LAPORAN TUGAS KECIL I
IF2211 STRATEGI ALGORITMA

Penyelesaian IQ Puzzler dengan Algoritma Brute Force

Disusun oleh:
Joel Hotlan Haris Siahaan
13523025

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2025

Daftar Isi

A. Algoritma Brute Force

Program ini menggunakan algoritma Brute Force untuk menyusun blok-blok dalam papan permainan IQ Puzzler. Algoritma yang digunakan adalah sebagai berikut:

1. Validasi input: Dilakukan validasi terhadap ukuran papan, jumlah blok, luas total blok, duplikasi blok, dan kemungkinan kesalahan input lainnya.
2. Peletakan blok: Blok-blok diletakkan secara berurutan, mulai dari blok pertama (A) dengan konfigurasi awal, kemudian dilanjutkan dengan blok-blok berikutnya.
3. Pergeseran blok: Jika ada blok yang tidak dapat diletakkan di bagian paling kiri atas, maka blok tersebut akan digeser ke kanan satu satuan hingga tidak dapat diletakkan di bagian paling kanan. Jika masih tidak dapat diletakkan, maka blok akan diletakkan ulang di bagian paling kiri dan diturunkan satu satuan ke bawah. Proses ini diulang hingga blok terakhir dapat diletakkan atau konfigurasi blok tidak dapat diletakkan di manapun.
4. Perubahan konfigurasi blok: Jika blok terakhir tidak dapat diletakkan, maka konfigurasi blok akan diubah dengan rotasi 90 derajat searah jarum jam, kemudian dicerminkan. Urutan perubahan konfigurasi selanjutnya adalah rotasi 180 derajat, pencerminan rotasi 180 derajat, rotasi 270 derajat searah jarum jam, dan pencerminan rotasi 270 derajat searah jarum jam.
5. Backtracking: Jika semua konfigurasi masih tidak dapat meletakkan blok di manapun, maka dilakukan backtracking untuk blok sebelumnya dan kembali ke langkah 2.
6. Solusi: Proses akan selesai jika diperoleh satu solusi untuk memenuhi papan dan semua blok sudah digunakan, atau jika semua percobaan sudah dilakukan namun masih belum memenuhi papan dengan menggunakan keseluruhan blok.

B. Source Code

Program ini menggunakan bahasa Java. Program ini memiliki struktur utama terdiri dari :

1. **Main.java**: Program utama

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

public class Main {
    Run | Debug
    public static void main(String[] args) {
        System.out.println(x:"---------------- IQ Puzzler ----------------");
        Scanner scanner = new Scanner(System.in);
        System.out.print(s:"Masukkan nama file test case (ex : tes): ");
        String fileName1 = scanner.nextLine();
        String fileName = ("../test/" + fileName1 +".txt");

        try {
            PuzzleSolver solver = new PuzzleSolver(fileName);
            long start = System.currentTimeMillis();
            boolean solved = solver.solve();
            long finish = System.currentTimeMillis();

            if (solved) {
                System.out.println();
                System.out.println(x:"Solusi ditemukan: ");
                solver.printBoard();

                System.out.println("Jumlah iterasi: " + solver.getIterationCount() + " kali");
                System.out.println("Waktu pencarian: " + (finish - start) + " ms");

                System.out.print(s:"Simpan Solusi? (Y/N): ");
                String save = scanner.nextLine();
                if (save.equalsIgnoreCase(anotherString:"Y")) {
                    System.out.print(s:"Masukkan nama file solusi tanpa .txt (ex : solusi1): ");
                    String fileName2 = scanner.nextLine();
                    String Solutionfile = ("../test/" + fileName2 +".txt");
                    solver.saveSolution(Solutionfile);
                    System.out.println(x:"Solusi berhasil disimpan.");
                }
                else {
                    System.out.println(x:"Solusi tidak disimpan.");
                }
            }
            else {
                System.out.println(x:"Solusi tidak ditemukan.");
            }
        } catch (IOException e) {
            System.err.println("File tidak ditemukan.: " + e.getMessage());
        }
        scanner.close();
    }
}
```

2. **Solver.java**: Menyelesaikan puzzle

```java
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

public class PuzzleSolver {
    private char[][] board;
    private List<Block> blocks;
    private int N, M;
    private int iterationCount;

    private void readInput(String fileName) throws IOException {
        BufferedReader reader = new BufferedReader(new FileReader(fileName));

