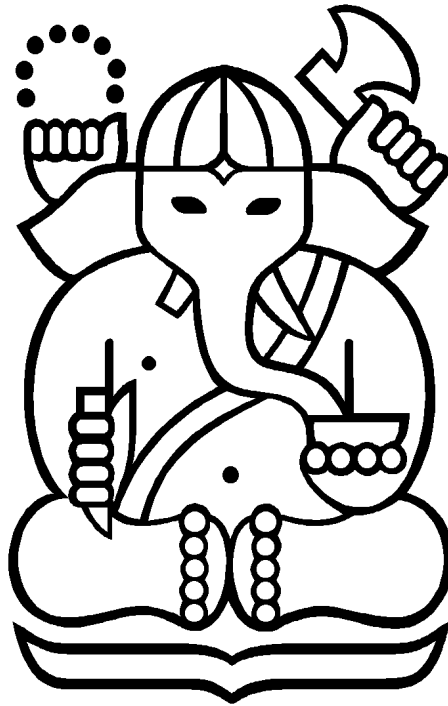


LAPORAN TUGAS KECIL 1
IF 2211 STRATEGI ALGORITMA
PENYELESAIAN IQ PUZZLE PRO DENGAN ALGORITMA BRUTE FORCE



Oleh :
Diyah Susan Nugrahani (13523080)

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2025

1. Algoritma *Brute Force*

Dalam menyelesaikan persoalan IQ puzzle pro menggunakan algoritma *brute force* saya menggunakan pendekatan *object oriented programming* dengan membagi persoalan puzzle ini sebagai objek-objek yang dapat berinteraksi. Saya mengidentifikasi persoalan puzzle ini memiliki dua buah objek utama yaitu *block puzzle* dan *grid* papan. Kedua objek ini yang akan berinteraksi dan diolah sampai didapatkan solusi yang bisa menampung seluruh *block puzzle* ke dalam *grid*.

Block dalam *puzzle* ini memiliki beberapa atribut berupa karakter huruf yang mewakili tiap *block* serta bentuk berupa matriks dua dimensi. Setiap *block* memiliki beberapa metode penting yang dapat dipanggil yaitu rotasi dan pencerminan yang nantinya akan digunakan untuk menyesuaikan posisi *block* agar dapat ditempatkan di dalam *grid*. Dalam hal ini saya hanya menggunakan *rotateClockWise* dan *mirrorHorizontally* karena pada dasarnya setiap *block* maksimal hanya memiliki delapan kemungkinan variasi bentuk dan hal tersebut dapat dicapai dengan mengombinasikan *rotateClockWise* dan *mirrorHorizontally*.

Grid atau papan yang akan digunakan sebagai tempat meletakkan *block* memiliki beberapa atribut berupa bentuk yang direpresentasikan sebagai matriks dua dimensi serta ukuran baris dan kolom. *Grid* ini memiliki beberapa metode penting yang dapat dipanggil yaitu *canPlaceBlock* yang berfungsi untuk mengecek apakah sebuah *block* dapat diletakkan dalam *grid*, *placeBlock* yang berfungsi untuk meletakkan *block* ke dalam *grid*, *printGrid* untuk mencetak solusi di terminal, *saveGrid* untuk menyimpan solusi sebagai sebuah file txt, dan *saveGridAsPNG* untuk menyimpan solusi dalam format gambar.

