TUGAS KECIL 1
IF2211 STRATEGI ALGORITMA

Penyelesaian IQ Puzzler Pro dengan Algoritma Brute Force

Disusun oleh :

Jonathan Levi 13523132

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2023

Daftar Isi

BAGIAN I
Algoritma *Brute Force*

Algoritma *Brute Force* adalah algoritma yang mencari segala variasi atau permutasi sampai jawaban dari masalah tersebut ditemukan. Pada permainan IQ Puzzler Pro, digunakan teknik *backtracking* yang juga merupakan metode *brute force*.

Gambar 1.1. Permainan IQ Puzzler Pro (Sumber: https://www.amazon.com.au/Children-Suitable-Teenagers-Thinking-Concentration/dp/B0CYGQVB2C)

Cara kerja IQ Puzzler Pro dengan algoritma *brute force* adalah sebagai berikut:

1. Program membaca file teks yang diberikan oleh pengguna, yaitu jumlah baris, jumlah kolom, banyaknya piece, mode atau *config*, dan bentuk piece.
    - Jika config yang dipakai adalah *custom*, maka program membaca bentuk papan (*board*) juga.
2. Piece yang dibaca disimpan sebagai kumpulan koordinat.
3. Semua *piece* dirotasikan dan dicerminkan, tiap hasil orientasi tersebut disimpan sebagai kumpulan koordinat dan akan digunakan nanti.
4. Program memasang *piece* pertama pada papan yang masih kosong.
5. Program lanjut mencari posisi untuk mencoba memasang *piece* selanjutnya pada papan tanpa tumpang tindih (*overlapping*) dengan piece lainnya.

- Jika *piece* tidak dapat dipasang, maka dipakai orientasi (rotasi atau pencerminan) lain dari *piece* tersebut, program mencari lokasi sampai *piece* tersebut dapat dipasang pada papan.
    5.1. Jika semua orientasi terpakai dan *piece* tersebut tetap tidak dapat dipasang di mana pun tanpa tumpang tindih, cabut *piece* sebelumnya dan coba pasang dengan orientasi lain dari *piece* tersebut (*backtracking*).
- Jika *piece* dapat dipasang, ulangi tahap 5 hingga *piece* terakhir dapat dipasang dan tidak tersisa satu pun slot atau *cell* pada papan.

6. Program berhenti atau menyimpan hasil dalam teks dan/atau gambar, sesuai keinginan pengguna.

BAGIAN II
Source Code

Program ini ditulis menggunakan bahasa pemrograman Java, dengan menggunakan *library* berikut:

