LAPORAN TUGAS KECIL I

IF2211 Strategi Algoritma

"Penyelesaian IQ Puzzler Pro dengan Algoritma Brute Force"

Dosen:

Ir. Rila Mandala, M. Eng, Ph. D.

Monterico Adrian, S. T., M. T.

Disusun Oleh:

18222086 Juan Sohuturon Arauna Siagian

PROGRAM STUDI TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

SEMESTER II TAHUN 2024/2025

KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa penulis ucapkan atas kesempatan dan keberhasilan dalam menyelesaikan Tugas Kecil 1 IF2211 Strategi Algoritma, Semester II tahun 2024/2025, yang berjudul "Penyelesaian IQ Puzzler Pro dengan Algoritma Brute Force". Laporan ini merupakan dokumentasi komprehensif dari proses pengembangan program yang dirancang untuk menemukan solusi paling optimal dalam permainan IQ Puzzler Pro menggunakan pendekatan algoritma *brute force*.

Tugas ini memberikan pengalaman belajar yang menarik dalam mengaplikasikan teori algoritma pada kasus nyata yang cukup kompleks. Penulis mengembangkan solusi yang efisien dan efektif, mulai dari tahap *design* algoritma hingga implementasi *code* dan pengujian.

Dengan dukungan dari dosen, asisten kelompok, input dari rekan mahasiswa, dan berbagai sumber daya yang tersedia, penulis telah menyelesaikan program untuk tugas kecil ini yang diharapkan dapat bekerja dengan baik dan efisien. Akhir kata, penulis mengucapkan terima kasih kepada semua yang telah berpartisipasi dan mendukung. Penulis berharap agar tugas kecil ini dapat memenuhi mata kuliah Strategi Algoritma dan dapat memberikan wawasan yang bermanfaat bagi pembaca serta menjadi sumber inspirasi untuk inovasi lebih lanjut.

Bandung, 24 Februari 2025,

Juan S.A.S.

DAFTAR ISI

BAB I
DESKRIPSI MASALAH

1.1. Algoritma Brute Force

Algoritma *brute force* adalah sebuah pendekatan yang langsung (*straightforward*) untuk memecahkan suatu masalah, biasanya didasarkan pada pernyataan masalah (*problem statement*) dan definisi konsep yang dilibatkan. Algoritma brute force memecahkan masalah dengan sangat sederhana, langsung dan dengan cara yang jelas (*obvious way*).

Kelebihan Algoritma Brute Force:

1) Algoritma brute force dapat digunakan untuk memecahkan hampir sebagian besar masalah.
2) Sederhana dan mudah dimengerti.
3) Menghasilkan algoritma yang layak untuk beberapa masalah penting seperti pencarian, pengurutan, pencocokan string, perkalian matriks.
4) Menghasilkan algoritma baku (standar) untuk tugas-tugas komputasi seperti penjumlahan/perkalian N buah bilangan, menentukan elemen minimum atau maksimum di tabel.

Kelemahan Algoritma Brute Force:

1) Jarang menghasilkan algoritma yang mangkus/efektif.
2) Lambat sehingga tidak dapat diterima.
3) Tidak sekreatif teknik pemecahan masalah lainnya.

Cara Kerja Algoritma Brute Force:

1) Definisikan masalah yang ingin diselesaikan dan parameter-parameter yang terlibat.
2) *Generate* semua kemungkinan solusi tanpa mempertimbangkan keefisienan.

3) Evaluasi setiap solusi untuk memeriksa apakah memenuhi kriteria keberhasilan.
4) Pilih solusi yang paling optimal atau memenuhi kriteria terbaik dari semua solusi yang dihasilkan.
5) Implementasikan solusi yang dipilih dan analisis performa serta efisiensinya.
6) Optimalkan algoritma jika diperlukan untuk meningkatkan performa atau mengurangi kompleksitasnya.
7) Uji coba algoritma pada berbagai kasus uji untuk memastikan hasil yang benar dan sesuai dengan ekspektasi.
8) Ulangi proses jika diperlukan untuk meningkatkan performa atau menangani masalah baru yang muncul.
9) Gunakan algoritma *brute force* secara efektif untuk menyelesaikan masalah, dengan memperhitungkan kebutuhan dan keterbatasan yang ada.

