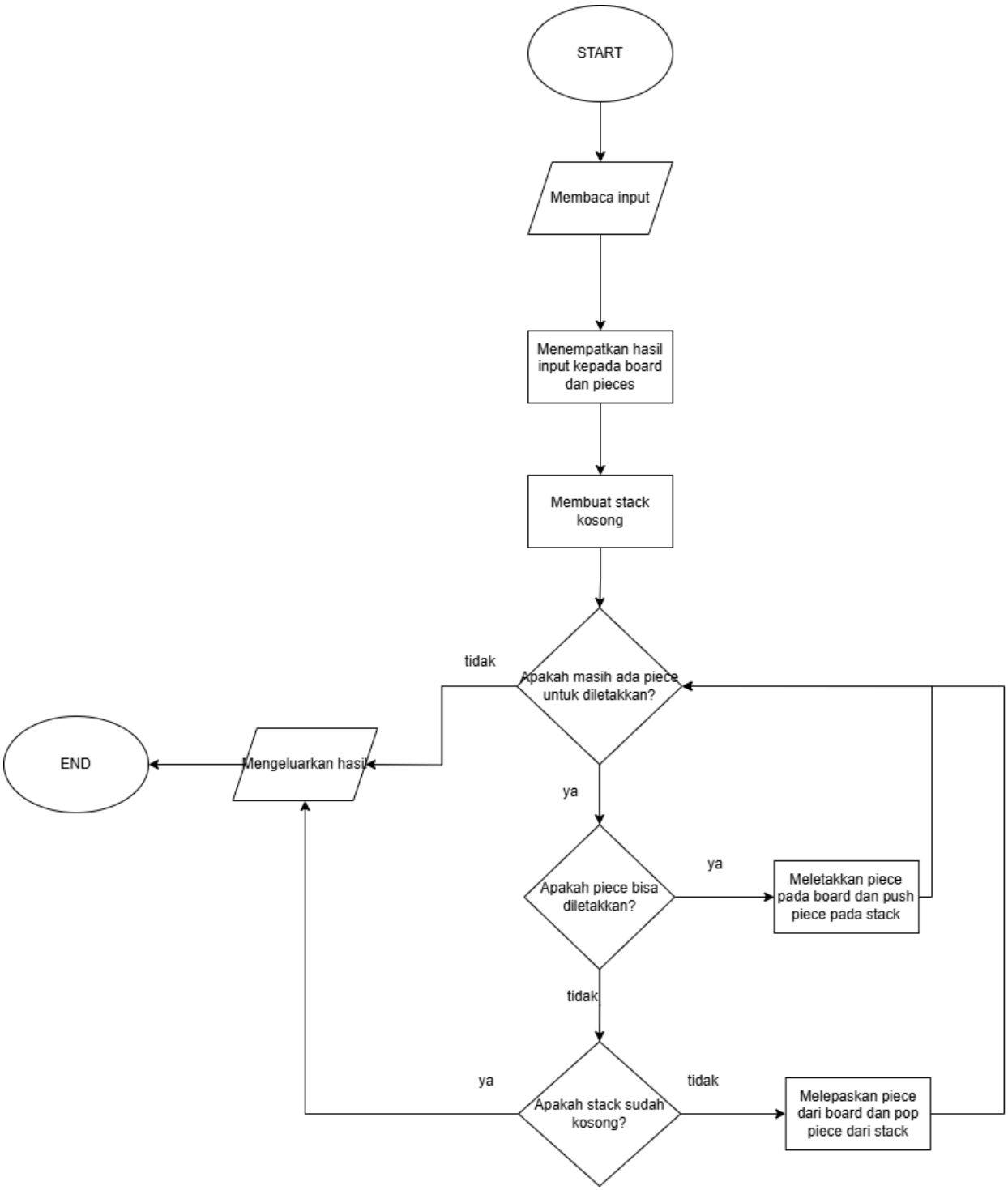
Tugas Kecil #1 - IF2211 Strategi Algoritma

Farrell Jabaar Altafataza
10122057
K1

Daftar Isi

Penjelasan Program



Seperti pada alur program di atas, program akan membaca input dari .txt file yang diberikan. Proses tersebut akan membaca informasi mengenai ukuran dari *board* ($n \; x \; m$), banyak *pieces* (p), dan juga kasus konfigurasi (DEFAULT, CUSTOM, dan PYRAMID). Sayangnya,

program ini hanya bisa menjalankan kasus DEFAULT. Selain itu, program ini juga akan membaca bentuk2 dari *pieces* yang diberikan dari input file.

