LAPORAN TUGAS KECIL 1 STRATEGI ALGORITMA

Karya tulis ini disusun untuk memenuhi tugas

Mata Kuliah IF2211 Strategi Algoritma

Semester II Tahun Akademik 2024/2025

Disusun oleh:

Anella Utari Gunadi

13523078

SEKOLAH TINGGI ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

FEBRUARI 2025

Daftar Isi

Deskripsi Masalah

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.

Tugas kecil 1 ini memerintahkan untuk menemukan cukup satu solusi dari permainan IQ Puzzler Pro dengan menggunakan algoritma Brute Force atau menampilkan bahwa solusi tidak ditemukan jika tidak ada solusi yang mungkin dari puzzle.

Algoritma Brute Force

Pada program IQPuzzlerPro ini, algoritma brute force yang digunakan adalah dengan meletakkan satu per satu puzzle ke setiap blok papan beserta rotasi dan flip serta melakukan backtracking pada puzzle sebelumnya jika puzzle tidak dapat diletakkan di semua blok papan dengan berbagai rotasi dan flip. Mulanya, program utama akan memanggil fungsi solve dengan indeks 0, yaitu mengambil puzzle paling pertama untuk dicoba diletakkan di papan. Pada fungsi solve, indeks akan bertambah jika puzzle dapat diletakkan dan akan me-*return false* jika puzzle pertama (indeks 0) tidak dapat diletakkan di semua kemungkinan.

```
if (!solve(0, allpuzzles, board, N, M, colorMap, outputtxt)) {
                String nosolution = "Tidak ada solusi.";
                System.out.println(nosolution);
                outputtxt.append(nosolution).append("\n");
            }
```

Pada fungsi solve, algoritma brute force yang digunakan adalah dengan mencoba meletakkan puzzle tersebut di seluruh posisi pada papan dimulai dari posisi paling ujung kiri atas lalu diiterasi hingga ujung kanan bawah. Di setiap iterasinya, puzzle menggunakan fungsi canPlace yang berfungsi untuk mengecek apakah papan dapat ditempatkan oleh puzzle yang sedang dicek. Misalkan puzzle A akan ditempatkan pada blok papan[0][0], jika puzzle A tidak dapat ditempatkan,

maka program akan melakukan iterasi untuk melakukan rotasi dengan memanggil fungsi rotate. Iterasi yang dilakukan untuk merotasi puzzle adalah dari 0 hingga 3 kali. Pada rotasi ke-0, puzzle tetap pada bentuk awal. Pada rotasi ke-1, puzzle dirotasi sebesar 90 derajat searah jarum jam, dan seterusnya hingga rotasi ke-3. Pada iterasi rotasi tersebut, jika puzzle A yang telah dirotasi tetap tidak dapat dilakukan, puzzle A akan di-*flip* dengan fungsi flipHorizontal dan dicoba ditempatkan kembali. Jika semua rotasi dan flip telah dilakukan tetapi puzzle A tetap tidak bisa ditempatkan, program akan melanjutkan iterasi ke blok papan[0][1], dan seterusnya.

```
       for (int i=0;i<N;i++) {                          # panjang blok papan
             for (int j=0;j<M;j++) {                    # lebar blok papan
                  for (int k=0;k<4;k++) {               # coba semua rotasi
                      char[][] rotated = rotate(puzzle,k);
                      char[][] flippedH = flipHorizontal(rotated);
                      if (canPlace(rotated, board, i, j, N, M)) {
                          placePuzzle(rotated, board, i, j);
                          if (nextPuzzle(rotated, board, i, j, N, M, index, allpuzzles, colorMap,
outputtxt)) return true;
       # backtrack jika puzzle berikutnya tidak dapat ditempatkan di semua posisi
                               removePuzzle(rotated, board, i, j);
                           }


                           if (canPlace(flippedH, board, i, j, N, M)) {      # untuk flip puzzle
                               placePuzzle(flippedH, board, i, j);
                                 if (nextPuzzle(flippedH, board, i, j, N, M, index, allpuzzles,
colorMap, outputtxt)) return true;
                               removePuzzle(flippedH, board, i, j);
                           }
                       }
                  }
             }
             return false;
```