- java.io
- java.nio
- java.util
- java.awt
- javax.imageio

File Main.java

```java
import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.nio.file.Files;
import java.nio.file.NoSuchFileException;
import java.nio.file.Paths;
import java.util.*;

public class Main {

    static int N,M,P;
    static char[][] board;
    static List<Piece> pieces = new ArrayList<>();
    static boolean solutionFound = false;
    static char[][] finalBoard;
    static long iterCount = 0;

    static class Piece {
        char letter;
        List<int[]> baseShape;
        List<List<int[]>> orientations;

        public Piece(List<String> lines) {
            letter = lines.get(0).trim().charAt(0);
            baseShape = new ArrayList<>();
            for (int i=0; i<lines.size(); i++) {
                String line = lines.get(i);
                for (int j=0; j<line.length(); j++) {
                    if (line.charAt(j)==letter) baseShape.add(new int[]{i,j});
                }
            }
            acuanTopLeft(baseShape);
            orientations = generateOrientations(baseShape);
        }

        // Buat titik paling kiri atas = (0,0)
        private void acuanTopLeft(List<int[]> shape) {
            int minRow = 999999999;
            int minCol = 999999999;
            for (int[] p : shape) {
                if (p[0]<minRow) minRow = p[0];
                if (p[1]<minCol) minCol = p[1];
            }
            for (int[] p : shape) {
                p[0] -= minRow;
                p[1] -= minCol;
            }
        }
```

```java
    // Rotasi dan pencerminan
    private List<List<int[]>> generateOrientations(List<int[]> base) {
        Set<String> unique = new HashSet<>();
        List<List<int[]>> result = new ArrayList<>();
        for (int r = 0; r < 4; r++) {
            List<int[]> rotated = rotate(base, r);
            acuanTopLeft(rotated);
            String key = coordString(rotated);
            if (!unique.contains(key)) {
                unique.add(key);
                result.add(copyShape(rotated));
            }
            // Tambahkan orientasi mirror (horizontal)
            List<int[]> mirrored = mirror(rotated);
            acuanTopLeft(mirrored);
            key = coordString(mirrored);
            if (!unique.contains(key)) {
                unique.add(key);
                result.add(copyShape(mirrored));
            }
        }
        return result;
    }

    // Rotasi searah jarum jam sebanyak n kali
    private List<int[]> rotate(List<int[]> shape, int n) {
        List<int[]> newShape = copyShape(shape);
        for (int t=0; t<n; t++) {
            for (int[] p : newShape) {
                int temp = p[0];
                p[0] = p[1];
                p[1] = -temp;
            }
            acuanTopLeft(newShape);
        }
        return newShape;
    }

    // Pencerminan vertikal (terhadap sumbu horizontal)
    private List<int[]> mirror(List<int[]> shape) {
        List<int[]> res = new ArrayList<>();
        for (int[] p : shape) {
            res.add(new int[]{p[0], -p[1]});
        }
        return res;
    }

    // Konversi koordinat dalam format string "(x,y)"
    private String coordString(List<int[]> shape) {
        List<String> pts = new ArrayList<>();
        for (int[] p : shape) {
            pts.add(p[0] + "," + p[1]);
        }
        Collections.sort(pts);
        return String.join(";", pts);
    }

    // Copy list titik
    private List<int[]> copyShape(List<int[]> shape) {
        List<int[]> copy = new ArrayList<>();
        for (int[] p : shape) {
            copy.add(new int[]{p[0], p[1]});
        }
        return copy;
    }
}
```

```java
// pencarian solusi dengan brute force/backtracking
static void solve(int pieceIdx) {
    //jika semua piece berhasil dipasang
    if (pieceIdx == pieces.size()) {
        solutionFound = true;
        finalBoard = new char[N][M];
        for (int i=0; i<N; i++) {
            for (int j=0; j<M; j++) {
                finalBoard[i][j] = board[i][j];
            }
        }
        return;
    }
    Piece piece = pieces.get(pieceIdx);
    for (List<int[]> orientation : piece.orientations) {
        for (int i=0; i<N; i++) {
            for (int j=0; j<M; j++) {
                iterCount++;
                if (canPlace(orientation, i, j)) {
                    place(orientation, i, j, piece.letter);
                    solve(pieceIdx + 1);
                    if (solutionFound) return;
                    place(orientation, i, j, ',');
                }
            }
        }
    }
}

// Menempatkan/menghapus piece pada board
static void place(List<int[]> orientation, int row, int col, char ch) {
    for (int[] p : orientation) {
        int r = row + p[0];
        int c = col + p[1];
        board[r][c] = ch;
    }
}
static boolean canPlace(List<int[]> orientation, int row, int col) {
    for (int[] p : orientation) {
        int r = row + p[0];
        int c = col + p[1];
        if (r<0||r>=N||c<0||c>=M||board[r][c]!=',') return false;
    }
    return true;
}
```

```java
// Print board dengan warna
static void printColored(char[][] boardOutput) {
    Map<Character, String> colorMap = new HashMap<>();
    String[] colors = {
        "\u001B[38;5;196m", // merah 1
        "\u001B[38;5;46m",  // hijau 1
        "\u001B[38;5;21m",  // biru 1
        "\u001B[38;5;226m", // kuning 1
        "\u001B[38;5;129m", // ungu 1
        "\u001B[38;5;51m",  // biru 2
        "\u001B[38;5;208m", // orange 1
        "\u001B[38;5;118m", // hijau 2
        "\u001B[38;5;27m",  // biru 3
        "\u001B[38;5;220m", // kuning 2
        "\u001B[38;5;99m",  // magenta
        "\u001B[38;5;201m", // pink
        "\u001B[38;5;33m",  // biru 4
        "\u001B[38;5;160m", // merah 2
        "\u001B[38;5;202m", // orange 2
        "\u001B[38;5;82m",  // hijau 3
        "\u001B[38;5;34m",  // hijau 4
        "\u001B[38;5;94m",  // coklat
        "\u001B[38;5;135m", // ungu 2
        "\u001B[38;5;165m", // ungu 3
        "\u001B[38;5;75m",  // biru 5
        "\u001B[38;5;123m", // biru 6
        "\u001B[38;5;161m", // ungu 4
        "\u001B[38;5;179m", // cream
        "\u001B[38;5;141m", // ungu 5
        "\u001B[38;5;76m"   // hijau 5
    };
    int idx = 0;
    for (Piece p: pieces) {
        if (!colorMap.containsKey(p.letter)) {
            colorMap.put(p.letter, colors[idx]);
            idx++;
        }
    }
    String resetColor = "\u001B[0m";
    for (int i=0; i<boardOutput.length; i++) {
        for (int j=0; j<boardOutput[i].length; j++) {
            char ch = boardOutput[i][j];
            if ((ch!=',') && (colorMap.containsKey(ch))) System.out.print(colorMap.get(ch)+ch+resetColor);
            else System.out.print(ch);
        }
        System.out.println();
    }
}

// Save solusi ke file
static void saveSolution(String filename, char[][] solutionBoard/*, long searchTime, long iterations*/) {
    try (PrintWriter writer = new PrintWriter(new File(filename))) {
        for (int i=0; i<solutionBoard.length; i++) {
            writer.println(new String(solutionBoard[i]));
        }
        writer.println();
        // writer.println("Waktu pencarian: " + searchTime + " ms");
        // writer.println("Banyak kasus yang ditinjau: " + iterations);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