        String[] dims = reader.readLine().split(regex:" ");
        N = Integer.parseInt(dims[0]);
        M = Integer.parseInt(dims[1]);
        int P = Integer.parseInt(dims[2]);

        String type = reader.readLine();

        if (N <= 0 || M <= 0){
            System.out.println(x:"Error: Ukuran papan tidak valid");
            System.exit(status:0);
        } else {
            board = new char[N][M];
            for (char [] row : board) {
                Arrays.fill(row, val:'.');
            }
        }
        blocks = new ArrayList<>();
        Set<Character> usedSymbols = new HashSet<>();

        String currentSymbol = null;
        List<String> currentShape = new ArrayList<>();

        String line;
        while ((line = reader.readLine()) != null) {
            if (line.isEmpty()) continue;

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

            if (!Character.isUpperCase(firstChar) || firstChar < 'A' || firstChar > 'Z') {
                System.out.println("Karakter blok "+ firstChar + " tidak valid. Masukkan karakter antara A-Z. ");
                System.exit(status:0);
            }

            if (currentSymbol == null || firstChar != currentSymbol.charAt(index:0)) {
                if (!currentShape.isEmpty()) {
                    // Validasi: Pastikan blok tidak kosong
                    if (!isValidCharacter(currentShape)) {
                        System.out.println("Blok " + currentSymbol + " tidak valid!");
                        System.exit(status:0);
                    }

                    blocks.add(new Block(currentSymbol.charAt(index:0), new ArrayList<>(currentShape)));
                    currentShape.clear();
                }
                currentSymbol = String.valueOf(firstChar);

                if (usedSymbols.contains(currentSymbol.charAt(index:0))) {
                    System.out.println("Blok " + currentSymbol + " duplikasi.");
                    System.exit(status:0);
```

```
            }
            usedSymbols.add(currentSymbol.charAt(index:0));
        }
        currentShape.add(line);
    }

    if (!currentShape.isEmpty()) {
        if (!isValidCharacter(currentShape)) {
            System.out.println("Bentuk blok " + currentSymbol + " tidak valid!");
            System.exit(status:0);
        }
        blocks.add(new Block(currentSymbol.charAt(index:0), new ArrayList<>(currentShape)));
    }

    reader.close();

    if (blocks.size() != P) {
        System.out.println("Jumlah blok tidak sesuai dengan masukan P=" + P );
        System.exit(status:0);
    }

    int totalBlockCells = 0;
    for (Block block : blocks) {
        totalBlockCells += block.coordinates.size();
    }

    int totalBoardCells = N * M;
    if (totalBlockCells < totalBoardCells) {
        System.out.println("Blok kurang! Total blok masukkan = " + totalBlockCells + ", tetapi papan memerlukan " + totalBoard
        System.exit(status:0);
    } else if (totalBlockCells > totalBoardCells) {
        System.out.println("Blok lebih! Total blok masukkan = " + totalBlockCells + ", tetapi papan hanya memiliki " + totalBo
        System.exit(status:0);
```

```
private boolean isValidCharacter(List<String> shape) {
    for (String row : shape) {
        for (char c : row.toCharArray()) {
            if (Character.isUpperCase(c) && c >= 'A' && c <= 'Z') {
                return true;
            }
        }
    }
    return false;
}

private boolean canPlace(Block block, int startX, int startY) {
    for (int[] cell : block.coordinates) {
        int x = startX + cell[0];
        int y = startY + cell[1];
        if (x < 0 || x >= N || y < 0 || y >= M || board[x][y] != '.') {
            return false;
        }
    }
    return true;
}

private void placeBlock(Block block, int startX, int startY, char symbol) {
    for (int[] cell : block.coordinates) {
        int x = startX + cell[0];
        int y = startY + cell[1];
        board[x][y] = symbol;
    }
}
```

```java
private void removeBlock(Block block, int startX, int startY) {
    for (int[] cell : block.coordinates) {
        int x = startX + cell[0];
        int y = startY + cell[1];
        board[x][y] = '.';
    }
}


private boolean isValidBoard() {
    int filledCells = 0;
    for (char[] row : board) {
        for (char cell : row) {
            if (cell != '.') filledCells++;
        }
    }
    return filledCells == (N * M);
}
    public void printBoard() {
    for (char[] row : board) {
        for (char cell : row) {
            System.out.print(Utility.getColor(cell) + " ");
        }
        System.out.println();
    }
}

public PuzzleSolver(String filename) throws IOException {
    readInput(filename);
}
```

