

# **IF2121 - Logika Komputasional**

## **Tugas Kecil**

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



**Disusun Oleh :**

13523069 - Mochammad Fariz Rifqi Rizqulloh

**Program Studi Teknik Informatika  
Sekolah Teknik Elektro dan Informatika  
Institut Teknologi Bandung  
2024**

## **Daftar Isi**

Daftar Isi.....	2
BAB 1: Pendahuluan.....	3
1.1. Deskripsi Tugas.....	3
1.2. Ilustrasi Kasus.....	3
BAB 2 : Pembahasan.....	5
2.1. Algoritma Brute Force.....	5
2.2. Brute Force Untuk Penyelesaian Masalah.....	5
2.3. Pseudocode.....	7
BAB 3 : Implementasi.....	8
3.1. Framework.....	8
3.2. Struktur Kode.....	8
3.3. Source Code.....	8
4.1. Kasus Solusi Ada.....	8
4.2. Kasus Solusi Tidak Ada.....	8
Lampiran.....	8

## BAB 1: Pendahuluan

### 1.1. Deskripsi Tugas

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 anda adalah 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.

### 1.2. Ilustrasi Kasus

Diberikan sebuah wadah berukuran  $11 \times 5$  serta 12 buah blok puzzle dan beberapa blok telah ditempatkan dengan bentuk sebagai berikut



Awal Permainan Game IQ Puzzler Pro

Gambar 1. (Sumber: <https://www.smartgamesusa.com>)

Pemain berusaha untuk mengisi bagian papan yang kosong dengan menggunakan blok yang tersedia. Permainan dinyatakan selesai jika pemain mampu mengisi seluruh papan dengan blok (dalam tugas kecil ini ada kemungkinan pemain tidak dapat mengisi seluruh papan)



Kondisi Permainan Selesai

Gambar 2. (Sumber: <https://www.smartgamesusa.com>)

## BAB 2 : Pembahasan

### 2.1. Algoritma Brute Force

Algoritma bruteforce adalah suatu pendekatan yang lurus atau lempang (straightforward) untuk memecahkan suatu persoalan. Algoritma brute force memecahkan persoalan secara sangat sederhana, langsung, jelas caranya, serta mudah dipahami.

Contoh beberapa algoritma bruteforce juga digunakan pada permasalahan permasalahan berikut :

- Menghitung  $a^n$
- Menghitung  $n!$
- Mengalikan dua buah matriks, A dan B, berukuran  $n \times n$
- Uji Bilangan Prima
- Pengurutan elemen
- Algoritma brute force pencocokan string
- Mencari Pasangan Titik yang Jaraknya Terdekat (Closest Pair Problem)

Algoritma brute force umumnya tidak “cerdas” dan tidak sangkil, karena ia membutuhkan cost komputasi yang besar dan waktu yang lama dalam penyelesaiannya. Kata “force” mengindikasikan “tenaga” ketimbang “otak” Kadang-kadang algoritma brute force disebut juga algoritma naif (naïve algorithm). Algoritma brute force lebih cocok untuk persoalan yang ukuran masukannya ( $n$ ) kecil.

Kelebihan dari algoritma bruteforce antara lain adalah bahwa algoritma brute force dapat diterapkan untuk memecahkan hampir sebagian besar masalah (wide applicability). Selain itu, algoritma brute force sederhana dan mudah dimengerti, serta algoritma brute force menghasilkan algoritma baku (standard) untuk tugas-tugas komputasi seperti penjumlahan/perkalian  $n$  buah bilangan, menentukan elemen minimum atau maksimum di dalam senarai (larik). Di sisi lain, algoritma bruteforce jarang menghasilkan algoritma yang mangkus, umumnya lambat untuk masukan berukuran besar sehingga tidak dapat diterima, serta tidak sekonstruktif/sekreatif strategi pemecahan masalah lainnya.

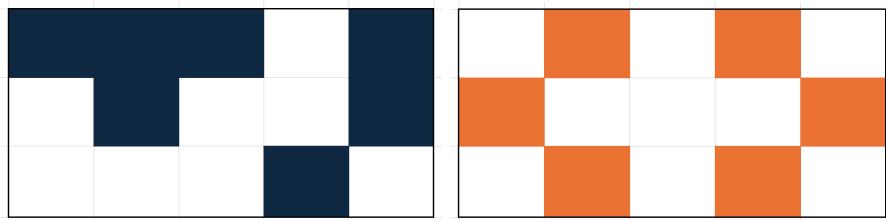
## 2.2. Brute Force Untuk Penyelesaian Masalah

Pendekatan brute force digunakan dalam penyelesaian masalah puzzle karena menjamin pencarian solusi dengan mencoba semua kemungkinan secara sistematis, tanpa memerlukan heuristik atau pendekatan kompleks lainnya. Metode ini sangat cocok untuk masalah dengan ruang solusi terbatas atau ketika solusi optimal tidak dapat ditemukan dengan cara yang lebih efisien. Selain itu, brute force sering digunakan sebagai pendekatan awal dalam pengembangan algoritma sebelum mencari metode yang lebih optimal, serta berguna dalam verifikasi hasil dari teknik yang lebih kompleks.

Strategi bruteforce yang digunakan dengan memanfaatkan fakta bahwa setiap petak harus diisi oleh hanya satu keping puzzle. Jika dalam meletakkan kepingan puzzle mengakibatkan adanya 2 kepingan puzzle mengisi satu petak yang sama, atau terdapat kepingan puzzle yang berada di luar papan permainan, artinya kita tidak bisa meletakkan kepingan puzzle pada posisi tersebut. Sehingga kita harus mencoba cara lain, bisa dengan merotasi kepingan puzzle tersebut, membalik kepingan puzzle tersebut, mengubah posisi peletakan kepingan puzzle tersebut, atau mengubah kepingan puzzle tersebut.

Sebelum melakukan algoritma bruteforce, input yang diberikan oleh pengguna akan diproses terlebih dahulu untuk didapatkan kepingan-kepingan puzzle yang sesuai. Dalam proses ini, juga terdapat beberapa validasi yang dilakukan, antara lain

1. Validasi kesesuaian format input
2. Total luas kepingan puzzle tidak sama dengan luas papan permainan
3. Dimensi dari kepingan puzzle terlalu besar, melebihi dimensi papan
4. Terdapat 2 atau lebih kepingan puzzle dengan representasi alphabet yang sama
5. Terdapat lebih dari 26 kepingan puzzle
6. Terdapat kepingan puzzle yang tidak *connected* (tidak nyambung), beberapa contoh kasus tersebut adalah kepingan puzzle seperti berikut :



Gambar 3. Contoh kepingan invalid (1)

#### 7. Kesesuaian jumlah kepingan puzzle serta variabel P dalam input

Setelah validasi-validasi tersebut, akan dilakukan pre-prosessing terlebih dahulu, untuk mempersiapkan algoritma bruteforce. Akan dibuat 2 class, yaitu class Block, yang merepresentasikan papan permainan, serta class Tetromino, yang merepresentasikan kepingan puzzle yang digunakan. Suatu papan permainan dianggap sebagai array 2 dimensi dengan element integer, dimana value **-1** berarti posisi tersebut masih belum diisi, sementara value lainnya menyatakan bahwa posisi tersebut diisi oleh kepingan puzzle dengan ID sesuai dengan value tersebut. Suatu kepingan puzzle akan dianggap array 2 dimensi dengan element boolean, value **true** menyatakan bahwa terdapat kepingan puzzle di posisi tersebut, sementara value **false** menyatakan bahwa tidak terdapat kepingan puzzle di posisi tersebut. Selain itu, setiap kepingan puzzle juga memiliki atribut **N**, **M**, yakni ukuran dari puzzle tersebut, serta **ID**, yang menyatakan ID dari puzzle tersebut.

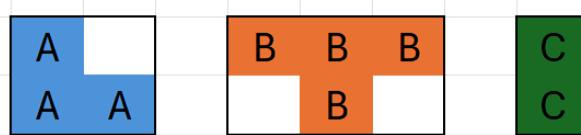
Setelah pre-processing selesai dilakukan, algoritma bruteforce akan dilakukan sampai menemukan 1 solusi, atau tidak ditemukan sama sekali solusi yang memenuhi. Langkah-langkah strategi bruteforce tersebut adalah sebagai berikut

1. Mulai dari posisi  $(i, j) = (1, 1)$ , yaitu baris ke-1 dari atas, kolom ke-1 dari kiri.
2. Coba untuk memasang sebuah kepingan puzzle yang **belum pernah dipakai sebelumnya** pada posisi  $(i, j)$ . Pastikan bahwa kepingan tersebut **harus** mengisi posisi  $(i, j)$ . Apabila tidak ada kepingan puzzle yang dapat ditempatkan untuk mengisi posisi tersebut, coba untuk merotasi kepingan-kepingan puzzle. Apabila masih tidak ada kepingan puzzle yang dapat ditempatkan untuk mengisi posisi tersebut, coba untuk membalik (*flip*) kepingan puzzle-puzzle tersebut. Jika masih tidak ada kepingan puzzle yang dapat ditempatkan, maka tinggalkan kasus tersebut. Kasus tersebut adalah kasus

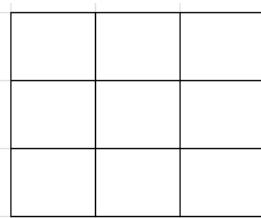
*dead-end*, yakni kasus yang tidak mungkin diselesaikan. Lakukan *back-tracking* pada keadaan sebelumnya, dan mengubah *state* puzzle yang ditempatkan pada posisi sebelumnya. Pengubahan *state* dapat dilakukan dengan merotasi puzzle, menggeser puzzle (selama puzzle tersebut masih menempati posisi  $(i, j)$ ), melakukan flip pada puzzle, atau mengganti kepingan puzzle menuju kepingan yang lain.

3. Check apakah semua posisi sudah pada papan permainan telah diisi oleh kepingan puzzle atau belum. Jika semua posisi telah diisi, program bisa langsung dihentikan, dan cetak hasil ke layar
4. Apabila tidak ada pelanggaran (tidak ada 2 atau lebih kepingan puzzle yang menempati posisi papan permainan yang sama, atau terdapat bagian dari kepingan puzzle yang melewaati papan permainan), traversal ke kanan sampai ditemukan posisi yang belum terisi oleh puzzle atau sudah sampai ujung kanan dari board. Jika sudah sampai ujung kanan dari board, turun 1 posisi ke bawah. Ulangi langkah ini sampai ditemukan posisi yang belum terisi oleh puzzle.
5. Jika sudah menemukan posisi  $(i, j)$  yang belum diisi oleh kepingan puzzle, ulangi langkah 2 sampai langkah 5

Berikut adalah visualisasi sederhana pada algoritma bruteforce yang digunakan :