```java
static void solveMain(String fileName) {
    Scanner sc = new Scanner(System.in);
    try {
        // Membaca seluruh baris dari file
        List<String> allLines = Files.readAllLines(Paths.get(fileName));
        if (allLines.size()<2) {
            System.out.println("File tidak valid.");
            sc.close();
            return;
        }

        // Baris pertama: N M P
        String[] parts = allLines.get(0).trim().split("\\s+");
        if (parts.length < 3) {
            System.out.println("Baris pertama harus berisi tiga angka: N M P");
            sc.close();
            return;
        }
        N = Integer.parseInt(parts[0]);
        M = Integer.parseInt(parts[1]);
        P = Integer.parseInt(parts[2]);
        String config = allLines.get(1).trim().toLowerCase();

        int currentIdx = 2;
        int pieceCellTotal = 0;
        if (config.equals("custom")) {
            board = new char[N][M];
            for (int i=0; i<N; i++) {
                String line = allLines.get(currentIdx);
                currentIdx++;
                for (int j=0; j<M; j++) {
                    char cell = line.charAt(j);
                    cell = Character.toLowerCase(cell);
                    if (cell=='x') {
                        board[i][j] = ',';
                        pieceCellTotal++;
                    } else if (cell=='.') board[i][j] = ' ';
                    else {
                        System.out.println("Karakter pada file teks tidak valid.");
                        sc.close();
                        return;
                    }
                }
            }
        } else if (config.equals("default")) {
            board = new char[N][M];
            for (int i=0; i<N; i++) {
               for (int j=0; j<M; j++) {
                   board[i][j] = ',';
               }
            }
        } else {
            System.out.println("Konfigurasi pada file teks tidak valid.");
            sc.close();
            return;
        }
```

```
        List<String> remainingLines = new ArrayList<>();
        for (int i=currentIdx; i<allLines.size(); i++) {
            String line = allLines.get(i);
            if (line.trim().isEmpty()) continue;
            remainingLines.add(line);
        }

        // Pengelompokan baris untuk tiap piece.
        int idx = 0;
        for (int i=0; i<P; i++) {
            if (idx >= remainingLines.size()) break;
            List<String> pieceLines = new ArrayList<>();
            String firstLine = remainingLines.get(idx);
            pieceLines.add(firstLine);
            char pieceChar = firstLine.trim().charAt(0);
            idx++;
            while (idx < remainingLines.size()) {
                String nextLine = remainingLines.get(idx);
                if ((!nextLine.trim().isEmpty()) && (nextLine.trim().charAt(0)==pieceChar)) {
                    pieceLines.add(nextLine);
                    idx++;
                } else break;
            }
            pieces.add(new Piece(pieceLines));
        }

