LAPORAN TUGAS KECIL 1
IF2211 STRATEGI ALGORITMA

PENYELESAIAN IQ PUZZLER PRO DENGAN ALGORITMA *BRUTE FORCE*

Oleh :

Muhammad Timur Kanigara

13523055

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2025

Daftar Isi

BAB I

Deskripsi Program

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**.

(Paragraf di atas dikutip dari :
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/Tucil1-Stima-2025.pdf).

BAB II

Algoritma Program

Pada program ini digunakan algoritma brute-force untuk menemukan semua solusi yang mungkin untuk menyelesaikan permainan IQ Puzzler Pro. Adapun langkah-langkah yang digunakan pada program yaitu:

1. **Alur Kerja Program** Program dimulai dengan membaca file input yang berisi:
    a. Ukuran papan (N x M)
    b. Jumlah bentuk yang harus ditempatkan (P)
    c. Mode penempatan awal (misalnya, "DEFAULT")
    d. Representasi bentuk dengan karakter huruf

    Setelah membaca input, program menginisialisasi papan dengan tanda titik ('.') untuk menandakan posisi kosong. Kemudian, program mencoba menyelesaikan puzzle dengan menggunakan algoritma brute force.

2. **Implementasi Algoritma Brute Force** Algoritma brute force bekerja dengan mencoba setiap kemungkinan peletakan bentuk pada papan hingga menemukan solusi yang memenuhi syarat. Berikut langkah-langkah utama dalam algoritma ini:
    a. **Mencoba Menempatkan Setiap Bentuk:**
        i. Program mengambil satu bentuk dan mencoba menempatkannya di semua posisi yang mungkin pada papan.
        ii. Untuk setiap posisi, program mengecek apakah bentuk dapat ditempatkan tanpa tumpang tindih dengan bentuk lain.
    b. **Menghasilkan Semua Orientasi Bentuk:**
        i. Untuk setiap bentuk, program menghasilkan semua kemungkinan rotasi (0°, 90°, 180°, 270°) dan pencerminan.
        ii. Hal ini dilakukan untuk memastikan bahwa bentuk dapat ditempatkan dalam semua orientasi yang valid.
    c. **Rekursi dan Backtracking:**
        i. Jika suatu bentuk dapat ditempatkan, program melanjutkan dengan menempatkan bentuk berikutnya secara rekursif.
        ii. Jika menemukan bahwa tidak ada solusi dari posisi tertentu, program melakukan backtracking dengan menghapus bentuk terakhir yang ditempatkan dan mencoba kemungkinan lain.
    d. **Menghitung Jumlah Percobaan:**
        i. Program mencatat jumlah iterasi yang telah dilakukan selama proses pencarian solusi.
        ii. Ini membantu dalam memahami kompleksitas dan efisiensi pencarian brute force

yang dilakukan.

3. **Fitur Tambahan**
    a. **Menampilkan Solusi dengan Warna:** Program menampilkan hasil dengan warna berbeda untuk setiap bentuk, menggunakan ANSI escape codes untuk memberikan visualisasi yang lebih jelas.
    b. **Menyimpan Solusi ke File:** Pengguna dapat menyimpan solusi dalam file dengan memberikan nama file yang diinginkan.

Algoritma brute force yang digunakan dalam program ini memastikan bahwa semua kemungkinan solusi dicoba hingga ditemukan solusi yang valid. Meskipun pendekatan ini sederhana dan menjamin solusi jika ada, kelemahannya adalah kompleksitas waktu yang tinggi, terutama jika ukuran papan dan jumlah bentuk meningkat. Oleh karena itu, untuk skala yang lebih besar, optimasi seperti pruning atau algoritma lain seperti backtracking heuristik dapat meningkatkan efisiensi pencarian solusi.

