksperimen

Test Case 1

```
PS C:\Users\Alfansya\Documents\GitHub\Tucil1_13523005\bin> java src.Main
Masukkan path file input: tc1.txt
Solusi ditemukan:
AGGGD
AABDD
CCBBE
CFFEE
FFFEE
Execution time: 197 ms
Attempts : 5390931

Save solution to txt file? (y/n):
```

Test Case 2

```
Masukkan path file input: tc2.txt
Solusi ditemukan:
AAAEEEF
AAAAAHF
BBBBBHF
CCCIIHF
CDDIFFF
CDDGGGG
CCCGGGG
Execution time: 16 ms
Attempts : 12306
```

Test Case 3

```
Masukkan path file input: tc3.txt
Execution time: 2 ms
Attempts : 2952

Save solution to txt file? (y/n): n
Solution unsaved.
```

Test Case 4

```
Masukkan path file input: tc4.txt
Tidak Membuat Bonus Custom.

Solusi DEFAULT
Solusi ditemukan:
AGGGD
AABDD
CCBBE
CFFEE
FFFEE
Execution time: 210 ms
Attempts : 5390931

Save solution to txt file? (y/n): y
```

Test Case 5

```
Masukkan path file input: tc5.txt
Error: Pastikan inputan N M P benar
```

Test Case 6

```
Masukkan path file input: tc6.txt
Solusi ditemukan:
AABBBC
DBBEEC
DDDEEC
Execution time: 17 ms
Attempts : 44
```

Test Case 7

```
Masukkan path file input: tc7.txt
Solusi ditemukan:
AAAAGG
AAAEGG
BBBEEE
BBCCCC
BDDCCF
DDDFFF
Execution time: 47 ms
Attempts : 507703

Save solution to txt file? (y/n): y
Solution saved to: test/output/solution_tc7.txt
```

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	✓	