Setelah puzzle A dapat ditempatkan pada papan, program akan melanjutkan ke puzzle berikutnya dengan menambahkan indeks pada fungsi solve. Misalkan puzzle berikutnya adalah puzzle B, lakukan hal yang sama seperti sebelumnya yaitu dengan meletakkan puzzle B ke semua blok papan dan melakukan iterasi rotasi dan flip di setiap tempat. Jika semua tempat pada papan tidak bisa ditempatkan oleh puzzle B, program akan melakukan backtrack dengan menghapus puzzle sebelumnya, yaitu puzzle A, pada papan dengan memanggil fungsi removePuzzle, kemudian melanjutkan iterasi pada puzzle A setelah tempat yang sebelumnya ditempatkan. Misalkan sebelumnya puzzle A ditempatkan di papan[0][3], maka puzzle A akan dihapus terlebih dahulu pada papan lalu dilanjutkan iterasi untuk blok papan selanjutnya, misalnya papan[0][4], dan seterusnya.

```
if (solve(index + 1, allpuzzles, board, N, M, colorMap, outputtxt)) {
            return true;
        }
        removePuzzle(puzzle, board, i, j);              # backtrack
        return false;
```

Hasil akhir dari program ini akan menghasilkan dua kemungkinan, yaitu memiliki solusi atau tidak memiliki solusi. Program akan mengeluarkan output tidak memiliki solusi jika seluruh puzzle telah ditempatkan pada papan tetapi masih terdapat blok papan yang kosong atau jika blok papan telah terisi oleh puzzle tetapi masih terdapat puzzle yang belum terpakai atau jika ada puzzle yang tidak dapat ditempatkan dimana-mana walaupun sudah melakukan backtracking ke semua puzzle sebelumnya. Program akan mengeluarkan output memiliki solusi beserta hasil akhir dari papan IQPuzzlerPro jika seluruh puzzle telah berhasil ditempatkan di papan dan setiap blok papan telah terisi oleh puzzle-puzzle tersebut.