Pranala *Repository* : github.com/tkanigara/Tucil1_13523055

BAB III
Source Code

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


public class Main {
    private static int N, M, P;
    private static String S;
    private static ArrayList<String> puzzleShapes = new ArrayList<>();
    private static char[][] board;
    private static boolean solutionFound = false;
    private static long iterationCount = 0; // Menyimpan jumlah percobaan
    private static final String ANSI_RESET = "\u001B[0m"; // Reset warna
    private static HashMap<Character, String> colorMap = new HashMap<>();


    private static final String[] ANSI_COLORS = {
        "\u001B[31m", // Merah
        "\u001B[32m", // Hijau
        "\u001B[33m", // Kuning
        "\u001B[34m", // Biru
        "\u001B[35m", // Ungu
```

```
        "\u001B[36m", // Cyan
        "\u001B[91m", // Merah terang
        "\u001B[92m", // Hijau terang
        "\u001B[93m", // Kuning terang
        "\u001B[94m", // Biru terang
        "\u001B[95m", // Ungu terang
        "\u001B[96m", // Cyan terang
        "\u001B[97m"  // Putih terang
    };

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Masukkan nama file test case (contoh: test_case.txt): ");
        String fileName = scanner.nextLine();

        try {
            readInputFile(fileName);
            initializeColors(); // Beri warna unik untuk setiap huruf
            board = new char[N][M];
            initializeBoard();

            long startTime = System.currentTimeMillis();
```

```
            solvePuzzle(0);
            long endTime = System.currentTimeMillis();

            if (!solutionFound) {
                System.out.println("Tidak ada solusi yang ditemukan.");
            }

            System.out.println("Waktu pencarian: " + (endTime - startTime) + " ms");
            System.out.println("Banyak kasus yang ditinjau: " + iterationCount);
            System.out.print("Apakah anda ingin menyimpan solusi? (ya/tidak): ");
            String saveOption = scanner.nextLine();
            if (saveOption.equalsIgnoreCase("ya")) {
                saveSolution();
            }

        } catch (FileNotFoundException e) {
            System.out.println("Error: File tidak ditemukan di folder 'test'.");
            System.out.println("Pastikan:");
            System.out.println("1. File " + fileName + " ada di dalam folder
'test'");
            System.out.println("2. Folder 'test' ada di lokasi yang sama dengan
program");
            String projectPath = new
```

```
File(System.getProperty("user.dir")).getParent();

            System.out.println("Path yang dicari: " + new File(projectPath +
File.separator + "test").getAbsolutePath());

        }

    }



    private static void readInputFile(String fileName) throws FileNotFoundException {

        String projectPath = new File(System.getProperty("user.dir")).getParent();

        File testDir = new File(projectPath + File.separator + "test");

        File file = new File(testDir, fileName);

        Scanner fileScanner = new Scanner(file);

        puzzleShapes.clear();



        N = fileScanner.nextInt();

        M = fileScanner.nextInt();

        P = fileScanner.nextInt();

        fileScanner.nextLine();

        S = fileScanner.nextLine();



        Character currentLetter = null;

        StringBuilder currentShape = new StringBuilder();



        while (fileScanner.hasNextLine()) {
```

```
            String line = fileScanner.nextLine().trim();
            if (!line.isEmpty()) {
                // Get first character of each line (the letter identifier)
                char firstChar = line.charAt(0);

                // If this is a new letter or first shape
                if (currentLetter == null || firstChar != currentLetter) {
                    // Save previous shape if exists
                    if (currentShape.length() > 0) {
                        puzzleShapes.add(currentShape.toString());
                    }
                    // Start new shape
                    currentShape = new StringBuilder();
                    currentLetter = firstChar;
                }

                // Add line to current shape
                if (currentShape.length() > 0) {
                    currentShape.append("\n");
                }
                // Add full line including the letter
                currentShape.append(line);
```

