LAPORAN TUGAS KECIL 1

IF2211 STRATEGI ALGORITMA

Penyelesaian IQ Puzzler Pro dengan Algoritma Brute Force

Disusun oleh:

Jethro Jens Norbert Simatupang – 13523081

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

JL. GANESA 10, BANDUNG 40132

2025

DAFTAR ISI

BAB 1

PENDAHULUAN

1.1. Latar Belakang

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.

Permainan dimulai dengan papan yang kosong. Pemain dapat meletakkan *piece* sedemikian sehingga tidak ada *piece* yang bertumpang tindih (kecuali dalam kasus 3D). Setiap *piece* dapat dirotasikan maupun dicerminkan. Puzzle dinyatakan selesai jika dan hanya jika papan terisi penuh dan seluruh *piece* berhasil diletakkan.

Pencarian solusi dari permainan *IQ Puzzler Pro* dapat dimodelkan sebagai masalah pencarian dalam ruang keadaan. Salah satu metode yang dapat digunakan untuk menemukan solusi permainan ini adalah algoritma brute force.

1.2. Tujuan

Tujuan tugas kecil ini adalah:

a. Mengimplementasikan algoritma brute force untuk menemukan solusi permainan *IQ Puzzler Pro*
b. Mengevaluasi keberadaan solusi untuk sebuah konfigurasi awal

1.3. Teori Singkat

Algoritma brute force adalah pendekatan yang lurus atau lempang (straightforward) untuk memecahkan suatu persoalan. Algoritma brute force adalah pendekatan pencarian solusi yang dilakukan dengan mencoba semua kemungkinan kombinasi hingga menemukan solusi yang benar atau memastikan bahwa tidak ada solusi yang mungkin. Algoritma ini tidak mengandalkan heuristik atau optimasi, melainkan menguji setiap kemungkinan solusi satu per satu secara sistematis.

Algoritma brute force menjadi salah satu metode yang dapat dimanfaatkan dalam permainan *IQ Puzzler Pro*. Dengan algoritma ini, setiap kemungkinan posisi masing-masing *piece* (termasuk rotasi dan pencerminannya) dicoba secara *backtracking* hingga ditemukan solusi permainan.

BAB 2

ALGORITMA

2.1. Notasi *Pseudocode*

a. public static void main(String[] args)

```
procedure main()
    input(sc)
    try
        readInput(sc) {Membaca input file}
        startTime ← currentTimeMillis() {Memulai timer}
        foundSolution ← solve(0) {Mencari solusi puzzle}
        elapsedTime ← currentTimeMillis() - startTime {Menghentikan timer}

        {Menyusun output}
        output ← StringBuilder()
        if foundSolution then
            output.append(printBoard())
        else
            output.append("Tidak ada solusi!\n")
        output.append("\nWaktu pencarian: ") + elapsedTime + " ms\n"
        output.append("\nBanyak kasus yang ditinjau: ") + iterations + "\n"

        {Menampilkan output}
        output(output)

        {Menyimpan solusi}
        output("Apakah anda ingin menyimpan solusi? (ya/tidak): ")
        input(response)
        response ← trim(lowercase(response))
        if response = "ya" then
            saveToFile(toString(output))

    catch IOException e
        output(getMessage(e))
    finally
        close(sc)
```

b. private static void saveToFile(String content)

```
procedure saveToFile(input content: string)
    input(scanner)
    output("Masukkan nama file output: ")
    input(fileName)
    outputPath ← "test/output/" + fileName

    try
        writer ← FileWriter(outputPath)

        {Menyimpan papan tanpa warna}
        write(writer, printBoardWithoutColors())

        {Menyimpan waktu dan jumlah kasus}
        write(writer, "\nWaktu pencarian: " + (currentTimeMillis() - startTime) +
" ms\n")
        write(writer, "\nBanyak kasus yang ditinjau: " + iterations + "\n")

        output("Solusi berhasil disimpan dalam " + fileName)
    catch IOException e
        output("Gagal menyimpan solusi: " + getMessage(e))
```