Source Program

Berikut adalah source program untuk IQPuzzlerPro dalam bahasa pemrograman Java.

```
import java.io.*;
import java.util.*;

public class IQPuzzlerPro {
    public static final String RESET = "\033[38;5;0m";
    public static final String[] ANSI_COLORS = {
        "\033[38;5;196m", "\033[38;5;226m", "\033[38;5;46m",
        "\033[38;5;208m", "\033[38;5;30m", "\033[38;5;99m",
        "\033[38;5;51m", "\033[38;5;18m", "\033[38;5;81m",
```

```
        "\033[38;5;90m","\033[38;5;21m", "\033[38;5;201m",
        "\033[38;5;206m","\033[38;5;130m","\033[38;5;218m",
        "\033[38;5;88m", "\033[38;5;180m", "\033[38;5;100m",
        "\033[38;5;220m", "\033[38;5;22m", "\033[38;5;87m",
        "\033[38;5;147m", "\033[38;5;15m", "\033[38;5;89m",
        "\033[38;5;240m", "\033[38;5;252m"
    };
    private static int totalAttempts = 0;
    public static void main(String[] args) throws FileNotFoundException, IOException{
        Map<Character, String> colorMap = new HashMap<>();
        for (int i = 0; i < 26; i++) {
            colorMap.put((char) ('A' + i), ANSI_COLORS[i % ANSI_COLORS.length]);
        }

        Scanner scanner = new Scanner(System.in);
        System.out.print("Masukkan nama file txt: ");
        String file = scanner.nextLine();

        StringBuilder outputtxt = new StringBuilder();

        try (BufferedReader input = new BufferedReader(new FileReader(file))){
            String[] baris1 = input.readLine().split(" ");
            if (baris1.length < 3) {
                System.out.println("Error: Format file tidak sesuai.");
                return;
            }

            int N,M,P;
            try{
                N = Integer.parseInt(baris1[0]);
                M = Integer.parseInt(baris1[1]);
                P = Integer.parseInt(baris1[2]);
            } catch (NumberFormatException e){
                System.out.println("N, M, atau P bukan angka valid. Silakan coba
lagi.");
                return;
            }

            char[][] board = new char[N][M];
            for (int i = 0; i < N; i++) {
                Arrays.fill(board[i], '.'); // Isi semua sel dengan spasi (kosong)
            }

            String S = input.readLine();
            S = S.trim();
            System.out.println();

            if (!S.equals("DEFAULT")){
                System.out.println("Mohon maaf, tidak tersedia jenis kasus " + S +
".");
                return;
            }

            List<char[][]> allpuzzles = new ArrayList<>();
            List<String> currentPuzzle = new ArrayList<>();
```

```
            Character lastFirstChar = null;

            while (input.ready()) {
                String line = input.readLine();
                if (line.trim().isEmpty()) continue;

                int firstCharIndex = findFirstChar(line);
                if (firstCharIndex == -1) continue;

                char firstChar = line.charAt(firstCharIndex);

                if (lastFirstChar == null) {
                    lastFirstChar = firstChar;
                }

                if (firstChar == lastFirstChar) {
                    currentPuzzle.add(line);
                } else {
                    if (!currentPuzzle.isEmpty()) {
                        allpuzzles.add(convertStringtoChar(currentPuzzle));
                        currentPuzzle.clear();
                    }
                    currentPuzzle.add(line);
                    lastFirstChar = firstChar;
                }
            }

            if (!currentPuzzle.isEmpty()) {
                allpuzzles.add(convertStringtoChar(currentPuzzle));
            }

            input.close();

            if (allpuzzles.size() != P){
                System.out.println("Banyaknya puzzle tidak sama dengan nilai P.
Silakan coba lagi.");
                return;
            }

            if (allpuzzles.isEmpty()) {
                System.out.println("Error: Tidak ada puzzle yang terbaca dari file.");
                return;
            }

            long startTime = System.nanoTime();

            System.out.println("Mencari solusi...");

            if (!solve(0, allpuzzles, board, N, M, colorMap, outputtxt)) {
                String nosolution = "Tidak ada solusi.";
                System.out.println(nosolution);
                outputtxt.append(nosolution).append("\n");
            }
```

```
            long endTime = System.nanoTime();
            long totalWaktu = endTime - startTime;
            String waktu = "Waktu pencarian: " + (totalWaktu / 1_000_000.0) + " ms";
            System.out.println(waktu);
            outputtxt.append(waktu).append("\n");

            System.out.println();
            outputtxt.append("\n");
            String totalKasus = "Banyak kasus yang ditinjau: " + totalAttempts;
            System.out.println(totalKasus);
            outputtxt.append(totalKasus).append("\n");

            System.out.println();
            System.out.println("Apakah anda ingin menyimpan solusi? (ya/tidak)");
            String simpan = scanner.nextLine().trim().toLowerCase();
            System.out.println();

            while (!simpan.equals("ya") && !simpan.equals("tidak")){
                System.out.println("Input tidak valid. Silakan masukan kembali.");
                System.out.println("Apakah anda ingin menyimpan solusi? (ya/tidak)");
                simpan = scanner.nextLine().trim().toLowerCase();
            }
            System.out.println();

            if (simpan.equals("ya")){
                System.out.print("Masukkan nama file output txt: ");
                String outputFile = scanner.nextLine();

                String outputDir = "../test";
                File outputFolder = new File(outputDir);

                if (!outputFolder.exists()) {
                    outputFolder.mkdir();
                }

                String outputFilePath = outputDir + "/" + outputFile;

                try (BufferedWriter outputWriter = new BufferedWriter(new
FileWriter(outputFilePath))) {
                    outputWriter.write(outputtxt.toString());
                    System.out.println("Output berhasil disimpan ke: " +
outputFilePath);
                    System.out.println();
                } catch (IOException e) {
                    System.out.println("Error menyimpan file: " + e.getMessage());
                    System.out.println();
                }
            }
            scanner.close();
            System.out.println("IQPuzzlerPro telah selesai. Terima kasih telah
bermain.");
        }
    }

    private static int findFirstChar(String line) {
```