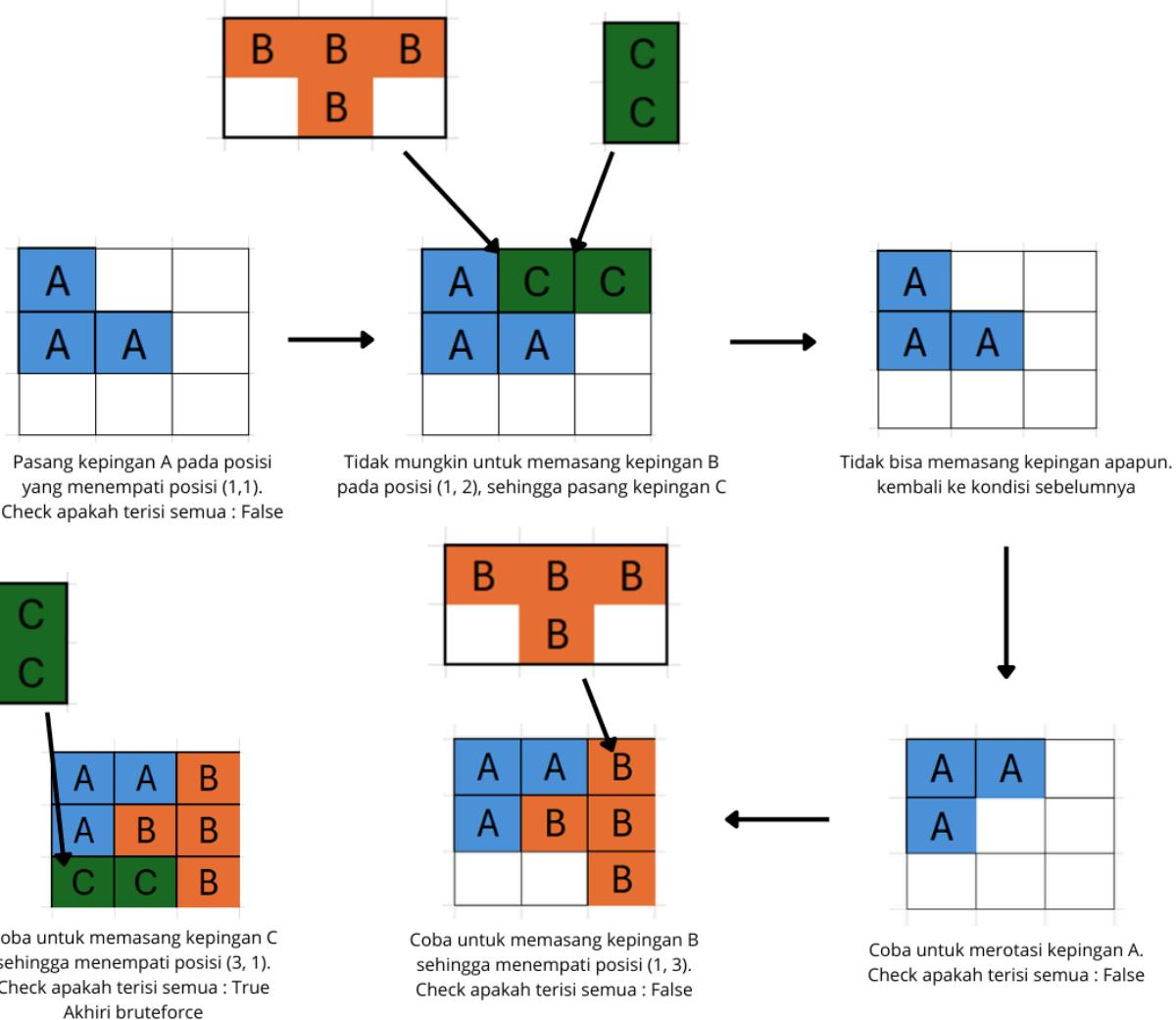


Gambar 4. Kepingan puzzle yang akan digunakan



Gambar 5. Papan permainan yang akan digunakan

Berikut adalah proses penyelesaian puzzle tersebut dengan algoritma bruteforce yang telah dijelaskan sebelumnya :



Algoritma brute force memastikan bahwa semua kasus akan diuji, sehingga memberikan jaminan bahwa solusi yang ditemukan adalah benar dan lengkap. Dengan mencoba setiap kemungkinan tanpa pengecualian, metode ini tidak melewatkkan potensi solusi yang mungkin tersembunyi dalam ruang pencarian. Meskipun tidak efisien untuk masalah dengan kompleksitas tinggi, brute force tetap menjadi pendekatan yang andal untuk skenario dengan ruang solusi terbatas atau ketika tidak ada metode heuristik yang lebih baik tersedia.

### 2.3. Pseudocode

```

FUNCTION findSolution(posx : integer, posy : integer, used : array of boolean)
  IF check() THEN
    printBlock()
    foundSolution = TRUE
    RETURN
  ENDIF

  FOR cur_tetromino FROM 1 TO tetrominoCount DO
    IF NOT used[cur_tetromino] THEN
      used[cur_tetromino] = TRUE
      tetromino = tetrominos[cur_tetromino]

      { coba semua kemungkinan rotasi dari tetromino }
      FOR rot FROM 0 TO 3 DO
        { coba semua kemungkinan flip dari tetromino }
        FOR flip FROM 0 TO 1 DO
          {coba semua kemungkinan posisi dari tetromino agar mengisi posisi (posx, posy) pada board }
          FOR shiftx FROM -tetromino.n + 1 TO tetromino.n - 1 DO
            FOR shifty FROM -tetromino.m + 1 TO tetromino.m - 1 DO
              bruteForceCount = bruteForceCount + 1

              { apabila tetromino dalam diletakkan, maka lanjut untuk meletakkan tetromino selanjutnya }
              IF placeTetromino(tetromino, posx, posy, posx + shiftx, posy + shifty) THEN
                nextx = posx
                nexty = posy + 1

                { cari posisi selanjutnya yang masih belum diisi oleh kepingan tetromino }
                WHILE nextx <= n DO
                  WHILE nexty <= m AND arr[nextx][nexty] != -1 DO
                    nexty = nexty + 1
                  ENDWHILE

                  IF nexty <= m THEN
                    BREAK
                  ENDIF

                  nexty = 1
                  nextx = nextx + 1
                ENDWHILE

                { rekursi untuk meletakkan kepingan puzzle lain pada posisi yang belum terisi }
                findSolution(nextx, nexty, used)
              ENDIF
            ENDFOR
          ENDFOR
        ENDFOR
      ENDFOR
    ENDIF
  ENDFOR
ENDFUNCTION

```

```

        IF foundSolution THEN
            RETURN
        ENDIF
        { apabila masih belum ada solusi, hapus kepingan puzzle yang sekarang }
        removeTetromino(tetromino, posx + shiftx, posy + shifty)
    ENDIF
ENDFOR
ENDFOR

    tetromino = tetromino.flip()
ENDFOR

    tetromino = tetromino.rotate()
ENDFOR
used[cur_tetromino] = FALSE
ENDIF
ENDFOR
END FUNCTION

```

## 2.4. Analisis Kompleksitas Waktu

Misalkan N adalah lebar papan permainan, M adalah panjang papan permianan, serta P adalah banyak kepingan puzzle, Dalam kasus terburuk, akan terdapat P posisi yang perlu diisi oleh kepingan kepingan puzzle tersebut, dimana

- Terdapat P puzzle yang dapat diletakkan pada posisi tersebut, dikurangi dengan banyak puzzle yang telah dipakai
- Terdapat 4 kemungkinan rotasi
- Terdapat 2 kemungkinan *flip*

Sehingga, banyak pemasangan puzzle yang mungkin dalam kasus terburuk adalah :

$$T(N) = 8P * 8(P - 1) * 8(P - 2) * \dots * 16 * 8$$

$$T(N) = 8^P * P!$$

Dalam tiap pemasangannya, akan dilakukan operasi *assignment* sebanyak luas dari kepingan puzzle tersebut, dimana total luas kepingan puzzle tersebut adalah  $N*M$ , maka dapat diambil kesimpulan bahwa kompleksitas waktu dari algoritma ini adalah  $O(8^P * P! * N * M)$

Perlu diperhatikan bahwa analisis kompleksitas waktu tersebut masih tidak akurat, dimana penghitungan tersebut masih belum menghitung banyaknya “pergeseran” yang bisa dilakukan untuk tiap puzzle.

## BAB 3 : Implementasi

### 3.1. Framework

Dalam pengembangan program puzzle solver ini, saya menggunakan Java sebagai bahasa pemrograman utama. Java dipilih karena sifatnya yang portabel, performanya yang stabil, serta ekosistemnya yang luas, yang memungkinkan pengembangan aplikasi dengan struktur yang bersih dan modular. Dengan paradigma object-oriented programming (OOP), Java memudahkan pengelolaan komponen-komponen dalam solver, seperti representasi puzzle, algoritma pencarian solusi, dan pengelolaan UI.

Untuk antarmuka grafis, saya menggunakan JavaFX, yang memberikan fleksibilitas dalam membuat tampilan interaktif dan responsif. JavaFX memungkinkan saya untuk menampilkan visualisasi puzzle secara dinamis, termasuk proses penyelesaiannya secara real-time. Dengan fitur seperti Canvas, Scene Graph, dan Animation API, saya dapat memberikan pengalaman pengguna yang lebih intuitif dengan animasi yang halus dan tata letak yang fleksibel.

Sebagai sistem build, saya memilih Gradle untuk mengelola dependensi dan otomatisasi build. Gradle memudahkan pengelolaan pustaka eksternal yang digunakan dalam proyek, seperti JavaFX SDK, serta memastikan proses kompilasi, testing, dan packaging berjalan secara efisien. Dengan Gradle, saya juga dapat mengonfigurasi proyek agar lebih modular, sehingga memudahkan pemeliharaan dan pengembangan lebih lanjut.

Kombinasi Java, JavaFX, dan Gradle memberikan fondasi yang kuat untuk program puzzle solver ini, memungkinkan pengembangan yang skalabel, efisien, dan mudah dikelola.

### 3.2. Struktur Kode

Dalam implementasi solusi, source code dipecah menjadi tiga bagian utama untuk memastikan modularitas dan keterpisahan tanggung jawab dalam program.

1. App.java berperan sebagai controller utama dalam solver ini. File ini bertanggung jawab dalam menampilkan antarmuka grafis menggunakan JavaFX, menerima dan memproses input dari pengguna, melakukan validasi input, serta melakukan parsing input agar sesuai dengan format yang digunakan dalam sistem. Selain itu, App.java juga bertugas memanggil fungsi-fungsi utama dari kelas lain untuk menjalankan proses penyelesaian puzzle dan menampilkan hasilnya secara visual.
2. Tetromino.java berisi definisi class Tetromino, yang merepresentasikan bentuk-bentuk yang digunakan dalam penyelesaian puzzle. File ini mencakup atribut seperti bentuk dan orientasi tetromino serta metode yang digunakan untuk manipulasi, seperti rotasi dan

flipping. Dengan pendekatan ini, setiap tetromino dapat dikelola secara independen, memungkinkan fleksibilitas dalam pencarian solusi.

3. Block.java merepresentasikan papan permainan, tempat tetromino ditempatkan selama proses penyelesaian. File ini mencakup atribut yang menggambarkan kondisi papan serta metode untuk menempatkan dan menghapus tetromino, memeriksa validitas posisi, serta melacak status permainan. Dengan pemisahan ini, manipulasi papan dapat dilakukan secara terpisah tanpa mempengaruhi logika pengolahan lainnya.