        // cek apakah dipastikan tidak muat atau dipastikan ada kelebihan slot/cell
        int totalCells = 0;
        for (Piece p : pieces) {
            totalCells += p.baseShape.size();
        }
        if (config.equals("custom")) {
            if (totalCells!=pieceCellTotal) {
                System.out.println("Solusi tidak ditemukan.");
                sc.close();
                return;
            }
        } else if (config.equals("default")) {
            if (totalCells != N*M) {
                System.out.println("Solusi tidak ditemukan.");
                sc.close();
                return;
            }
        }

        long startTime = System.currentTimeMillis();
        solve(0);
        long endTime = System.currentTimeMillis();
        long searchTime = endTime - startTime;

        if (solutionFound) {
            System.out.println("\nSolusi ditemukan.");
            printColored(finalBoard);
            System.out.println("\nWaktu pencarian: " + searchTime + " ms");
            System.out.println("Banyak kasus yang ditinjau: " + iterCount);
        } else System.out.println("Solusi tidak ditemukan.");
```

```
            String answer;
            do {
                System.out.print("\nApakah anda ingin menyimpan solusi? (Y/n): ");
                answer = sc.nextLine().trim().toLowerCase();
                if (answer.equals("y")) {
                    System.out.print("Masukkan nama file teks output (tanpa \".txt\"): ");
                    String outFileName = "../test/"+sc.nextLine().trim()+".txt";
                    saveSolution(outFileName, finalBoard/*, searchTime, iterCount*/);
                    System.out.println("Solusi berhasil disimpan ke " + outFileName);
                }
            } while ((!answer.equals("n"))&&(!answer.equals("y")));

            String answerImg;
            do {
                System.out.print("\nApakah anda ingin menyimpan solusi sebagai gambar? (Y/n): ");
                answerImg = sc.nextLine().trim().toLowerCase();
                if (answerImg.equals("y")) {
                    System.out.print("Masukkan nama file gambar output (tanpa \".jpg\"): ");
                    String outFileName = "../test/"+sc.nextLine().trim()+".jpg";
                    ImageSaver.saveAsImage(outFileName, finalBoard, 100);
                }
            } while ((!answerImg.equals("n"))&&(!answerImg.equals("y")));
            sc.close();

        } catch (FileNotFoundException e) {
            System.out.println("File tidak ditemukan.");
        } catch (NoSuchFileException e) {
            System.out.println("File tidak ditemukan.");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Masukkan nama file teks (tanpa \".txt\"): ");
        String fileName = "../test/"+sc.nextLine().trim()+".txt";
        solveMain(fileName);
        sc.close();
    }
}
```

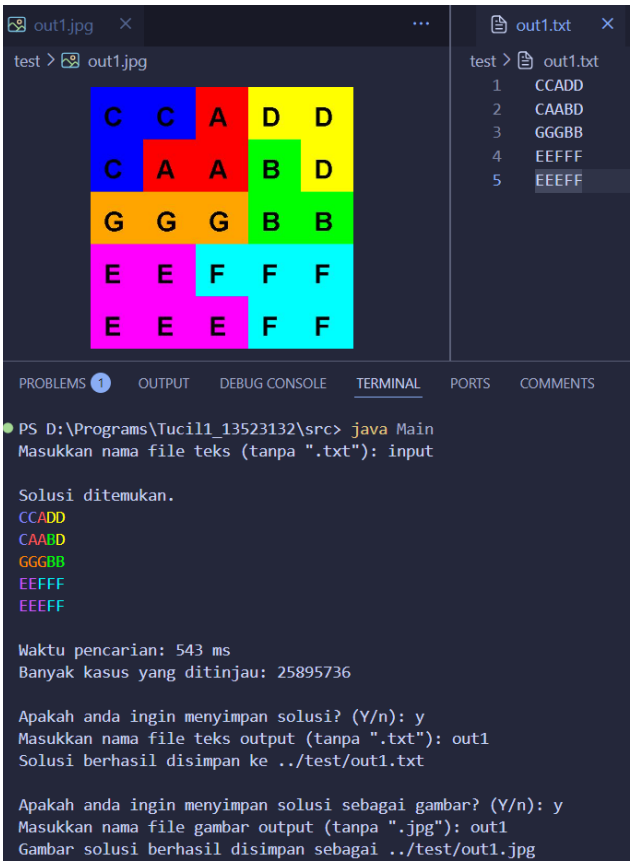
File SaveImg.java:

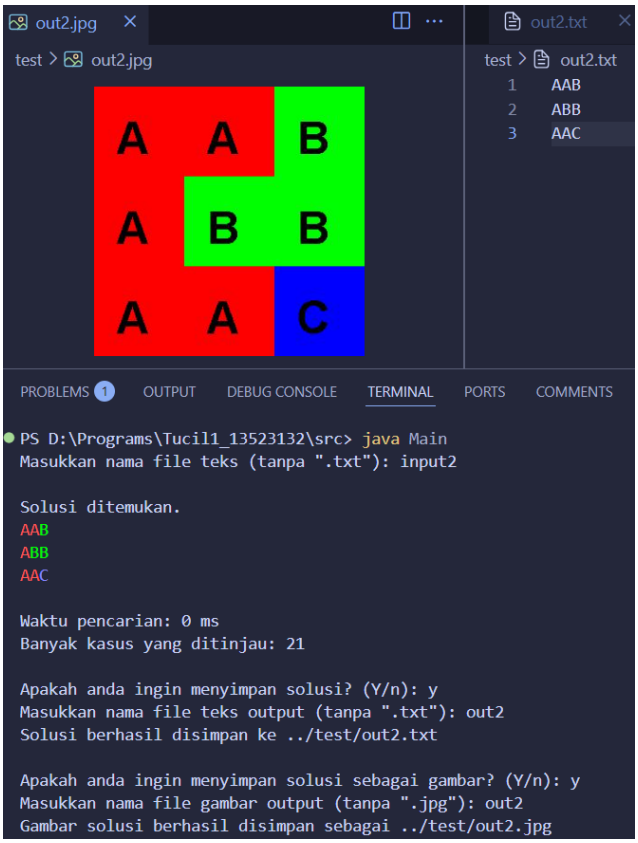
```java
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics2D;
import java.awt.image.BufferedImage;
import java.awt.BasicStroke;
import javax.imageio.ImageIO;
import java.io.File;
import java.util.HashMap;
import java.util.Map;