1.2. IQ Puzzler Pro

IQ Puzzler Pro adalah permainan teka-teki logika yang dirancang oleh **SmartGames**, yang bertujuan untuk mengasah keterampilan berpikir logis, pemecahan masalah, dan kemampuan spasial. Permainan ini memiliki tiga mode tantangan, yaitu menyusun kepingan di papan secara horizontal dalam bentuk 2D, menyusun dalam pola diagonal yang lebih sulit, serta membangun struktur piramida dalam mode 3D. Dengan total **120 tantangan** yang bervariasi dari tingkat mudah hingga sangat sulit, pemain harus menyusun kepingan berbentuk unik berdasarkan petunjuk awal yang diberikan. Permainan ini memiliki konsep yang mirip dengan **Tetris**, tetapi lebih kompleks dan membutuhkan strategi yang lebih matang. Selain itu, desainnya yang kompak dengan kotak penyimpanan membuatnya mudah dibawa dan dimainkan di mana saja.

Program ini dibuat untuk menyelesaikan puzzle IQ Puzzler Pro dengan menggunakan algoritma *brute force* yang mengimplementasikan *searching* dan *pruning* yang efisien.

1.3. Langkah-Langkah Penyelesaian IQ Puzzler Pro dengan Pendekatan Brute Force

Langkah-langkah penyelesaian IQ Puzzler Pro dengan algoritma *brute force* yang digunakan pada program ini adalah sebagai berikut:

1) *Parsing input*

Input yang diberikan dalam bentuk *file* akan dibaca dan diolah di dalam program. Kepingan puzzle (*piece*) akan disimpan kedalam sebuah *class* yang akan menyimpan seluruh kombinasi (rotasi dan translasi) dari *piece* dalam bentuk *array* 2 dimensi.

2) Recursive steps

Inti utama dari pendekatan *brute force* ini adalah rekursif. Pendekatan rekursif memungkinkan kita untuk melakukan penelusuran *search space* yang lebih sistematis dan tidak redundan. Sehingga jumlah kasus yang ditinjau pun bisa diminimalisir.

Recursive steps pada program ini didefinisikan dengan menempatkan sebuah *piece* keatas papan. Penempatan *piece* tersebut bisa dilakukan di bagian papan yang masih kosong, dan setiap kemungkinan posisi dari *piece* dipertimbangkan. Setelah kita memilih dan meletakkan sebuah *piece*, kita akan melakukan tahapan yang sama untuk *piece* berikutnya. Apabila kita sampai ke sebuah kondisi dimana tidak mungkin meletakkan *piece* yang tersisa, maka kita akan masuk ke fase *backtracking*.

3) Backtracking

Jika kita berada pada posisi dimana tidak tersedia tempat yang valid untuk meletakkan *piece* berikutnya, kita akan mengubah posisi dari *piece*

terakhir yang kita letakkan, dan mencoba tahapan yang sama dengan konfigurasi papan yang baru.

4) Stop condition

Program akan berakhir ketika seluruh pemanggilan fungsi rekursif telah selesai. Dan ada dua kemungkinan dimana hal ini terjadi, yaitu ketika ditemukan sebuah solusi yang valid atau ketika seluruh *search space* telah ditelusuri.

BAB II
SOURCE CODE & IMPLEMENTASI

2.1 GitHub Repository

Source code dari program ini dapat diakses melalui GitHub https://github.com/JuanSign/IQPP-Solver.

2.2 Library

Penggunaan berbagai *library* memiliki peran penting dalam memfasilitasi berbagai fungsi dan fitur yang diimplementasikan dalam program seperti manipulasi data, interaksi dengan *user* melalui input dan output, serta pemrosesan string. Pada program ini, *library* yang digunakan hanya berasal dari *standard library* Java dan digunakan untuk pengolahan data, penerimaan input, *error handling*, dan penciptaan gambar solusi.

Pada Main.java,

```
import java.util.Scanner;
```

Library digunakan untuk menerima *input* dari *user* melalui *command-line*.

Pada Piece.java,

```
import java.lang.StringBuilder;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashSet;
import java.util.List;
import java.util.Set;
```

Library digunakan untuk fungsionalitas struktur data dan manipulasi *string*.

Pada State.java,

```
import java.io.IOException;

import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashSet;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.Set;
```

Library digunakan untuk fungsionalitas struktur data dan operasi I/O dengan file.

Pada BoardImageGenerator.java,

```
import javax.imageio.ImageIO;
import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
```

```
import java.util.HashMap;
import java.util.Map;
```

Library digunakan untuk fungsionalitas penciptaan gambar.

