LAPORAN TUGAS KECIL 1 IF2211 STRATEGI ALGORITMA

Penyelesaian IQ Puzzler Pro dengan Algoritma *Brute Force*

Disusun Oleh:

Athian Nugraha Muarajuang 13523106

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2025

DAFTAR ISI

BAB I
DESKRIPSI MASALAH

1.1 IQ Puzzler Pro

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.

1.2 Algoritma Brute Force

Algoritma *Brute Force* adalah Algoritma dengan pendekatan yang lempang (straightforward) untuk memecahkan suatu persoalan. Algoritma *Brute Force* biasanya didasarkan pada pernytaan pada persoalan (*Problem Statement*) dan konsep yang dilibatkan. Algoritma *brute force* memecahkan persoalan dengan sangat sederhana, *straight forward*, dan mudah dipahami caranya.
Algoritma *brute force* umumnya tidak cerdas dan tidak efisien karena membutuhkan volume komputasi yang besar dan durasi eksekusi yang memakan banyak waktu. Karena itu, algoritma *brute force* kadang-kadang disebut sebagai algoritma naif, dimana masih banyak algoritma lainnya yang lebih cerdas. Algoritma *brute force* lebih cocok digunakan untuk persoalan dengan *size* input (n) yang kecil serta biasa sebagai basis pembanding dengan algoritma lain yang lebih efisien.

BAB II
IMPLEMENTASI ALGORITMA BRUTE FORCE

Implementasi yang digunakan dalam solving IQ Puzzler Pro ini adalah *exhaustive search*. Berikut langkah-langkah program dalam solving IQ Puzzler Pro:

1. Program menerima input file berekstensi .txt yang berisi format untuk informasi board dan puzzle pieces. Board diinisialisasi dengan jumlah baris dan kolom yang tersedia pada file .txt. input merupakan board custom, akan menerima input berupa gambaran custom board. Lalu, setiap puzzle piece diinisialisasikan berdasarkan bentuk dan ukuran baris dan kolomnya. puzzle piece juga diinisialisasikan semua kemungkinan orientasinya ke dalam sebuah list (hal ini dilakukan dengan bantuan hash set agar isi list orientasi-orientasi puzzle piece-nya tidak redundan). Semua puzzle piece yang telah diinisialisasikan dimasukkan ke dalam list of puzzle piece dan dimasukkan sebagai atribut dari board.
2. Algoritma solver menggunakan pendekatan rekursif. Dimulai dari puzzle piece pertama, dilakukan traversal untuk setiap orientasi puzzle piece-puzzle piece tersebut lalu cek secara traversal juga per petak pada board. Jika puzzle piece valid untuk ditempatkan, maka akan ditempatkan lalu secara rekursif melanjutkan ke puzzle berikutnya. Ketika tidak ada petak valid dan tidak ada orientasi yang tepat untuk menempatkan puzzle piece, maka program akan melakukan *backtracking* ke puzzle piece sebelumnya untuk melanjutkan pencarian traversal.
3. program selesai antara dua kasus:
   a. Ketika rekursif mencapai base case, dimana sudah semua puzzle piece ditempatkan di board. dalam base case ini akan dicek jika board sudah penuh, maka program telah selesai. Sebaliknya, jika belum penuh maka program akan melakukan backtrack dan melakukan seluruh kemungkinan penempatan puzzle, secara tidak langsung menyatakan jika puzzle tidak dapat diselesaikan.
   b. Ketika seluruh puzzle piece telah dicari penempatannya secara traversal dan tidak ada yang valid.
4. Ketika selesai menyelesaikan puzzle-nya, program akan memberikan output hasil penyelesaian puzzle beserta angka banyak kasus yang dicoba dan durasi eksekusi program menyelesaikan puzzle. Program akan memberi pilihan untuk user menyimpan output dengan file txt, lalu gambar. Setelah itu, user dipersilakan untuk memilih antara melanjutkan program atau keluar dari program.

BAB III
SOURCE CODE

Dalam program ini, terbuat lima kelas: Main.java, Board.java, PuzzlePiece.java, IOPuzzlerFile.java, dan Solver.java

3.1. Main.java

```
public class Main {

    public static void main(String[] args) {
        while (true) {
            Board board = IOPuzzlerFile.readInputFile();
            Solver solver = new Solver(board);
            long startTime = System.currentTimeMillis();
            if (solver.getBoard().getValidGrids() !=
PuzzlePiece.getCount()) {
                System.out.println("No solution");
            }
            boolean result = solver.solvePuzzle(0);
            long endTime = System.currentTimeMillis() -
startTime;
            if (result) {
                solver.getBoard().printColored();
            } else {
                System.out.println("No solution.");
            }
            System.out.println("Number of cases examined: " +
solver.getIterationCount() + " cases");
            System.out.println("Execution time (ms): " +
endTime + " ms");
            IOPuzzlerFile.promptSaveSolution(solver, endTime);

IOPuzzlerFile.promptSaveImageSolution(solver.getBoard());
            IOPuzzlerFile.endConfirmation();
        }
    }
}
```

3.2. Solver.java

```
public class Solver {
    private Board board;
    private long iterationCount = 0;

    public Solver(Board board) {
        this.board = board;
    }

    public boolean solvePuzzle(int puzzlePieceIndex) {
        if (puzzlePieceIndex ==
board.getPieces().size()) {
            if (board.isFull()) return true;
            return false;
        }

        PuzzlePiece puzzlePiece =
board.getPieces().get(puzzlePieceIndex);
        for (char[][] shape :
puzzlePiece.getOrientations()) {
            for (int i = 0; i < board.getRows(); i++) {
                for (int j = 0; j < board.getCols();
j++) {
                    iterationCount++;
                    if (board.canPlace(shape, i, j)) {
```

```
                                board.placePiece(shape, i, j,
puzzlePieceIndex);
                                if (solvePuzzle(puzzlePieceIndex
+ 1)) return true;
                                board.removePiece(shape, i, j,
puzzlePiece.getId());
                           }
                      }
                 }
            }
            return false;
     }

     public Board getBoard() {
            return board;
     }

     public long getIterationCount() {
            return iterationCount;
     }

}
```