class ImageSaver {
    public static void saveAsImage(String filename, char[][] board, int cellPixel) {
        int rows = board.length;
        int cols = board[0].length;
        int width = cols * cellPixel;
        int height = rows * cellPixel;

        BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
        Graphics2D g2d = image.createGraphics();

        // bg hitam
        g2d.setColor(Color.BLACK);
        g2d.fillRect(0, 0, width, height);

        Map<Character, Color> colorMap = new HashMap<>();
        colorMap.put('A', new Color(255, 0, 0));
        colorMap.put('B', new Color(0, 255, 0));
        colorMap.put('C', new Color(0, 0, 255));
        colorMap.put('D', new Color(255, 255, 0));
        colorMap.put('E', new Color(255, 0, 255));
        colorMap.put('F', new Color(0, 255, 255));
        colorMap.put('G', new Color(255, 165, 0));
        colorMap.put('H', new Color(152, 251, 152));
        colorMap.put('I', new Color(0, 0, 205));
        colorMap.put('J', new Color(255, 215, 0));
        colorMap.put('K', new Color(139, 0, 139));
        colorMap.put('L', new Color(255, 105, 180));
        colorMap.put('M', new Color(0, 128, 128));
        colorMap.put('N', new Color(128, 0, 0));
        colorMap.put('O', new Color(255, 69, 0));
        colorMap.put('P', new Color(0, 128, 0));
        colorMap.put('Q', new Color(0, 139, 139));
        colorMap.put('R', new Color(139, 69, 19));
        colorMap.put('S', new Color(138, 43, 226));
        colorMap.put('T', new Color(128, 0, 128));
        colorMap.put('U', new Color(135, 206, 235));
        colorMap.put('V', new Color(176, 196, 222));
        colorMap.put('W', new Color(199, 21, 133));
        colorMap.put('X', new Color(255, 131, 0));
        colorMap.put('Y', new Color(221, 160, 221));
        colorMap.put('Z', new Color(0, 191, 255));
```

```
        // Gambar
        for (int i=0; i<rows; i++) {
            for (int j=0; j<cols; j++) {
                char ch = board[i][j];
                int x = j * cellPixel;
                int y = i * cellPixel;
                if (ch == ' ') {
                    g2d.setColor(Color.GRAY);
                    g2d.fillRect(x, y, cellPixel, cellPixel);
                    g2d.setColor(Color.BLACK);
                    g2d.setStroke(new BasicStroke(2.0f));
                    g2d.drawLine(x, y, x + cellPixel, y + cellPixel);
                    g2d.drawLine(x + cellPixel, y, x, y + cellPixel);
                    g2d.setColor(Color.BLACK);
                    g2d.drawRect(x, y, cellPixel, cellPixel);
                    g2d.setStroke(new BasicStroke(1.0f));
                } else if (ch != ',') {
                    Color c = colorMap.getOrDefault(ch, Color.DARK_GRAY);
                    g2d.setColor(c);
                    g2d.fillRect(x, y, cellPixel, cellPixel);
                }
            }
        }