```java
public boolean solve() {
    System.out.println(x:"Pencarian sedang berlangsung");
    return recursiveSolve(index:0);
}

private boolean recursiveSolve(int index) {
    if (index == blocks.size()) return isValidBoard();
    Block block = blocks.get(index);
    List<Block> orientations = block.generateOrientations();
    for (Block orient : orientations) {
        for (int a = 0; a < N; a++) {
            for (int c = 0; c < M; c++) {
                if (canPlace(orient, a, c)) {
                    placeBlock(orient, a, c, block.symbol);
                    clearScreen();
                    printBoard();
                    try { Thread.sleep(millis:0); } catch (InterruptedException e) { Thread.currentThread().interrupt(); }

                    if (recursiveSolve(index + 1)){
                        return true;
                    }
                    removeBlock(orient, a, c);
                    clearScreen();
                    printBoard();
                    try { Thread.sleep(millis:0); } catch (InterruptedException e) { Thread.currentThread().interrupt(); }
                    iterationCount++;
                }
            }
        }
    }
    return false;
}
```

```java
    public void saveSolution(String fileName) throws IOException {
        BufferedWriter writer = new BufferedWriter(new FileWriter(fileName));
        for (char[] row : board) {
            for (char cell : row) {
                writer.write(cell + " ");
            }
            writer.newLine();
        }
        writer.close();
    }

    public int getIterationCount() {
        return iterationCount;
    }

    public char[][] getBoard() {
        return board;
    }

    private void clearScreen() {
        System.out.print(s:"\033[H\033[2J");
        System.out.flush();
    }
}
```

3. **Block.java**: Mengelola kelas blok

```java
import java.util.*;

public class Block {
    public char symbol;
    public List<int[]> coordinates;

    public Block(char symbol, List<String> shape) {
        this.symbol = symbol;
        this.coordinates = parseShape(shape);
    }

    private List<int[]> parseShape(List<String> shape) {
        List<int[]> coord = new ArrayList<>();
        for (int i = 0; i < shape.size(); i++) {
            for (int j = 0; j < shape.get(i).length(); j++) {
                if (shape.get(i).charAt(j) != '.') {
                    coord.add(new int[]{i, j});
                }
            }
        }
        return coord;
    }

    public List<Block> generateOrientations() {
        Set<String> seen = new HashSet<>();
        List<Block> orientations = new ArrayList<>();
        char[][] currentShape = toMatrix(this.coordinates);

        for (int i = 0; i < 4; i++) {
            String shapeStr = matrixToString(currentShape);

            if (seen.add(shapeStr)) {
                orientations.add(new Block(this.symbol, matrixToShape(currentShape)));

                // Cerminkan dan cek bentuk uniknya
                char[][] flipped = flip(currentShape);
```

```
                shapeStr = matrixToString(flipped);
                if (seen.add(shapeStr)) {
                    orientations.add(new Block(this.symbol, matrixToShape(flipped)));
                }
            }
            currentShape = rotate(currentShape);
        }
        return orientations;
    }

    private char[][] rotate(char[][] matrix) {
        int rows = matrix.length, cols = matrix[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 - i - 1] = matrix[i][j];
            }
        }
        return rotated;
    }

    private char[][] flip(char[][] matrix) {
        int rows = matrix.length, cols = matrix[0].length;
        char[][] flipped = new char[rows][cols];