2.3 Global Variable

Penggunaan variabel global memiliki tujuan khusus dalam menyimpan informasi yang dibutuhkan secara luas di berbagai bagian dalam program ini. Variabel global digunakan untuk menyimpan nilai-nilai seperti ukuran papan, ukuran *piece*, jumlah iterasi, dan solusi dari permasalahan, dan kemudian dapat diakses dan dimanipulasi oleh fungsi-fungsi utama program. Berikut merupakan variabel global yang digunakan dalam program ini untuk mendukung fungsi-fungsi program secara keseluruhan:

```
private int N, M, P;
private String S;

private List<Piece> pieces;
public char[][] board;
public int combinations = 0;
```

Dimana *Piece* adalah representasi dari *piece* yang terdapat pada puzzle. Piece dibuat menjadi sebuah Class untuk mempermudah manipulasi dan penyimpanan data.

```
public class Piece{
    public List<char[][]> contents;
    public char identifier;
    ……
}
```

2.4 Main Program Functions

Fungsi-fungsi utama dalam program ini bertanggung jawab untuk mengatur alur utama program, mulai dari menerima input pengguna hingga menangani proses pencarian solusi dari puzzle. Fungsi-fungsi utama ini merupakan inti dari program yang langsung terlibat dalam menjalankan algoritma pencarian jalur optimal yang diimplementasikan.

`ReadInput()dan ParseInput()`

Kedua fungsi berikut bertujuan untuk membaca data input dari file yang diberikan dan melakukan *parsing* dan validasi terhadap data input yang diberikan. Jika data yang diberikan valid, maka program akan menyimpan data-data yang dibutuhkan kedalam bentuk struktur data yang paling sesuai.

```
public void ReadInput(String inputFileName) throws Exception {
        Path inputFilePath = Paths.get("test/inputs/", inputFileName);

        if (!Files.exists(inputFilePath)) {
            throw new IOException("[ERROR][IO] : Input file not found!");
        }

        List<String> inputFileContent;
        try {
            inputFileContent = Files.readAllLines(inputFilePath);
        } catch (IOException e) {
            throw new IOException("[ERROR][IO] : ERROR Reading file", e);
        }

        try {
            ParseInput(inputFileContent);
        } catch (Exception e){
            throw e;
        }
    }
```

```
    private void ParseInput(List<String> inputFileContent) throws
Exception {
        if(inputFileContent.size() < 2){
            throw new Exception("[ERROR][INPUT] : Input must be at least
two lines.");
        }

        String[] config = inputFileContent.get(0).trim().split("\\s+");
        if(config.length != 3){
            throw new Exception(String.format("[ERROR][INPUT] : Invalid
input at line 1 [%s]", inputFileContent.get(0)));
        }

        try{
            N = Integer.parseInt(config[0]);
            M = Integer.parseInt(config[1]);
            P = Integer.parseInt(config[2]);

            board = new char[N][M];
            for(int i = 0; i < N; i++){
                for(int j = 0; j < M; j++){
                    board[i][j] = ' ';
                }
            }
        }catch(NumberFormatException e){
            throw new Exception("[ERROR][INPUT] : N, M, P must be an
integer!");
        }

        S = inputFileContent.get(1).trim();
        if(!S.equals("DEFAULT")){
            throw new Exception("[ERROR][INPUT] : S must be DEFAULT");
        }

        List<List<String>> rawPieces = new ArrayList<>();
```

```
        Set<Character> charSet = new HashSet<>();
        char curID = '\0';
        for(int i = 2; i < inputFileContent.size(); i++){
            String inputLine = inputFileContent.get(i);
            Set<Character> uniqueChar = new HashSet<>();
            for(char c : inputLine.toCharArray()){
                if(!Character.isWhitespace(c)){
                    uniqueChar.add(c);
                }
            }
            if(uniqueChar.size() != 1){
                throw new Exception(String.format("[ERROR][INPUT] :
Invalid input at line %d [%s]", i+1, inputLine));
            }else{
                char id = uniqueChar.iterator().next();
                if(id == curID){
                    rawPieces.get(rawPieces.size()-1).add(inputLine);
                }else{
                    rawPieces.add(new ArrayList<>(List.of(inputLine)));
                    if(charSet.contains(id)){
                        throw new Exception(String.format("[ERROR][INPUT]
: Duplicate piece identifier [%c]", id));
                    }else{
                        curID = id;
                        charSet.add(curID);
                    }
                }
            }
        }
        if(rawPieces.size() != P){
            throw new Exception(String.format("[ERROR][INPUT] : Given %d
pieces but P is %d", rawPieces.size(), P));
        }

        pieces = new ArrayList<>();
        for(List<String> rawPiece : rawPieces){
```

```
            pieces.add(new Piece(rawPiece));
        }
    }
```

