# Penyelesaian IQ Puzzler Pro dengan Algoritma Brute Force

Laporan Tugas Kecil 1 IF2211 Strategi Algoritma



#### **Disusun Oleh:**

12821046 – Fardhan Indrayesa

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

#### 1. Algoritma brute force

#### Psedocode

```
Procedure puzzleSolver(input M, N, P: integer; input blocks: array; output result:
boolean)
Deklarasi
 puzzle board: array[M][N] of char
 transforms: array of array
index: integer
 result: boolean
Algoritma
transforms <- generateTransform(blocks)</pre>
result <- false
 index <- 0
 i <- 0
 while (i < size(transforms)) and (not result) do
 puzzle board <- generateBoard(M, N)</pre>
 result <- placeBlocks(puzzle board, transforms, index)</pre>
 if not result then
  shiftBlocks(transforms)
 endif
 i <- i + 1
 endwhile
endprocedure
```

```
Procedure placeBlocks(input puzzle: array[M][N] of char; input transform: array of
array; input index: integer; output success: boolean)
Deklarasi
block: array
row, col: integer
success: boolean
Algoritma
If index = size(transform) then
 success <- not stillEmpty(puzzle)</pre>
 return success
 endif
 for each block in transform[index] do
 row <- 0
  while row <= (size(puzzle) - size(block)) do
  col <- 0
   while col <= (size(puzzle[0]) - size(block[0])) do</pre>
   if canPlace(puzzle, block, row, col) then
    placeBlock(puzzle, block, row, col)
     success <- placeBlocks(puzzle, transform, index + 1)</pre>
     if success then
      return true
     removeBlock(puzzle, block, row, col)
    endif
    col <- col + 1
```

```
endwhile
row <- row + 1
endwhile
endfor

success <- false
return success
endprocedure</pre>
```

#### Penjelasan algoritma Brute Force:

- 1. Setiap blok yang tersedia, akan diperiksa, apakah dapat dimasukan dalam papan atau tidak. Pemeriksaan dilakukan dari kiri ke kanan (dimulai dari kiri atas papan).
- 2. Jika dapat dimasukkan, letakkan blok di papan tersebut. Tapi, jika tidak dapat dimasukkan, periksa di cell selanjutanya.
- 3. Lalu, jika masih tidak ada yang cocok, maka akan dicoba variasi (rotasi/flip) dari blok tersebut, ulangi hingga seluruh variasi sudah dicoba.
- 4. Jika seluruh blok dan variasinya sudah diperiksa di seluruh cell dan masih tidak ada yang cocok atau masih ada cell yang kosong, maka tidak ada solusi untuk papan tersebut.
- 5. Jika tidak ada solusi, geser (shift) urutan puzzle dari kanan ke kiri, lalu setiap sudah di shift, akan diperiksa lagi apakah papan mempunyai solusi atau tidak.
- 6. Jika papan sudah terisi penuh dan tidak ada blok yang tersisa, maka solusi berhasil ditemukan.