        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                flipped[i][cols - j - 1] = matrix[i][j];
            }
        }
        return flipped;
    }
```

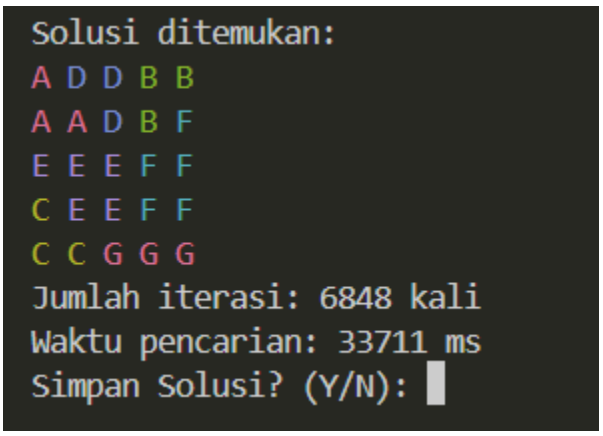
```
    private char[][] toMatrix(List<int[]> coords) {
        int maxX = 0, maxY = 0;
        for (int[] c : coords) {
            maxX = Math.max(maxX, c[0]);
            maxY = Math.max(maxY, c[1]);
        }

        char[][] shape = new char[maxX + 1][maxY + 1];
        for (char[] row : shape) Arrays.fill(row, val:'.');

        for (int[] c : coords) {
            shape[c[0]][c[1]] = this.symbol;
        }
        return shape;
    }

    private String matrixToString(char[][] matrix) {
        StringBuilder sb = new StringBuilder();
        for (char[] row : matrix) {
            sb.append(new String(row)).append(str:"\n");
        }
        return sb.toString();
    }

    private List<String> matrixToShape(char[][] matrix) {
        List<String> result = new ArrayList<>();
        for (char[] row : matrix) {
            result.add(new String(row));
        }
        return result;
    }
}
```

4. **Utility.java**: Utilitas untuk memberi warna pada blok.

```
public class Utility {
    public static String getColor(char c) {
        if (c == '.') return "\u001B[37m.\u001B[0m";
        int color = 31 + (c - 'A') % 6; // 6 warna berbeda
        return "\u001B[" + color + "m" + c + "\u001B[0m";
    }
}
```

C. Uji Coba Program

1. **Test Case 1 (valid input, terdapat penyelesaian)**
   5 5 7
   DEFAULT
   A
   AA
   BB
   B
   C
   CC
   D
   DD
   EE
   EE
   E
   F
   FF
   FF
   GGG



2. **Test Case 2 (valid input, terdapat penyelesaian)**
   3 3 3
   DEFAULT
   A
   AA
   B
   BB
   CCC

```
Solusi ditemukan:
A B B
A A B
C C C
Jumlah iterasi: 1 kali
Waktu pencarian: 33 ms
Simpan Solusi? (Y/N):
```

3. **Test Case 3 (valid input terdapat penyelesaian)**
   4 4 4
   DEFAULT
   A
   AA
   A
   B
   BB
   BB
   C
   CC
   DDDD

```
Solusi ditemukan:
A C C D
A A C D
A B B D
B B B D
Jumlah iterasi: 57 kali
Waktu pencarian: 330 ms
Simpan Solusi? (Y/N):
```

4. **Test Case 4 (valid input terdapat penyelesaian)**
   5 3 4
   DEFAULT
   A
   AA
   A
   B
   BB
   C
   C
   CC
   D
   D
   DD

```
Solusi ditemukan:
A C C
A A C
A B C
D B B
D D D
Jumlah iterasi: 0 kali
Waktu pencarian: 37 ms
Simpan Solusi? (Y/N):
```

5. **Test Case 5 (valid input, terdapat penyelesaian)**
   5 5 7
   DEFAULT
   AA
   AA
   BB
   C
   CC
   D
   DD
   E
   EE
   E
   F
   F
   FF
   G
   GG
   GG

```
Solusi ditemukan:
A A B B E
A A C E E
F D C C E
F D D G G
F F G G G
Jumlah iterasi: 16 kali
Waktu pencarian: 163 ms
Simpan Solusi? (Y/N):
```

6. **Test Case 6 (terdapat karakter tidak)**
   3 3 3
   DEFAULT
   A

AA
BB
B
;;;

```
--------------- IQ Puzzler ---------------
Masukkan nama file test case (ex : tes): tc6
Karakter blok ; tidak valid. Masukkan karakter antara A-Z.
```

7. **Test Case 7 (papan tidak)**

-5 -3 4
DEFAULT
A
AA
A
B
BB
C
C
CC
D
D
DD

```
---------------- IQ Puzzler ----------------
Masukkan nama file test case (ex : tes): tc7
Error: Ukuran papan tidak valid
```

D. 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	✓	

Link Repository Github : https://github.com/jhotlann/Tucil1_13523025