3.3. Board.java

```
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
```

```java
public class Board {
    // ATTRIBUTES
    private char[][] grid;
    private List<PuzzlePiece> pieces;
    private String config;
    private int validGrids;

    public static final String[] ANSI_COLORS = {
        "\u001B[30m", // Hitam
        "\u001B[31m", // Merah
        "\u001B[32m", // Hijau
        "\u001B[33m", // Kuning
        "\u001B[34m", // Biru
        "\u001B[35m", // Ungu
        "\u001B[36m", // Cyan
        "\u001B[37m", // Putih
        "\u001B[90m", // Hitam terang
        "\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
        "\u001B[40m", // Background Hitam
        "\u001B[41m", // Background Merah
        "\u001B[42m", // Background Hijau
        "\u001B[43m", // Background Kuning
        "\u001B[44m", // Background Biru
        "\u001B[45m", // Background Ungu
        "\u001B[46m", // Background Cyan
        "\u001B[47m", // Background Putih
        "\u001B[100m", // Background Hitam terang
        "\u001B[107m"  // Background Putih terang
    };

    public static final String ANSI_RESET = "\u001B[0m";


    // CONSTRUCTOR
    public Board(int rows, int cols, String config,
List<PuzzlePiece> pieces) {
        this.grid = new char[rows][cols];
        for (int i = 0; i < rows; i++) {
            Arrays.fill(grid[i], '.');
        }
        this.pieces = pieces;
        this.config = config;
        this.validGrids = rows * cols;
    }

    // SELECTOR
    public String getConfig() {
        return config;
    }

    public int getValidGrids() {
        return validGrids;
    }

    public List<PuzzlePiece> getPieces() {
```

```java
        return pieces;
    }

    public int getCols() {
        return grid[0].length;
    }

    public int getRows() {
        return grid.length;
    }

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

    // Setter
    public void setGrid(char[][] grid) {
        this.grid = grid;
    }

    public void setConfig(String config) {
        this.config = config;
    }

    public void setPieces(List<PuzzlePiece> pieces) {
        this.pieces = pieces;
    }

    public void setValidGrids(int validGrids) {
        this.validGrids = validGrids;
    }
    // FUNCTION
    public boolean canPlace(char[][] pieceMat, int r,
int c) {
        int pr = pieceMat.length;
        int pc = pieceMat[0].length;
        int br = grid.length;
        int bc = grid[0].length;
        if (r + pr > br || c + pc > bc) return false;
        for (int i = 0; i < pr; i++) {
            for (int j = 0; j < pc; j++) {
                if (pieceMat[i][j] != '.' &&
grid[r+i][c+j] != '.') {
                    return false;
                }
            }
        }
        return true;
    }
    public boolean isEmpty() {
        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 boolean isFull() {
        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 placePiece(char[][] piece, int r, int c,
int puzzlePieceIdx) {
        int puzzleHeight = piece.length;
        int puzzleWidth = piece[0].length;
        for (int i = r; i < r + puzzleHeight; i++) {
            for (int j = c; j < c + puzzleWidth; j++) {
                if (grid[i][j] != '.') continue;
                grid[i][j] = piece[i-r][j-c];
            }
        }
    }

    public void removePiece(char[][] piece, int r, int
c, char id) {
        int puzzleHeight = piece.length;
        int puzzleWidth = piece[0].length;
        for (int i = r; i < r + puzzleHeight; i++) {
            for (int j = c; j < c + puzzleWidth; j++) {
                if (grid[i][j] == id) grid[i][j] = '.';
            }
        }
    }
    public void printColored() {
        Map<Character, String> colorMap = new
HashMap<>();
```

```
        int rows = grid.length;
        int cols = grid[0].length;
        int colorIndex = 0;
        for (int i = 0; i < rows; i++){
            for (int j = 0; j < cols; j++){
                char ch = grid[i][j];
                if(ch != '.' &&
!colorMap.containsKey(ch)){
                    colorMap.put(ch,
ANSI_COLORS[colorIndex % ANSI_COLORS.length]);
                    colorIndex++;
                }
            }
        }
        for (int i = 0; i < rows; i++){
            for (int j = 0; j < cols; j++){
                char ch = grid[i][j];
                if(ch == '.')
                    System.out.print(ch);
                else
                    System.out.print(colorMap.get(ch) +
ch + ANSI_RESET + " ");
            }
            System.out.println();
        }
    }



}
```

3.4. PuzzlePiece.java

```
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

public class PuzzlePiece {
    // ATTRIBUTES
    private char id;
    private char[][] shape;
    private List<char[][]> orientations;
    private static int count = 0;

    // CONSTRUCTORS
    public PuzzlePiece(char id, char[][] shape) {
        this.id = id;
        this.shape = shape;
        this.orientations = generateAllOrientations();
    }

    private List<char[][]> generateAllOrientations() {
        List<char[][]> result = new ArrayList<>();
        Set<String> seen = new HashSet<>();

        char[][] current = shape;
        for (int i = 0; i < 4; i++) {
            addIfUnique(current, result, seen);
            char[][] flipped =
flipHorizontally(current);
```