#### 2. Source code

#### Main.java

```
public static void main(String[] args) {
        String caseFile = "case 4.txt";
        String outputFile = "output case 1.txt";
        readData data = getData("src\\" + caseFile);
        int M = data.M;
        int N = data.N;
        int P = data.P;
        String kasus = data.kasus;
        Map<Character, List<String[]>> blocks_data = data.blocks;
        List<String[][]> blocks = new ArrayList<>();
        for (List<String[]> block : blocks data.values()) {
            blocks.add(block.toArray(new String[0][]));
        1
        String[][] puzzle board = new String[M][N];
        List<List<String[][]>> transforms = generateTransform(blocks);
        boolean result = false;
        int index = 0;
        totalAttempts = 0;
        long sTime = System.nanoTime();
        for (int i = 0; i < transforms.size(); <math>i++) { //
            puzzle board = generateBoard(M, N);
            if (i == 0) {
                printMatrix(puzzle board);
            result = placeBlocks(puzzle board, transforms, index);
            if (result) { // Jika solusi sudah ditemukan, keluar dari
loop shift
                break;
            }
            shiftBlocks(transforms); // Jika solusi belum ditemukan,
geser urutan penempatan block
                                      // (block A menjadi urutan akhir,
block B jadi urutan pertama, dst.)
        long eTime = System.nanoTime();
        System.out.println("\nSolusi:");
        if (result) {
            printMatrix(puzzle_board);
        else {
            System.out.println("Puzzle tidak memiliki solusi.");
        System.out.println("\nWaktu pencarian " + (eTime - sTime)/1000000
+ " ms.");
        System.out.println("\nTotal Kasus: " + totalAttempts + "\n");
        new writeData("test\\" + outputFile, puzzle board, result);
    }
```

```
public static readData getData(String pathFile) {
        return new readData(pathFile);
      public static writeData importOutput(String namefile, String[][]
puzzle) {
//
     }
    public static String[][] cloneBoard(String[][] board) {
        int rows = board.length, cols = board[0].length;
        String[][] newBoard = new String[rows][cols];
        for (int i = 0; i < rows; i++) {
            newBoard[i] = Arrays.copyOf(board[i], cols);
        1
        return newBoard;
    }
    public static String[][] generateBoard(int m, int n) {
        String[][] puzzle board = new String[m][n];
        for (int i = 0; i < m; i++) {</pre>
            Arrays.fill(puzzle board[i], " ");
        return puzzle board;
    }
    public static String coloredFont(String val) {
        if (!val.equals(" ")) {
            return Color[Math.abs(val.hashCode()) % Color.length];
        return "#";
    public static void printMatrix(String[][] matrix) {
        for (String[] row : matrix) {
            for (String value : row) {
                System.out.print(coloredFont(value) + value + Reset);
            System.out.println();
        }
    }
    public static boolean placeBlocks(String[][] puzzle,
List<List<String[][]>> transform, int index) {
        if (index == transform.size()) {    // Jika sudah semua block
berhasil ditempatkan, akan diperiksa,
            return (!stillEmpty(puzzle)); // apakah masih ada cell yang
kosong.
        for (String[][] block : transform.get(index)) {
            int nRows = block.length, nCols = block[0].length;
            for (int row = 0; row <= (puzzle.length - nRows); row++) {</pre>
                for (int col = 0; col <= (puzzle[0].length - nCols);</pre>
col++) {
                    totalAttempts++;
```

```
if (canPlace(puzzle, block, row, col)) {
                                                                  // Jika
block dapat diletakkan di cell papan,
                        placeBlock(puzzle, block, row, col);
tempatkan block di cell tersebut
                        if (placeBlocks(puzzle, transform, index + 1)) {
// memeriksa block selanjutnya
                            return true; // true jika sudah tidak ada
cell vang kosong
                        1
                        removeBlock(puzzle,block,row,col); // hapus block
jika masih ada cell yang kosong,
                                                            // lalu
periksa block tersebut di cell selanjutnya
                }
            }
        } // Jika masih tidak ada yang cocok, maka coba periksa variasi
block tersebut
        return false; // false jika masih ada cell yang kosong
    public static boolean canPlace(String[][] puzzle, String[][] piece,
int sRow, int sCol) {
        int nRow = piece.length, nCol = piece[0].length;
        for (int i = 0; i < nRow; i++) {
            for (int j = 0; j < nCol; j++) {
                if (!piece[i][j].equals(" ") &&
!puzzle[sRow+i][sCol+j].equals(" ")) {
                    return false;
        1
        return true;
    public static void placeBlock(String[][] puzzle, String[][] piece,
int sRow, int sCol) {
        int nRow = piece.length, nCol = piece[0].length;
        Set<String> puzzle unique = new HashSet<>();
        Set<String> piece unique = new HashSet<>();
        for (String[] puzzleRow : puzzle) {
            puzzle unique.addAll(Arrays.asList(puzzleRow));
        for (String[] pieceRow : piece) {
            piece unique.addAll(Arrays.asList(pieceRow));
        if (!(puzzle unique.containsAll(piece unique))) { // mencegah
block duplikat
            for (int i = 0; i < nRow; i++) {
                for (int j = 0; j < nCol; j++) {
                    if (!piece[i][j].equals(" ")) {
                        puzzle[sRow + i][sCol + j] = piece[i][j];
                }
            }
        }
    }
```

```
public static void removeBlock(String[][] board, String[][] block,
int row, int col) {
        for (int i = 0; i < block.length; <math>i++) {
            for (int j = 0; j < block[0].length; <math>j++) {
                if (!block[i][j].equals(" ")) {
                    if (board[row + i][col + j].equals(block[i][j])) {
                        board[row + i][col + j] = " ";
                    //board[row + i][col + i] = " ";
                }
            }
        }
    }
    public static boolean stillEmpty(String[][] puzzle) {
        for (String[] puzzleRow : puzzle) {
            for (String element : puzzleRow) {
                if (element.equals(" ")) {
                    return true;
                }
            }
        }
        return false;
    public static void shiftBlocks(List<List<String[][]>> blocks) {
        blocks.add(blocks.removeFirst());
   public static List<List<String[][]>>
generateTransform(List<String[][]> blocks) {
        List<List<String[][]>> transform = new ArrayList<>();
        for (String[][] block : blocks) {
            List<String[][]> otherBlock = new ArrayList<>();
            otherBlock.add(block);
            otherBlock.add(rotate90(block));
            otherBlock.add(rotate180(block));
            otherBlock.add(rotate270(block));
            String[][] flipped = flipH(rotate270(block));
            otherBlock.add(flipped);
            otherBlock.add(rotate90(flipped));
            otherBlock.add(rotate180(flipped));
            otherBlock.add(rotate270(flipped));
            transform.add(otherBlock);
        return transform;
    public static String[][] rotate90(String[][] block) {
        int nRows = block.length, nCols = block[0].length;
        String[][] rotatedBlock = new String[nCols][nRows];
        for (int i = 0; i < nRows; i++) {
            for (int j = 0; j < nCols; j++) {
                rotatedBlock[j][nRows - 1 - i] = block[i][j];
        return rotatedBlock;
    }
```

```
public static String[][] rotate180(String[][] block) {
    return rotate90(rotate90(block));
}

public static String[][] rotate270(String[][] block) {
    return rotate90(rotate180(block));
}

public static String[][] flipH(String[][] block) {
    int nRows = block.length, nCols = block[0].length;
    String[][] flippedBlock = new String[nRows][nCols];
    for (int i = 0; i < nRows; i++) {
        for (int j = 0; j < nCols; j++) {
            flippedBlock[i][nCols - 1 - j] = block[i][j];
        }
    }
    return flippedBlock;
}</pre>
```