c. static void readInput(Scanner sc) throws IOException

```
procedure readInput(input sc: scanner) throws IOException
    output("Masukkan nama file test case: ")
    input(fileName)
    inputPath ← "test/input/" + fileName

    try
        br ← BufferedReader(FileReader(inputPath))
    catch FileNotFoundException e
        throw IOException("File tidak ditemukan!")

    {Membaca baris pertama untuk mendapatkan ukuran papan dan jumlah piece}
    firstLine ← readLine(br)
    if firstLine = null then
        throw IOException("Format file input tidak valid!")

    dims ← split(firstLine, " ")
    if length(dims) < 3 then
        throw IOException("Format file input tidak valid!")

    N ← toInteger(dims[0])
    M ← toInteger(dims[1])
    P ← toInteger(dims[2])

    {Membaca baris "DEFAULT"}
    defaultLine ← readLine(br)
    if defaultLine = null or defaultLine ≠ "DEFAULT" then
        throw IOException("Format file input tidak valid!")

    {Inisialisasi papan}
    board ← array[N][M] of char
    while each row in board do
        fill(row, ".")

    {Membaca piece-piece}
    previousSymbol ← "\0"
    shapeLines ← List<String>()

    while (line ← readLine(br)) ≠ null do
        {Mencari simbol piece pada baris saat ini}
        currentSymbol ← "\0"
        while each ch in toCharArray(line) do
            if ch ≠ " " then
                currentSymbol ← ch
                break

        {Jika simbol berubah dan shapeLines tidak kosong, piece sebelumnya
diproses}
        if currentSymbol ≠ previousSymbol and not empty(shapeLines) then
            processPiece(shapeLines, previousSymbol)
            clear(shapeLines)

        {Menambahkan baris ke shapeLines}
        append(shapeLines, line)
        previousSymbol ← currentSymbol

    {Memproses piece terakhir}
    if not empty(shapeLines) then
        processPiece(shapeLines, previousSymbol)

    close(br)

    {Memvalidasi jumlah piece}
```

```
    depend on size(pieces)
        size(pieces) > P : throw IOException("Piece berlebih!")
        size(pieces) < P : throw IOException("Piece kurang!")

    {Memvalidasi total ukuran piece}
    totalSize ← 0
    while each piece in pieces do
        while each row in piece do
            while each cell in row do
                if cell then totalSize ← totalSize + 1

    if totalSize ≠ N * M then
        throw IOException("Ukuran piece tidak sesuai!")
```

d. static void processPiece(List<String> shapeLines, char label) throws IOException

```
procedure processPiece(input shapeLines: List, input label: character) throws
IOException
    if shapeLines.isEmpty() then {Memastikan piece tidak kosong}
        throw IOException("Potongan puzzle tidak memiliki bentuk!")

    pieceSymbols.add(label) {Menyimpan simbol piece}

    {Menentukan ukuran maksimum baris dan kolom dari bentuk piece}
    shapeRows ← shapeLines.size()
    shapeCols ← 0
    while s in shapeLines do
        shapeCols ← max(shapeCols, s.length())

    {Inisialisasi array boolean untuk merepresentasikan bentuk piece}
    shape ← array(shapeRows, shapeCols) of boolean

    {Mengisi array boolean berdasarkan karakter dalam shapeLines}
    r traversal [0 .. shapeRows - 1]
        row ← shapeLines.get(r)
        c traversal [0 .. row.length() - 1]
            if row.charAt(c) = label then
                shape[r][c] ← true

    pieces.add(shape) {Menambahkan bentuk piece ke daftar pieces}
```