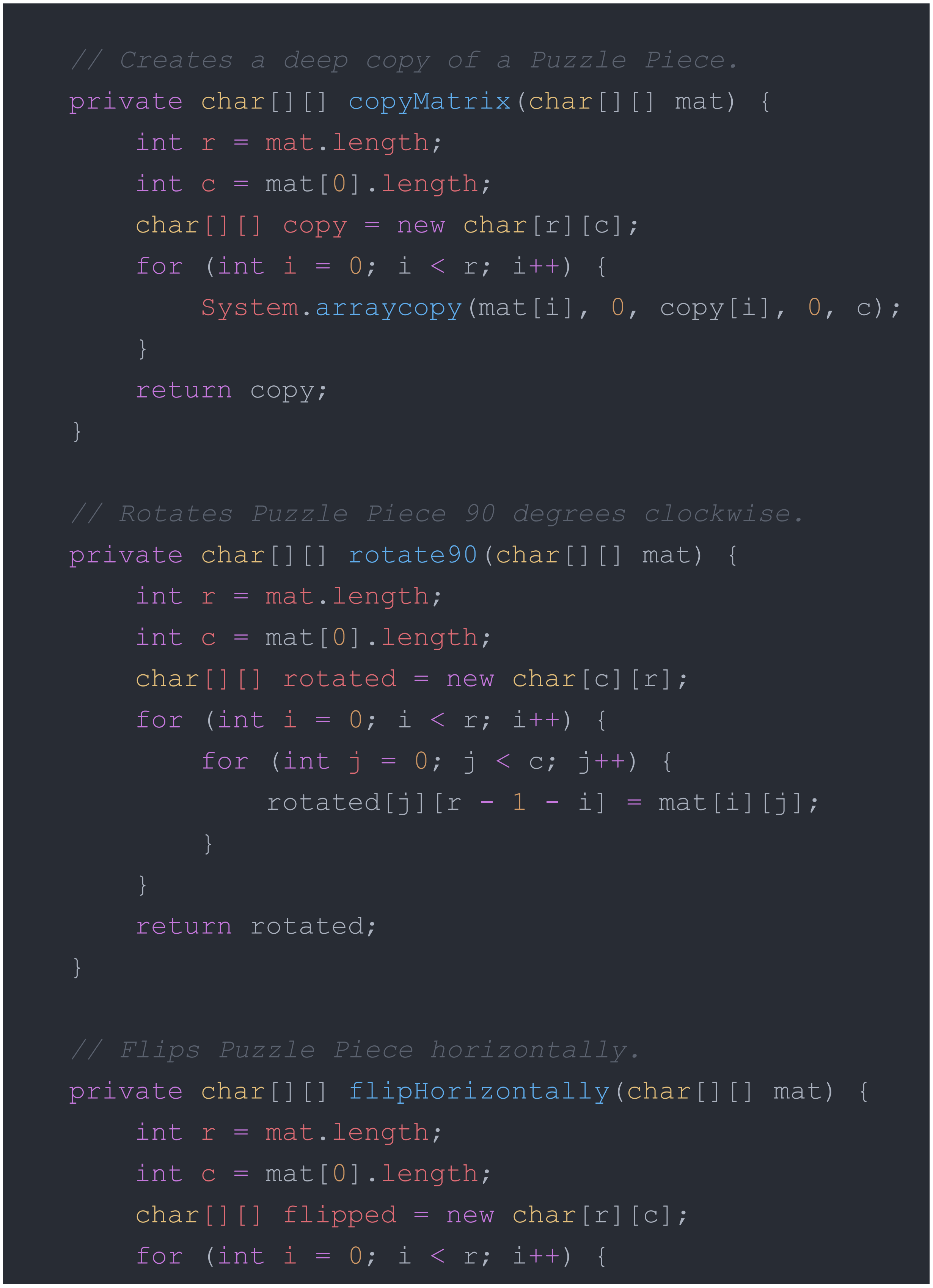
```
            addIfUnique(flipped, result, seen);
            current = rotate90(current);
        }

        return result;
    }

    // NOTE: generateAllOrientations() HELPER
    // Add Puzzle Piece orientation to the list if it is
unique.
    private void addIfUnique(char[][] mat,
List<char[][]> list, Set<String> seen) {
        String key = matrixToString(mat);
        if (!seen.contains(key)) {
            seen.add(key);
            list.add(copyMatrix(mat));
        }
    }

    // Converts Puzzle Piece into a String. (for
HashSet's Keys)
    private String matrixToString(char[][] mat) {
        StringBuilder sb = new StringBuilder();
        for (char[] row : mat) {
            sb.append(new String(row));
            sb.append("\n");
        }
        return sb.toString();
    }

    // Creates a deep copy of a Puzzle Piece.
```

```java
    private char[][] copyMatrix(char[][] mat) {
        int r = mat.length;
        int c = mat[0].length;
        char[][] copy = new char[r][c];
        for (int i = 0; i < r; i++) {
            System.arraycopy(mat[i], 0, copy[i], 0, c);
        }
        return copy;
    }

    // Rotates Puzzle Piece 90 degrees clockwise.
    private char[][] rotate90(char[][] mat) {
        int r = mat.length;
        int c = mat[0].length;
        char[][] rotated = new char[c][r];
        for (int i = 0; i < r; i++) {
            for (int j = 0; j < c; j++) {
                rotated[j][r - 1 - i] = mat[i][j];
            }
        }
        return rotated;
    }

    // Flips Puzzle Piece horizontally.
    private char[][] flipHorizontally(char[][] mat) {
        int r = mat.length;
        int c = mat[0].length;
        char[][] flipped = new char[r][c];
        for (int i = 0; i < r; i++) {
            for (int j = 0; j < c; j++) {
                flipped[i][c - 1 - j] = mat[i][j];
            }
        }
        return flipped;
    }

    //SELECTOR
    public char getId() {
        return id;
    }

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

    public List<char[][]> getOrientations() {
        return orientations;
    }

    public static int getCount() {
        return count;
    }

    // METHODS
    public static void addCount(int add) {
        count += add;
    }

    public static void resetCount() {
        count = 0;
    }
```

```
}
```