GenerateTransformation()

Fungsi berikut bertujuan untuk menerima input *piece* dan melakukan transformasi untuk menemukan seluruh konfigurasi yang memungkinkan. Fungsi berikut juga bertanggung jawab untuk memastikan bahwa konfigurasi *piece* yang disimpan adalah *unique* dan tidak redundan.

```
    private static List<char[][]> GenerateTransformation(char[][] matrix)
{
        Set<String> seen = new HashSet<>();
        List<char[][]> uniqueTransformations = new ArrayList<>();

        char[][] current = matrix;
        for (int i = 0; i < 4; i++) {
            String key = MatrixToString(current);
            if (seen.add(key)) {
                uniqueTransformations.add(current);
            }
            current = Rotate(current);
        }

        current = Mirror(matrix);
        for (int i = 0; i < 4; i++) {
            String key = MatrixToString(current);
            if (seen.add(key)) {
                uniqueTransformations.add(current);
            }
            current = Rotate(current);
        }

        return uniqueTransformations;
```

```
    }

    private static char[][] Rotate(char[][] matrix) {
        int N = matrix.length, M = matrix[0].length;
        char[][] rotated = new char[M][N];
        for (int r = 0; r < N; r++) {
            for (int c = 0; c < M; c++) {
                rotated[c][N - 1 - r] = matrix[r][c];
            }
        }
        return rotated;
    }

    private static char[][] Mirror(char[][] matrix) {
        int N = matrix.length, M = matrix[0].length;
        char[][] mirrored = new char[N][M];
        for (int r = 0; r < N; r++) {
            mirrored[r] = matrix[N - 1 - r].clone();
        }
        return mirrored;
    }

    private static String MatrixToString(char[][] matrix) {
        StringBuilder sb = new StringBuilder();
        for (char[] row : matrix) {
            sb.append(new String(row)).append("\n");
        }
        return sb.toString();
    }
```

`generateImage()`

Fungsi berikut bertujuan untuk menciptakan sebuah gambar yang merepresentasikan solusi yang ditemukan. Gambar akan disimpan kedalam sebuah *file .png*.

```java
public static void generateImage(char[][] board, String fileName) {
        int rows = board.length;
        int cols = board[0].length;
        int width = cols * CELL_SIZE;
        int height = rows * CELL_SIZE;

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

        Map<Character, Color> colorMap = generateColorMap();

        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                char c = board[i][j];
                Color color = colorMap.getOrDefault(c, Color.WHITE);

                g2d.setColor(color);
                g2d.fillRect(j * CELL_SIZE, i * CELL_SIZE, CELL_SIZE,
CELL_SIZE);

                g2d.setColor(Color.BLACK);
                g2d.drawRect(j * CELL_SIZE, i * CELL_SIZE, CELL_SIZE,
CELL_SIZE);
            }
        }

        g2d.dispose();

        try {
            ImageIO.write(image, "png", new File(fileName));
            System.out.println("Image saved as: " + fileName);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

```
    private static Map<Character, Color> generateColorMap() {
        Map<Character, Color> colorMap = new HashMap<>();
        Color[] colors = {
            Color.RED, Color.BLUE, Color.GREEN, Color.YELLOW, Color.CYAN,
            Color.MAGENTA, Color.ORANGE, Color.PINK, Color.LIGHT_GRAY,
Color.GRAY,
            new Color(128, 0, 128), new Color(255, 165, 0), new Color(0,
128, 128),
            new Color(128, 128, 0), new Color(255, 105, 180), new
Color(139, 69, 19),
            new Color(0, 255, 127), new Color(0, 191, 255), new Color(147,
112, 219),
            new Color(255, 222, 173), new Color(0, 255, 255), new
Color(255, 20, 147),
            new Color(218, 165, 32), new Color(70, 130, 180), new
Color(152, 251, 152),
            new Color(176, 224, 230)
        };