```
        }
    }

    // Add final shape
    if (currentShape.length() > 0) {
        puzzleShapes.add(currentShape.toString());
    }

    fileScanner.close();    }

private static void initializeColors() {
    List<String> colorList = new ArrayList<>(Arrays.asList(ANSI_COLORS));
    Collections.shuffle(colorList); // Acak warna agar tidak berurutan

    int colorIndex = 0;
    for (String shape : puzzleShapes) {
        for (char c : shape.toCharArray()) {
            if (Character.isLetter(c) && !colorMap.containsKey(c)) {
                colorMap.put(c, colorList.get(colorIndex % colorList.size()));
                colorIndex++;
            }
        }
```

```
        }
    }

    private static void initializeBoard() {
        board = new char[N][M];
        for (int i = 0; i < N; i++) {
            for (int j = 0; j < M; j++) {
                board[i][j] = '.';
            }
        }
    }

    private static boolean solvePuzzle(int pieceIndex) {
        if (pieceIndex == P) {
            if (isBoardFull()) {
                solutionFound = true;
                System.out.println("\nSolution found!");
                printBoard();
                return true;
            }
            return false;
```

```
        }

        String piece = puzzleShapes.get(pieceIndex);
        ArrayList<String> orientations = getAllOrientations(piece);

        for (String rotatedPiece : orientations) {
            int height = rotatedPiece.split("\n").length;
            int width = rotatedPiece.split("\n")[0].length(); // Ambil panjang baris pertama

            for (int i = 0; i <= N - height; i++) { // Hanya cek posisi yang bisa menampung blok
                for (int j = 0; j <= M - width; j++) {
                    iterationCount++; // Hitung setiap percobaan menempatkan blok

                    if (canPlacePiece(rotatedPiece, i, j)) {
                        placePiece(rotatedPiece, i, j, (char) ('A' + pieceIndex));
                        if (solvePuzzle(pieceIndex + 1)) {
                            return true;
                        }
                        removePiece(rotatedPiece, i, j);
                    }
                }
```

```
            }
        }

        return false;
    }

    private static ArrayList<String> getAllOrientations(String piece) {
        HashSet<String> uniqueOrientations = new HashSet<>();
        ArrayList<String> orientations = new ArrayList<>();

        // Tambahkan bentuk asli
        uniqueOrientations.add(piece);
        orientations.add(piece);

        // Tambahkan pencerminan horizontal dan vertikal
        String mirroredHoriz = mirror(piece);
        String mirroredVert = mirrorVertically(piece);

        if (uniqueOrientations.add(mirroredHoriz)) {
            orientations.add(mirroredHoriz);
        }
        if (uniqueOrientations.add(mirroredVert)) {
```

```
        orientations.add(mirroredVert);
    }

    // Tambahkan rotasi 90°, 180°, 270° untuk setiap bentuk
    String current = piece;
    for (int i = 0; i < 3; i++) {
        current = rotateRight(current);
        if (uniqueOrientations.add(current)) {
            orientations.add(current);
        }
    }

    // Ulangi rotasi untuk bentuk-bentuk yang dipantulkan
    current = mirroredHoriz;
    for (int i = 0; i < 3; i++) {
        current = rotateRight(current);
        if (uniqueOrientations.add(current)) {
            orientations.add(current);
        }
    }

    return orientations;
```

```
}


private static String mirrorVertically(String piece) {
    String[] lines = piece.split("\n");
    List<String> flipped = new ArrayList<>(Arrays.asList(lines));
    Collections.reverse(flipped);
    return String.join("\n", flipped);
}


private static String rotateRight(String piece) {
    if (piece == null || piece.isEmpty()) {
        return piece;
    }

    String[] lines = piece.split("\n");
    if (lines.length == 0 || lines[0].isEmpty()) {
        return piece;
    }

    int height = lines.length;
    int width = lines[0].length();
    char[][] rotated = new char[width][height];
```