Algoritma yang saya gunakan untuk mencari solusi *puzzle* ini adalah algoritma *brute force* dengan pendekatan rekursif. Pertama program akan membaca *input* dari file txt dan kemudian melakukan *mapping* dengan menyimpan setiap *block* sesuai dengan hurufnya. Lalu program akan mencoba meletakkan *block* mulai dari indeks pertama dan dicek apakah *block* dapat diletakkan di *grid*. Jika dapat diletakkan maka akan lanjut mencoba ke *block* berikutnya dengan rekursi. Apabila ternyata *block* tidak dapat diletakkan maka akan dicoba variasi bentuk lain dengan cara dirotasi dan dicerminkan. Jika masih tidak bisa diletakkan maka dicoba geser ke *grid* lainnya. Apabila *block* masih tidak dapat diletakkan maka akan dilakukan *backtracking* dengan cara mengubah variasi posisi *block* sebelumnya dengan dirotasi dan dicerminkan sampai ditemukan posisi yang sesuai. Ketika seluruh *block* berhasil diletakkan di dalam *grid*, rekursi akan berhenti yang berarti solusi berhasil ditemukan dan program akan mengeluarkan *output* berupa bentuk solusi, banyak percobaan, dan durasi pencarian solusi. Berikut merupakan *pseudocode* dari algoritma *brute force* yang saya gunakan.

```
function boolean solutionBF (Grid grid, Map blocks, int    currentBlockIndex, int count) → boolean
    if (currentBlockIndex >= blocks.size) then
        → true
    Block currentBlock ← blocks[currentBlockIndex]
    Char[][] originalGrid ← copyGrid
```

```

traversal [0...rows]
  traversal [0...cols]
    traversal [0...3]
      count ← count + 1
      if (canPlaceBlock(currentBlock, row, col) then
        placeBlock(currentBlock, row, col)
        if (solutionBF(grid, blocks, currentBlockIndex + 1, count) then
          → true
          setGrid(originalGrid)
          {jika gagal maka dicoba rotasi}
          currentBlock.rotateClockWise

          {variasi dicerminkan terlebih dahulu}
          currentBlock.mirrorHorizontally
          traversal [0...3]
            count ← count + 1
            if (canPlaceBlock(currentBlock, row, col) then
              placeBlock(currentBlock, row, col)
              if (solutionBF(grid, blocks, currentBlockIndex + 1, count) then
                → true
                setGrid(originalGrid)
                {jika gagal maka dicoba rotasi}
                currentBlock.rotateClockWise

            → false

```

2. Source Code

Berikut merupakan *source code* lengkap program yang telah saya buat. Untuk menjalankan program ini bisa dilakukan dengan mengakses tautan *repository* yang terlampir.

2.1 Block.java

```

package component;

public class Block {
    private char letter;
    private char[][] shape;

    public Block(char letter) {
        this.letter = letter;
        this.shape = new char[0][0];
    }

    public char[][] getShape() {
        return shape; // representasi blok sebagai matriks 2D
    }

    public void addRow(String row) {
        int currentRows = shape.length;
        int rowLength = row.length();

        int maxCols = 0;
        if (currentRows > 0) {
            maxCols = shape[0].length;
        }
    }

```

```

        maxCols = Math.max(maxCols, rowLength);

        // matriks dengan dimensi baru
        char[][] newShape = new char[currentRows + 1][maxCols];

        for (int i = 0; i < currentRows; i++) {
            for (int j = 0; j < maxCols; j++) {
                newShape[i][j] = (j < shape[i].length) ? shape[i][j] : '.';
            }
        }

        for (int j = 0; j < maxCols; j++) {
            newShape[currentRows][j] = (j < rowLength) ? row.charAt(j) : '.';
        }

        shape = newShape;
    }

    public void rotateClockwise() {
        int rows = shape.length;
        int cols = shape[0].length;

        char[][] rotatedShape = new char[cols][rows];

        // rotasi matriks 90 derajat searah jarum jam
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                rotatedShape[j][rows - 1 - i] = shape[i][j];
            }
        }

        shape = rotatedShape;
    }

    public void mirrorHorizontally() {
        int rows = shape.length;
        for (int r = 0; r < rows / 2; r++) {
            char[] temp = shape[r];
            shape[r] = shape[rows - 1 - r];
            shape[rows - 1 - r] = temp;
        }
    }

    @Override
    public String toString() {
        StringBuilder sb = new StringBuilder("Block ").append(letter).append(": \n");
        for (char[] row : shape) {
            sb.append(new String(row)).append("\n");
        }
        return sb.toString();
    }