### 3.3. Source Code

#### 3.3.1. App.java

```
package org.example;

import java.io.BufferedReader;
import java.io.File;
import java.io.*;
import java.nio.file.*;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;
import java.util.Queue;

import org.checkerframework.common.returnsreceiver.qual.This;

import javafx.application.Application;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.scene.text.Text;
import javafx.stage.FileChooser;
import javafx.stage.FileChooser.ExtensionFilter;
import javafx.stage.Stage;
import javafx.util.Pair;
```

```
public class App extends Application {
    int N = 0, M = 0, P = 0, i;
    String s;
    ArrayList<boolean[][]> tetrominos = new ArrayList<boolean[][]>();
    ArrayList<TetrominoPage> tetrominoPages = new
    ArrayList<TetrominoPage>();
    boolean[] isSaved = new boolean[26];
    String ConfigPath;
    boolean[][] Board;

    public String getGreeting() {
        return "Hello World!";
    }

    public void loadConfig(String path) {
        int n = -1, m = -1, p = -1;
        String S = "";
        ArrayList<String> UnprocessedTetrominos = new
        ArrayList<String>();

        boolean first = true;
        boolean second = true;
        String filePath = path;
        try (BufferedReader reader = new BufferedReader(new
        FileReader(filePath))) {
            String line;
            while ((line = reader.readLine()) != null) {
                if (first) {
                    String[] tokens = line.trim().split("\\s+");
                    try {
                        if (tokens.length != 3) {
                            Alert alert = new
                            Alert(Alert.AlertType.ERROR);
                            alert.setTitle("Error Dialog");
                            alert.setHeaderText("Invalid input format
: line input 1");
                            alert.setContentText("Please input the
correct format");
                            alert.showAndWait();
                            throw new
                            IllegalArgumentException("Invalid input format");
                        }
                        n = Integer.parseInt(tokens[0]);
                    } catch (NumberFormatException e) {
                        System.out.println("Number format error: " + e.getMessage());
                    }
                }
                if (second) {
                    m = Integer.parseInt(tokens[1]);
                }
                if (third) {
                    p = Integer.parseInt(tokens[2]);
                }
                first = false;
            }
        } catch (IOException e) {
            System.out.println("File read error: " + e.getMessage());
        }
    }
}
```

```

        m = Integer.parseInt(tokens[1]);
        p = Integer.parseInt(tokens[2]);
        first = false;
    } catch (NumberFormatException e) {
        Alert alert = new
Alert(Alert.AlertType.ERROR);
        alert.setTitle("Error Dialog");
        alert.setHeaderText("Invalid input format :
line input 1");
        alert.setContentText("Please input the
correct format");
        alert.showAndWait();
        throw new IllegalArgumentException("Invalid
input format");
    }
}
else if (second) {
    S = line;
    if (S.compareTo("DEFAULT") != 0 &&
S.compareTo("CUSTOM") != 0 && S.compareTo("PYRAMID") != 0) {
        Alert alert = new
Alert(Alert.AlertType.ERROR);
        alert.setTitle("Error Dialog");
        alert.setHeaderText("Invalid input format :
input line 2");
        alert.setContentText("Please input the
correct format");
        alert.showAndWait();
        throw new IllegalArgumentException("Invalid
input format");
    }
    second = false;
}
else {
    UnprocessedTetrominos.add(line);
}
}
} catch (IOException e) {
    e.printStackTrace();
}

if (p > 26) {
    Alert alert = new Alert(Alert.AlertType.ERROR);
    alert.setTitle("Error Dialog");
    alert.setHeaderText("Invalid input P");
}

```

```

        alert.setContentText("P must be less than or equal to
26");
        alert.showAndWait();
        throw new IllegalArgumentException("Invalid input P : P
must be less than or equal to 26");
    }

    System.out.println("n : " + n);
    System.out.println("m : " + m);
    System.out.println("p : " + p);
    System.out.println("s : " + S);
    System.out.println("UnprocessedTetrominos : ");
    for (String tetromino : UnprocessedTetrominos) {
        System.out.println(tetromino);
    }

    Tetromino[] localtetrominos =
ProcessTetrominos(UnprocessedTetrominos, p);
    System.out.println("ProcessedTetrominos : ");
    for (int i = 1; i <= p; i++) {
        System.out.println("Tetromino " + localtetrominos[i].id +
" : ");
        System.out.println("n : " + localtetrominos[i].n);
        System.out.println("m : " + localtetrominos[i].m);
        localtetrominos[i].printTetromino();
    }

    int max_length = 0;
    for (int i = 1; i <= p; i++) {
        max_length = Math.max(max_length,
Math.max(localtetrominos[i].n, localtetrominos[i].m));
    }
    if (max_length > Math.max(n, m)) {
        Alert alert = new Alert(Alert.AlertType.ERROR);
        alert.setTitle("Error Dialog");
        alert.setHeaderText("Invalid Tetromino");
        alert.setContentText("Tetromino is too big");
        alert.showAndWait();
        throw new IllegalArgumentException("Invalid Tetromino :
Tetromino is too big");
    }

    String configPath =
Paths.get("input.txt").toAbsolutePath().toString();
    System.out.println("configPath : " + configPath);

```

```

        File configFile = new File(configPath);
        try (FileWriter writer = new FileWriter(configFile)) {
            writer.write(n + " " + m + " " + p + "\n");
            writer.write("DEFAULT\n");

            for (int i = 1; i <= p; i++) {
                for (int j = 1; j <= localtetrominos[i].n; j++) {
                    for (int k = 1; k <= localtetrominos[i].m; k++) {
                        if (localtetrominos[i].get(j, k)) {
                            writer.write((char)('A' +
localtetrominos[i].id - 1));
                        }
                        else {
                            writer.write(" ");
                        }
                    }
                    writer.write("\n");
                }
            }

            System.out.println("Config saved at: " +
configFile.getAbsolutePath());
        } catch (IOException e) {
            e.printStackTrace();
        }

        startBruteforce();
    }

    public void writeConfig() {
        String configPath =
Paths.get("input.txt").toAbsolutePath().toString();

        File configFile = new File(configPath);
        try (FileWriter writer = new FileWriter(configFile)) {
            writer.write(this.N + " " + this.M + " " + this.P +
"\n");
            writer.write("DEFAULT\n");

            for (int i = 0; i < this.P; i++) {
                for (int j = 0; j < tetrominos.get(i).length; j++) {
                    for (int k = 0; k < tetrominos.get(i)[j].length;
k++) {

```

```

                if (tetrominos.get(i)[j][k]) {
                    writer.write((char)('A' + i));
                }
                else {
                    writer.write(" ");
                }
            }
            writer.write("\n");
        }
    }

    System.out.println("Config saved at: " +
configFile.getAbsolutePath());
} catch (IOException e) {
    e.printStackTrace();
}
}

class TetrominoPage {
    int n, m;
    boolean[][] arr;
    int id;
    Button openTetrominoConfig;
    Button submitTetromino;
    TextField nField;
    TextField mField;
    Stage tetrominoStage;
    GridPane tetrominoGrid;
    VBox tetrominoLayout;
    Scene tetrominoScene;

    //constructor
    public TetrominoPage(int id) {
        this.n = 5;
        this.m = 5;
        this.arr = new boolean[5][5];
        for (int i = 0; i < 5; i++) {
            for (int j = 0; j < 5; j++) {
                this.arr[i][j] = false;
            }
        }
        this.id = id;

        Label nLabel = new Label("N : ");

```

```

nField = new TextField();
nField.setText(" " + this.n);
Label nLabel = new Label("N : ");
nLabel.setAlignment(Pos.CENTER);
Label mLabel = new Label("M : ");
mLabel.setAlignment(Pos.CENTER);
TextField mField = new TextField();
mField.setText(" " + this.m);

HBox n_HBox = new HBox(10, nLabel, nField);
n_HBox.setAlignment(Pos.CENTER);
HBox m_HBox = new HBox(10, mLabel, mField);
m_HBox.setAlignment(Pos.CENTER);
Button saveSize = new Button("Save Size");
saveSize.setOnAction(e -> {
    try {
        int befn = this.n;
        int befm = this.m;

        this.n = Integer.parseInt(nField.getText());
        this.m = Integer.parseInt(mField.getText());

        //change the array
        boolean[][] temp_arr = new
boolean[this.n][this.m];
        for (int i = 0; i < this.n; i++) {
            for (int j = 0; j < this.m; j++) {
                if (i < befn && j < befm) {
                    temp_arr[i][j] = this.arr[i][j];
                }
                else {
                    temp_arr[i][j] = false;
                }
            }
        }
        this.arr = temp_arr;

        //refresh the page
        this.tetrominoStage.close();
        this.tetrominoStage = null;

        this.openTetrominoConfig.fire();
    } catch (NumberFormatException ex) {
        System.out.println("Invalid input format");
    }
});
```

```

VBox NMBox = new VBox(10, n_HBox, m_HBox, saveSize);
NMBox.setStyle("-fx-padding: 20; -fx-alignment: center;
-fy-alignment: center;");

this.openTetrominoConfig = new Button("Block " + (id+1));
this.openTetrominoConfig.setOnAction(event -> {
    this.tetrominoGrid = new GridPane();
    int tetrominoRows = this.n;
    int tetrominoCols = this.m;

    for (int j = 0; j < tetrominoRows; j++) {
        for (int k = 0; k < tetrominoCols; k++) {
            Button cellButton = new Button();
            cellButton.setMinSize(30, 30);
            updateCellAppearance(cellButton,
this.arr[j][k]);
            int row = j;
            int col = k;
            cellButton.setOnAction(cellButtonAction -> {
                this.arr[row][col] = !this.arr[row][col];
                updateCellAppearance(cellButton,
this.arr[row][col]);
            });
            this.tetrominoGrid.add(cellButton, k, j);
        }
    }

    this.submitTetromino = new Button("Save Tetromino");
    this.submitTetromino.setOnAction(submitEvent -> {
        tетrominos.set(id, this.arr);
        isSaved[id] = true;
        this.tetrominoStage.close();
    });

    this.tetrominoLayout = new VBox(20, NMBox,
this.tetrominoGrid, this.submitTetromino);
    this.tetrominoLayout.setStyle("-fx-padding: 20;");
    this.tetrominoLayout.setAlignment(Pos.CENTER);
    NMBox.setAlignment(Pos.CENTER);
    tetrominoGrid.setAlignment(Pos.CENTER);

    this.tetrominoScene = new Scene(this.tetrominoLayout,
400, 400);

    this.tetrominoStage = new Stage();

```

```

        this.tetrominoStage.setScene(tetrominoScene);
        this.tetrominoStage.setTitle("Tetromino " + (id+1));
        this.tetrominoStage.show();
    });
}

//destructor
void destroy() {
    System.out.println("Destroying TetrominoPage: " + id);

    // Close the stage (if open)
    if (tetrominoStage != null) {
        tetrominoStage.close();
    }

    // Nullify references to help garbage collection
    openTetrominoConfig = null;
    nField = null;
    mField = null;
    arr = null;
}
}

private void updateCellAppearance(Button cellButton, boolean state) {
    if (state) {
        cellButton.setStyle("-fx-background-color: #0000FF;
-fx-border-color: black;");
    } else {
        cellButton.setStyle("-fx-background-color: #ADD8E6;
-fx-border-color: black;");
    }
}

@Override
public void start(Stage primaryStage) {
    //startBruteforce();

    for (int now = 0; now < 26; now++) {
        isSaved[now] = false;
    }

    Button uploadConfig = new Button ("Upload Config");
    Label fileLabel = new Label("No file selected");
    uploadConfig.setOnAction(e -> {

```

```

        FileChooser fileChooser = new FileChooser();
        fileChooser.getExtensionFilters().add(new
ExtensionFilter("Text Files", "*.txt"));

        File file = fileChooser.showOpenDialog(primaryStage);
        if (file != null) {
            fileLabel.setText("Selected File: " +
file.getName());
            System.out.println("File Path: " +
file.getAbsolutePath());

            ConfigPath = file.getAbsolutePath();
        }
        else {
            fileLabel.setText("No file selected");
        }
    });

GridPane BoardGridPane = new GridPane();

Button load = new Button("Load Config");
load.setOnAction(e -> {
    if (ConfigPath == null) {
        Alert alert = new Alert(Alert.AlertType.ERROR);
        alert.setTitle("Error Dialog");
        alert.setHeaderText("Invalid input file");
        alert.setContentText("Please upload the config file
first");
        alert.showAndWait();
        throw new IllegalArgumentException("You haven't
upload the config file, please upload the config file first");
    }

    this.loadConfig(ConfigPath);

});

```

Button startButton = new Button("Start Finding Solution!");  
startButton.setOnAction(e -> {  
 this.writeConfig();  
  
