

Laporan Tugas Kecil 1 IF2211 Strategi Algoritma
Semester II Tahun 2024/2025

**Penyelesaian IQ Puzzler Pro
dengan Algoritma Brute Force**



Fajar Kurniawan – 13523027

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

DAFTAR ISI

DAFTAR ISI	2
BAB I: DESKRIPSI MASALAH.....	3
BAB II: ALGORITMA BRUTE FORCE UNTUK PENYELESAIAN PERMAINAN IQ PUZZLER PRO	4
BAB III: IMPLEMENTASI PROGRAM DENGAN BAHASA JAVA.....	5
BAB IV: EKSPERIMEN	14
LAMPIRAN.....	21

BAB I: DESKRIPSI MASALAH

IQ Puzzler Pro adalah permainan papan yang diproduksi oleh perusahaan Smart Games. Tujuan dari permainan ini adalah untuk 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. Pemain harus mengisi seluruh area papan menggunakan semua 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.

Permainan dimulai dengan papan yang kosong. Pemain dapat meletakkan blok puzzle sedemikian sehingga tidak ada blok yang bertumpang tindih. Setiap blok puzzle dapat dirotasikan maupun dicerminkan. Puzzle dinyatakan selesai jika dan hanya jika papan terisi penuh dan seluruh blok puzzle berhasil diletakkan.



Gambar 1. IQ Puzzler Pro

(Sumber: <https://www.takealot.com/iq-puzzle-travel-games-cognitive-skill-building-brain-game-toy-f/PLID95733096>)

Laporan ini membahas penyelesaian dari permainan IQ Puzzler Pro dengan menggunakan algoritma *brute force*.

BAB II: ALGORITMA BRUTE FORCE UNTUK PENYELESAIAN PERMAINAN IQ PUZZLER PRO

Algoritma brute force dapat digunakan untuk menyelesaikan permainan IQ Puzzler Pro dengan mencoba semua kombinasi peletakan dari seluruh blok yang ada dengan semua kemungkinan rotasi dan pencerimeranan dari masing-masing blok hingga tercapai suatu kondisi permainan selesai (jika ada). Algoritma brute force kali ini diimplementasikan dengan menggunakan *back tracking* seperti dari *slide* kuliah Strategi Algoritma halaman 46 dalam menyelesaikan sudoku [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/02-Algorithm-Brute-Force-\(2025\)-Bag1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/02-Algorithm-Brute-Force-(2025)-Bag1.pdf).

1. Langkah Pertama

Coba meletakkan blok pada koordinat paling kiri atas papan, jika tidak bisa maka kita geser blok ke kanan. Ketika sudah mentok di ujung kanan maka kembali ke ujung kiri tetapi turun 1 kotak/lubang/sel. Jika blok sudah mencapai sudut kanan bawah tetapi belum ditemukan posisi yang valid (atau posisi valid tapi verifikasi pada langkah ketiga gagal) maka piece dirotasi/dicerminkan. Proses ini diulang untuk semua lokasi penempatan yang valid dengan masing-masing penempatan valid akan bercabang ke sejumlah percobaan lanjutan menggunakan langkah kedua agar semua kemungkinan solusi dieksplorasi.

2. Langkah Kedua

Setelah didapatkan posisi penempatan blok yang valid maka dilanjutkan ke blok kedua, ketiga, dan seterusnya hingga semua blok digunakan dengan mengulangi langkah pertama pada blok kedua.

3. Langkah Ketiga

Setelah pengulangan langkah pertama dan kedua dilakukan sejumlah sekian kali, jika semua blok telah digunakan tetapi permainan tidak selesai, maka mundur satu langkah dan dicoba kemungkinan selanjutnya. Misal blok terakhir yang diletakkan akan dirotasi dan dicoba diletakkan pada papan dari ujung kiri atas hingga kanan bawah. Namun, jika semua blok telah digunakan dan solusi yang dapat menyelesaikan permainan ditemukan, program akan dihentikan dan solusi ditampilkan.