        for (int i = 0; i < 26; i++) {
            colorMap.put((char) ('A' + i), colors[i]);
        }
        return colorMap;
    }
```

2.5 Algorithm

Algoritma solusi terdiri dari sebuah fungsi rekursif Solve() dan tiga buah helper method, Check(), Put(), dan UnPut(). Fungsi rekursif tersebut akan meletakkan *piece* secara satu persatu dan melakukan *backtracking* jika tidak terdapat tempat yang valid untuk meletakkan *piece* berikutnya. Fungsi Solve() juga akan mengembalikan sebuah boolean ke *parent caller*nya untuk mengindikasikan keberhasilan dari *search* yang sedang berjalan.

```
public boolean Solve(int id){
    if(id == P) return true;
    for(char[][] p : pieces.get(id).contents){
        for(int i = 0; i <= N - p.length; i++){
            for(int j = 0; j <= M - p[0].length; j++){
                if(Check(p, i, j)){
                    Put(p, i, j);
                    combinations++;
                    if(Solve(id+1)){
                        return true;
                    }
                    UnPut(p, i, j);
                }
            }
        }
    }
    return false;
}

private boolean Check(char[][] p, int i, int j){
    for(int ii = 0; ii < p.length; ii++){
        for(int jj = 0; jj < p[ii].length; jj++){
            if(board[i+ii][j+jj] != ' ' && p[ii][jj] != ' '){
                return false;
            }
        }
    }
    return true;
}

private void Put(char[][] p, int i, int j){
    for(int ii = 0; ii < p.length; ii++){
        for(int jj = 0; jj < p[ii].length; jj++){
            if(p[ii][jj] != ' '){
                board[i+ii][j+jj] = p[ii][jj];
            }
```

```
            }
        }
    }

    private void UnPut(char[][] p, int i, int j){
        for(int ii = 0; ii < p.length; ii++){
            for(int jj = 0; jj < p[ii].length; jj++){
                if(p[ii][jj] != ' '){
                    board[i+ii][j+jj] = ' ';
                }
            }
        }
    }
```

BAB III

HASIL EKSEKUSI PROGRAM

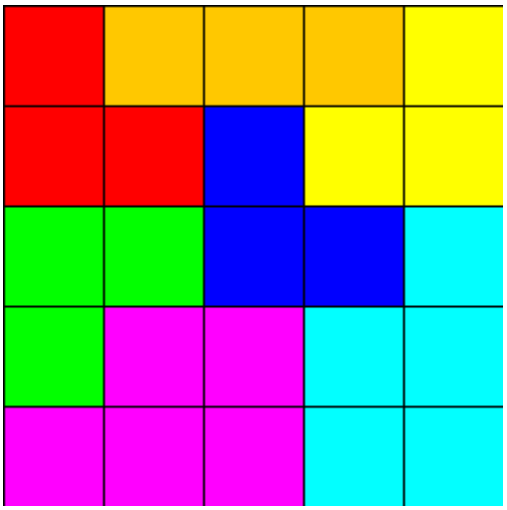
3.1 TC 1

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

Solusi :

```
PS C:\Users\juans\work\project\IQPP-Solver> java -cp bin Main
IQPP Solver.
Enter input file name (.txt) : testcase_1.txt
A|G|G|G|D|
A|A|B|D|D|
C|C|B|B|E|
C|F|F|E|E|
F|F|F|E|E|