3.5. IOPuzzlerFile.java

```
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

public class PuzzlePiece {
    // ATTRIBUTES
    private char id;
    private char[][] shape;
    private List<char[][]> orientations;
    private static int count = 0;

    // CONSTRUCTORS
    public PuzzlePiece(char id, char[][] shape) {
        this.id = id;
        this.shape = shape;
        this.orientations = generateAllOrientations();
    }

    private List<char[][]> generateAllOrientations() {
        List<char[][]> result = new ArrayList<>();
        Set<String> seen = new HashSet<>();

        char[][] current = shape;
        for (int i = 0; i < 4; i++) {
            addIfUnique(current, result, seen);
            char[][] flipped =
flipHorizontally(current);
            addIfUnique(flipped, result, seen);
            current = rotate90(current);
        }

        return result;
    }

    // NOTE: generateAllOrientations() HELPER
    // Add Puzzle Piece orientation to the list if it is
unique.
    private void addIfUnique(char[][] mat,
List<char[][]> list, Set<String> seen) {
        String key = matrixToString(mat);
        if (!seen.contains(key)) {
            seen.add(key);
            list.add(copyMatrix(mat));
        }
    }

    // Converts Puzzle Piece into a String. (for
HashSet's Keys)
    private String matrixToString(char[][] mat) {
        StringBuilder sb = new StringBuilder();
        for (char[] row : mat) {
            sb.append(new String(row));
            sb.append("\n");
        }
        return sb.toString();
    }
```



```java
// Creates a deep copy of a Puzzle Piece.
private char[][] copyMatrix(char[][] mat) {
    int r = mat.length;
    int c = mat[0].length;
    char[][] copy = new char[r][c];
    for (int i = 0; i < r; i++) {
        System.arraycopy(mat[i], 0, copy[i], 0, c);
    }
    return copy;
}

// Rotates Puzzle Piece 90 degrees clockwise.
private char[][] rotate90(char[][] mat) {
    int r = mat.length;
    int c = mat[0].length;
    char[][] rotated = new char[c][r];
    for (int i = 0; i < r; i++) {
        for (int j = 0; j < c; j++) {
            rotated[j][r - 1 - i] = mat[i][j];
        }
    }
    return rotated;
}

// Flips Puzzle Piece horizontally.
private char[][] flipHorizontally(char[][] mat) {
    int r = mat.length;
    int c = mat[0].length;
    char[][] flipped = new char[r][c];
    for (int i = 0; i < r; i++) {
        for (int j = 0; j < c; j++) {
            flipped[i][c - 1 - j] = mat[i][j];
        }
    }
    return flipped;
}

//SELECTOR
public char getId() {
    return id;
}

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

public List<char[][]> getOrientations() {
    return orientations;
}

public static int getCount() {
    return count;
}

// METHODS
public static void addCount(int add) {
    count += add;
}

public static void resetCount() {
    count = 0;
```

```
        }


}
```

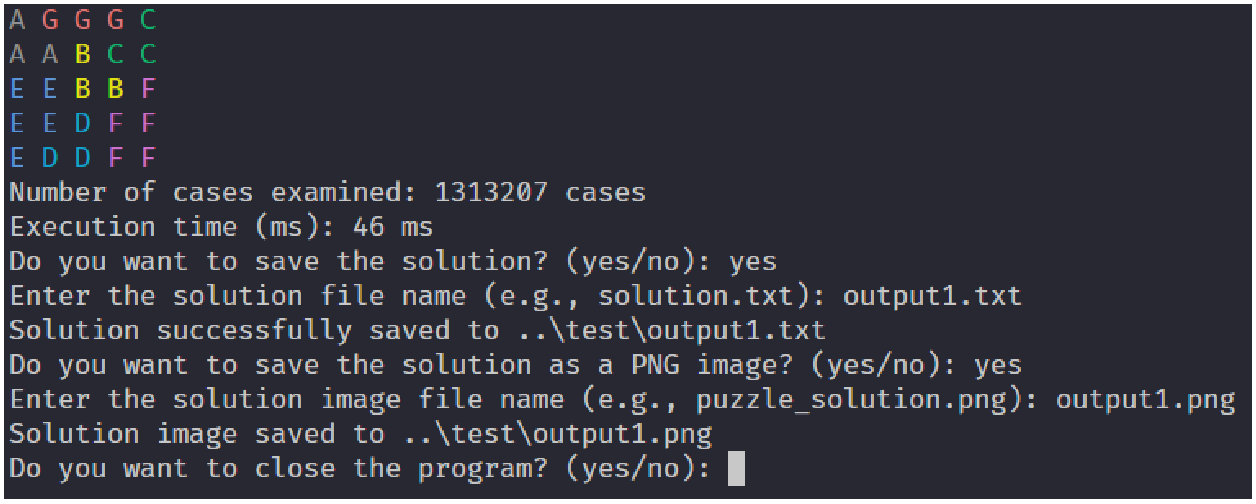
BAB IV
EKSPERIMEN

1. *input1.txt*

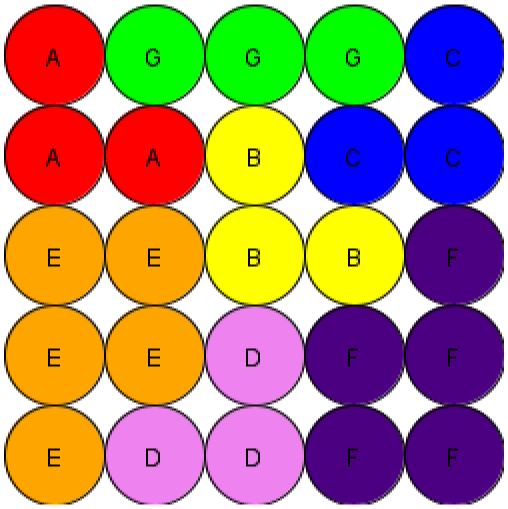
Input:

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

Output:



```
A G G G C
A A B C C
E E B B F
E E D F F
E D D F F
Number of cases examined: 1313207 cases
Execution time (ms): 46 ms
Do you want to save the solution? (yes/no): yes
Enter the solution file name (e.g., solution.txt): output1.txt
Solution successfully saved to ..\test\output1.txt
Do you want to save the solution as a PNG image? (yes/no): yes
Enter the solution image file name (e.g., puzzle_solution.png): output1.png
Solution image saved to ..\test\output1.png
Do you want to close the program? (yes/no):
```

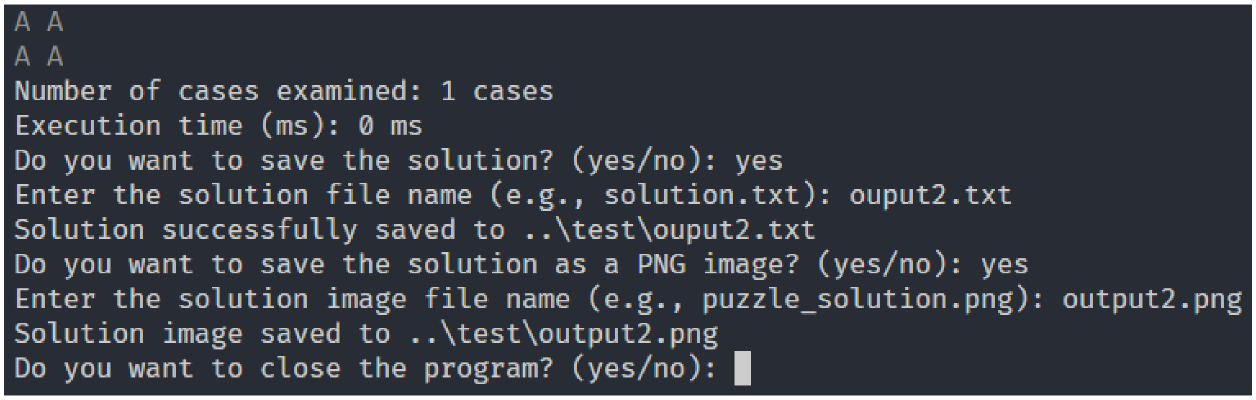


2. *input2.txt*

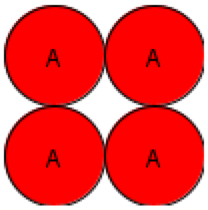
Input:

```
2 2 1
DEFAULT
AA
AA
```

Output:



```
A A
A A
Number of cases examined: 1 cases
Execution time (ms): 0 ms
Do you want to save the solution? (yes/no): yes
Enter the solution file name (e.g., solution.txt): ouput2.txt
Solution successfully saved to ..\test\ouput2.txt
Do you want to save the solution as a PNG image? (yes/no): yes
Enter the solution image file name (e.g., puzzle_solution.png): output2.png
Solution image saved to ..\test\output2.png
Do you want to close the program? (yes/no):
```

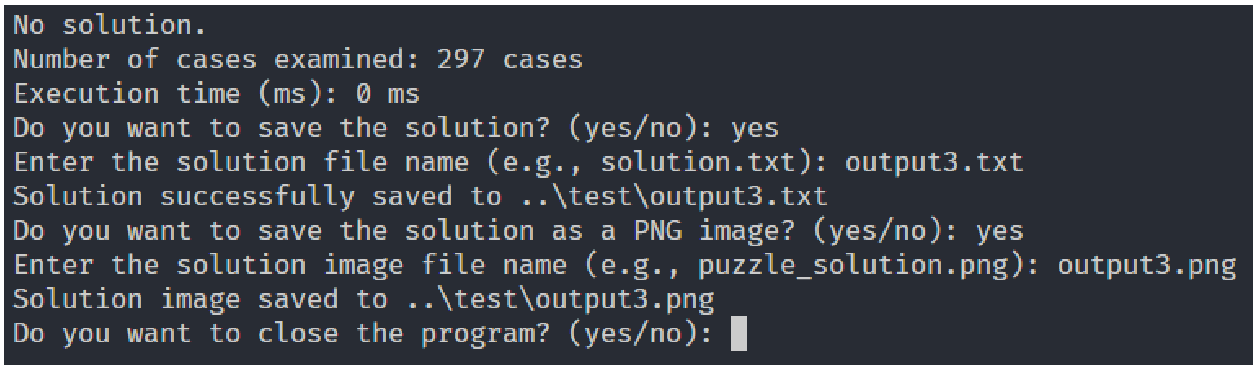


3. *input3.txt*

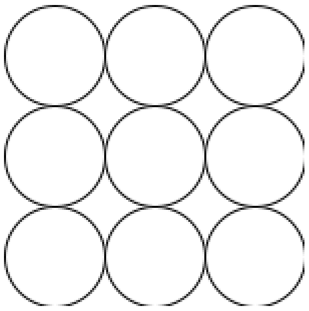
Input:

```
3 3 2
DEFAULT
AA
AA
BB
BBB
```