```
        for (int i = 0; i < height; i++) {
            for (int j = 0; j < width; j++) {
                if (j < lines[i].length()) {
                    rotated[j][height - 1 - i] = lines[i].charAt(j);
                } else {
                    rotated[j][height - 1 - i] = ' ';
                }
            }
        }
        return charArrayToString(rotated);
    }

    private static String mirror(String piece) {
        String[] lines = piece.split("\n");
        StringBuilder mirrored = new StringBuilder();

        for (String line : lines) {
            mirrored.append(new StringBuilder(line).reverse()).append("\n");
        }

        return mirrored.toString().trim();
```

```
}


private static String charArrayToString(char[][] arr) {
    StringBuilder sb = new StringBuilder();
    for (int i = 0; i < arr.length; i++) {
        for (int j = 0; j < arr[i].length; j++) {
            sb.append(arr[i][j]);
        }
        if (i < arr.length - 1) sb.append("\n");
    }
    return sb.toString();
}


private static boolean canPlacePiece(String piece, int row, int col) {
    if (row < 0 || col < 0) return false;


    String[] lines = piece.split("\n");
    int height = lines.length;
    int width = 0;


    // Hitung lebar maksimum blok
    for (String line : lines) {
```

```
            width = Math.max(width, line.length());
        }

        // Cek apakah blok keluar dari batas papan
        if (row + height > N || col + width > M) return false;

        boolean hasAdjacent = false;
        for (int i = 0; i < height; i++) {
            for (int j = 0; j < lines[i].length(); j++) {
                if (lines[i].charAt(j) != ' ') {
                    if (board[row + i][col + j] != '.') {
                        return false; // Bentrok dengan blok lain
                    }
                    if (checkAdjacent(row + i, col + j)) {
                        hasAdjacent = true;
                    }
                }
            }
        }

        return (row == 0 && col == 0) || hasAdjacent; // Blok pertama bebas, lainnya
harus adjacent
```

```
}


private static boolean checkAdjacent(int row, int col) {
    int[][] directions = {{-1,0}, {1,0}, {0,-1}, {0,1}};
    for (int[] dir : directions) {
        int newRow = row + dir[0];
        int newCol = col + dir[1];
        if (newRow >= 0 && newRow < N && newCol >= 0 && newCol < M) {
            if (board[newRow][newCol] != '.') {
                return true;
            }
        }
    }
    return false;
}


private static void placePiece(String piece, int row, int col, char c) {
    String[] lines = piece.split("\n");
    for (int i = 0; i < lines.length && row + i < N; i++) {
        String line = lines[i];
        for (int j = 0; j < line.length() && col + j < M; j++) {
            if (Character.isLetter(line.charAt(j))) {
```

```
                board[row + i][col + j] = c;
            }
        }
    }
}

private static void removePiece(String piece, int row, int col) {
    String[] lines = piece.split("\n");
    for (int i = 0; i < lines.length; i++) {
        for (int j = 0; j < lines[i].length(); j++) {
            if (Character.isLetter(lines[i].charAt(j))) {
                board[row + i][col + j] = '.';
            }
        }
    }
}

private static boolean isBoardFull() {
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < M; j++) {
            if (board[i][j] == '.') {
```

```
                    return false;
                }
            }
        }
        return true;
    }

    private static void printBoard() {
        for (int i = 0; i < N; i++) {
            for (int j = 0; j < M; j++) {
                char c = board[i][j];
                if (c == '.') {
                    System.out.print(c + " ");
                } else {
                    System.out.print(colorMap.getOrDefault(c, ANSI_RESET) + c +
ANSI_RESET + " ");
                }
            }
            System.out.println();
        }
        System.out.println();
    }
```

```
    private static void saveSolution() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Masukkan nama file untuk menyimpan solusi (contoh:
solusi1.txt): ");
        String fileName = scanner.nextLine().trim();

        try {
            String projectPath = new
File(System.getProperty("user.dir")).getParent();
            File testDir = new File(projectPath + File.separator + "test");
            File solutionFile = new File(testDir, fileName);

            java.io.PrintWriter writer = new java.io.PrintWriter(solutionFile);
            for (int i = 0; i < N; i++) {
                for (int j = 0; j < M; j++) {
                    writer.print(board[i][j]);
                }
                writer.println();
            }
            writer.close();
            System.out.println("Solusi telah disimpan dalam file: " +
```