```

```
}  
}
```

2.2 Grid.java

```
package component;  
import java.awt.*;  
import java.awt.image.BufferedImage;  
import java.io.File;  
import java.io.FileWriter;  
import java.io.IOException;  
import java.util.LinkedHashMap;  
import java.util.Map;  
import javax.imageio.ImageIO;  
  
public class Grid {  
    private char[][] grid;  
    private int rows;  
    private int cols;  
  
    public Grid (int rows, int cols){  
        this.rows = rows;  
        this.cols = cols;  
        grid = new char[rows][cols];  
        for (int i = 0; i < rows; i++) {  
            for (int j = 0; j < cols; j++) {  
                grid[i][j] = '.';  
            }  
        }  
    }  
  
    public int getRows(){  
        return rows;  
    }  
  
    public int getCols(){  
        return cols;  
    }  
  
    public boolean canPlaceBlock(Block block, int startRow, int startCol) {  
        char[][] shape = block.getShape();  
        int blockRows = shape.length;  
        int blockCols = shape[0].length;  
  
        for (int i = 0; i < blockRows; i++) {  
            for (int j = 0; j < blockCols; j++) {  
                if (shape[i][j] != '.') {  
                    int gridRow = startRow + i;  
                    int gridCol = startCol + j;  
  
                    // cek apakah posisi di dalam grid dan tidak terisi
```

```

        if (gridRow >= rows || gridCol >= cols || grid[gridRow][gridCol]
!= '.') {
            return false;
        }
    }
}

return true;
}

public void placeBlock(Block block, int startRow, int startCol) {
    char[][] shape = block.getShape();
    int blockRows = shape.length;
    int blockCols = shape[0].length;

    for (int i = 0; i < blockRows; i++) {
        for (int j = 0; j < blockCols; j++) {
            if (shape[i][j] != '.') {
                grid[startRow + i][startCol + j] = shape[i][j];
            }
        }
    }
}

public char[][] copyGrid() {
    char[][] copy = new char[rows][cols];
    for (int i = 0; i < rows; i++) {
        System.arraycopy(grid[i], 0, copy[i], 0, cols);
    }
    return copy;
}

public void setGrid(char[][] newGrid) {
    for (int i = 0; i < rows; i++) {
        System.arraycopy(newGrid[i], 0, grid[i], 0, cols);
    }
}

public void printGrid() {
    Map<Character, String> colorMap = new LinkedHashMap<>();
    String resetColor = "\u001B[0m";

    // print dengan warna
    for (int i = 0; i < 26; i++) {
        char block = (char) ('A' + i);
        String color = "\u001B[38;5;" + (i*18) + "m";
        colorMap.put(block, color);
    }

    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {

```

```

        char cell = grid[i][j];
        String color = colorMap.getOrDefault(cell, "\u001B[37m");
        System.out.print(color + cell + " " + resetColor);
    }
    System.out.println();
}

}

public void saveGrid(String filename) {
    try (FileWriter writer = new FileWriter(filename)) {
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                char cell = grid[i][j];
                writer.write(cell + " ");
            }
            writer.write(System.lineSeparator());
        }
        System.out.println();
        System.out.println("Berhasil menyimpan solusi.");
    } catch (IOException e) {
        System.out.println("Error saat menyimpan solusi : " + e.getMessage());
    }
}

public void saveGridAsPNG(String filename) {
    final int size = 30;
    final int padding = 2;

    // dimensi
    int width = cols * size;
    int height = rows * size;

    GraphicsEnvironment ge = GraphicsEnvironment.getLocalGraphicsEnvironment();
    GraphicsDevice gd = ge.getDefaultScreenDevice();
    GraphicsConfiguration gc = gd.getDefaultConfiguration();

    BufferedImage image = gc.createCompatibleImage(width, height,
Transparency.OPAQUE);
    Graphics2D g2d = image.createGraphics();

    // agar text tidak pecah
    g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
    g2d.setRenderingHint(RenderingHints.KEY_TEXT_ANTIALIASING,
RenderingHints.VALUE_TEXT_ANTIALIAS_ON);

    // background
    g2d.setPaint(java.awt.Color.WHITE);
    g2d.fillRect(0, 0, width, height);

    Map<Character, Color> colorMap = new LinkedHashMap<>();

