

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**

2024

1. Algoritma brute force

Pseudocode

```
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)
result <- false
index <- 0

i <- 0
while (i < size(transforms)) and (not result) do
  puzzle_board <- generateBoard(M, N)
  result <- placeBlocks(puzzle_board, transforms, index)
  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)
  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
      if canPlace(puzzle, block, row, col) then
        placeBlock(puzzle, block, row, col)
        success <- placeBlocks(puzzle, transform, index + 1)
        if success then
          return true
        endif
        removeBlock(puzzle, block, row, col)
      endif
      col <- col + 1
    endwhile
    row <- row + 1
  endwhile
endfor
```

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

success <- false
return success

endprocedure
```

Penjelasan algoritma Brute Force:

1. Setiap blok yang tersedia, akan diperiksa, apakah dapat dimasukkan 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 selanjutnya.
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

```
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
    };

    static int totalAttempts = 0;
```

```

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][]));
    }

    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(); 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);
            }

            return newBoard;
        }

        public static String[][] generateBoard(int m, int n) {
            String[][] puzzle_board = new String[m][n];
            for (int i = 0; i < m; i++) {
                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++) {
                    for (int col = 0; col <= (puzzle[0].length - nCols);
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 yang kosong
            }
            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;
            }
        }
    }
    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; i++) {
            for (int j = 0; j < block[0].length; 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 + j] = " ";
                }
            }
        }
    }

    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;
    }
}

```

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

```

import java.io.FileWriter;
import java.io.IOException;
import java.util.Arrays;

public class writeData {
    public writeData(String nameFile, String[][] puzzle, boolean result)
    {
        try {
            FileWriter writer = new FileWriter(nameFile);
            if (result) {
                for (String[] rowVal : puzzle) {
                    for (String val : rowVal) {
                        writer.write(val);
                    }
                    writer.write("\n");
                }
            }
            else {
                writer.write("Tidak ada solusi.");
            }
        }
        catch (IOException e) {
            e.printStackTrace();
        }
    }
}

```

```

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

```

3. Hasil

Case 1

<div data-bbox="453 696 531 978"> <pre> 5 5 7 DEFAULT A AA B BB C CC D DD EE EE E FF FF F GGG </pre> </div> <div data-bbox="450 996 534 1039">Input</div>	<div data-bbox="957 696 1235 987"> <pre> Solusi: AGGGD AABDD CCBBE CFFEE FFFE </pre> <p>Waktu pencarian 199 ms.</p> <p>Total Kasus: 2703813</p> </div> <div data-bbox="1040 1003 1150 1046">Output</div>
---	--

Case 2

<div data-bbox="453 1178 531 1662"> <pre> 5 11 12 DEFAULT AA A A BBB BB C CC CC D DDDD EE EEE F F FFF G GGGG H HH H III I J JJ JJ KKK K K L LL </pre> </div> <div data-bbox="450 1682 534 1724">Input</div>	<div data-bbox="844 1178 1351 1655"> <pre> Solusi: AABBBGGCKKK DAHBBGCCKEK DAHFFGJCCEE DDIHFGJJLLE DIIIFFFJJLE </pre> <p>Waktu pencarian 152322 ms.</p> <p>Total Kasus: 1481738752</p> </div> <div data-bbox="1040 1671 1150 1713">Output</div>
---	---

Case 3

<pre> 5 5 4 DEFAULT AA A BB CCC C DDD D </pre> <p>Input</p>	<pre> Solusi: Puzzle tidak memiliki solusi. Waktu pencarian 30577 ms. Total Kasus: 201141216 </pre> <p>Output</p>
---	---

Case 4

<pre> 4 5 6 DEFAULT A A A BB B C C C D D DD E EE E FFF F </pre> <p>Input</p>	<pre> Solusi: ABBFC ABFFC AEEFD EEDDD Waktu pencarian 13 ms. Total Kasus: 4693 </pre> <p>Output</p>
--	---

Case 5

<pre> 4 5 7 DEFAULT AA A BB B C C C DD DD EE E FF GG </pre> <p>Input</p>	<pre> Solusi: AABBC ADDBC DDEEC FFEgg Waktu pencarian 12 ms. Total Kasus: 53 </pre> <p>Output</p>
--	---

Case 6

<pre> 4 4 5 DEFAULT AA A B BB BB C C DD D E E E </pre> <p>Input</p>	<pre> Solusi: AABE ABBE DBBE DDCC Waktu pencarian 12 ms. Total Kasus: 1265 </pre> <p>Output</p>
---	---

Case 7

<pre>5 5 6 DEFAULT A AA B B B CC C C DD DD D EEEE FF FFF</pre> <p>Input</p>	<pre>Solusi: Puzzle tidak memiliki solusi. Waktu pencarian 120566 ms. Total Kasus: 190009064</pre> <p>Output</p>
--	--

4. Link github

https://github.com/fardhan248/IQ_Puzzler_Pro.git

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	✓	