Lalu, informasi-informasi tersebut digunakan untuk mengisi nilai dari *piece* dan *board. Class Piece* merepresentasikan setiap potongan puzzle dengan semua variasi dari potongan tersebut. Variasi-variasi tersebut yaitu rotasi dan juga *flip*. Variasi-variasi tersebut dilakukan dengan tujuan untuk menggunakan segala kemungkinan penempatan dari setiap potongan.

Class Board berisi grid 2 dimensi yang merepresentasikan papan dalam puzzle ini. Selain itu, *Board* ini juga digunakan untuk mengelola penempatan *Piece* dan *backtracking* menggunakan struktur data *stack*. *Board* dapat mencoba untuk menempatkan potongan puzzle pada suatu titik tertentu dan memeriksa apakah terjadi *overlap* saat penempatan potongan. Lalu, *Board* juga dapat melepaskan potongan dari suatu titik tertentu dan memeriksa apakah seluruh papan telah terisi oleh potongan puzzle atau belum. Selain itu, *Board* juga dapat menjalankan *backtracking* untuk memeriksa semua kemungkinan penempatan potongan puzzle hingga mendapatkan solusi. Pada akhir program, program akan mengeluarkan output berupa solusi jika berhasil, dan akan mengeluarkan pemberitahuan jika gagal menemukan solusi.

Isi Program dan Pengetesan Algoritma

Isi Program

```java
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
import java.util.Stack;

class Input {
    public static int n, m, p;
    public static String s;
    public static List<Piece> pieces = new ArrayList<Piece>();
}

```

```java
class Piece {
    private String name;
    private String[] shape;
    private String color;

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

    public String getName() {
        return name;
    }

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

    public String getColor() {
        return color;
    }

    public String[] rotate() {
        int rows = shape.length;
        int cols = shape[0].length();
        String[] rotated = new String[cols];

        for (int i = 0; i < cols; i++) {
            StringBuilder sb = new StringBuilder(rows);
            for (int j = rows - 1; j >= 0; j--) {
                sb.append(shape[j].charAt(i));
            }
            rotated[i] = sb.toString();
        }
        return rotated;
    }

```

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

    @Override
    public String toString() {
        StringBuilder sb = new StringBuilder();
        sb.append(name).append("\n");
        for (String line : shape) {
            sb.append(line).append("\n");
        }
        return sb.toString();
    }
}
```

```
class Board {
    private char[][] grid;
    private Stack<Piece> pieceStack;
    private int caseCount;

    public Board(int n, int m) {
        grid = new char[n][m];
        pieceStack = new Stack<>();
        caseCount = 0;
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < m; j++) {
                grid[i][j] = '.';
            }
        }
    }

    public boolean placePiece(Piece piece, int row, int col) {
        String[] shape = piece.getShape();
        int pieceHeight = shape.length;
        int pieceWidth = shape[0].length();

        if (row < 0 || col < 0 || row + pieceHeight > grid.length || col + pieceWidth > grid[0].length) {
            return false;
        }

        for (int i = 0; i < pieceHeight; i++) {
            for (int j = 0; j < pieceWidth; j++) {
                if (shape[i].charAt(j) != ' ' && grid[row + i][col + j] != '.' && shape[i].charAt(j) != '.') {
                    return false;
                }
            }
        }

```

```
        for (int i = 0; i < pieceHeight; i++) {
            for (int j = 0; j < pieceWidth; j++) {
                if (shape[i].charAt(j) != ' ') {
                    grid[row + i][col + j] = shape[i].charAt(j);
                }
            }
        }

        pieceStack.push(piece);
        return true;
    }

    public void removePiece(Piece piece, int row, int col) {
        String[] shape = piece.getShape();
        int pieceHeight = shape.length;
        int pieceWidth = shape[0].length();

        for (int i = 0; i < pieceHeight; i++) {
            for (int j = 0; j < pieceWidth; j++) {
                if (shape[i].charAt(j) != ' ') {
                    grid[row + i][col + j] = '.';
                }
            }
        }
    }

    public boolean isFilled() {
        for (int i = 0; i < grid.length; i++) {
            for (int j = 0; j < grid[0].length; j++) {
                if (grid[i][j] == '.') {
                    return false;
                }
            }
        }
        return true;
    }
```