e. static boolean solve(int index)

```
function solve(index: integer) → boolean
    if index = P then {Jika semua piece telah ditempatkan, solusi ditemukan}
        → true

    iterations ← iterations + 1
    label ← pieceSymbols.get(index) {Mendapatkan simbol untuk piece saat ini}

    {Mencoba semua transformasi untuk piece saat ini}
    while shape in generateTransformations(pieces.get(index)) do
        r traversal [0 .. N - shape.length]
            c traversal [0 .. M - shape[0].length]
                if canPlace(shape, r, c) then {Mencoba semua posisi untuk
transformasi saat ini}
                    placePiece(shape, r, c, label)
                    if solve(index + 1) then → true {Mencoba piece berikutnya}
                    removePiece(shape, r, c) {Backtracking}

    → false
```

f. static boolean canPlace(boolean[][] shape, int row, int col)

```
function canPlace(shape: array [][] of boolean, row: integer, col: integer) → boolean
    {Iterasi melalui semua sel dalam bentuk piece}
    r traversal [0 .. shape.length - 1]
        c traversal [0 .. shape[r].length - 1]
            if shape[r][c] and board[row + r][col + c] ≠ "." then {Jika posisi papan sudah terisi, maka tidak bisa ditempatkan}
                → false

    → true
```

g. static void placePiece(boolean[][] shape, int row, int col, char label)

```
procedure placePiece(shape: array [][] of boolean, row: integer, col: integer, label: character)
    {Iterasi melalui semua sel dalam bentuk piece}
    r traversal [0 .. shape.length - 1]
        c traversal [0 .. shape[r].length - 1]
            if shape[r][c] then {Jika sel dalam shape bernilai true, piece ditempatkan pada posisi yang sesuai}
                board[row + r][col + c] ← label
```

h. static void removePiece(boolean[][] shape, int row, int col)

```
procedure removePiece(shape: array [][] of boolean, row: integer, col: integer)
    {Iterasi melalui semua sel dalam bentuk piece}
    r traversal [0 .. shape.length - 1]
        c traversal [0 .. shape[r].length - 1]
            if shape[r][c] then {Jika sel dalam shape bernilai true, hapus dari papan dengan mengganti kembali ke "."}
                board[row + r][col + c] ← "."
```

i. static List<boolean[][]> generateTransformations(boolean[][] shape)

```
function generateTransformations(shape:  array [][] of boolean)
    {Membuat daftar untuk menyimpan semua transformasi (rotasi dan flip)}
    transformations ← list()
    i traversal [0 .. 3]
        shape ← rotate90(shape)
        transformations.add(shape)
        transformations.add(flip(shape))
    → transformations
```

j. static boolean[][] rotate90(boolean[][] shape)

```
function rotate90(shape: array [][] of boolean)
    {Memutar piece 90 derajat searah jarum jam}
    rows ← length(shape)
    cols ← length(shape[0])
    rotated ← array[cols][rows]
    i traversal [0 .. rows - 1]
        j traversal [0 .. cols - 1]
```

```
            rotated[j][rows - i - 1] ← shape[i][j]

→ rotated
```

k. static boolean[][] flip(boolean[][] shape)

```
function flip(shape: array [][] of boolean)
    {Membalik/merefleksi piece}
    rows ← length(shape)
    cols ← length(shape[0])
    flipped ← array[rows][cols]

    i traversal [0 .. rows – 1]
        j traversal [0 .. cols – 1]
            flipped[i][cols - j - 1] ← shape[i][j]

→ flipped
```