Image saved as: test\solutions\testcase_1.png
Execution time: 8 ms
Combinations tried : 7393
```



3.2 TC 2

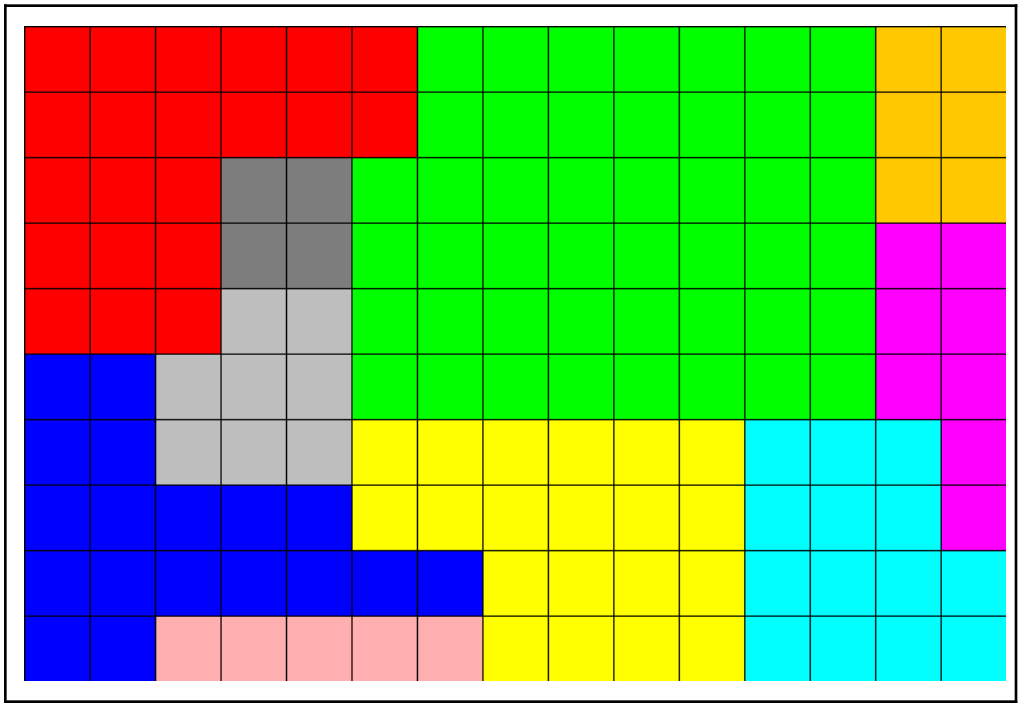
```
10 15 10
DEFAULT
AAAAAA
AAAAAA
AAA
AAA
AAA
BB
BB
BBBBB
BBBBBBB
BB
 CCCCCCC
 CCCCCCC
CCCCCCCC
CCCCCCCC
CCCCCCCC
CCCCCCCC
DDDDDD
DDDDDD
  DDDD
```

```
 DDDD
EEE
EEE
EEEE
EEEE
FF
FF
FF
 F
 F
GG
GG
GG
HHHHH
 II
III
III
JJ
JJ
```

Solusi :

```
PS C:\Users\juans\work\project\IQPP-Solver> java -cp bin Main
IQPP Solver.
Enter input file name (.txt) : testcase_2.txt
A|A|A|A|A|A|C|C|C|C|C|C|C|G|G|
A|A|A|A|A|A|C|C|C|C|C|C|C|G|G|
A|A|A|J|J|C|C|C|C|C|C|C|C|G|G|
A|A|A|J|J|C|C|C|C|C|C|C|C|F|F|
A|A|A|I|I|C|C|C|C|C|C|C|C|F|F|
B|B|I|I|I|C|C|C|C|C|C|C|C|F|F|
B|B|I|I|I|D|D|D|D|D|D|E|E|E|F|
B|B|B|B|B|D|D|D|D|D|D|E|E|E|F|
B|B|B|B|B|B|B|D|D|D|D|E|E|E|E|
B|B|H|H|H|H|H|D|D|D|D|E|E|E|E|

Image saved as: test\solutions\testcase_2.png
Execution time: 7 ms
Combinations tried : 1951
```



3.3 TC 3

```
10 15 10
DEFAULT
AAAAAA
AAAAAAA
AAA
AAA
AAA
BBB
BB
BBBBB
BBBBBBB
BB
 CCCCCCC
 CCCCCCC
CCCCCCCC
CCCCCCCC
CCCCCCCCC
CCCCCCCC
```

```
DDDDDD
DDDDDD
  DDDD
  DDDDD
EEE
EEEE
EEEE
EEEE
FFF
FF
FF
 F
 F
GG
GGG
GG
HHHHHH
 II
IIII
III
JJJ
JJ
```

Solusi :