BAB III: IMPLEMENTASI PROGRAM DENGAN BAHASA JAVA

Algoritma pada Bab II diimplementasikan dengan bahasa Java menggunakan *Object Oriented Programming*. Maka dari itu terdapat 5 kelas yang saling berinteraksi satu sama lain yang diwujudkan dalam 5 file berbeda, dengan masing-masing file memiliki nama yang sama dengan nama kelasnya yaitu: **Block.java**, **Board.java**, **Solve.java**, **Game.java**, dan **Main.java**. Kelas Block merepresentasikan objek blok, kelas Board merepresentasikan papan, kelas Solve merepresentasikan algoritma brute force, dan kelas Main sebagai program utama yang menggabungkan kelas-kelas yang lain.

Block.java

Kelas Block memiliki tiga atribut yaitu 2D matrix of char yang merepresentasikan balok itu sendiri, symbol yang menunjukkan karakter penyusun blok, list of 2D matrix of char yang menyimpan variasi blok yang dirotasikan dan dicerminkan untuk mengurangi beban komputasi sehingga tidak perlu melakukan rotasi dan pencerminan berkali-kali. Terdapat method constructor yang membuat objek Block dari sebuah input string pada parameternya.

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

public class Block {

    char[][] block;
    char symbol;

    List<char[][]> variasi;
    int varIndex;

    public Block(String blockStr) {
        String[] lines = blockStr.split("\n");
        // split the strings by newline to list of string
        block = new char[lines.length][lines[0].length()];
        // create 2D array of char with the same size as String[] lines
        for (String line : lines) {
            for (int j = 0; j < line.length(); j++) {
                if (line.charAt(j) != ' ') {
                    symbol = line.charAt(j);
                    // set symbol of block
                    break;
                }
            }
        }
        for (int i = 0; i < lines.length; i++) {
            block[i] = lines[i].toCharArray();
            // convert each string of lines to array of char and put it to block[i]
        }

        variasi = new ArrayList<>();
        makeVariasi();
        varIndex = 0;
    }
}
```

Kemudian ada method makeVariasi untuk membuat variasi yang disebutkan di atas. Method getter currentVar dan method setter nextVar. Method printBlock untuk mengoutput blok sebagai teks dan method rotate serta mirror untuk merotasi dan mencerminkan blok.

```

public void makeVariasi() {
    variasi.add(block);
    char[][] newBlock = block;
    for (int i = 0; i < 3; i++) {
        newBlock = rotate(newBlock);
        variasi.add(newBlock);
    }
    newBlock = mirror(block);
    variasi.add(newBlock);
    for (int i = 0; i < 3; i++) {
        newBlock = rotate(newBlock);
        variasi.add(newBlock);
    }
}

public char[][] currentVar() {
    return variasi.get(varIndex);
}

public void nextVar() {
    varIndex = (varIndex + 1) % variasi.size();
}

public void printBlock() {
    for (char[] chars : this.currentVar()) {
        for (char bChar : chars) {
            System.out.print(bChar);
        }
        System.out.println();
    }
}

public char[][] rotate(char[][] block) {
    char[][] newBlock = new char[block[0].length]
    [block.length]; //angnya sama lebarnya ditukar
    for (int i = 0; i < block.length; i++) {
        for (int j = 0; j < block[0].length; j++) {
            newBlock[j][block.length - 1 - i] = block[i][j];
            // rotate 90 degree clockwise
        }
    }
    return newBlock;
}

public char[][] mirror(char[][] block) {
    char[][] newBlock = new char[block.length]
    [block[0].length]; // dan lebar tetap
    for (int i = 0; i < block.length; i++) {
        for (int j = 0; j < block[0].length; j++) {
            newBlock[i][block[0].length - 1 - j] = block[i][j];
            // flip horizontally
        }
    }
    return newBlock;
}
}

```

Board.java

Kelas Board memiliki 4 atribut yaitu 2D matrix of char untuk merepresentasikan papan. Atribut height dan width untuk lebar dan panjang papan. String mode untuk menentukan apakah papan default (persegi) atau kustom.