l. static String printBoard()

```
function printBoard()
    {Daftar warna ANSI untuk setiap piece}
    colors ← [
        "\u001B[31m", {Merah}
        "\u001B[32m", {Hijau}
        "\u001B[33m", {Kuning}
        "\u001B[34m", {Biru}
        "\u001B[35m", {Magenta}
        "\u001B[36m", {Cyan}
        "\u001B[37m", {Putih}
        "\u001B[91m", {Merah Terang}
        "\u001B[92m", {Hijau Terang}
        "\u001B[93m", {Kuning Terang}
        "\u001B[94m", {Biru Terang}
        "\u001B[95m", {Magenta Terang}
        "\u001B[96m"  {Cyan Terang}
    ]

    {Reset warna ke default setelah mencetak}
    resetColor ← "\u001B[0m"

    {Map untuk menyimpan warna berdasarkan label piece}
    colorMap ← dictionary()

    i traversal [0 .. length(pieceSymbols) – 1]
        colorMap[pieceSymbols[i]] ← colors[i mod length(colors)]

    {Menyusun string papan}
    sb ← ""

    {Mencetak papan dengan warna}
    while row in board do
        while c in row do
            if c ≠ "." then
                sb ← sb + colorMap[c] + c + " " + resetColor
            else
                sb ← sb + ". "
        sb ← sb + "\n"

   → sb
```

m. static String printBoardWithoutColors()

```
function printBoardWithoutColors()
    {Menyusun string papan tanpa warna (untuk save file)}
    sb ← ""

    while row in board do
        while c in row do
            sb ← sb + c + " "
        sb ← sb + "\n"

→ sb
```

2.2. Penjelasan Algoritma

Algoritma yang dibuat menggunakan algoritma brute force dengan metode *backtracking*. Berikut adalah penjelasan langkah-langkahnya:

a. Inisialisasi dan Pembacaan Input

Program dimulai dengan membaca input dari pengguna, yaitu nama file yang berisi spesifikasi puzzle. File ini menyimpan ukuran papan (N × M), jumlah *piece* (P), serta bentuk setiap *piece* dalam format karakter. Program membaca file ini dan memproses setiap *piece* menjadi representasi matriks boolean dan menginisialisasi papan sebagai matriks karakter N × M dengan tanda titik (.) untuk posisi yang kosong.

Fungsi utama yang berperan:

- readInput(sc): Membaca isi file dan mengonversi setiap *piece* menjadi representasi matriks
- processPieces(shapeLines, char label): Mengubah *piece* dari format teks ke dalam format array dua dimensi (boolean matrix)

b. Validasi Data

Setelah membaca data, program melakukan validasi, seperti memastikan jumlah *piece* sesuai dengan yang diharapkan dan total luas semua *piece* sama dengan luas papan. Jika ada ketidaksesuaian, program akan mengeluarkan pesan kesalahan.

Fungsi yang berperan:

- readInput(sc): Membaca isi file sekaligus memvalidasi nama dan isi file

c. *Backtracking* untuk Menyusun Puzzle (Algoritma Brute Force)

Program menggunakan algoritma *backtracking* untuk mencoba menyusun *piece* puzzle di papan. Algoritma ini merupakan algoritma brute force yang memeriksa setiap kemungkinan posisi dan transformasi dari *piece* yang tersedia secara berurut. Untuk setiap *piece*, program mencoba semua kemungkinan transformasi (rotasi 90°, 180°, 270°, serta flip). Setiap transformasi diuji di setiap posisi untuk melihat apakah *piece* tersebut dapat ditempatkan di papan tanpa tumpang tindih. Jika suatu *piece* dapat ditempatkan, program mengisinya di papan menggunakan simbol yang sesuai. Jika solusi ditemukan, program berhenti. Namun, jika tidak suatu *piece* tidak dapat ditempatkan di posisi manapun dan dalam bentuk transformasi manapun, program akan menghapus *piece* yang ditempatkan terakhir dan kembali (*backtrack*) ke langkah sebelumnya untuk mencoba kemungkinan lain.

Fungsi yang berperan:

- solve(index): Fungsi utama yang menjalankan *backtracking*
- canPlace(shape, row, col): Mengecek apakah *piece* bisa ditempatkan di lokasi tertentu
- placePiece(shape, row, col, label): Menempatkan *piece* di papan

- removePiece(shape, row, col): Menghapus *piece* yang sudah ditempatkan untuk melakukan *backtrack*

1. rotate90(shape): Memutar *piece* 90° searah jarum jam
2. flip(shape): Membalik *piece*
3. generateTransformations(shape): Mencoba semua transformasi

d. Output Hasil

Jika solusi ditemukan, program mencetak papan hasil penyusunan. Jika tidak, program menginformasikan bahwa tidak ada solusi. Program juga mencatat waktu eksekusi dan jumlah iterasi yang dilakukan selama pencarian solusi.