```
PS C:\Users\juans\work\project\IQPP-Solver> java -cp bin Main
IQPP Solver.
Enter input file name (.txt) : testcase_3.txt
No solutions.
Execution time: 518 ms
Combinations tried : 164076
```

3.4 TC 4

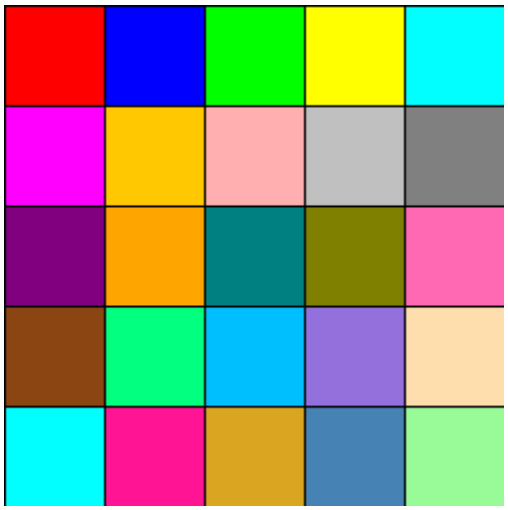
```
5 5 25
DEFAULT
A
B
```

C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y

Solusi :

```
PS C:\Users\juans\work\project\IQPP-Solver> java -cp bin Main
IQPP Solver.
Enter input file name (.txt) : testcase_4.txt
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|

Image saved as: test\solutions\testcase_4.png
Execution time: 0 ms
Combinations tried : 25
```



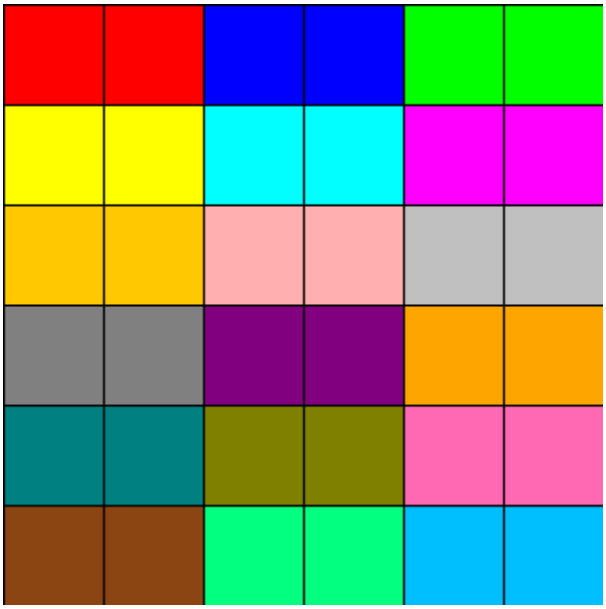
3.5 TC 5

```
6 6 18
DEFAULT
AA
BB
CC
DD
EE
FF
GG
HH
II
JJ
KK
LL
MM
NN
OO
PP
QQ
RR
```

Solusi :

```
PS C:\Users\juans\work\project\IQPP-Solver> java -cp bin Main
IQPP Solver.
Enter input file name (.txt) : testcase_5.txt
A|A|B|B|C|C|
D|D|E|E|F|F|
G|G|H|H|I|I|
J|J|K|K|L|L|
M|M|N|N|O|C|
P|P|Q|Q|R|R|