```

```

        for (int i = 1; i < 27; i++) {
            char block = (char) ('A' + i);
            int red = (i * 15) % 256;
            int green = (i * 30) % 256;
            int blue = (i * 45) % 256;
            Color color = new Color(red, green, blue);
            colorMap.put(block, color);
        }

        Color defaultColor = Color.WHITE;

        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                int x = j * size;
                int y = i * size;

                char cell = grid[i][j];
                if (cell != ' ') {

                    Color cellColor = colorMap.getOrDefault(cell, defaultColor);

                    g2d.setPaint(cellColor);
                    g2d.fillRect(x + padding, y + padding, size - 2*padding, size -
2*padding);

                    g2d.setPaint(java.awt.Color.BLACK);
                    g2d.drawRect(x + padding, y + padding, size - 2*padding, size -
2*padding);

                    g2d.setFont(new java.awt.Font("SansSerif", java.awt.Font.PLAIN,
size/2));

                    FontMetrics fm = g2d.getFontMetrics();
                    String text = String.valueOf(cell);
                    int textX = x + (size - fm.stringWidth(text))/2;
                    int textY = y + ((size - fm.getHeight())/2) + fm.getAscent();
                    g2d.setPaint(java.awt.Color.BLACK);
                    g2d.drawString(text, textX, textY);
                }
            }
        }

        g2d.dispose();

        // menyimpan gambar
        try {
            if (!filename.toLowerCase().endsWith(".png")) {
                filename += ".png";
            }
            File outputFile = new File(filename);
            ImageIO.write(image, "PNG", outputFile);
            System.out.println("Berhasil menyimpan gambar.");
        }
    }
}

```



```

    } catch (IOException e) {
        System.out.println("Error saat menyimpan gambar: " + e.getMessage());
    }
}
}

```

2.3 Solution.java

```

package component;
import java.util.Map;

public class Solution {
    public static boolean solutionBF(Grid grid, Map<Character, Block> blocks, int
currentBlockIndex, int count) {
        if (currentBlockIndex >= blocks.size()) {
            System.out.println("Banyak percobaan : " + count);
            // semua blok telah ditempatkan, puzzle selesai
            return true;
        }

        // ambil blok saat ini berdasarkan indeks
        Block currentBlock = (Block) blocks.values().toArray()[currentBlockIndex];

        char[][] originalGrid = grid.copyGrid();

        // coba semua posisi di grid
        for (int row = 0; row < grid.getRows(); row++) {
            for (int col = 0; col < grid.getCols(); col++) {
                // rotasi
                for (int rotation = 0; rotation < 4; rotation++) {
                    count += 1;
                    if (grid.canPlaceBlock(currentBlock, row, col)) {
                        grid.placeBlock(currentBlock, row, col);

                        // rekursi untuk blok berikutnya
                        if (solutionBF(grid, blocks, currentBlockIndex + 1, count)) {
                            return true;
                        }

                        // backtrack jika solusi gagal
                        grid.setGrid(originalGrid);
                    }

                    currentBlock.rotateClockwise();
                }

                // mirror
                currentBlock.mirrorHorizontally();
                for (int rotation = 0; rotation < 4; rotation++) {
                    count += 1;

```



```

        if (result == JFileChooser.APPROVE_OPTION) {
            File selectedFile = fileChooser.getSelectedFile();

            try (BufferedReader br = new BufferedReader(new
FileReader(selectedFile))) {
                // membaca dimensi
                String[] dimensions = br.readLine().trim().split(" ");
                int rows = Integer.parseInt(dimensions[0]);
                int cols = Integer.parseInt(dimensions[1]);

                int blockCount = Integer.parseInt(dimensions[2]);

                if (rows <= 0 || cols <= 0) {
                    valid = false;
                }

                // type
                String type = br.readLine().trim();

                // membaca blocks dan menyimpannya di sebuah map
                Map<Character, Block> blocks = new LinkedHashMap<>();
                Character currentBlock = null;
                int counter = 0;

                while (blocks.size() < blockCount || currentBlock != null &&
valid) {

                    String line = br.readLine();
                    if (line != null) {
                        // kalau ada kasus space maka akan diubah sebagai dot
                        line = line.replace(' ', '.');

                        if (!line.trim().isEmpty()) {
                            char firstChar = line.trim().charAt(0);

                            // jika ada block baru
                            if (currentBlock == null || (firstChar != '.' &&
currentBlock != firstChar)) {

                                currentBlock = firstChar;
                                blocks.putIfAbsent(currentBlock, new
Block(currentBlock));
                            }

                            Block block = blocks.get(currentBlock);
                            if (block != null) {
                                block.addRow(line);
                            } else {
                                System.err.println("Error: currentBlock tidak
ditemukan dalam blocks.");
                            }
                        }
                    }
                }
            }
        }
    }
}