```
public class Board {  
    // Board untuk meletakkan block  
  
    char[][] board;  
    int height;  
    int width;  
    String mode = "DEFAULT";  
}
```

Kemudian ada lebih dari dua puluh kode ANSI untuk memberikan warna pada teks di console

```
public static final String RESET = "\033[0m";  
public static final String RED = "\033[0;31m";  
public static final String BLACK = "\033[0;30m";  
public static final String GREEN = "\033[0;32m";  
public static final String YELLOW = "\033[0;33m";  
...
```

Selanjutnya ada method constructor dengan beberapa variasi.

```
public Board(int row, int col) {  
    board = new char[row][col];  
    // create 2D array and fill with '.'  
    for (int i = 0; i < row; i++) {  
        for (int j = 0; j < col; j++) {  
            board[i][j] = '.';  
        }  
    }  
    height = row;  
    width = col;  
}  
  
public Board(Board board) {  
    this.height = board.height;  
    this.width = board.width;  
    this.mode = board.mode;  
    this.board = new char[height][width];  
    for (int i = 0; i < height; i++) {  
        System.arraycopy(board.board[i], 0, this.board[i], 0, width);  
    }  
}  
  
public Board(char[][] board) {  
    this.height = board.length;  
    this.width = board[0].length;  
    this.board = new char[height][width];  
    for (int i = 0; i < height; i++) {  
        System.arraycopy(board[i], 0, this.board[i], 0, width);  
    }  
}
```

Lalu ada method setCustom untuk mengubah atribut mode menjadi CUSTOM. Ada method printBoard dan printColor untuk memberikan warna pada blok-blok yang terpasang di papan ketika diprint ke console. Ada method cekFit yang memanggil dua method berbeda lainnya

tergantung apakah atribut mode merupakan DEFAULT atau CUSTOM. Method tersebut digunakan untuk mengecek apakah suatu blok dapat diletakkan pada papan pada posisi tertentu.

```
public void setCustom() {
    mode = "CUSTOM";
}

public void printBoard() {
    for (char[] chars : board) {
        for (char bChar : chars) {
            printColor(bChar);
        }
        System.out.println();
    }
}

public static void printColor(char bChar) {
    switch (bChar) {
        case 'A':
            System.out.print(RED + bChar + RESET);
            break;
        case 'B':
            System.out.print(GREEN + bChar + RESET);
            break;
        case 'C':
            ...
    }
}

public boolean cekFit(Block block, int row, int col) {
    if (mode.equals("CUSTOM")) {
        return cekFitCustom(block, row, col);
    } else {
        return cekFitBlock(block, row, col);
    }
}

public boolean cekFitBlock(Block block, int row, int col) {
    for (int i = 0; i < block.currentVar().length; i++) {
        for (int j = 0; j < block.currentVar()[0].length; j++) {
            if (block.currentVar()[i][j] != ' ' && board[row + i][col +
j] != '.') {
                return false;
            }
        }
    }
    return true;
}

public boolean cekFitCustom(Block block, int row, int col) {
    for (int i = 0; i < block.currentVar().length; i++) {
        for (int j = 0; j < block.currentVar()[0].length; j++) {
            if (block.currentVar()[i][j] != ' ' && (board[row + i][col +
j] != '.' || board[row + i][col + j] == ' ')) {
                return false;
            }
        }
    }
    return true;
}
```

Kemudian ada method clearBoard untuk menghapus semua blok yang dipasang di papan. Method putBlock untuk meletakkan blok di papan. Method cekFull untuk mengecek apakah papan penuh.