//alert if not all tetromino is saved  
for (int now = 0; now < P; now++) {  
 if (!isSaved[now]) {  
 Alert alert = new Alert(Alert.AlertType.ERROR);

```

        alert.setTitle("Error Dialog");
        alert.setHeaderText("Invalid input P");
        alert.setContentText("Please save all
tetrominos");
        alert.showAndWait();
        throw new IllegalArgumentException("You haven't
save all tetromino, please save all tetrominos");
    }
}

//start the bruteforce algorithm
startBruteforce();
});

// HBox gridWrapper = new HBox(gridPane);
// gridWrapper.setAlignment(javafx.geometry.Pos.CENTER);

TextField nField = new TextField("N : " + N);
Label nLabel = new Label("N : ");
TextField mField = new TextField("M : " + M);
Label mLabel = new Label("M : ");
TextField pField = new TextField("P : " + P);
Label pLabel = new Label("P : ");
Button SubmitNMP = new Button("Submit");

HBox TetrominoHBox = new HBox(10);

SubmitNMP.setOnAction(e -> {
    try {
        int befP = P;
        N = Integer.parseInt(nField.getText());
        M = Integer.parseInt(mField.getText());
        P = Integer.parseInt(pField.getText());

        if (P > 26) {
            Alert alert = new Alert(Alert.AlertType.ERROR);
            alert.setTitle("Error Dialog");
            alert.setHeaderText("Invalid input P");
            alert.setContentText("P must be less than or
equal to 26");
            alert.showAndWait();
            throw new IllegalArgumentException("Invalid input
P : P must be less than or equal to 26");
        }
    }
}

```

```

        if (befP < P) {
            for (i = befP; i < P; i++) {
                tetrominos.add(new boolean[5][5]);
                tetrominoPages.add(new TetrominoPage(i));

TetrominoHBox.getChildren().add(tetrominoPages.get(i).openTetrominoCo
nfig);
        }
    }
    else {
        System.out.println("P : " + P);
        System.out.println("befP : " + befP);
        for (int now = befP-1; now >= P; now--) {
            System.out.println("now : " + now);

TetrominoHBox.getChildren().remove(TetrominoHBox.getChildren().size()
-1);
            tetrominos.remove(tetrominos.size()-1);
            tetrominoPages.get(now).destroy();
            tetrominoPages.remove(now);
        }
    }

    if (N >= 0 && M >= 0) {
        BoardGridPane.getChildren().clear();
        BoardGridPane.setStyle("-fx-padding: 20;
-fx-alignment: center;");
        Board = new boolean[N][M];
        for (int j = 0; j < N; j++) {
            for (int k = 0; k < M; k++) {
                Button cellButton = new Button();
                cellButton.setMinSize(30, 30);
                updateCellAppearance(cellButton,
Board[j][k]);
                int row = j;
                int col = k;
                cellButton.setOnAction(cellButtonAction
-> {
                    Board[row][col] = !Board[row][col];
                    updateCellAppearance(cellButton,
Board[row][col]);
                });
                BoardGridPane.add(cellButton, k, j);
            }
        }
    }
}

```

```

        }
    } catch (NumberFormatException ex) {
        System.out.println("Invalid input format");
    }
});

HBox n_HBox = new HBox(10, nLabel, nField);
HBox m_HBox = new HBox(10, mLabel, mField);
HBox p_HBox = new HBox(10, pLabel, pField);

Label ConfigureYourPuzzle = new Label("Configure Your
Puzzle!");
ConfigureYourPuzzle.setStyle("-fx-font-size: 20;
-fx-font-weight: bold; -fx-alignment: center;");

VBox NMPField = new VBox(10, n_HBox, m_HBox, p_HBox);
HBox NPMWrapper = new HBox(40, NMPField, SubmitNMP);
NPMWrapper.setStyle("-fx-padding: 20; -fx-alignment: center;
-fy-alignment: center;");

VBox layout = new VBox(20, uploadConfig, load, fileLabel,
NPMWrapper, ConfigureYourPuzzle, BoardGridPane, TetrominoHBox,
startButton);
layout.setStyle("-fx-padding: 20; -fx-alignment: center;");

TetrominoHBox.setAlignment(Pos.CENTER);

// Set up the stage
Scene scene = new Scene(layout, 800, 800);
primaryStage.setScene(scene);
primaryStage.setTitle("IQ-Puzzler Pro Solver");
primaryStage.show();
}

public static boolean isCapitalLetter(char c) {
    return c >= 'A' && c <= 'Z';
}

public static boolean isSameTetromino(String a, String b) {
    char aChar = '$', bChar = '$';
    for (int i = 0; i < a.length(); i++) {
        if (isCapitalLetter(a.charAt(i))) {
            if (aChar != '$' && aChar != a.charAt(i)) {
                throw new IllegalArgumentException("Invalid
Tetromino : Exist different characters in the same Tetromino");

```

```

        }
        else aChar = a.charAt(i);
    }
}
for (int i = 0; i < b.length(); i++) {
    if (b.charAt(i) != ' ') {
        if (bChar != '$' && bChar != b.charAt(i)) {
            throw new IllegalArgumentException("Invalid
Tetromino : Exist different characters in the same Tetromino");
        }
        else bChar = b.charAt(i);
    }
}
return aChar == bChar;
}

public static char getCharID(String s) {
    for (int i = 0; i < s.length(); i++) {
        if (isCapitalLetter(s.charAt(i))) {
            return s.charAt(i);
        }
    }
    throw new IllegalArgumentException("Invalid Tetromino : No
capital letter found");
}

public static Tetromino processTetromino(List<String> tetromino)
{
    int id = 0;
    int n = tetromino.size();
    int m = 0;
    for (int i = 0; i < n; i++) {
        m = Math.max(m, tetromino.get(i).length());
    }

    int rootn = -1, rootm = -1;

    boolean[][] arr = new boolean[n][m];
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            if (j < tetromino.get(i).length()) {
                arr[i][j] =
isCapitalLetter(tetromino.get(i).charAt(j));
                if (isCapitalLetter(tetromino.get(i).charAt(j)))

```

```

{
    rootn = i;
    rootm = j;
    id = tetromino.get(i).charAt(j) - 'A' + 1;
}
else if (tetromino.get(i).charAt(j) != ' ') {
    throw new IllegalArgumentException("Invalid
Tetromino : Found invalid character");
}
else {
    arr[i][j] = false;
}
}

if (rootn == -1 || rootm == -1) {
    throw new IllegalArgumentException("Invalid Tetromino :
Found empty tetromino");
}

//Check wether given tetromino is connected or not
boolean[][] visited = new boolean[n][m];
for (int i = 0; i < n; i++) {
    for (int j = 0; j < m; j++) {
        visited[i][j] = false;
    }
}

Queue<Pair<Integer, Integer>> q = new
LinkedList<Pair<Integer, Integer>>();
q.add(new Pair<Integer, Integer>(rootn, rootm));
visited[rootn][rootm] = true;

while (!q.isEmpty()) {
    Pair<Integer, Integer> p = q.poll();
    int x = p.getKey();
    int y = p.getValue();

    if (x > 0 && !visited[x-1][y] && arr[x-1][y]) {
        q.add(new Pair<Integer, Integer>(x-1, y));
        visited[x-1][y] = true;
    }
    if (x < n-1 && !visited[x+1][y] && arr[x+1][y]) {
        q.add(new Pair<Integer, Integer>(x+1, y));
    }
}
}

```

```

        visited[x+1][y] = true;
    }
    if (y > 0 && !visited[x][y-1] && arr[x][y-1]) {
        q.add(new Pair<Integer, Integer>(x, y-1));
        visited[x][y-1] = true;
    }
    if (y < m-1 && !visited[x][y+1] && arr[x][y+1]) {
        q.add(new Pair<Integer, Integer>(x, y+1));
        visited[x][y+1] = true;
    }

    //check diagonal
    if (x > 0 && y > 0 && !visited[x-1][y-1] &&
arr[x-1][y-1]) {
        q.add(new Pair<Integer, Integer>(x-1, y-1));
        visited[x-1][y-1] = true;
    }

    if (x > 0 && y < m-1 && !visited[x-1][y+1] &&
arr[x-1][y+1]) {
        q.add(new Pair<Integer, Integer>(x-1, y+1));
        visited[x-1][y+1] = true;
    }

    if (x < n-1 && y > 0 && !visited[x+1][y-1] &&
arr[x+1][y-1]) {
        q.add(new Pair<Integer, Integer>(x+1, y-1));
        visited[x+1][y-1] = true;
    }

    if (x < n-1 && y < m-1 && !visited[x+1][y+1] &&
arr[x+1][y+1]) {
        q.add(new Pair<Integer, Integer>(x+1, y+1));
        visited[x+1][y+1] = true;
    }
}

for (int i = 0; i < n; i++) {
    for (int j = 0; j < m; j++) {
        if (arr[i][j] && !visited[i][j]) {
            Alert alert = new Alert(Alert.AlertType.ERROR);
            alert.setTitle("Error Dialog");
            alert.setHeaderText("Invalid Tetromino");
            alert.setContentText("Tetromino is not
connected");
        }
    }
}

```

```

        alert.showAndWait();
        throw new IllegalArgumentException("Invalid
Tetromino : Tetromino is not connected");
    }
}

// System.out.println("n : " + n);
// System.out.println("m : " + m);
// System.out.println("arr : ");
// for (int i = 0; i < n; i++) {
//     for (int j = 0; j < m; j++) {
//         System.out.print(arr[i][j] + " ");
//     }
//     System.out.println();
// }

return new Tetromino(n, m, arr, id);
}

public static Tetromino[] ProcessTetrominos(ArrayList<String>
UnprocessedTetrominos, int tetrominoCount) {
    boolean[] isUsed = new boolean[26];
    for (int i = 0; i < 26; i++) {
        isUsed[i] = false;
    }
    Tetromino[] tetrominos = new Tetromino[tetrominoCount + 1];

    int count = 1;
    for (int i = 0; i < UnprocessedTetrominos.size(); i++) {
        boolean isBlank = true;
        for (int j = 0; j <
UnprocessedTetrominos.get(i).length(); j++) {
            if (UnprocessedTetrominos.get(i).charAt(j) != ' ') {
                isBlank = false;
                break;
            }
        }
        if (isBlank) continue;
        int j = i;

        int len_max = 0;
        while (isSameTetromino(UnprocessedTetrominos.get(i),
UnprocessedTetrominos.get(j))) {

```

```

        len_max = Math.max(len_max,
UnprocessedTetrominos.get(j).length());
        j++;
        if (j == UnprocessedTetrominos.size()) break;
    }

    //delete suffix
    while (true) {
        boolean blank = true;

        for (int k = i; k < j; k++) {
            if (UnprocessedTetrominos.get(k).length() <
len_max) continue;
            else {
                if
(UnprocessedTetrominos.get(k).charAt(len_max-1) != ' ') {
                    blank = false;
                    break;
                }
            }
        }

        if (blank) {
            for (int k = i; k < j; k++) {
                UnprocessedTetrominos.set(k,
UnprocessedTetrominos.get(k).substring(0, len_max-1));
            }
            len_max--;
        }
        else break;
    }

    //delete prefix
    while (true) {
        boolean blank = true;

        for (int k = i; k < j; k++) {
            if (UnprocessedTetrominos.get(k).length() == 0)
continue;
            else {
                if (UnprocessedTetrominos.get(k).charAt(0) !=
' ') {
                    blank = false;
                    break;
                }
            }
        }
    }
}