```

```

        } else {
            currentBlock = null;
        }
    } else {
        break;
    }
}

if (valid){
    // menghitung jumlah kotak
    for (Block block : blocks.values()) {
        char [][] shape = block.getShape();
        int baris = shape.length;
        int kolom = shape[0].length;
        for (int i=0; i<baris; i++){
            for (int j=0; j<kolom; j++){
                if(shape[i][j] != '.'){
                    counter += 1;
                }
            }
        }
    }
}

// validasi input
if (blocks.size() > 26){
    valid = false;
}
else if (blockCount != blocks.size()){
    valid = false;
}
else if (counter != rows * cols){
    valid = false;
}

if (type.equalsIgnoreCase("DEFAULT") && valid){
    // grid papan
    Grid grid = new Grid(rows, cols);
    // menghitung waktu
    long startTime = System.currentTimeMillis();
    if (Solution.solutionBF(grid, blocks, 0, 0)) {
        System.out.println("Puzzle berhasil diselesaikan.");
        grid.printGrid();
    } else {
        System.out.println("Tidak ada solusi untuk puzzle.");
    }
    long endTime = System.currentTimeMillis();

    long duration = endTime - startTime;
    System.out.println("Durasi : " + duration + " ms");
}

```

```

        Scanner scanner = new Scanner(System.in);

        System.out.println();
        System.out.print("Apakah anda ingin menyimpan solusi?
(ya/tidak) : ");

        String save = scanner.nextLine();

        if (save.equalsIgnoreCase("ya")){
            System.out.print("Masukkan nama file yang ingin disimpan
: ");

            String fileName = scanner.nextLine();
            grid.saveGrid("test/" + fileName);
            grid.saveGridAsPNG("test/" + fileName);
        }
    }
    else{
        valid = false;
        System.out.println("Input tidak valid!");
        System.out.println("Silahkan memasukkan test case lainnya!");




    }
} catch (IOException e) {
    System.out.println("Terjadi kesalahan saat membaca file: " +
e.getMessage());
}
} else {
    System.out.println("Tidak ada file yang dipilih.");
}
}
}
}



```



3. Tangkapan Layar

Berikut merupakan hasil dari *test case* yang telah saya buat. Terdapat beberapa tipe *test case* mulai dari yang sederhana hingga cukup kompleks. Selain itu juga terlampir hasil output gambar dari masing-masing *test case*.


<pre> 3 3 3 DEFAULT AAA A BB B C C </pre>	<pre> Banyak percobaan : 67 Puzzle berhasil diselesaikan: A A A B A C B B C Durasi : 12 ms Apakah anda ingin menyimpan solusi? (ya/tidak) : ya Berhasil menyimpan solusi. Berhasil menyimpan gambar. </pre>
---	---

	
4 4 4 DEFAULT AA B BB B CCCC C DD D DD	<p>Banyak percobaan : 95 Puzzle berhasil diselesaikan: A A D C D D D C D B C C B B B C Durasi : 23 ms Apakah anda ingin menyimpan solusi? (ya/tidak) : ya Berhasil menyimpan solusi. Berhasil menyimpan gambar.</p> 
5 5 6 DEFAULT AA A AA BB B CC CC C DD DD D E EEE FFF	<p>Banyak percobaan : 409 Puzzle berhasil diselesaikan: A A B E E A B B c E A A c c E D D D c c D D F F F Durasi : 22 ms Apakah anda ingin menyimpan solusi? (ya/tidak) : ya Berhasil menyimpan solusi. Berhasil menyimpan gambar.</p> 
4 6 5 DEFAULT A AAA BB B BB	<p>Banyak percobaan : 261 Puzzle berhasil diselesaikan: A C C C B B A A A C C B E E D D B B E E E D D D Durasi : 27 ms Apakah anda ingin menyimpan solusi? (ya/tidak) : ya Berhasil menyimpan solusi. Berhasil menyimpan gambar.</p>

<div>CC CCC DDD DD EE EEE</div>	
<div>3 7 6 DEFAULT A A AAA BB BB CC CC C DD D DD E F</div>	<div>Banyak percobaan : 214 Puzzle berhasil diselesaikan: A B B C E D D A B B C C D F A A A C C D D Durasi : 16 ms Apakah anda ingin menyimpan solusi? (ya/tidak) : ya Berhasil menyimpan solusi.</div> 
<div>6 6 8 DEFAULT A AA AA BBB B CCC CC DD D DD EE EEE FF F GGGG H HH HH</div>	<div>Banyak percobaan : 803 Puzzle berhasil diselesaikan: A B B B H C A A B H H C A A H H C C D D E E C F D E E E F F D D G G G G Durasi : 923 ms Apakah anda ingin menyimpan solusi? (ya/tidak) : ya Berhasil menyimpan solusi. Berhasil menyimpan gambar.</div> 