```

public void clearBoard() {
    for (int i = 0; i < board.length; i++) {
        for (int j = 0; j < board[0].length; j++) {
            board[i][j] = '.';
        }
    }
}

public void putBlock(Block block, int row, int col) {
    for (int i = 0; i < block.currentVar().length; i++) {
        for (int j = 0; j < block.currentVar()[0].length; j++) {
            if (block.currentVar()[i][j] != ' ') {
                board[row + i][col + j] = block.currentVar()[i][j];
            }
        }
    }
}

public boolean cekFull() {
    for (char[] chars : board) {
        for (int i = 0; i < board[0].length; i++) {
            if (chars[i] == '.') {
                return false;
            }
        }
    }
    return true;
}

```

Solve.java

Kelas Solve memiliki banyak atribut yaitu object Board yang digunakan sebagai papan permainan dan object Board lainnya untuk menyimpan solusi permainan. Kemudian ada list of Blocks untuk menyimpan blok-blok pada permainan, ada juga tries dan time yang berisi jumlah percobaan dan waktu yang digunakan. Boolean solved yang menunjukkan keadaan permainan. Lalu bWidth dan bHeight yaitu panjang dan lebar papan. Ada method constructor Solve. Terdapat method solve yang akan menghitung waktu yang diperlukan untuk menyelesaikan permainan dan memanggil fungsi solveNew yang merupakan method untuk menggunakan brute force pada permainan.

```

import java.util.List;
import java.util.Stack;

public class Solve {

    List<Block> blocks;
    Board board;
    Board solution;
    long tries;
    long time;
    boolean solved = false;
    int bWidth;
    int bHeight;

    public Solve(List<Block> blocks, Board board) {
        this.blocks = blocks;
        this.board = board;
        this.tries = 0;
        this.time = 0;
        this.bWidth = board.board[0].length;
        this.bHeight = board.board.length;
    }

    public boolean solve() {
        long startTime = System.currentTimeMillis();
        boolean result = solveNew();
        long endTime = System.currentTimeMillis();
        time = endTime - startTime;
        return result;
    }
}

```

Selanjutnya method `solveNew` yang merupakan fungsi procedural dengan memanfaatkan stack untuk mengeksplorasi semua kemungkinan yang ada dengan algoritma brute force pada Bab II. Mengapa menggunakan stack? Hal ini karena asisten melarang penggunaan rekursif pada QnA meskipun menurut saya dengan rekursi pun akan bisa melakukan brute force untuk mencoba semua solusi dan tidak ada bedanya dengan procedural. Bahkan saya melihat tugas kecil 1 Strategi Algoritma beberapa mahasiswa IF angkatan 21 yang menggunakan rekursif. Namun, karena asisten selalu benar maka saya menggantinya dengan procedural menggunakan stack agar tetap memudahkan saya untuk *back tracking*. Fungsi yang baru ini lah yang menjadi alasan mengapa namanya `solveNew`, bukan hanya `solve`. Kemudian ada kelas `Step` sebagai representasi state permainan saat ini yang akan di-*push* dan di-*pop* ke dan dari stack. Penggunaan stack dan kelas `Step` ini tentu sangat memudahkan untuk menyimpan semua percobaan dibandingkan menggunakan banyak array maupun list.

```
private boolean solveNew() {
    Stack<Step> stack = new Stack<>();
    stack.push(new Step(0, this.board));

    while (!stack.isEmpty()) {
        Step currentStep = stack.pop();
        int idx = currentStep.idx;
        Board currentBoard = currentStep.board;

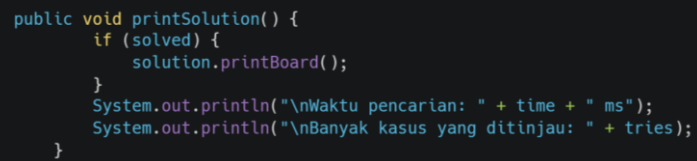
        if (idx == blocks.size()) {
            if (currentBoard.cekFull()) {
                solution = new Board(currentBoard);
                solved = true;
                return true;
            }
            continue;
        }