```
solutionFile.getAbsolutePath());

        } catch (FileNotFoundException e) {

            System.out.println("Gagal menyimpan solusi: " + e.getMessage());

        }

    }

}
```

BAB IV

Eksperimen

Berikut adalah beberapa contoh keluaran program:

A. Masukan tidak valid

```
PS C:\COOLYEAH\SMT 4\STIMA\Tucil 1\Tucil1_13523055\bin> java Main
Masukkan nama file test case (contoh: test_case.txt): apa.txt
Error: File tidak ditemukan di folder 'test'.
Pastikan:
1. File apa.txt ada di dalam folder 'test'
2. Folder 'test' ada di lokasi yang sama dengan program
Path yang dicari: C:\COOLYEAH\SMT 4\STIMA\Tucil 1\Tucil1_13523055\test
```

B. Masukan valid tapi tidak ada solusi yang memenuhi

```
PS C:\COOLYEAH\SMT 4\STIMA\Tucil 1\Tucil1_13523055\bin> java Main
Masukkan nama file test case (contoh: test_case.txt): no_sol.txt
Tidak ada solusi yang ditemukan.
Waktu pencarian: 540 ms
Banyak kasus yang ditinjau: 2704705
Apakah anda ingin menyimpan solusi? (ya/tidak): tidak
```

C. Masukan benar dan terdapat solusi yang memenuhi

Input	Output	Penjelasan
5 5 7 DEFAULT A AA B BB C CC D DD EE EE E FF FF F GGG	PS C:\COOLYEAH\SMT 4\STIMA\Tucil 1\Tucil1_13523055\bin> java Main Masukkan nama file test case (contoh: test_case.txt): 1.txt Solution found! A A D D F A B D F F C B B F F C C E E E G G G E E Waktu pencarian: 833 ms Banyak kasus yang ditinjau: 4267125 Apakah anda ingin menyimpan solusi? (ya/tidak): tidak	Test case dari spesifikasi

<table>
<tr>
<td>3 3 4
DEFAULT
BB
B
A
CC
CC
D</td>
<td>
<pre>PS C:\COOLYEAH\SMT 4\STIMA\Tucil 1\Tucil1_13523055\bin> java Main
Masukkan nama file test case (contoh: test_case.txt): 2.txt

Solution found!
A A B
A C C
D C C

Waktu pencarian: 11 ms
Banyak kasus yang ditinjau: 15
Apakah anda ingin menyimpan solusi? (ya/tidak): tidak</pre>
</td>
<td>Test case dengan piece puzzle yang sedikit serta papan yang kecil</td>
</tr>
<tr>
<td>4 4 5
DEFAULT
AAA
A
BB
C
C
C
C
DDD
D
EE</td>
<td>
<pre>PS C:\COOLYEAH\SMT 4\STIMA\Tucil 1\Tucil1_13523055\bin> java Main
Masukkan nama file test case (contoh: test_case.txt): 3.txt

Solution found!
A A A C
A B B C
D D D C
D E E C

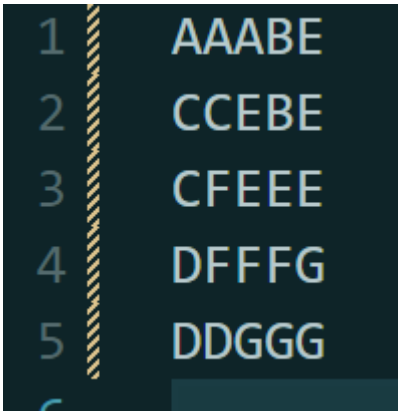
Waktu pencarian: 19 ms
Banyak kasus yang ditinjau: 26
Apakah anda ingin menyimpan solusi? (ya/tidak): tidak</pre>
</td>
<td>Test case dengan piece puzzle lebih banyak dan papan yang lebih besar</td>
</tr>
<tr>
<td>3 3 3
DEFAULT
AAA
BBB

CCC</td>
<td>
<pre>PS C:\COOLYEAH\SMT 4\STIMA\Tucil 1\Tucil1_13523055\bin> java Main
Masukkan nama file test case (contoh: test_case.txt): 4.txt