<pre> 5 6 6 DEFAULT AAA AA BB BB B C C CCC DDD D D D EE EE E FFF F </pre>	<pre> Banyak percobaan : 429 Puzzle berhasil diselesaikan: A A A E E E B B A A E E D B B F C C D B F F F C D D D D C C Durasi : 43 ms Apakah anda ingin menyimpan solusi? (ya/tidak) : ya Berhasil menyimpan solusi. Berhasil menyimpan gambar. </pre> 
<pre> 6 8 10 DEFAULT AA AA A BB B B C C C CCC DD D EE EEE FF F FF GGG G G HHH HH II I </pre>	<pre> Banyak percobaan : 1127 Puzzle berhasil diselesaikan: A A B B C D D J I A A B C D C J I I A B C C C J E E E F F G J J E E H H F G J J H H H F F G G G Durasi : 3594 ms Apakah anda ingin menyimpan solusi? (ya/tidak) : ya Berhasil menyimpan solusi. Berhasil menyimpan gambar. </pre> 

JJJJJ JJ																																														
5 9 11 DEFAULT A AA B BB CC DDD D D EEEE E FF FF F GG G G HH H H H II II I J JJ J KKK K	<div>Banyak percobaan : 1218 Puzzle berhasil diselesaikan: A B B C C E D D D A A B E E E E K D J J F F I I K K D H J J F F I I K G H H H F I G G G Durasi : 106814 ms Apakah anda ingin menyimpan solusi? (ya/tidak) : ya Berhasil menyimpan solusi. Berhasil menyimpan gambar.</div> <table><tr><td>A</td><td>B</td><td>B</td><td>C</td><td>C</td><td>E</td><td>D</td><td>D</td><td>D</td></tr><tr><td>A</td><td>A</td><td>B</td><td>E</td><td>E</td><td>E</td><td>E</td><td>K</td><td>D</td></tr><tr><td>J</td><td>J</td><td>F</td><td>F</td><td>I</td><td>I</td><td>K</td><td>K</td><td>D</td></tr><tr><td>H</td><td>J</td><td>J</td><td>F</td><td>F</td><td>I</td><td>I</td><td>K</td><td>G</td></tr><tr><td>H</td><td>H</td><td>H</td><td>H</td><td>F</td><td>I</td><td>G</td><td>G</td><td>G</td></tr></table>	A	B	B	C	C	E	D	D	D	A	A	B	E	E	E	E	K	D	J	J	F	F	I	I	K	K	D	H	J	J	F	F	I	I	K	G	H	H	H	H	F	I	G	G	G
A	B	B	C	C	E	D	D	D																																						
A	A	B	E	E	E	E	K	D																																						
J	J	F	F	I	I	K	K	D																																						
H	J	J	F	F	I	I	K	G																																						
H	H	H	H	F	I	G	G	G																																						
5 5 8 DEFAULT Y Y YYY Y X AA AAA I DD	<div>Banyak percobaan : 588 Puzzle berhasil diselesaikan: Y X Y A A Y Y Y A A I Y H S A D D H H H D D H N H Durasi : 26 ms Apakah anda ingin menyimpan solusi? (ya/tidak) : ya Berhasil menyimpan solusi. Berhasil menyimpan gambar.</div>																																													

DD S H HHH H H N	
---------------------------------	--

4. Tautan *Repository*
[DiyahSusan/Tucil1_13523080](https://github.com/DiyahSusan/Tucil1_13523080)

5. Lampiran

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 <i>Graphical User Interface</i> (GUI)		✓
6	Program dapat menyimpan solusi dalam bentuk file gambar	✓	
7	Program dapat menyelesaikan kasus konfigurasi <i>custom</i>		✓
8	Program dapat menyelesaikan kasus konfigurasi Piramida (3D)		✓
9	Program dibuat oleh saya sendiri	✓	