```

```

        }
    }

    if (blank) {
        for (int k = i; k < j; k++) {
            UnprocessedTetrominos.set(k,
UnprocessedTetrominos.get(k).substring(1));
        }
        len_max--;
    }
    else break;
}

System.out.println(tetrominoCount + " " + count + " : " +
i + " " + j);

char tetrominoID =
getCharID(UnprocessedTetrominos.get(i));
if (isUsed[tetrominoID - 'A']) {
    throw new IllegalArgumentException("Invalid Tetromino
: Multiple tetromino with same alphabet code");
}

if (count > tetrominoCount) {
    throw new IllegalArgumentException("Invalid Tetromino
: Too many tetrominos");
}
tetrominos[count++] =
processTetromino(UnprocessedTetrominos.subList(i, j));
isUsed[tetrominoID - 'A'] = true;

i = j-1;
}

if (count - 1 != tetrominoCount) {
    throw new IllegalArgumentException("Invalid Tetromino :
Number of tetrominos does not match with the given number");
}

return tetrominos;
}

public void startBruteforce() {

```

```
int n = -1, m = -1, p = -1;
String s = "";
ArrayList<String> UnprocessedTetrominos = new
ArrayList<String>();

boolean first = true;
boolean second = true;
String filePath = "input.txt";
try (BufferedReader reader = new BufferedReader(new
FileReader(filePath))) {
    String line;
    while ((line = reader.readLine()) != null) {
        if (first) {
            String[] tokens = line.trim().split("\\s+");
            try {
                if (tokens.length != 3) {
                    throw new
IllegalArgumentException("Invalid input format");
                }
                n = Integer.parseInt(tokens[0]);
                m = Integer.parseInt(tokens[1]);
                p = Integer.parseInt(tokens[2]);
                first = false;
            } catch (NumberFormatException e) {
                throw new IllegalArgumentException("Invalid
input format");
            }
        }
        else if (second) {
            s = line;
            second = false;
        }
        else {
            UnprocessedTetrominos.add(line);
        }
    }
} catch (IOException e) {
    e.printStackTrace();
}

System.out.println("n : " + n);
System.out.println("m : " + m);
System.out.println("p : " + p);
System.out.println("s : " + s);
```

```

System.out.println("UnprocessedTetrominos : ");
for (String tetromino : UnprocessedTetrominos) {
    System.out.println(tetromino);
}

Tetromino[] tetrominos =
ProcessTetrominos(UnprocessedTetrominos, p);
System.out.println("ProcessedTetrominos : ");
for (int i = 1; i <= p; i++) {
    System.out.println("Tetromino " + tetrominos[i].id + " :
");
    System.out.println("n : " + tetrominos[i].n);
    System.out.println("m : " + tetrominos[i].m);
    tetrominos[i].printTetromino();
}

int max_length = 0;
for (int i = 1; i <= p; i++) {
    max_length = Math.max(max_length,
Math.max(tetrominos[i].n, tetrominos[i].m));
}
if (max_length > Math.max(n, m)) {
    Alert alert = new Alert(Alert.AlertType.ERROR);
    alert.setTitle("Error Dialog");
    alert.setHeaderText("Invalid Tetromino");
    alert.setContentText("Tetromino is too big");
    alert.showAndWait();
    throw new IllegalArgumentException("Invalid Tetromino :
Tetromino is too big");
}

int totalarea = 0;
for (int i = 1; i <= p; i++) {
    for (int j = 1; j <= tetrominos[i].n; j++) {
        for (int k = 1; k <= tetrominos[i].m; k++) {
            if (tetrominos[i].get(j, k)) {
                totalarea++;
            }
        }
    }
}

int boardarea = 0;
for (int i = 0; i < n; i++) {
    for (int j = 0; j < m; j++) {

```

```

        if (!Board[i][j]) {
            boardarea++;
        }
    }

    //validate total area
    if (totalarea != boardarea) {
        Alert alert = new Alert(Alert.AlertType.ERROR);
        alert.setTitle("Error Dialog");
        alert.setHeaderText("Invalid Tetromino");
        alert.setContentText("Total area of tetrominos does not
match with the given area, " + totalarea + " vs " + n*m);
        alert.showAndWait();
        throw new IllegalArgumentException("Invalid Tetromino :
Total area of tetrominos does not match with the given area, " +
totalarea + " vs " + n*m);
    }

    Block block = new Block(tetrominos, n, m, p, Board);
    block.solve();
}

public static void main(String[] args) {
    launch(args);
}
}

```

### 3.3.2. Tetromino.java

```

package org.example;

public class Tetromino {
    int n, m;
    boolean[][] arr;
    int id;
    static int count = 1;

    public Tetromino(int n, int m, boolean arr[][][]) {
        this.n = n;
        this.m = m;
        this.arr = new boolean[n+2][m+2];
    }
}

```

```

        this.id = count++;

        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= m; j++) {
                this.arr[i][j] = arr[i-1][j-1];
            }
        }
    }

    public Tetromino(int n, int m, boolean arr[][][], int id) {
        this.n = n;
        this.m = m;
        this.arr = new boolean[n+2][m+2];
        this.id = id;

        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= m; j++) {
                this.arr[i][j] = arr[i-1][j-1];
            }
        }
    }

    public void set(int x, int y, boolean value) {
        this.arr[x][y] = value;
    }

    public boolean get(int x, int y) {
        return this.arr[x][y];
    }

    public boolean isValidPosition(int x, int y) {
        return x >= 1 && x <= this.n && y >= 1 && y <= this.m;
    }

    public Tetromino rotate() {
        boolean[][] newArr = new boolean[this.m][this.n];
        for (int i = 1; i <= this.m; i++) {
            for (int j = 1; j <= this.n; j++) {
                newArr[i-1][j-1] = this.arr[j][this.m-i+1];
            }
        }
        return new Tetromino(this.m, this.n, newArr, this.id);
    }

    public Tetromino flip() {
        boolean[][] newArr = new boolean[this.n][this.m];
        for (int i = 1; i <= this.n; i++) {
            for (int j = 1; j <= this.m; j++) {
                newArr[i-1][j-1] = this.arr[i][this.m-j+1];
            }
        }
    }
}

```

```

        }
    }
    return new Tetromino(this.n, this.m, newArr, this.id);
}

public void printTetromino() {
    for (int i = 1; i <= this.n; i++) {
        for (int j = 1; j <= this.m; j++) {
            if (this.arr[i][j]) {
                System.out.print("#");
            } else {
                System.out.print(" ");
            }
        }
        System.out.println();
    }
}
}

```

### 3.3.3. Block.java

```

package org.example;
import java.awt.Color;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;

import javax.imageio.ImageIO;

import javafx.scene.Scene;
import javafx.scene.SnapshotParameters;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.image.PixelFormat;
import javafx.scene.image.PixelReader;
import javafx.scene.image.WritableImage;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.StackPane;
import javafx.scene.layout.VBox;
import javafx.scene.shape.Rectangle;
import javafx.scene.text.Font;
import javafx.scene.text.FontWeight;
import javafx.scene.text.Text;

```

```
import javafx.scene.text.TextAlignment;
import javafx.stage.Stage;

public class Block {
    int n, m, tetrominoCount;
    int[][] arr;
    String[] color;
    int[] RGBinteger;
    Tetromino[] tetrominos;
    boolean foundSolution = false;
    int bruteForceCount = 0;
    long startTime;
    long endTime;

    public Block(Tetromino[] tetrominos, int n, int m, int tetrominoCount,
    boolean arr[][][]) {
        this.n = n;
        this.m = m;
        this.tetrominoCount = tetrominoCount;
        this.tetrominos = new Tetromino[tetrominoCount+2];

        this.arr = new int[n+2][m+2];
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= m; j++) {
                if (!arr[i-1][j-1]) {
                    this.arr[i][j] = -1;
                } else {
                    this.arr[i][j] = -2;
                }
            }
        }

        for (int i = 1; i <= tetrominoCount; i++) {
            this.tetrominos[i] = tetrominos[i];
        }

        this.color = new String[27];
        this.RGBinteger = new int[27];
        for (int i = 1; i <= 26; i++) {
            int j = (5*i % 26) + 1;
            float hue = (float) j / 26;
            Color tempcolor = Color.getHSBColor(hue, 1.0f, 1.0f);

            this.color[i] = "\u001B[38;2;" + tempcolor.getRed() + ";" +
tempcolor.getGreen() + ";" + tempcolor.getBlue() + "m";
            this.RGBinteger[i] = tempcolor.getRGB();
        }
    }
}
```

```
public void set(int x, int y, int value) {
    arr[x][y] = value;
}

public int get(int x, int y) {
    return arr[x][y];
}

public boolean isValidPosition(int x, int y) {
    return x >= 1 && x <= n && y >= 1 && y <= m && arr[x][y] == -1;
}

public boolean placeTetromino(Tetromino tetromino, int a, int b, int x,
int y) {
    boolean isPlaced = false;
    for (int i = 1; i <= tetromino.n; i++) {
        for (int j = 1; j <= tetromino.m; j++) {
            if (tetromino.get(i, j)) {
                if (!isValidPosition(x+i-1, y+j-1)) {
                    return false;
                }
            }
            if (x+i-1 == a && y+j-1 == b) {
                if (tetromino.get(i, j)) {
                    isPlaced = true;
                }
            }
        }
    }
    if (!isPlaced) {
        return false;
    }

    for (int i = 1; i <= tetromino.n; i++) {
        for (int j = 1; j <= tetromino.m; j++) {
            if (tetromino.get(i, j)) {
                set(x+i-1, y+j-1, tetromino.id);
            }
        }
    }
    return true;
}

public void removeTetromino(Tetromino tetromino, int x, int y) {
    for (int i = 1; i <= tetromino.n; i++) {
```

```

        for (int j = 1; j <= tетromino.m; j++) {
            if (tетromino.get(i, j)) {
                set(x+i-1, y+j-1, -1);
            }
        }
    }

public void printBlock() {
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= m; j++) {
            if (arr[i][j] < 0) {
                System.out.print("#");
            } else {
                System.out.print(color[arr[i][j]] + (char)('A' + arr[i][j] - 1) +
"\u001B[0m");
            }
        }
        System.out.println();
    }

//make a new stage
GridPane grid = new GridPane();
grid.setHgap(5.0);
grid.setVgap(5.0);

for (int i = 1; i <= n; i++) {
    for (int j = 1; j <= m; j++) {
        if (arr[i][j] < 0) {
            Rectangle rect = new Rectangle(50, 50);
            rect.setFill(javafx.scene.paint.Color.BLACK);
            rect.setStroke(javafx.scene.paint.Color.BLACK);
            rect.setStrokeWidth(1.0);
            grid.add(rect, j-1, i-1);
            continue;
        }
        Rectangle rect = new Rectangle(50, 50);
        Color tempcolor = new Color(RGBinteger[arr[i][j]]);

        javafx.scene.paint.Color FXcolor =
javafx.scene.paint.Color.rgb(tempcolor.getRed(), tempcolor.getGreen(),
tempcolor.getBlue());
        rect.setFill(FXcolor);

        rect.setStroke(javafx.scene.paint.Color.BLACK);
        rect.setStrokeWidth(1.0);

        Text text = new Text(25, 25, (char)('A' + arr[i][j] - 1) + "");
    }
}

```

```

        text.setAlignment(TextAlignment.CENTER);
        text.setFont(Font.font("Arial", FontWeight.BOLD, 20));
        StackPane Spane = new StackPane(rect, text);
        Spane.setStyle("-fx-alignment: center; -fy-alignment: center;");

        grid.add(Spane, j-1, i-1);
    }
}

this.endTime = System.currentTimeMillis();

Button SaveResult = new Button("Save as Image");
SaveResult.setOnAction(e -> {
    WritableImage image = new WritableImage((int) grid.getWidth(), (int)
grid.getHeight());
    grid.snapshot(new SnapshotParameters(), image);

    PixelReader pixelReader = image.getPixelReader();

    int width = (int) image.getWidth();
    int height = (int) image.getHeight();

    BufferedImage bufferedImage = new BufferedImage(width, height,
BufferedImage.TYPE_INT_ARGB);

    int[] pixels = new int[width * height];
    pixelReader.getPixels(0, 0, width, height,
PixelFormat.getIntArgbInstance(), pixels, 0, width);

    // Set pixel data
    bufferedImage.setRGB(0, 0, width, height, pixels, 0, width);

    File file = new File("result.png");
    try {
        ImageIO.write(bufferedImage, "png", file);
    } catch (IOException ex) {
        ex.printStackTrace();
    }

    String ResultPath = file.getAbsolutePath();

    //print path
    Alert alert = new Alert(Alert.AlertType.INFORMATION);
    alert.setTitle("Information Dialog");
    alert.setHeaderText(null);
    alert.setContentText("Image saved as result.png at : " + ResultPath);
    alert.showAndWait();
});