```
    public void printColoredBoard() {
        for (int i = 0; i < grid.length; i++) {
            for (int j = 0; j < grid[0].length; j++) {
                char pieceChar = grid[i][j];
                if (pieceChar != '.') {
                    System.out.print(getColorForPiece(pieceChar) + pieceChar + "\u001B[0m ");
                } else {
                    System.out.print(". ");
                }
            }
            System.out.println();
        }
    }

    private String getColorForPiece(char pieceChar) {
        String[] colors = {
            "\u001B[31m",
            "\u001B[32m",
            "\u001B[33m",
            "\u001B[34m",
            "\u001B[35m",
            "\u001B[36m",
            "\u001B[37m",
            "\u001B[90m",
            "\u001B[91m",
            "\u001B[92m",
            "\u001B[93m",
            "\u001B[94m",
            "\u001B[95m",
            "\u001B[96m",
            "\u001B[97m"
        };

        int index = (pieceChar - 'A') % colors.length;
        return colors[index];
    }
```

```java
public boolean backtrack(int pieceIndex) {
    caseCount++;
    if (pieceIndex >= Input.pieces.size()) {
        return isFilled();
    }

    Piece currentPiece = Input.pieces.get(pieceIndex);
    String[] originalShape = currentPiece.getShape();

    for (int rotation = 0; rotation < 4; rotation++) {
        String[] shapeToTry = originalShape;

        if (rotation == 1) {
            shapeToTry = currentPiece.rotate();
        } else if (rotation == 2) {
            shapeToTry = currentPiece.flip();
        } else if (rotation == 3) {
            shapeToTry = currentPiece.rotate();
            shapeToTry = new Piece(currentPiece.getName(), shapeToTry, currentPiece.getColor()).rotate();
        }

        for (int row = 0; row < Input.n; row++) {
            for (int col = 0; col < Input.m; col++) {
                if (placePiece(new Piece(currentPiece.getName(), shapeToTry, currentPiece.getColor()), row, col)) {
                    if (backtrack(pieceIndex + 1)) {
                        return true;
                    }
                    removePiece(new Piece(currentPiece.getName(), shapeToTry, currentPiece.getColor()), row, col);
                }
            }
        }
    }

    return false;
}
```

```java
    public int getCaseCount() {
        return caseCount;
    }

    public char[][] getGrid() {
        return grid;
    }
}
```

```java
public class Main {
    Run main | Debug main
    public static void main(String[] args) {
        long startTime = System.currentTimeMillis();

        Scanner input = new Scanner(System.in);
        System.out.println("Enter txt file name:");
        String fileName = input.next();

        String filePath = "../test/" + fileName;

        try {
            File file = new File(filePath);
            Scanner fileScanner = new Scanner(file);

            if (fileScanner.hasNextLine()) {
                String line = fileScanner.nextLine();
                String[] parts = line.trim().split("\\s+");

                if (parts.length >= 3) {
                    Input.n = Integer.parseInt(parts[0]);
                    Input.m = Integer.parseInt(parts[1]);
                    Input.p = Integer.parseInt(parts[2]);

                    if (Input.n <= 0 || Input.m <= 0 || Input.p <= 0) {
                        System.out.println("Error: N, M, and P must be positive integers.");
                        return;
                    }
                } else {
                    System.out.println("Error: The first line must contain at least three values for N, M, and P.");
                    return;
                }
            }
```

```
            while (fileScanner.hasNextLine()) {
                String pieceLine = fileScanner.nextLine().trim();

                if (pieceLine.isEmpty()) {
                    continue;
                }

                String pieceName = pieceLine;
                String[] shape = new String[]{pieceLine};

                String color = "\u001B[0m";
                Input.pieces.add(new Piece(pieceName, shape, color));
            }

            fileScanner.close();
        } catch (FileNotFoundException e) {
            System.out.println("Error: File not found.");
            return;
        }

        Board board = new Board(Input.n, Input.m);
        System.out.println("Initial Board:");
        board.printColoredBoard();

        long searchStartTime = System.currentTimeMillis();
        boolean allPiecesPlaced = board.backtrack(0);
        long searchEndTime = System.currentTimeMillis();

        if (allPiecesPlaced) {
            System.out.println("All pieces placed successfully.");
            System.out.println("Final Board:");
            board.printColoredBoard();
        } else {
            System.out.println("Failed to place all pieces.");
        }
```

```
        System.out.println("Waktu pencarian: " + (searchEndTime - searchStartTime) + " ms");
        System.out.println("Banyak kasus yang ditinjau: " + board.getCaseCount());