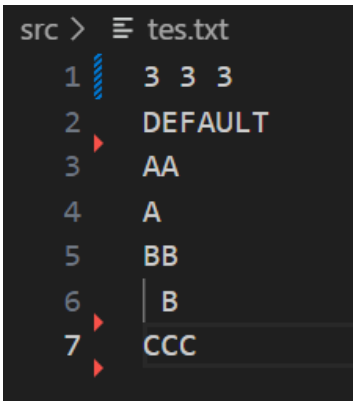
```
        for (int i = 0; i < line.length(); i++) {
            if (line.charAt(i) != ' ') return i;
        }
        return -1;
    }

    private static char[][] convertStringtoChar(List<String> puzzleStrings) {
        int rows = puzzleStrings.size();
        int cols = puzzleStrings.stream().mapToInt(String::length).max().orElse(0);
        char[][] puzzle = new char[rows][cols];

        for (int i = 0; i < rows; i++) {
            String row = puzzleStrings.get(i);
            for (int j = 0; j < cols; j++) {
                puzzle[i][j] = (j < row.length()) ? row.charAt(j) : ' ';
            }
        }
        return puzzle;
    }

    private static boolean solve(int index, List<char[][]> allpuzzles, char[][] board,
int N, int M, Map<Character, String> colorMap, StringBuilder outputtxt) {
        if (index == allpuzzles.size()) {
            if (hasEmptySpace(board,N,M)){
                return false;
            }

            printBoard(board, colorMap, outputtxt);
            return true;
        }

        char[][] puzzle = allpuzzles.get(index);

        for (int i=0;i<N;i++) {
            for (int j=0;j<M;j++) {
                for (int k=0;k<4;k++) {
                    char[][] rotated = rotate(puzzle,k);
                    char[][] flippedH = flipHorizontal(rotated);

                    if (canPlace(rotated, board, i, j, N, M)) {
                        placePuzzle(rotated, board, i, j);
                        if (nextPuzzle(rotated, board, i, j, N, M, index, allpuzzles,
colorMap, outputtxt)) return true;
                        removePuzzle(rotated, board, i, j);
                    }

                    if (canPlace(flippedH, board, i, j, N, M)) {
                        placePuzzle(flippedH, board, i, j);
                        if (nextPuzzle(flippedH, board, i, j, N, M, index, allpuzzles,
colorMap, outputtxt)) return true;
                        removePuzzle(flippedH, board, i, j);
                    }
                }
            }
        }
```

```
        return false;
    }

    private static boolean hasEmptySpace(char[][] board, int N, int M){
        for (int i=0;i<N;i++){
            for (int j=0;j<M;j++){
                if (board[i][j] == '.'){
                    return true;
                }
            }
        }
        return false;
    }

    private static boolean nextPuzzle(char[][] puzzle, char[][] board, int i, int j,
int N, int M, int index, List<char[][]> allpuzzles, Map<Character, String> colorMap,
StringBuilder outputtxt){
        if (solve(index + 1, allpuzzles, board, N, M, colorMap, outputtxt)) {
            return true;
        }
        removePuzzle(puzzle, board, i, j);
        return false;
    }

    private static boolean canPlace(char[][] puzzle, char[][] board, int x, int y, int
N, int M){
        totalAttempts++;
        for (int i=0;i<puzzle.length;i++){
            for (int j=0;j<puzzle[i].length;j++){
                if (puzzle[i][j] != ' ' && (x+i >= N || y+j >= M || board[x+i][y+j] !=
'.')){
                    return false;
                }
            }
        }
        return true;
    }

    private static void placePuzzle (char[][] puzzle, char[][] board, int x, int y){
        for (int i=0;i<puzzle.length;i++){
            for (int j=0;j<puzzle[i].length;j++){
                if (puzzle[i][j] != ' '){
                    board[x+i][y+j] = puzzle[i][j];
                }
            }
        }
    }

    private static void removePuzzle (char[][] puzzle, char[][] board, int row, int
col){
        for (int i=0;i<puzzle.length;i++){
            for (int j=0;j<puzzle[i].length;j++){
                if (puzzle[i][j] != ' '){
                    board[row+i][col+j] = '.';
                }
```