```

```

HBox GridHBox = new HBox(20, grid);
GridHBox.setStyle("-fx-padding:20px; -fx-alignment:center; -fy-alignment:center;");

Label resultLabel = new Label("Execution time : " + (this.endTime - this.startTime) + " ms\nCase searched : " + this.bruteForceCount);
resultLabel.setStyle("-fx-font-size: 20px; -fx-font-weight: bold; -fx-text-fill: #000000;");

VBox ResultLayout = new VBox(20, resultLabel, GridHBox, SaveResult);
ResultLayout.setStyle("-fx-background-color:hsl(0, 0.00%, 100.00%); -fx-padding:20px; -fx-priority:20px; -fx-alignment: center; -fy-alignment:center;");

Scene resultScene = new Scene(ResultLayout, 600, 600);
Stage resultStage = new Stage();
resultStage.setScene(resultScene);
resultStage.show();
}

public void findSolution(int i, int j, boolean[] used) {
    if (i == n+1) {
        printBlock();
        this.foundSolution = true;
    }
    else {
        for (int cur_tetromino = 1; cur_tetromino <= this.tetrominoCount; cur_tetromino++) {
            if (!used[cur_tetromino]) {
                used[cur_tetromino] = true;
                Tetromino tetromino = this.tetrominos[cur_tetromino];
                for (int rot = 0; rot < 4; rot++) {
                    for (int flip = 0; flip < 2; flip++) {
                        for (int shiftx = -tetromino.n+1; shiftx <= tetromino.n-1; shiftx++) {
                            for (int shifty = -tetromino.m+1; shifty <= tetromino.m-1; shifty++) {
                                this.bruteForceCount++;
                                if (placeTetromino(tetromino, i, j, i+shiftx, j+shifty)) {
                                    int nexti = i;
                                    int nextj = j+1;

                                    while (nexti <= this.n) {
                                        while (nextj <= this.m && arr[nexti][nextj] != -1) {
                                            nextj++;
                                        }
                                        if (nextj > this.m) {
                                            nextj = 1;
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}

```

```

        nexti++;
    }
    else {
        break;
    }
}

findSolution(nexti, nextj, used);
if (this.foundSolution) return;
removeTetromino(tetromino, i+shiftx, j+shifty);
}
}
}
if (flip != 1) tetromino = tetromino.flip();
}
if (rot != 3) tetromino = tetromino.rotate();
}
used[cur_tetromino] = false;
}
}
}
}

public void solve() {
boolean[] used = new boolean[this.tetrominoCount+1];
this.foundSolution = false;
this.bruteForceCount = 0;
this.startTime = System.currentTimeMillis();

printBlock();

int posi = 1, posj = 1;
while (posi <= this.n) {
    while (posj <= this.m && arr[posi][posj] != -1) {
        posj++;
    }
    if (posj > this.m) {
        posj = 1;
        posi++;
    }
    else {
        break;
    }
}
System.out.println("Starting from " + posi + " " + posj);
findSolution(posi, posj, used);

System.out.println("Execution time : " + (this.endTime -
this.startTime) + " ms");

```

```
System.out.println("Case searched : " + this.bruteForceCount);
if (!this.foundSolution) {
    System.out.println("No solution found");

    this.endTime = System.currentTimeMillis();

    Label resultLabel = new Label("No Solution Found\nExecution time : "
+ (this.endTime - this.startTime) + " ms\nCase searched : " +
this.bruteForceCount);
    resultLabel.setStyle("-fx-font-size: 20px; -fx-font-weight: bold;
-fx-text-fill: #000000;");

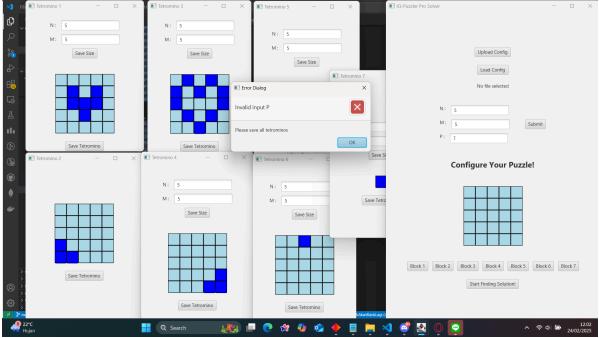
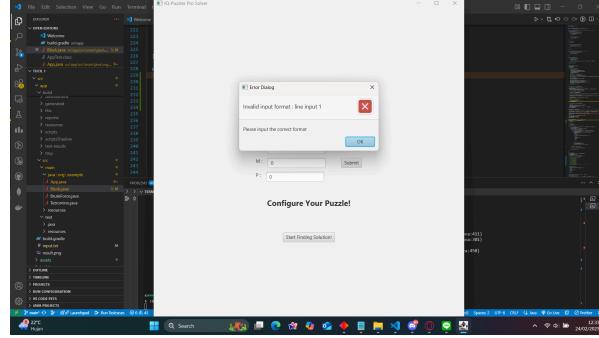
    VBox ResultLayout = new VBox(20, resultLabel);
    ResultLayout.setStyle("-fx-background-color:hsl(0, 0.00%, 100.00%);
-fy-padding:20px; -fx-padding:20px; -fx-alignment: center; -fy-alignment:
center;");

    Scene resultScene = new Scene(ResultLayout, 400, 400);
    Stage resultStage = new Stage();
    resultStage.setScene(resultScene);
    resultStage.show();
}
}
}
```

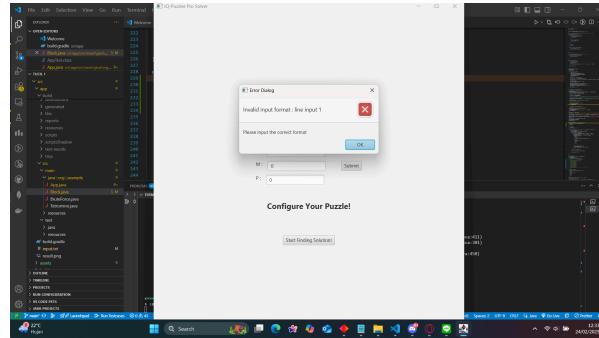
## BAB 4 : Pengujian

### 4.1. Kasus Invalid

#### 4.1.1. Config Manual : Kepingan Puzzle Belum Disimpan

Input	Output
	

#### 4.1.2. Config Input : Format Input Invalid

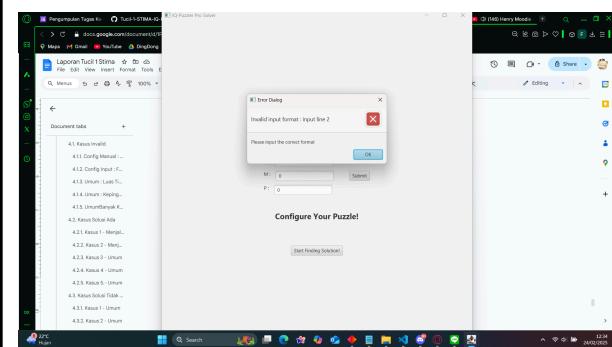
Input	Output
6 6 2 2 hahahahahih A A A A B B B B C C D D D E	

E	
E	
FFF	
F	
GG	
G	

Input	Output
-------	--------

5 5 7	
A	
AA	
A	
B	
B	
B	
B	
B	
C	
C	
C	
D	
D	
D	
E	
E	
E	
FFF	
F	
GG	
G	

Input	Output
-------	--------



#### 4.1.3. Umum : Luas Tidak Sesuai

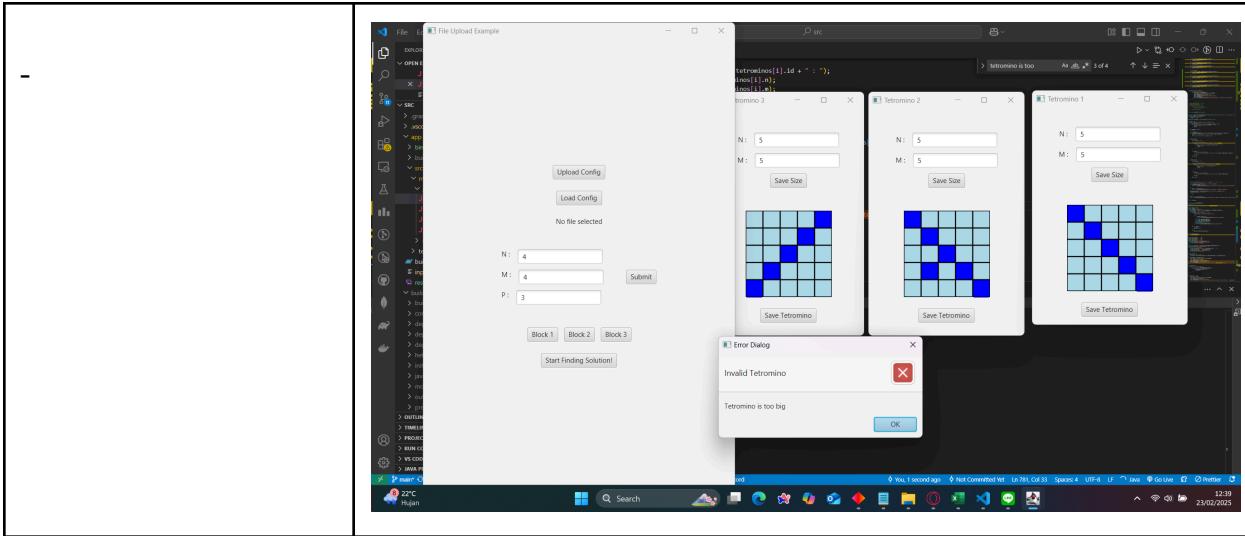
Input	Output
-------	--------

<pre> 5 5 5 DEFAULT AAAAA AAAAA BBBBB BBBBB CCCCC CCCCC DDDDD EEEE </pre>	<p>The screenshot shows the IQ-Puzzler Pro Solver application window. On the left is a file explorer-like sidebar with various project files and folders. In the center, there's a large dark area representing a puzzle board. A small error dialog box is open in the upper right corner, stating "Invalid Tetromino" and "Total area of tetrominos does not match with the given area, 40 vs 25". Below the dialog is a "Configure Your Puzzle!" section with input fields for M and P, and a "Start Finding Solution!" button.</p>
---	--

#### 4.1.4. Umum : Kepingan Puzzle Terlalu Besar

Input	Output
<pre> 5 5 5 DEFAULT AAAAAAA BBBBB CCCCC DDDDD EEE </pre>	<p>The screenshot shows the IQ-Puzzler Pro Solver application window. On the left is a file explorer-like sidebar with various project files and folders. In the center, there's a large dark area representing a puzzle board. A small error dialog box is open in the upper right corner, stating "Invalid Tetromino" and "Tetromino is too big". Below the dialog is a "Configure Your Puzzle!" section with input fields for M and P, and a "Start Finding Solution!" button.</p>