        // huruf di tiap sel
        g2d.setFont(new Font("Arial", Font.BOLD, cellPixel / 2));
        for (int i=0; i<rows; i++) {
            for (int j=0; j<cols; j++) {
                char ch = board[i][j];
                if (ch != ',') {
                    int x = j * cellPixel;
                    int y = i * cellPixel;
                    g2d.setColor(Color.BLACK);
                    g2d.drawString(Character.toString(ch), x + cellPixel / 4, y + (3 * cellPixel) / 4);
                }
            }
        }

        g2d.dispose();

        try {
            ImageIO.write(image, "jpg", new File(filename));
            System.out.println("Gambar solusi berhasil disimpan sebagai " + filename);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

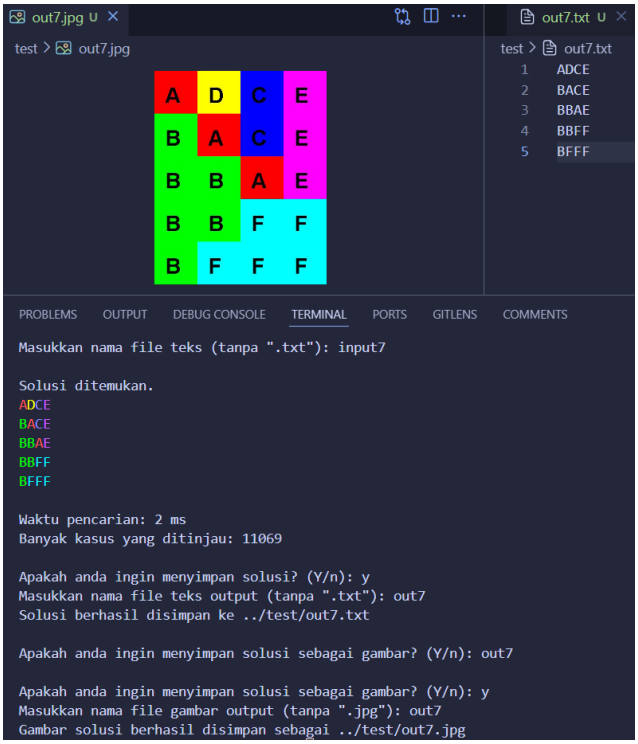
BAGIAN III

Test Case

Test Case	Input	Output
1	5 5 7 DEFAULT A AA B BB C CC D DD E EE EE FF FF F GGG	

<table>
<tr>
<td>2</td>
<td>3 3 3
DEFAULT
AA
A
AA
B
BB
C</td>
<td> 
</td>
</tr>
<tr>
<td>3</td>
<td>4 5 6
DEFAULT
AA
A
AA
BB
B
BB
C
DD
E
EE
E
F
FF</td>
<td> 
</td>
</tr>
</table>

4	6 8 9 DEFAULT AAA A AA A AAAA BB C D DDDD EE FF F FF GGG GGG GG G H HHH H II III III	
5	6 10 12 DEFAULT JJJ JJJJ J QQ QQ S TTT TTT OOO O PPPP PPP PP M M MMMM	

	MM NN AAA A KK K KKK LL LLLLL U	
6 (Diagonal)	5 4 6 DEFAULT A A A BBBB BB C C D EEE F FF FF	

7 (Lebih banyak diagonal)	4 6 9 DEFAULT A A A Q Q Q Q Q D D DDD CC B B P P P R T T Z	
8 (*Custom*)	5 7 5 CUSTOM ...X... .XXXXX. XXXXXXX .XXXXX. ...X... A AAA BB BBB CCCC C D EEE E	
9 (*Custom*)	7 10 16 CUSTOM	

	XXXXXXXXXX XXXXX.XXXX XXXX...XXX XXX.....XX XXXX...XXX XXXXX.XXXX XXXXXXXXXX AA AA A A B B B C CC EEE E E D GGG F FF Z Z Y YY Q PPP P P P P SSS SS UU JJ J L L LL LL WWW W	

BAGIAN IV
Lampiran

Pranala repositori: https://github.com/RealNath/Tucil1_13523132

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	✓	