        Block piece = blocks.get(idx);
        for (int k = 0; k < 8; k++) {
            int pWidth = piece.currentVar()[0].length;
            int jarak = 0;
            for (int m = 0; m < pWidth; m++) {
                if (piece.currentVar()[0][m] != ' ') {
                    jarak = m;
                    break;
                }
            }
            int height = bHeight - piece.currentVar().length;
            int width = bWidth - pWidth;
            for (int i = 0; i <= height; i++) {
                for (int j = 0; j <= width; j++) {
                    if (currentBoard.board[i][j + jarak] == '.' && currentBoard.cekFit(piece, i, j)) {
                        Board newBoard = new Board(currentBoard);
                        newBoard.putBlock(piece, i, j);
                        stack.push(new Step(idx + 1, newBoard));
                        tries++;
                    }
                }
                tries++;
            }
            piece.nextVar();
        }
    }
    return false;
}

private static class Step {
    int idx;
    Board board;

    Step(int idx, Board board) {
        this.idx = idx;
        this.board = board;
    }
}
```

Terakhir, ada method `printSolution` untuk mengoutput solusi yang didapatkan (jika ada) beserta waktu dan jumlah percobaan yang diperlukan.



```
public void printSolution() {  
    if (solved) {  
        solution.printBoard();  
    }  
    System.out.println("\nWaktu pencarian: " + time + " ms");  
    System.out.println("\nBanyak kasus yang ditinjau: " + tries);  
}
```

Game.java

Kelas Game memiliki sejumlah atribut yang diperlukan untuk menggunakan kelas Board dan Board serta Solve. Namun, ada object kelas Scanner yang akan diisi dengan object Scanner dari Main agar tidak terjadi error karena penutupan stream I/O System.in. Ada method constructor, kemudian method loadFromTxt untuk membaca input test case permainan dari file *.txt. Kemudian ada method solve untuk memanggil method solve dari kelas Solve. Ada addPadding untuk menyamaratakan whitespace pada blok yang ada sehingga memudahkan algoritma pengecekan pemasangan blok pada papan. Method saveSolusi untuk menyimpan solusi ke file. Method runGame untuk memanggil method-method lainnya untuk mengambil input, melakukan solve, dan menyimpan output. Kemudian ada method createImage untuk membuat gambar dari solusi yang ada. Selanjutnya ada method char2color untuk mengubah karakter menjadi kotak-kotak berwarna untuk digunakan oleh method createImage.

```
public class Game {

    int boardW;
    int boardH;
    int blockCount;
    String mode;
    List<Block> blocks = new ArrayList<>();
    Board board;
    Solve solve;
    Scanner scan;

    public Game(Scanner scan) {
        boardW = 0;
        boardH = 0;
        blockCount = 0;
        mode = "";
        this.scan = scan;
    }

    public void loadFromTxt(String path)

    public void solve()

    private List<String> addPadding(List<String> lines)

    private void saveSolusi()

    public void runGame()

    private static void createImage(Board board, int scale)

    private static Color char2color(char c)
```

Main.java

Kelas Main adalah program utama dengan kalang (*loop*) utama. Ia akan meminta input user dan akan menjalankan permainan dengan membuat instance dari kelas Game jika user menginput angka 1. Sedangkan program akan berhenti dan keluar jika user menginput angka 2. Terdapat juga validasi input karena input yang valid hanyalah 1 atau 2.

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {

        System.out.println("=====");
        System.out.println("IQ PUZZLER PRO!!! :D");
        System.out.println("=====");

        Scanner scan = new Scanner(System.in);

        while (true) {
            String input;
            System.out.println("\nKetik salah satu nomor di bawah untuk:");
            System.out.println("1. Mulai\n2. Exit\n");

            input = scan.nextLine();

            if (input.equals("1")) {
                Game game = new Game(scan);
                game.runGame();
            } else if (input.equals("2")) {
                System.out.println("Anda keluar.\n");
                break;
            } else {
                System.out.println("Input tidak valid\n");
            }
        }
    }
}
```

BAB IV: EKSPERIMEN

Menu Utama

```
=====
IQ PUZZLER PRO!!! :D
=====

Ketik salah satu nomor di bawah untuk:
1. Mulai
2. Exit
```

Keluar dari Program

```
Ketik salah satu nomor di bawah untuk:
1. Mulai
2. Exit

2
Anda keluar.

Process finished with exit code 0
```