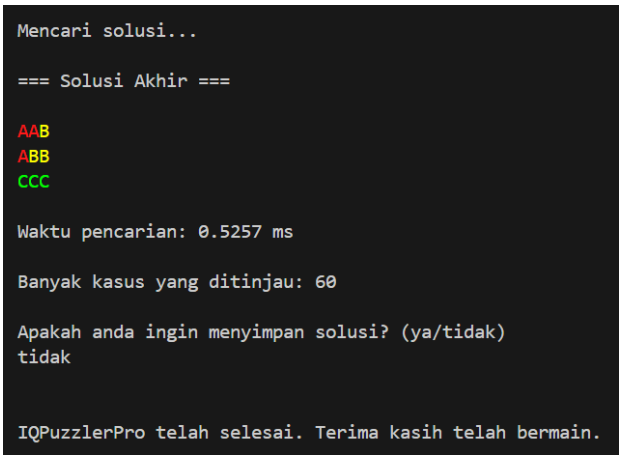
```
            }
        }
    }

    private static char[][] rotate(char[][] puzzle, int putar){
        char[][] rotated = puzzle;
        for (int i=0;i<putar;i++){
            int row = rotated.length;
            int col = rotated[0].length;
            char[][] newRotated = new char[col][row];

            for (int j=0;j<row;j++){
                for (int k=0;k<col;k++){
                    newRotated[k][row-j-1] = rotated[j][k];
                }
            }
            rotated = newRotated;
        }
        return rotated;
    }

    private static char[][] flipHorizontal(char[][] puzzle){
        int row = puzzle.length;
        char[][] flipped = new char[row][];
        for (int i=0;i<row;i++){
            flipped[i] = puzzle[row-i-1];
        }
        return flipped;
    }

    private static void printBoard (char[][] board, Map<Character, String> colorMap,
StringBuilder outputtxt){
        String printsolusi = "\n=== Solusi Akhir ===";
        System.out.println(printsolusi);
        System.out.println();
        outputtxt.append(printsolusi).append("\n");
        outputtxt.append("\n");

        for (int i = 0; i < board.length; i++){
            StringBuilder solusi = new StringBuilder();
            StringBuilder solusitxt = new StringBuilder();

            for (int j = 0; j < board[i].length; j++){
                char c = board[i][j];
                String color = colorMap.get(c);
                solusi.append(color).append(c).append("\u001B[0m");
                solusitxt.append(c);
            }
            System.out.println(solusi.toString());
            outputtxt.append(solusitxt.toString()).append("\n");
        }
        System.out.println();
        outputtxt.append("\n");
    }
}
```

Test Case

Test Case 1



Gambar 1. Input test case 1



Gambar 2. Output test case 1

Test Case 2

src > tes.txt

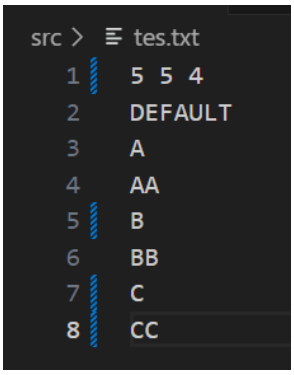
```
5 5 7
DEFAULT
A
AA
B
BB
C
CC
D
DD
EE
EE
E
FF
FF
F
GGG
```

Gambar 3. Input test case 2

```
Masukkan nama file txt: tes.txt

Mencari solusi...

=== Solusi Akhir ===

ABBCC
AABCD
EEEDD
EEFFF
GGGFF

Waktu pencarian: 39.8091 ms

Banyak kasus yang ditinjau: 250298

Apakah anda ingin menyimpan solusi? (ya/tidak)
ya


Masukkan nama file output txt: output.txt
Output berhasil disimpan ke: ../test/output.txt

IQPuzzlerPro telah selesai. Terima kasih telah bermain.
```