Fungsi yang berperan:

- printBoard(): Mencetak papan

e. Menyimpan Solusi ke File

Setelah solusi ditampilkan, pengguna diberikan opsi untuk menyimpannya ke dalam file. Jika pengguna memilih untuk menyimpan, program meminta nama file yang diinput pengguna dan menyimpan hasil penyusunan papan tanpa warna serta informasi waktu dan jumlah iterasi dalam file output.

Fungsi yang berperan:

- printBoardWithoutColors(): Menyusun papan tanpa warna untuk disimpan ke file
- saveToFile(content): Menyimpan hasil ke dalam file

BAB 3

SOURCE CODE

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

public class IQPuzzlerPro {
    static int N, M, P;
    static char[][] board;
    static List<boolean[][]> pieces = new ArrayList<>();
    static List<Character> pieceSymbols = new ArrayList<>();
    static long iterations = 0;
    static long startTime;

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        try {
            readInput(sc); // Membaca input file
            startTime = System.currentTimeMillis(); // Memulai timer
            boolean foundSolution = solve(0); // Mencari solusi puzzle
            long elapsedTime = System.currentTimeMillis() - startTime; // Menghentikan timer

            // Menyusun output
            StringBuilder output = new StringBuilder();
            if (foundSolution) {
                output.append(printBoard());
            } else {
                output.append("Tidak ada solusi!\n");
            }
            output.append("\nWaktu pencarian: ").append(elapsedTime).append(" ms\n");
            output.append("\nBanyak kasus yang ditinjau: ").append(iterations).append("\n");

            // Menampilkan output
            System.out.println(output);

            // Menyimpan solusi
            System.out.print("Apakah anda ingin menyimpan solusi? (ya/tidak): ");
            String response = sc.nextLine().trim().toLowerCase();
            if (response.equals("ya")) {
                saveToFile(output.toString());
            }

        } catch (IOException e) {
            System.out.println(e.getMessage());
        } finally {
            sc.close();
        }
    }

    private static void saveToFile(String content) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Masukkan nama file output: ");
        String fileName = scanner.nextLine();
        String outputPath = "test/output/" + fileName;

        try (FileWriter writer = new FileWriter(outputPath)) {
            // Menyimpan papan tanpa warna
            writer.write(printBoardWithoutColors());

            // Menyimpan waktu dan jumlah kasus
            writer.write("\nWaktu pencarian: " + (System.currentTimeMillis() - startTime) + " ms\n");
            writer.write("\nBanyak kasus yang ditinjau: " + iterations + "\n");

            System.out.println("Solusi berhasil disimpan dalam " + fileName);
        } catch (IOException e) {
            System.out.println("Gagal menyimpan solusi: " + e.getMessage());
        }
    }
```

```java
static void readInput(Scanner sc) throws IOException {
    System.out.print("Masukkan nama file test case: ");
    String fileName = sc.nextLine(); // Meminta input nama file
    String inputPath = "test/input/" + fileName;

    BufferedReader br;
    try {
        br = new BufferedReader(new FileReader(inputPath));
    } catch (FileNotFoundException e) {
        throw new IOException("File tidak ditemukan!");
    }

    // Membaca baris pertama untuk mendapatkan ukuran papan dan jumlah piece
    String firstLine = br.readLine();
    if (firstLine == null) {
        throw new IOException("Format file input tidak valid!");
    }
    String[] dims = firstLine.split(" ");
    if (dims.length < 3) {
        throw new IOException("Format file input tidak valid!");
    }
    N = Integer.parseInt(dims[0]);
    M = Integer.parseInt(dims[1]);
    P = Integer.parseInt(dims[2]);