Solution found!
A A A
B B B
C C C

Waktu pencarian: 11 ms
Banyak kasus yang ditinjau: 6
Apakah anda ingin menyimpan solusi? (ya/tidak): tidak</pre>
</td>
<td>Test case dangan input baris kosong, hasilnya tidak akan dianggap</td>
</tr>
<tr>
<td>5 5 7
DEFAULT
AAA
B
B
C
CC
D
DD
EEE
E E
E
FF
F
F
GGG
G</td>
<td>
Terminal:
<pre>PS C:\COOLYEAH\SMT 4\STIMA\Tucil 1\Tucil1_13523055\bin> java Main
Masukkan nama file test case (contoh: test_case.txt): 5.txt

Solution found!
A A A B E
C C E B E
C F E E E
D F F F G
D D G G G

Waktu pencarian: 76 ms
Banyak kasus yang ditinjau: 29252
Apakah anda ingin menyimpan solusi? (ya/tidak): ya
Masukkan nama file untuk menyimpan solusi (contoh: solusi1.txt): solusi.txt
Solusi telah disimpan dalam file: C:\COOLYEAH\SMT 4\STIMA\Tucil 1\Tucil1_13523055\test\solusi.txt</pre>
solusi.txt:


</td>
<td>Test case dengan salah satu piece memiliki spasi di barisnya. Test case ini dilengkapi dengan test jika menyimpan solusi ke folder “test” dengan nama file sesuai input user.</td>
</tr>
</table>

<table>
<tr>
<td>6 6 8
DEFAULT
AA
AA
BBB
BB
CC
C
C
DD
DD
D
EEE
E
FFF
F
F
GG
GG
G
HHH
H</td>
<td>Terminal:
<pre>PS C:\COOLYEAH\SMT 4\STIMA\Tucil 1\Tucil1_13523055\bin> java Main
Masukkan nama file test case (contoh: test_case.txt): 6.txt

Solution found!
A A B B B G
A A B B G G
E E C C G G
F E C D H H
F E C D D H
F F F D D H

Waktu pencarian: 140 ms
Banyak kasus yang ditinjau: 341698
Apakah anda ingin menyimpan solusi? (ya/tidak): ya
Masukkan nama file untuk menyimpan solusi (contoh: solusi1.txt): solusi1.txt
Solusi telah disimpan dalam file: C:\COOLYEAH\SMT 4\STIMA\Tucil 1\Tucil1_13523055\test\solusi1.txt</pre>
solusi1.txt:
<pre>1 AABBBG
2 AABBGG
3 EECCGG
4 FECDHH
5 FECDDH
6 FFFDDH</pre></td>
<td>Test case dengan total piece 8 dan board yang lebih besar yaitu 6x6. Test case ini dilengkapi dengan test jika menyimpan solusi ke folder “test” dengan nama file sesuai input user.</td>
</tr>
<tr>
<td>6 6 9
DEFAULT
AA
AA
BBB
B B
BBB
CC
C
C
DD
DD
EEE
E
F
F
F
GG
G
G
HHH
H
I</td>
<td>
<pre>PS C:\COOLYEAH\SMT 4\STIMA\Tucil 1\Tucil1_13523055\bin> java Main
Masukkan nama file test case (contoh: test_case.txt): 7.txt

Solution found!
A A B B B F
A A B I B F
C C B B B F
C G D D E E
C G D D H E
G G H H H E

Waktu pencarian: 49 ms
Banyak kasus yang ditinjau: 11522
Apakah anda ingin menyimpan solusi? (ya/tidak): tidak</pre>
</td>
<td>Test case ini dilengkapi dengan piece yang memiliki suatu lubang di tengahnya, yaitu pada piece huruf “B”</td>
</tr>
</table>