Mulai Permainan

```
Ketik salah satu nomor di bawah untuk:
1. Mulai
2. Exit

1
Masukkan nama file test case:
src/input.txt
```

1. Test Case 1

Input

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

```
FF
F
GGG
```

Hasil:

```
Board height: 5
Board width: 5
Mode: DEFAULT
Block count: 7

FFDDE
FFDEE
FBBEE
CCBAA
CGGGA

Waktu pencarian: 52 ms

Banyak kasus yang ditinjau: 2693789
Apakah anda ingin menyimpan solusi? (ya/tidak)
```

2. Test Case 2 (Bonus Konfigurasi Custom)

Input

```
5 7 5
CUSTOM
...X...
.XXXXX.
XXXXXXX
.XXXXX.
...X...
A
AAA
BB
BBB
CCCC
C
D
EEE
E
```

Hasil

```

Board height: 5
Board width: 7
Mode: CUSTOM
Block count: 5

  C
 ECCC
DEAABBB
EEABB
  A

Waktu pencarian: 1 ms

Banyak kasus yang ditinjau: 1518
Apakah anda ingin menyimpan solusi? (ya/tidak)

```

3. Test Case 3 (Output Berupa Gambar)

Input

```

4 4 4
CUSTOM
X..X
X.XX
XXX.
..XX
Z
YYY
Y Y
XX
VV

```

Hasil

```

Board height: 4
Board width: 4
Mode: CUSTOM
Block count: 4

X  Z
X YY
VY
  YY

Waktu pencarian: 1 ms

Banyak kasus yang ditinjau: 7544
Apakah anda ingin menyimpan solusi? (ya/tidak)
ya
Simpan dalam gambar atau teks? (ketik png/txt)
png
Gambar disimpan: img_solusi8.png

```


Output gambar:



4. Test Case 4

Input

```
3 3 2
DEFAULT
AAA
A A
AAA
B
```

Hasil

```
Board height: 3
Board width: 3
Mode: DEFAULT
Block count: 2

AAA
ABA
AAA

Waktu pencarian: 1 ms

Banyak kasus yang ditinjau: 96
Apakah anda ingin menyimpan solusi? (ya/tidak)
```

5. Test Case 5

Input

```
3 3 1
DEFAULT
AAA
A A
AAA
```

Hasil

```
Board height: 3
Board width: 3
Mode: DEFAULT
Block count: 1

Waktu pencarian: 2 ms

Banyak kasus yang ditinjau: 16
No solution found
```

6. Test Case 6

Input

```
4 4 4
DEFAULT
AAA
BBBB
CC
C
CC
DD
D
D
```

Hasil

```
Board height: 4
Board width: 4
Mode: DEFAULT
Block count: 4

Waktu pencarian: 14 ms

Banyak kasus yang ditinjau: 340864
No solution found
```

7. Test Case 7

Input

```
5 5 7
DEFAULT
A
AA
B
BB
C
CC
D
DD
EE
```

```
EE
E
FF
FF
F
GG
```

Hasil

```
Board height: 5
Board width: 5
Mode: DEFAULT
Block count: 7

Waktu pencarian: 178494 ms

Banyak kasus yang ditinjau: 13100853104
No solution found
```

Menyimpan Solusi dalam Teks

Input

```
4 4 4
DEFAULT
AAAA
BBBB
CCCC
DDDD
```

Hasil

```
Board height: 4
Board width: 4
Mode: DEFAULT
Block count: 4

DCBA
DCBA
DCBA
DCBA

Waktu pencarian: 0 ms

Banyak kasus yang ditinjau: 184
Apakah anda ingin menyimpan solusi? (ya/tidak)
ya
Simpan dalam gambar atau teks? (ketik png/txt)
txt
Solution saved to solusi6.txt
```

Output File

```
DCBA  
DCBA  
DCBA  
DCBA
```

LAMPIRAN

Github Repository

[Fajar2k5/Tucil1_13523027: Tugas Kecil 1 Strategi Algoritma 2025](#)

Tabel Spesifikasi

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	✓	