    // Membaca baris "DEFAULT"
    String defaultLine = br.readLine();
    if (defaultLine == null || !defaultLine.equals("DEFAULT")) {
        throw new IOException("Format file input tidak valid!");
    }

    // Inisialisasi papan
    board = new char[N][M];
    for (char[] row : board) Arrays.fill(row, '.');

    // Membaca piece-piece
    char previousSymbol = '\0'; // Menyimpan simbol piece sebelumnya
    List<String> shapeLines = new ArrayList<>(); // Menyimpan bentuk piece saat ini
    String line;
    while ((line = br.readLine()) != null) {
        // Mencari simbol piece pada baris saat ini
        char currentSymbol = '\0';
        for (char ch : line.toCharArray()) {
            if (ch != ' ') {
                currentSymbol = ch;
                break;
            }
        }
        // Jika simbol berubah dan shapeLines tidak kosong, piece sebelumnya diproses
        if (currentSymbol != previousSymbol && !shapeLines.isEmpty()) {
            processPiece(shapeLines, previousSymbol);
            shapeLines.clear(); // Mengosongkan shapeLines untuk piece berikutnya
        }
        // Menambahkan baris ke shapeLines
        shapeLines.add(line);
        previousSymbol = currentSymbol; // Memperbarui label sebelumnya
    }

    // Memproses piece terakhir
    if (!shapeLines.isEmpty()) {
        processPiece(shapeLines, previousSymbol);
    }

    br.close();

    // Memvalidasi jumlah piece
    if (pieces.size() > P) {
        throw new IOException("Piece berlebih!");
    } else if (pieces.size() < P) {
        throw new IOException("Piece kurang!");
    }

    // Memvalidasi total ukuran piece
    int totalSize = 0;
    for (boolean[][] piece : pieces) {
        for (boolean[] row : piece) {
            for (boolean cell : row) {
                if (cell) totalSize++;
            }
        }
    }
    if (totalSize != N * M) {
        throw new IOException("Ukuran piece tidak sesuai!");
    }
}
```

```java
static void processPiece(List<String> shapeLines, char label) throws IOException {
    if (shapeLines.isEmpty()) { // Memastikan piece tidak kosong
        throw new IOException("Potongan puzzle tidak memiliki bentuk!");
    }

    pieceSymbols.add(label); // Menyimpan simbol piece

    // Menentukan ukuran maksimum baris dan kolom dari bentuk piece
    int shapeRows = shapeLines.size();
    int shapeCols = 0;
    for (String s : shapeLines) {
        shapeCols = Math.max(shapeCols, s.length());
    }

    // Inisialisasi array boolean untuk merepresentasikan bentuk piece
    boolean[][] shape = new boolean[shapeRows][shapeCols];

    // Mengisi array boolean berdasarkan karakter dalam shapeLines
    for (int r = 0; r < shapeRows; r++) {
        String row = shapeLines.get(r);
        for (int c = 0; c < row.length(); c++) {
            if (row.charAt(c) == label) {
                shape[r][c] = true;
            }
        }
    }
    pieces.add(shape); // Menambahkan bentuk piece ke daftar pieces
}

static boolean solve(int index) {
    if (index == P) return true; // Jika semua piece telah ditempatkan, solusi ditemukan
    iterations++;
    char label = pieceSymbols.get(index); // Mendapatkan simbol untuk piece saat ini

    for (boolean[][] shape : generateTransformations(pieces.get(index))) { // Mencoba semua transformasi untuk piece saat ini
        for (int r = 0; r <= N - shape.length; r++) {
            for (int c = 0; c <= M - shape[0].length; c++) {
                if (canPlace(shape, r, c)) { // Mencoba semua posisi untuk transformasi saat ini
                    placePiece(shape, r, c, label);
                    if (solve(index + 1)) return true; // Mencoba piece berikutnya
                    removePiece(shape, r, c); // Backtracking
                }
            }
        }
    }
    return false;
}