        System.out.println("Apakah anda ingin menyimpan solusi? (ya/tidak)");
        String saveResponse = input.next().trim().toLowerCase();

        if (saveResponse.equals("ya")) {
            String outputFileName = "solution_" + fileName.replace(".txt", "") + ".txt";
            try {
                FileWriter writer = new FileWriter(outputFileName);
                for (int i = 0; i < board.getGrid().length; i++) {
                    for (int j = 0; j < board.getGrid()[0].length; j++) {
                        writer.write(board.getGrid()[i][j]);
                    }
                    writer.write("\n");
                }
                writer.close();
                System.out.println("Solusi telah disimpan dalam file " + outputFileName);
            } catch (IOException e) {
                System.out.println("Error: Gagal menyimpan solusi.");
            }
        } else {
            System.out.println("Solution file not saved.");
        }

        long endTime = System.currentTimeMillis();
        long elapsedTime = endTime - startTime;
        System.out.println("Total waktu eksekusi: " + elapsedTime + " milliseconds");
    }
}
```

Pengetesan Algoritma

Test Case 1

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

```
Enter txt file name:
1.txt
Initial Board:
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .
All pieces placed successfully.
Final Board:
A A A B C
B B C C D
D D E E E
E E F F F
F F G G G
Waktu pencarian: 0 ms
Banyak kasus yang ditinjau: 16
Apakah anda ingin menyimpan solusi? (ya/tidak)
```

Test Case 2

```
3 3 3
DEFAULT
AA
B
B
BB
CC
C
```

```
Enter txt file name:
2.txt
Initial Board:
. . .
. . .
. . .
All pieces placed successfully.
Final Board:
A A B
B B B
C C C
Waktu pencarian: 0 ms
Banyak kasus yang ditinjau: 7
Apakah anda ingin menyimpan solusi? (ya/tidak)
```

Test Case 3

```
6 6 8
DEFAULT
SS
SS
PPPP
PPP
PP
QQ
QQQ
RR
```

```
XX
XX
Y
YYY
ZZ
ZZ
W
WWW
```

```
Enter txt file name:
3.txt
Initial Board:
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
All pieces placed successfully.
Final Board:
S S S S P P
P P P P Q Q
P P P Q Q Q
R R X X X X
Y Y Y Y Z Z
Z Z W W W W
Waktu pencarian: 0 ms
Banyak kasus yang ditinjau: 17
Apakah anda ingin menyimpan solusi? (ya/tidak)
```

Test Case 4

```
4 4 5
DEFAULT
VVV
V
E
E
III
II
TTT
```

```
HH
```

```
Enter txt file name:
4.txt
Initial Board:
. . . .
. . . .
. . . .
. . . .
All pieces placed successfully.
Final Board:
V V V V
E E I I
I I I H
T T T H
Waktu pencarian: 0 ms
Banyak kasus yang ditinjau: 9
Apakah anda ingin menyimpan solusi? (ya/tidak)
```

Test Case 5

```
4 4 2
DEFAULT
AAA
AAA
AAA
BBBB
BBB
```

```
Enter txt file name:
5.txt
Initial Board:
. . . .
. . . .
. . . .
. . . .
All pieces placed successfully.
Final Board:
A A A B
A A A B
A A A B
B B B B
Waktu pencarian: 0 ms
Banyak kasus yang ditinjau: 6
Apakah anda ingin menyimpan solusi? (ya/tidak)
```

Test Case 6

```
7 5 6
DEFAULT
PPPPP
PPP
QQQQ
QQ
RRRRR
RRRR
SS
SS
SS
T
UUUU
U
```

```
Enter txt file name:
6.txt
Initial Board:
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .
All pieces placed successfully.
Final Board:
P P P P P
P P P Q Q
Q Q Q Q T
R R R R R
R R R R S
S S S S S
U U U U U
Waktu pencarian: 3 ms
Banyak kasus yang ditinjau: 113
Apakah anda ingin menyimpan solusi? (ya/tidak)
```

Test Case 7

```
6 4 7
DEFAULT
NN
NN
NN
M
MMM
G
GGG
TT
EE
D
K
KKK
L
```

```
Initial Board:
. . . .
. . . .
. . . .
. . . .
. . . .
. . . .
All pieces placed successfully.
Final Board:
N N N N
N N M G
M M M D
G G G K
T T E E
K K K L
Waktu pencarian: 0 ms
Banyak kasus yang ditinjau: 14
Apakah anda ingin menyimpan solusi? (ya/tidak)
```

Lampiran

[Github Repository]