Output:



```
No solution.
Number of cases examined: 297 cases
Execution time (ms): 0 ms
Do you want to save the solution? (yes/no): yes
Enter the solution file name (e.g., solution.txt): output3.txt
Solution successfully saved to ..\test\output3.txt
Do you want to save the solution as a PNG image? (yes/no): yes
Enter the solution image file name (e.g., puzzle_solution.png): output3.png
Solution image saved to ..\test\output3.png
Do you want to close the program? (yes/no):
```



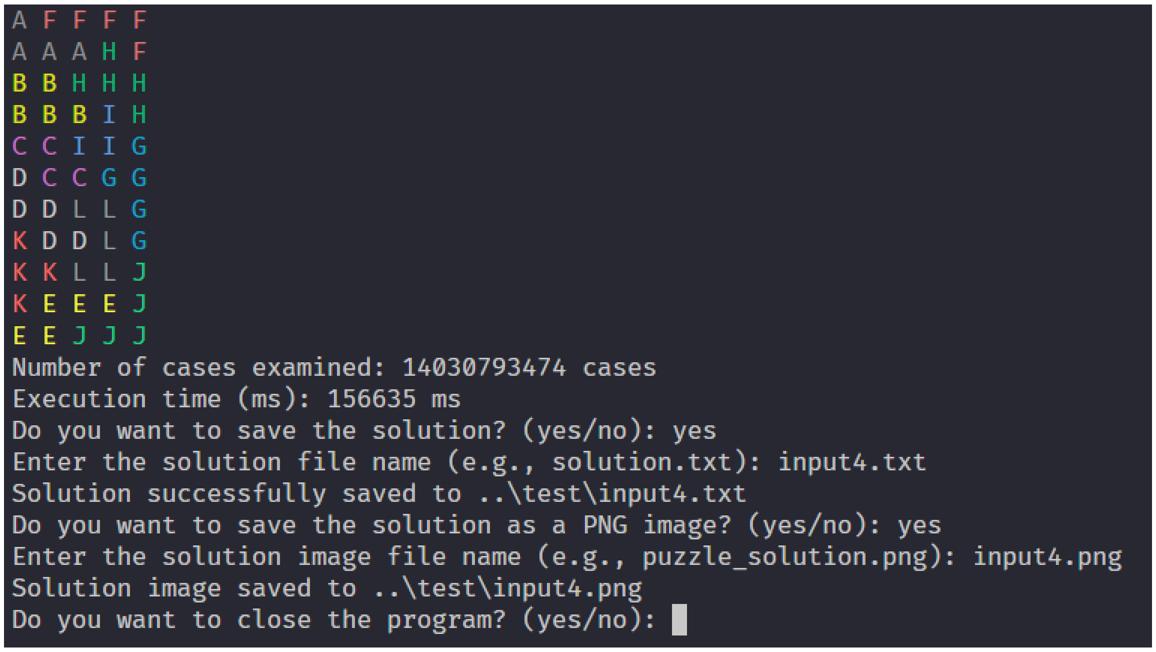
4. *input4.txt*

Input:

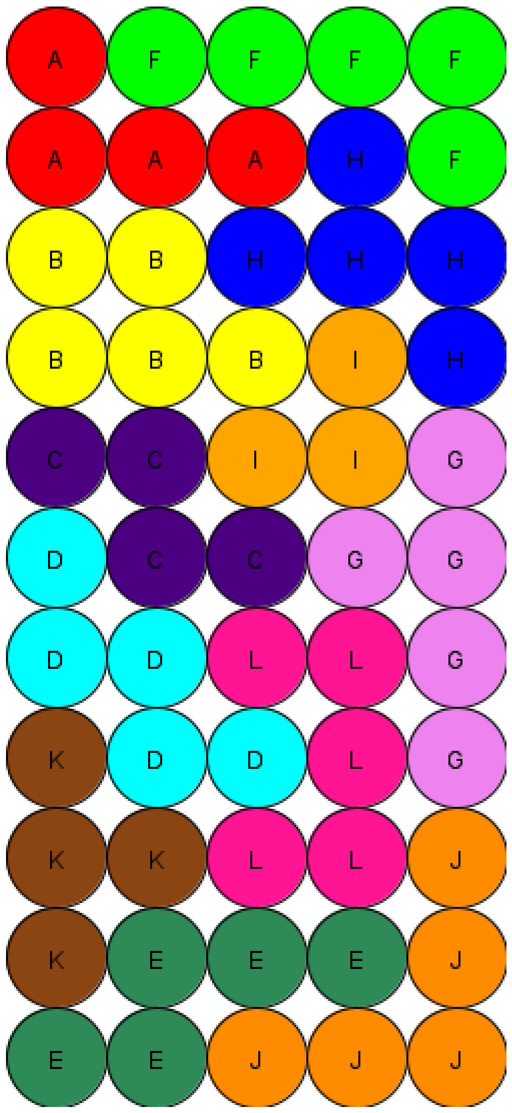
```
11 5 12
DEFAULT
A
AAA
BB
BBB
CC
 CC
D
```

```
DD
 DD
EEE
  EE
F
FFFF
GGGG
  G
 H
HH
 HH
II
I
J
J
JJJ
K
KK
K
LL
L
LL
```