static boolean canPlace(boolean[][] shape, int row, int col) {
    // Iterasi melalui semua sel dalam bentuk piece
    for (int r = 0; r < shape.length; r++) {
        for (int c = 0; c < shape[r].length; c++) {
            if (shape[r][c] && board[row + r][col + c] != '.') { // Jika posisi papan sudah terisi, maka tidak bisa ditempatkan
                return false;
            }
        }
    }
    return true;
}

static void placePiece(boolean[][] shape, int row, int col, char label) {
    // Iterasi melalui semua sel dalam bentuk piece
    for (int r = 0; r < shape.length; r++) {
        for (int c = 0; c < shape[r].length; c++) {
            if (shape[r][c]) { // Jika sel dalam shape bernilai true, piece ditempatkan pada posisi yang sesuai
                board[row + r][col + c] = label;
            }
        }
    }
}

static void removePiece(boolean[][] shape, int row, int col) {
    // Iterasi melalui semua sel dalam bentuk piece
    for (int r = 0; r < shape.length; r++) {
        for (int c = 0; c < shape[r].length; c++) {
            if (shape[r][c]) { // Jika sel dalam shape bernilai true, hapus dari papan dengan mengganti kembali ke '.'
                board[row + r][col + c] = '.';
            }
        }
    }
}
```

```java
    static List<boolean[][]> generateTransformations(boolean[][] shape) {
        // Membuat daftar untuk menyimpan semua transformasi (rotasi dan flip)
        List<boolean[][]> transformations = new ArrayList<>();
        for (int i = 0; i < 4; i++) {
            shape = rotate90(shape);
            transformations.add(shape);
            transformations.add(flip(shape));
        }
        return transformations;
    }

    static boolean[][] rotate90(boolean[][] shape) {
        // Memutar piece 90 derajat searah jarum jam
        int rows = shape.length, cols = shape[0].length;
        boolean[][] rotated = new boolean[cols][rows];
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                rotated[j][rows - i - 1] = shape[i][j];
            }
        }
        return rotated;
    }

    static boolean[][] flip(boolean[][] shape) {
        // Membalik/merefleksi piece
        int rows = shape.length, cols = shape[0].length;
        boolean[][] flipped = new boolean[rows][cols];
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                flipped[i][cols - j - 1] = shape[i][j];
            }
        }
        return flipped;
    }

    static String printBoard() {
        // Daftar warna ANSI untuk setiap piece
        String[] colors = {
            "\u001B[31m", // Merah
            "\u001B[32m", // Hijau
            "\u001B[33m", // Kuning
            "\u001B[34m", // Biru
            "\u001B[35m", // Magenta
            "\u001B[36m", // Cyan
            "\u001B[37m", // Putih
            "\u001B[91m", // Merah Terang
            "\u001B[92m", // Hijau Terang
            "\u001B[93m", // Kuning Terang
            "\u001B[94m", // Biru Terang
            "\u001B[95m", // Magenta Terang
            "\u001B[96m"  // Cyan Terang
        };

        // Reset warna ke default setelah mencetak
        String resetColor = "\u001B[0m";

        // Map untuk menyimpan warna berdasarkan label piece
        Map<Character, String> colorMap = new HashMap<>();
        for (int i = 0; i < pieceSymbols.size(); i++) {
            colorMap.put(pieceSymbols.get(i), colors[i % colors.length]);
        }

        // Menyusun string papan
        StringBuilder sb = new StringBuilder();