Input	Output
-------	--------



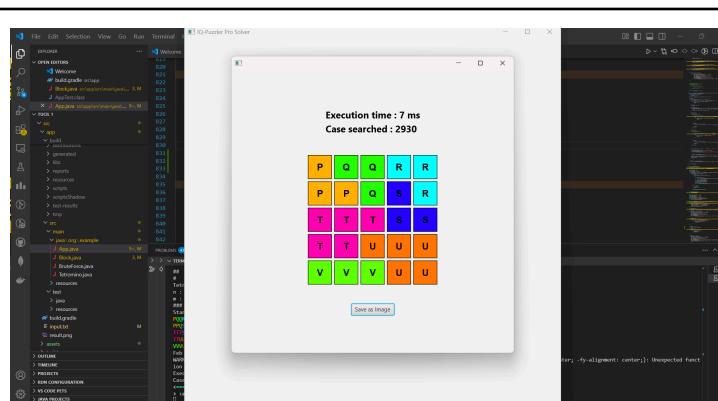
#### 4.1.5. Umum : Banyak Kepingan Puzzle Lebih dari 26

Input	Output
5 6 30	
DEFAULT	
A	
B	
C	
D	
E	
F	
G	
H	
I	
J	
K	
L	
M	
N	
O	
P	
Q	
R	
S	
T	

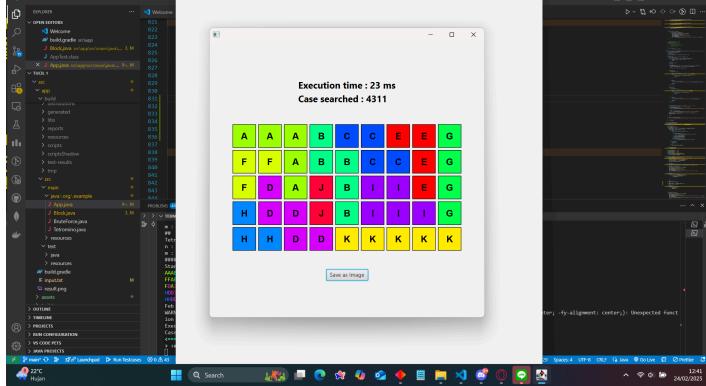
U V W X Y Z # \$ % &	
---	--

## 4.2. Kasus Solusi Ada

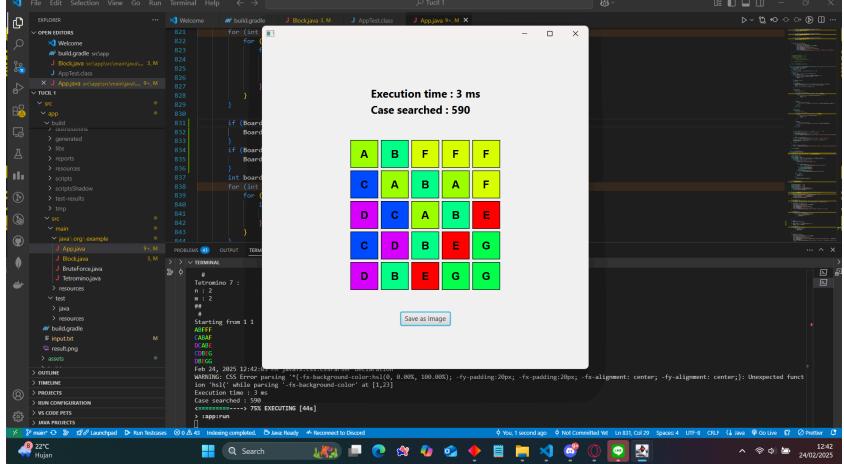
### 4.2.1. Kasus 1 - Umum

Input	Output
5 5 7	
DEFAULT	
P	
PP	
Q	
QQ	
R	
RR	
S	
SS	
TT	
TT	
T	
UU	
UU	
U	
VVV	

#### 4.2.2. Kasus 2 - Umum

Input	Output
5 9 11 DEFAULT AAA A A B BB B B CC CC DD DD D EE E E FF F G G G HH H I II II JJ KKKK	

#### 4.2.3. Kasus 3 - Umum

Input	Output																														
5 9 11 DEFAULT AAA A A B BB B B CC CC DD DD D EE E E FF F G G G HH H I II II JJ KKKK	 <p>The screenshot shows a Java IDE interface with several tabs open. The central part of the screen displays a 7x7 grid representing a Tetris board. The grid contains various colored blocks (A through G) representing different tetrominoes. The board state is as follows:</p> <table border="1"> <tr> <td>A</td><td>B</td><td>F</td><td>F</td><td>F</td> </tr> <tr> <td>C</td><td>A</td><td>B</td><td>A</td><td>F</td> </tr> <tr> <td>D</td><td>C</td><td>A</td><td>B</td><td>E</td> </tr> <tr> <td>C</td><td>D</td><td>B</td><td>E</td><td>G</td> </tr> <tr> <td>D</td><td>B</td><td>E</td><td>G</td><td>G</td> </tr> <tr> <td></td><td></td><td></td><td></td><td></td> </tr> </table> <p>Below the board, there is a message: "Save as image". The IDE's status bar at the bottom shows the date and time as "Feb 24, 2025 13:42:31" and "12:42".</p>	A	B	F	F	F	C	A	B	A	F	D	C	A	B	E	C	D	B	E	G	D	B	E	G	G					
A	B	F	F	F																											
C	A	B	A	F																											
D	C	A	B	E																											
C	D	B	E	G																											
D	B	E	G	G																											