Gambar 4. Output test case 2

Test Case 3

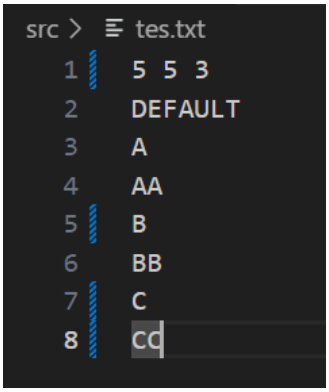


Gambar 5. Input test case 3

```
Masukkan nama file txt: tes.txt

Banyaknya puzzle tidak sama dengan nilai P. Silakan coba lagi.
```

Gambar 6. Output test case 3

Test Case 4

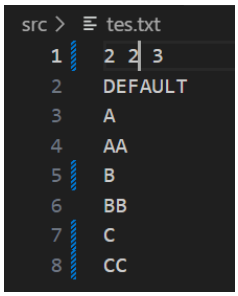


Gambar 7. Input test case 4

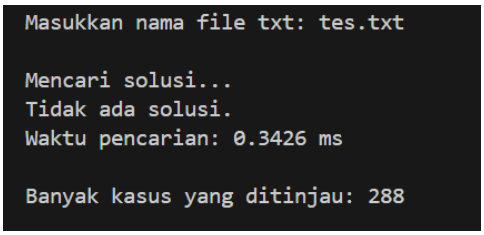
```
Masukkan nama file txt: tes.txt

Mencari solusi...
Tidak ada solusi.
Waktu pencarian: 197.9214 ms

Banyak kasus yang ditinjau: 2301000
```

Gambar 8. Output test case 4

Test Case 5

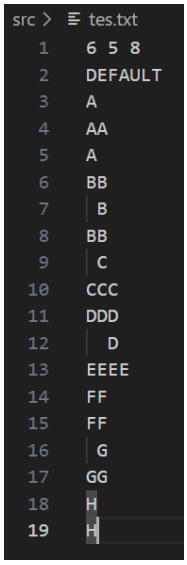


Gambar 9. Input test case 5



Gambar 10. Output test case 5

Test Case 6



Gambar 11. Input test case 6

```
Masukkan nama file txt: tes.txt

Mencari solusi...

=== Solusi Akhir ===

ABBGG
AABGC
ABBCC
EEEEC
DDDFF
HHDFF

Waktu pencarian: 740.1317 ms

Banyak kasus yang ditinjau: 4411019

Apakah anda ingin menyimpan solusi? (ya/tidak)
ya


Masukkan nama file output txt: output.txt
Output berhasil disimpan ke: ../test/output.txt

IQPuzzlerPro telah selesai. Terima kasih telah bermain.
```

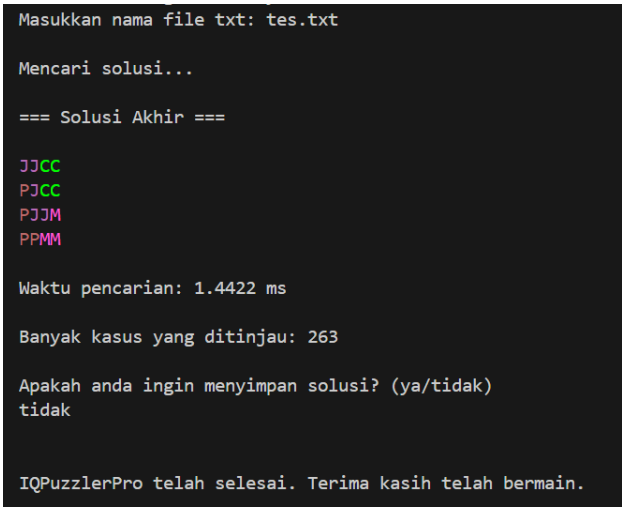
Gambar 12. Output test case 6

Test Case 7

src > tes.txt

```
4 4 4
DEFAULT
JJ
 J
 JJ
  P
PPP
CC
CC
MM
 M
```

Gambar 13. Input test case 7



Gambar 14. Output test case 7

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 *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	✓	

- Link Repository Github: https://github.com/anellautari/Tucil1_13523078.git