Output:



```
A F F F F
A A A H F
B B H H H
B B B I H
C C I I G
D C C G G
D D L L G
K D D L G
K K L L J
K E E E J
E E J J J
Number of cases examined: 14030793474 cases
Execution time (ms): 156635 ms
Do you want to save the solution? (yes/no): yes
Enter the solution file name (e.g., solution.txt): input4.txt
Solution successfully saved to ..\test\input4.txt
Do you want to save the solution as a PNG image? (yes/no): yes
Enter the solution image file name (e.g., puzzle_solution.png): input4.png
Solution image saved to ..\test\input4.png
Do you want to close the program? (yes/no): 
```



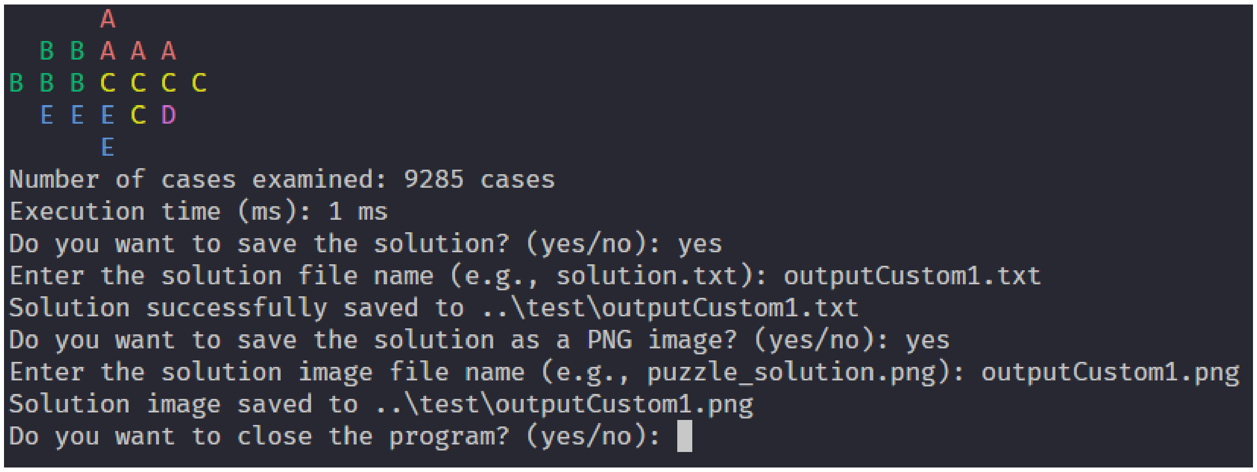
5. *inputCustom1.txt*

Input:

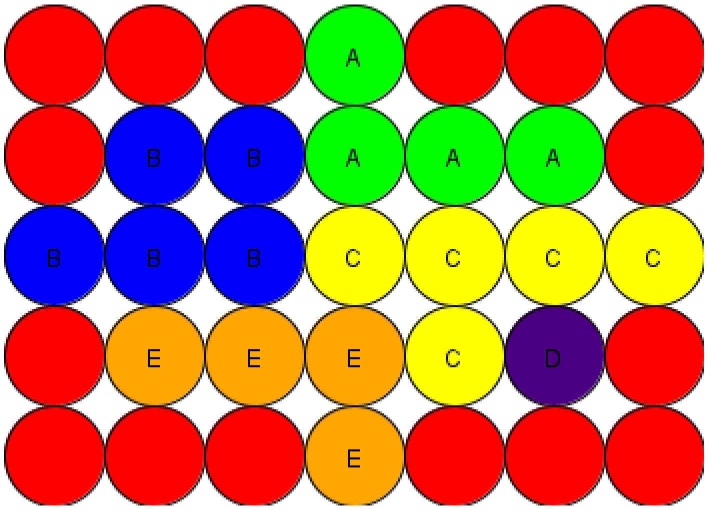
```
5 7 5
CUSTOM
...X...
.XXXXX.
XXXXXXX
.XXXXX.
...X...
A
AAA
BB
BBB
```

```
CCCC
 C
D
EEE
E
```

Output:



```
      A
  B B A A A
B B B C C C C
  E E E C D
      E
Number of cases examined: 9285 cases
Execution time (ms): 1 ms
Do you want to save the solution? (yes/no): yes
Enter the solution file name (e.g., solution.txt): outputCustom1.txt
Solution successfully saved to ..\test\outputCustom1.txt
Do you want to save the solution as a PNG image? (yes/no): yes
Enter the solution image file name (e.g., puzzle_solution.png): outputCustom1.png
Solution image saved to ..\test\outputCustom1.png
Do you want to close the program? (yes/no):
```



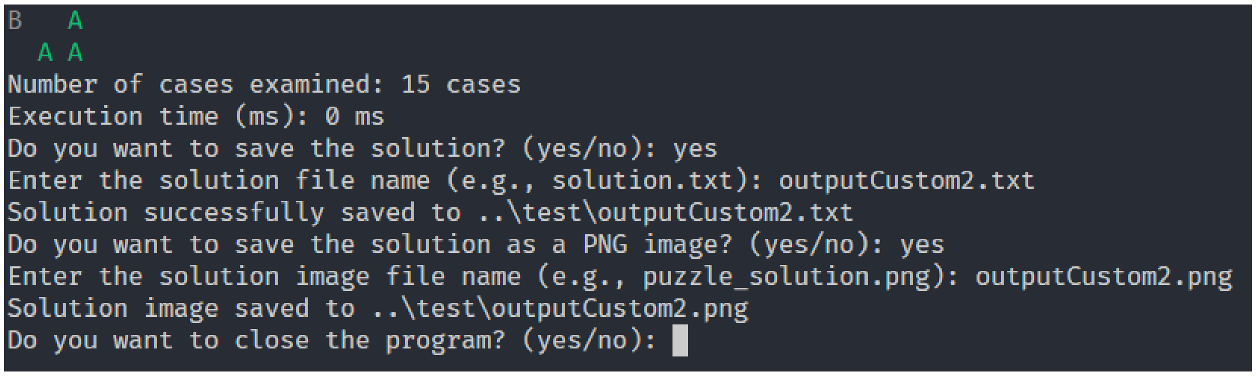
6. *InputCustom2.txt*

Input:

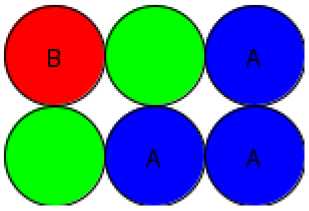
```
2 3 2
CUSTOM
X.X
.XX
AA
A
```

```
B
```

Output:



```
B   A
  A A
Number of cases examined: 15 cases
Execution time (ms): 0 ms
Do you want to save the solution? (yes/no): yes
Enter the solution file name (e.g., solution.txt): outputCustom2.txt
Solution successfully saved to ..\test\outputCustom2.txt
Do you want to save the solution as a PNG image? (yes/no): yes
Enter the solution image file name (e.g., puzzle_solution.png): outputCustom2.png
Solution image saved to ..\test\outputCustom2.png
Do you want to close the program? (yes/no): 
```

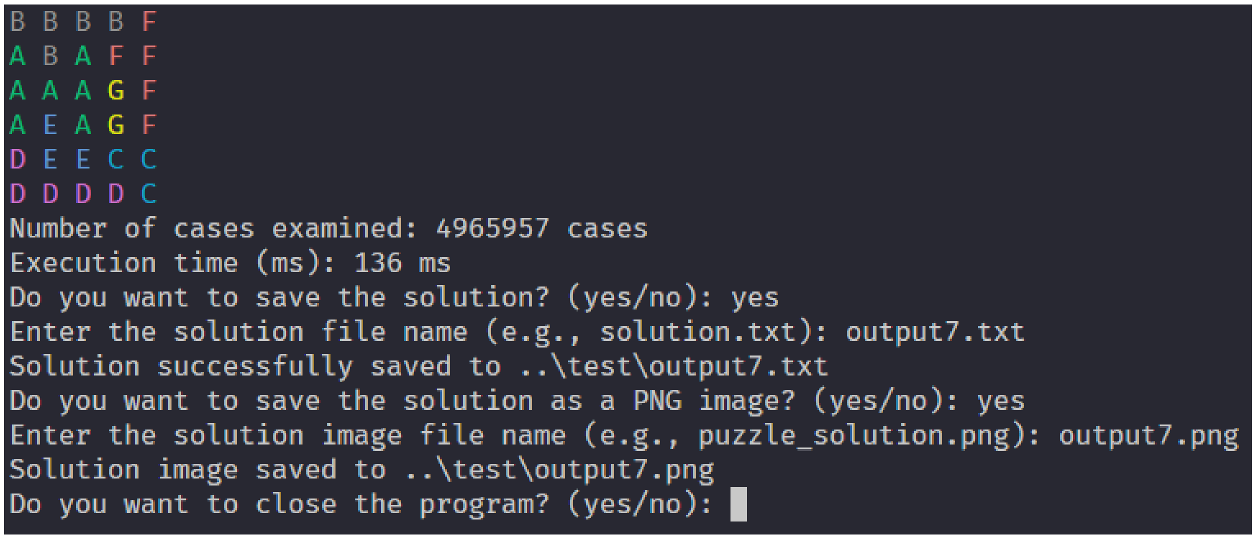


7. *Input6.txt*

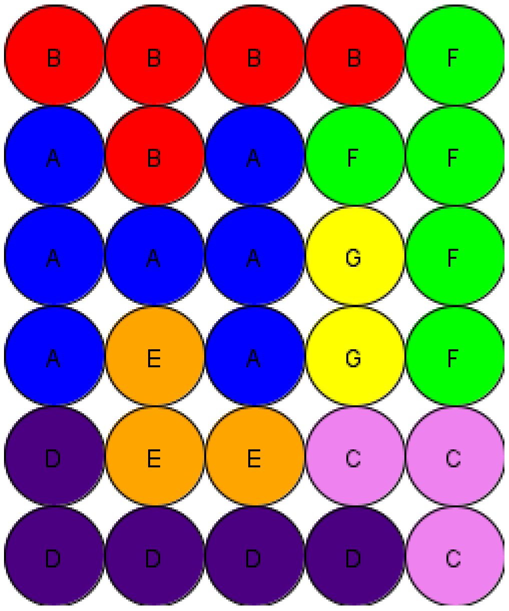
Input:

```
6 5 8
DEFAULT
A A
AAA
A A
BBBB
 B
CC
C
DDDD
D
EE
E
FFFF
  F
GG
HHHH
```

Output:



```
B B B B F
A B A F F
A A A G F
A E A G F
D E E C C
D D D D C
Number of cases examined: 4965957 cases
Execution time (ms): 136 ms
Do you want to save the solution? (yes/no): yes
Enter the solution file name (e.g., solution.txt): output7.txt
Solution successfully saved to ..\test\output7.txt
Do you want to save the solution as a PNG image? (yes/no): yes
Enter the solution image file name (e.g., puzzle_solution.png): output7.png
Solution image saved to ..\test\output7.png
Do you want to close the program? (yes/no): 
```



DAFTAR PUSTAKA

1. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/02-Algoritma-Brute-Force-(2025)-Bag1.pdf
2. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/Tucil1-Stima-2025.pdf

LAMPIRAN

Link repository Github:

https://github.com/Starath/Tucil1_13523106

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	✔	