        // Mencetak papan dengan warna
        for (char[] row : board) {
            for (char c : row) {
                if (c != '.') {
                    sb.append(colorMap.get(c)).append(c).append(" ").append(resetColor);
                } else {
                    sb.append(". ");
                }
            }
            sb.append("\n");
        }

        return sb.toString();
    }

    static String printBoardWithoutColors() {
        // Menyusun string papan tanpa warna (untuk save file)
        StringBuilder sb = new StringBuilder();
        for (char[] row : board) {
            for (char c : row) {
                sb.append(c).append(" ");
            }
            sb.append("\n");
        }

        return sb.toString();
    }

}
```

BAB 4

EKSPERIMEN

4.1. Test Case 1

Input:

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

Output:

```
Masukkan nama file test case: input1.txt
A A B B D
A E B D D
E E C C F
E E C F F
G G G F F

Waktu pencarian: 13 ms

Banyak kasus yang ditinjau: 1714
```

4.2. Test Case 2

Input:

```
5 6 5
DEFAULT
   C
   C
   C
   C
CCCC
DDDD
   D
   D
   D
   D
B
AAA
A A
AAA
EEEEE
```

Output:

```
Masukkan nama file test case: input2.txt
D D D D D E
C A A A D E
C A B A D E
C A A A D E
C C C C C E

Waktu pencarian: 32 ms

Banyak kasus yang ditinjau: 6535
```

4.3. Test Case 3

Input:

```
5 6 5
DEFAULT
   C
   C
   C
   C
CCCC
DDDD
   D
   D
   D
   D
B
AAA
A A
AAA
EEEEE
```

Output:

```
Masukkan nama file test case: input3.txt
A A A E C C H
B B A E E C C
B B F F E G C
B D D F E G C
D D F F F G C

Waktu pencarian: 28 ms

Banyak kasus yang ditinjau: 3202
```

4.4. Test Case 4

Input:

```
4 7 5
DEFAULT
 AA
AAAAA
 A
 BBB
   B
BBBB
C
C
C
C
FF
F
EEE
 EE
```

Output:

```
Masukkan nama file test case: input4.txt
B B B B A F F
B A A A A A F
B B B A A E E
C C C C E E E

Waktu pencarian: 0 ms

Banyak kasus yang ditinjau: 6
```

4.5. Test Case 5

Input:

```
3 3 2
DEFAULT
 A
AAA
CCC
C C
```

Output:

```
Masukkan nama file test case: input5.txt
A C C
A A C
A C C

Waktu pencarian: 0 ms

Banyak kasus yang ditinjau: 2
```

4.6. Test Case 6

Input:

```
4 6 6
DEFAULT
 F
FFFFF
C
CC
CCC
C
DD
D
BB
B
EE
 E
GG
```

Output:

```
Masukkan nama file test case: input6.txt
F F F F F E
B B C F E E
B C C D D G
C C C C D G

Waktu pencarian: 1 ms

Banyak kasus yang ditinjau: 32
```

4.7. Test Case 7

Input:

```
4 4 4
DEFAULT
AAAA
 AA
B
BB
 B
C
C
 D
DD
D
```

Output:

```
Masukkan nama file test case: input7.txt
Tidak ada solusi!

Waktu pencarian: 9 ms

Banyak kasus yang ditinjau: 2233
```

4.8. Test Case 8

Input:

```
3 3 3
DEFAULT
AA
 AA
B
BBB
```

Output:

```
Masukkan nama file test case: input8.txt
Piece kurang!
```

4.9. Test Case 9

Input:

```
3 3 3
DEFAULT
AA
 AA
C
B
BBBB
```

Output:

```
Masukkan nama file test case: input9.txt
Ukuran piece tidak sesuai!
```

4.10. Test Case 10

Input:

```
3 6 5
AAA
A
BB
 BB
CC
DDD
DD
 D
EE
```

Output:

```
Masukkan nama file test case: input10.txt
Format file input tidak valid!
```

LAMPIRAN

1. **Repository GitHub**
   Berikut adalah pranala ke repository GitHub yang berisi kode sumber dan dokumentasi proyek ini:
   https://github.com/JethroJNS/Tucil1_13523081.git
2. **Tabel Evaluasi**

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	✓	