#### readData.java

```
import java.util.*;
public class Main {
    public static final String Reset = "\u001B[0m";
    public static final String[] Color = {
             "\u001B[30m", // Black]
             "\u001B[31m", // Red
"\u001B[32m", // Green
             "\u001B[33m", // Yellow "\u001B[34m", // Blue
             "\u001B[35m", // Purple
             "\u001B[36m", // Cyan
             "\u001B[37m", // White
    };
import java.util.*;
import java.io.File;
import java.io.FileNotFoundException;
public class readData {
    int M = 0; int N = 0; int P = 0;
    String kasus = null;
    Map<Character, List<String[]>> blocks = new HashMap<>();
    public readData(String pathFile) {
         try {
             File caseFile = new File(pathFile);
             Scanner reader = new Scanner(caseFile);
             Character dummyChar = null;
             List<String[]> dummy matrix = new ArrayList<>();
             int rowNumber = 0;
             while (reader.hasNextLine()) {
```

```
String rowLine = reader.nextLine();
                rowNumber++;
                if (rowNumber == 1) {
                    String[] oneLine = rowLine.split(" ");
                    this.M = Integer.parseInt(oneLine[0]);
                    this.N = Integer.parseInt(oneLine[1]);
                    this.P = Integer.parseInt(oneLine[2]);
                else if (rowNumber == 2) {
                    this.kasus = rowLine;
                }
                else {
                    char firstChar = rowLine.trim().charAt(0);
                    if (dummyChar == null || firstChar != dummyChar) {
                        if (dummyChar != null) {
                            this.blocks.put(dummyChar, dummy matrix);
                            dummy matrix = new ArrayList<>();
                        dummyChar = firstChar;
                    }
                    dummy matrix.add(rowLine.split(""));
                }
            }
            if (dummyChar != null) {
                this.blocks.put(dummyChar, dummy matrix);
            }
            reader.close();
        catch (FileNotFoundException e) {
            System.out.println("File not found!");
            e.printStackTrace();
        }
    }
}
```