#### 4.2.4. Kasus 4 - Umum

Input	Output

26 3 26

DEFAULT

AA

A

BB

B

CC

C

DD

D

E

EE

F

FF

GG

G

H

HH

II

I

JJ

J

K

KK

LL

L

M

MM

NN

N

OO

O

P

PP

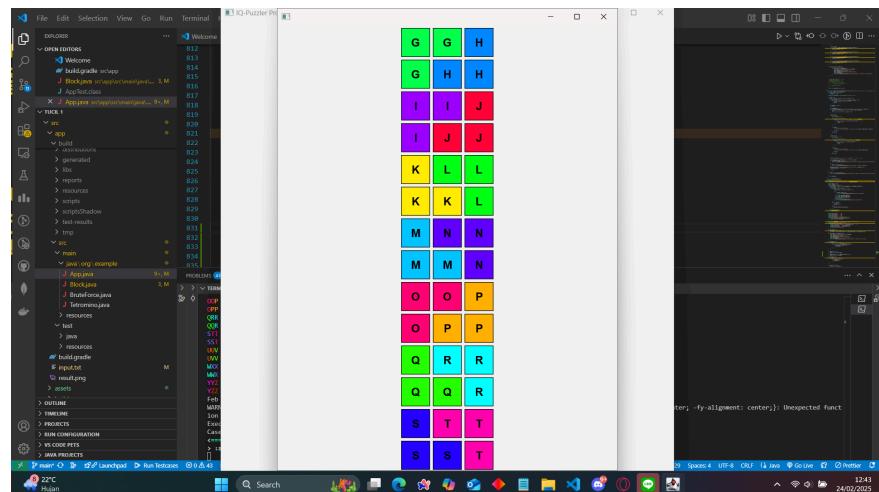
Q

QQ

R

RR

S



SS	
TT	
T	
UU	
U	
V	
VV	
W	
WW	
XX	
X	
YY	
Y	
Z	
ZZ	

10 12 8

DEFAULT

R

R

AAAAAAA

AAAAAAA

AAA

AAAAA

AAAAA

AAAAAAAAA

AAAAAAAAA

AAAAAAA

AAA AAA

AAA AAA

II

II

II

HHH

H

BBB

BBBBB

BBBBBB

KKKKK

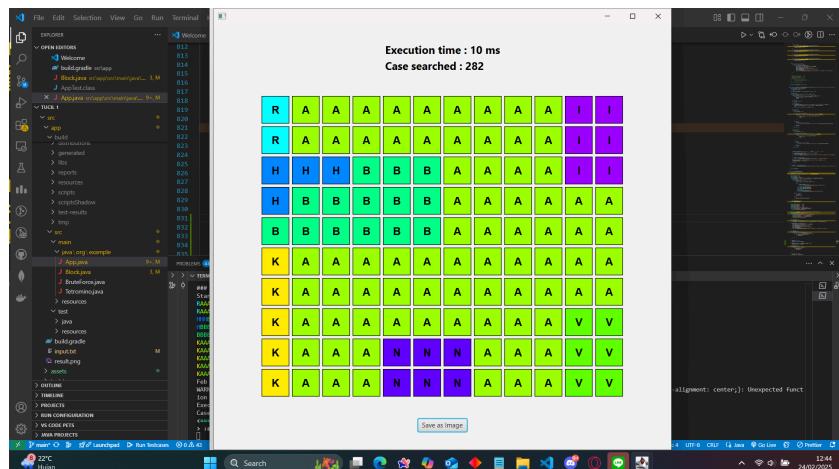
VV

VV

VV

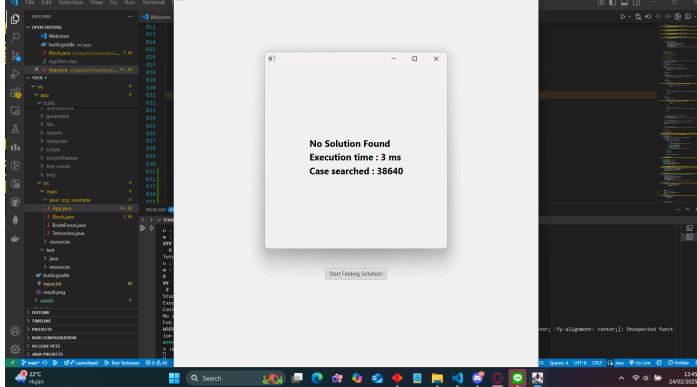
NNN

NNN



### 4.3. Kasus Solusi Tidak Ada

#### 4.3.1. Kasus 1 - Umum

Input	Output
4 4 4 DEFAULT AA AA BB BB CCC C D DD D	

#### 4.3.2. Kasus 2 - Umum

Input	Output
-------	--------

5 5 5

DEFAULT

AAA

AA

B

B

B

B

B

C

C

C

C

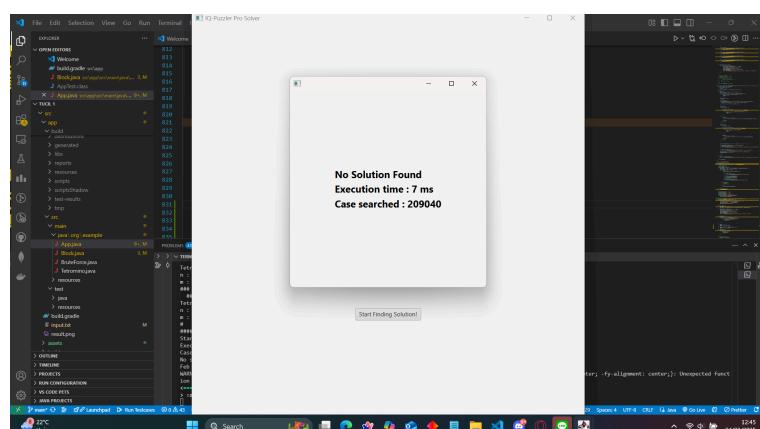
C

DDD

DD

E

EEEE



#### 4.3.3. Kasus 3 - Umum

Input	Output
-------	--------

6 6 6

DEFAULT

AAA

AAA

BBB

BBB

CCC

CCC

DDD

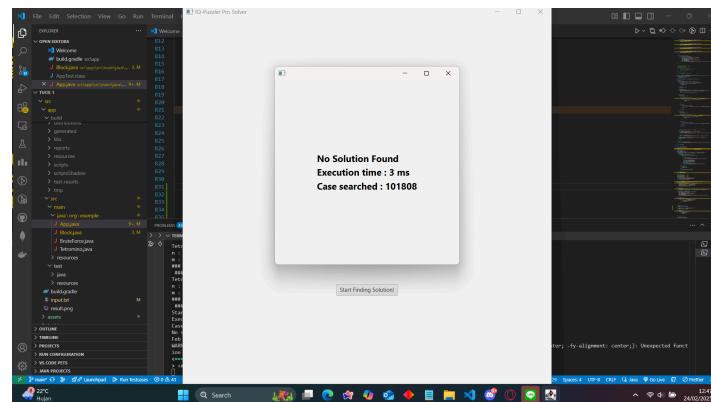
DDD

EEE

EEE

FFF

FFF



#### 4.3.4. Kasus 4 - Umum

Input	Output
-------	--------

7 7 7

DEFAULT

AAAA

A

A

A

BBBB

B

B

B

CCCC

C

C

C

DDDD

D

D

D

EEEE

E

E

E

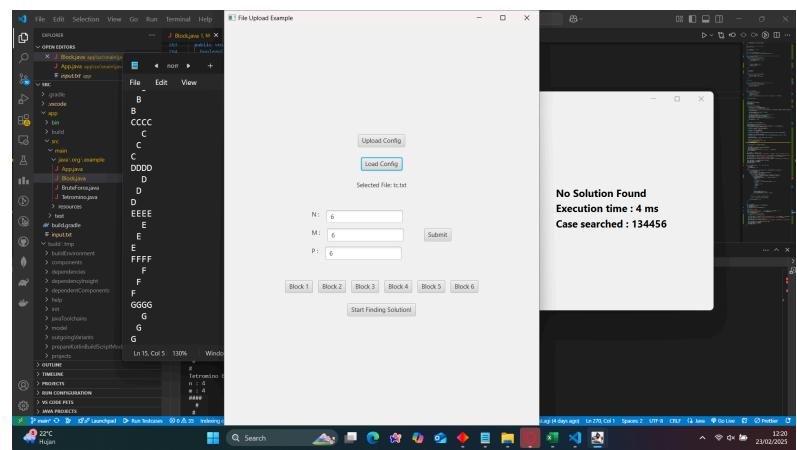
FFFF

F

F

F

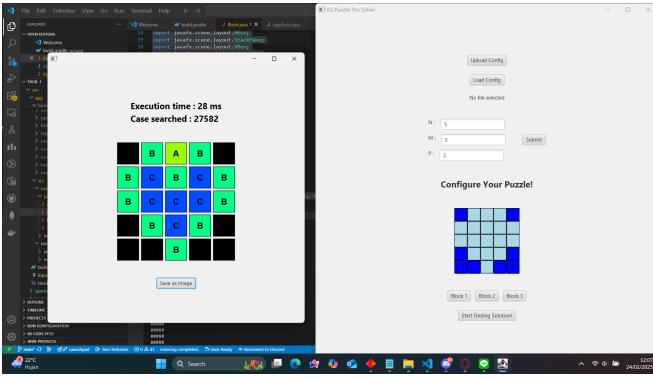
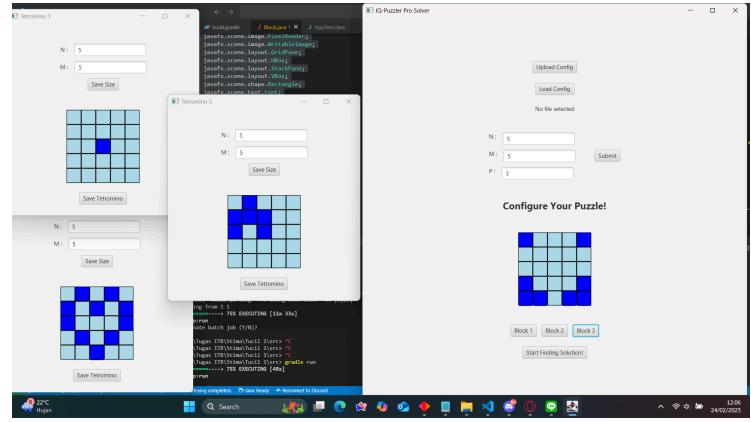
GGGG



G	
G	
G	

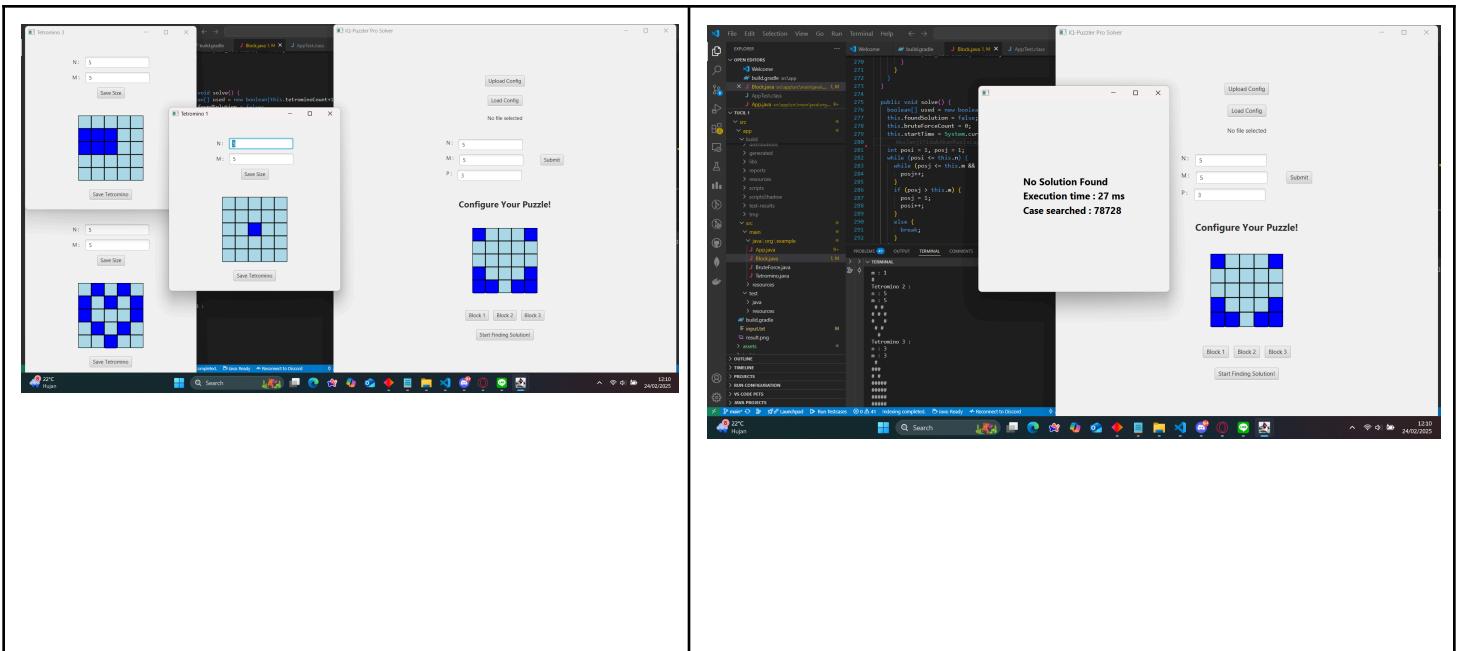
## 4.4. Kasus Papan Permainan Custom

### 4.4.1. Kasus 1 - Umum

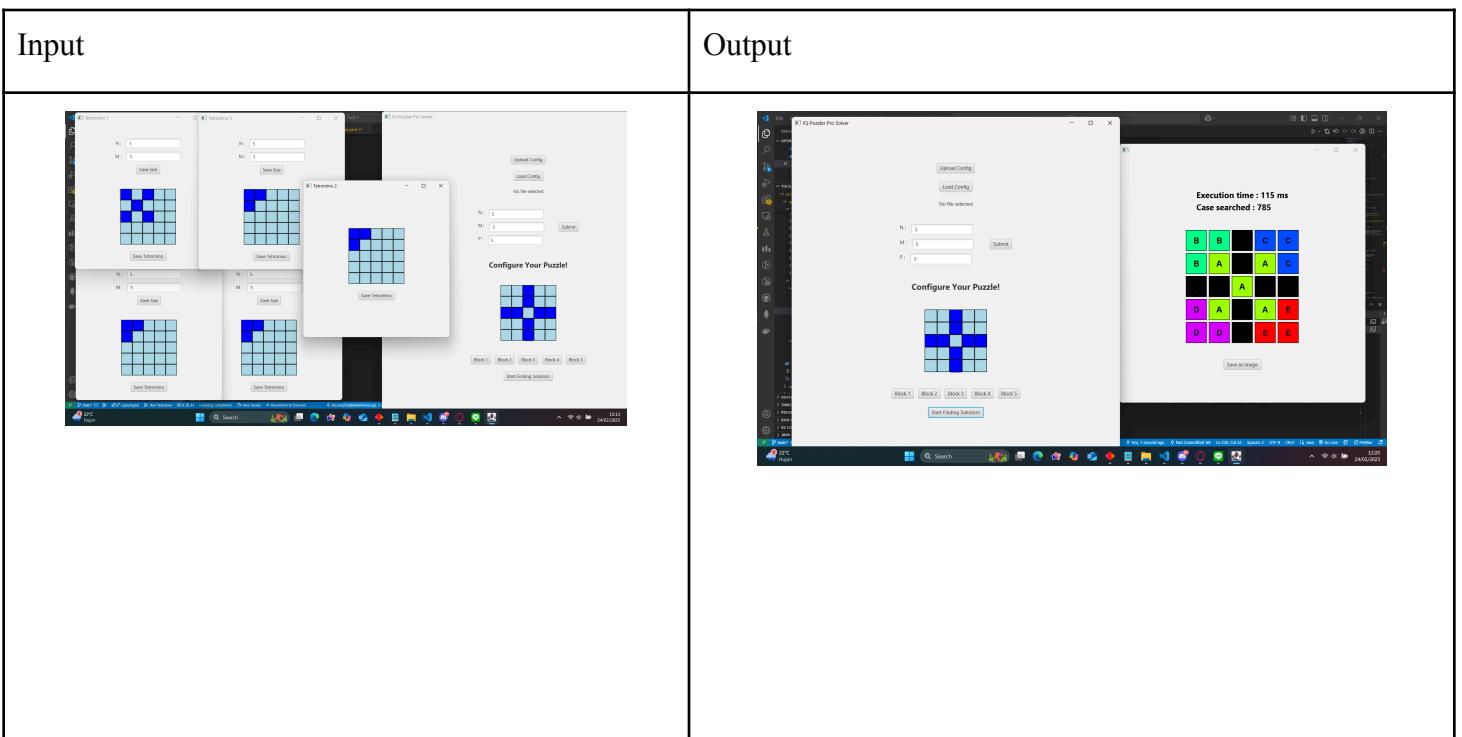
Input	Output
	

### 4.4.2. Kasus 2 - Umum

Input	Output



#### 4.4.3. Kasus 3 - Umum



#### 4.4.4. Kasus 4 - Umum

Input	Output

#### 4.4.5. Kasus 5 - Umum

Input	Output

## Lampiran

Link Github : <https://github.com/Fariz36/Tucil-1-STIMA-IQ-Puzzler-Pro-Solver>

Tabel Poin :

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	(✓)	

## **Referensi**

[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/02-Algoritma-Brute-Force-\(2025\)-Bag1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/02-Algoritma-Brute-Force-(2025)-Bag1.pdf)

[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/03-Algoritma-Brute-Force-\(2025\)-Bag2.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/03-Algoritma-Brute-Force-(2025)-Bag2.pdf)

<https://www.smartgamesusa.com>