#### writeData.java

```
    writer.close();
    System.out.println("File disimpan sebagai " + nameFile);
}
catch (IOException e) {
    System.out.println("An error occured!");
    e.printStackTrace();
}
}
```

### 3. Hasil

### Case 1

```
Solusi:

AGGGD

AA

B

BB

CC

CC

D

D

D

D

EE

EE

EF

FF

FF

GGG

Input

Solusi:

AGGGD

AABDD

CCBBE

CFFEE

FFFEE

Waktu pencarian 199 ms.

Total Kasus: 2703813

Output
```

#### Case 2

```
Solusi:

AABBBGGCKKK

DAHBGCCKEK

DAHFGJCCEE

DDIHFGJJLLE

DIIIFFFJJLE

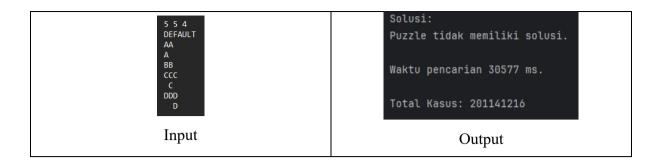
Waktu pencarian 152322 ms.

Total Kasus: 1481738752

Input

Output
```

Case 3



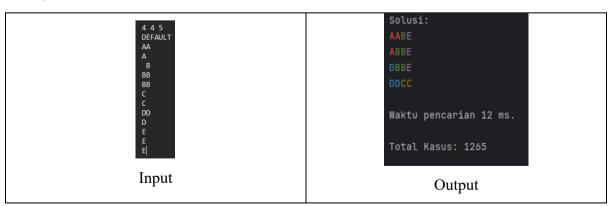
### Case 4



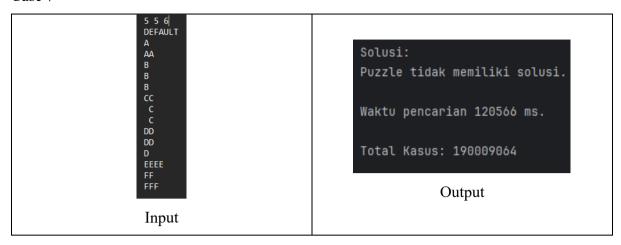
## Case 5



## Case 6



# Case 7



# 4. Link github

https://github.com/fardhan248/IQ\_Puzzler\_Pro.git

No	Poin	Ya	Tidak
1	Program berhasil dikompilasi tanpa kesalahan	<b>√</b>	
2	Program berhasil dijalankan	<b>√</b>	
3	Solusi yang diberikan program benar dan mematuhi aturan permainan	✓	
4	Program dapat membaca masukan berkas .txt serta menyimpan solusi dalam berkas .txt	<b>√</b>	
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	<b>√</b>	