Image saved as: test\solutions\testcase_5.png
Execution time: 0 ms
Combinations tried : 18
```

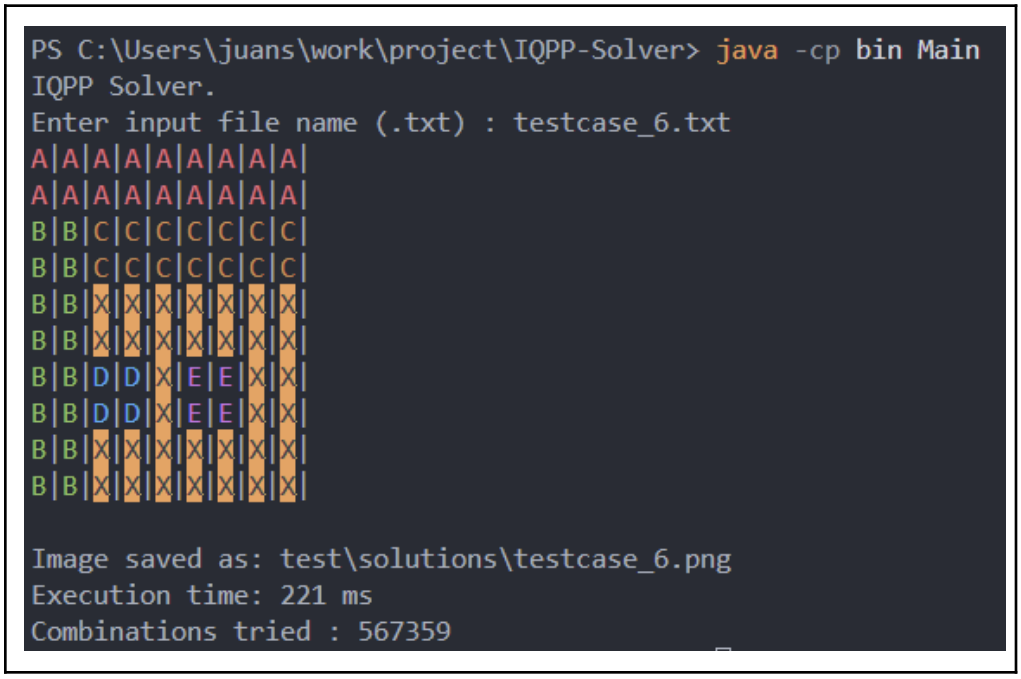


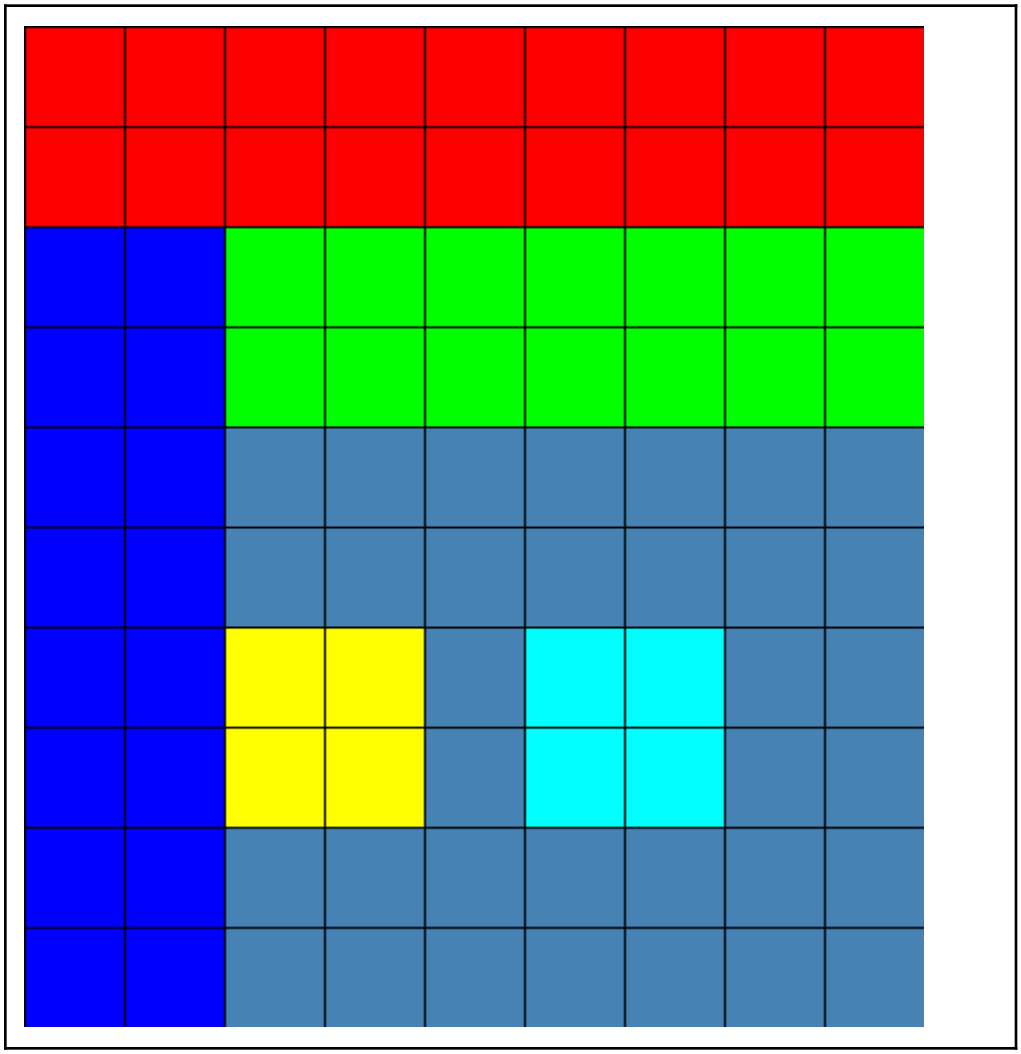
3.6 TC 6

```
10 9 6
DEFAULT
AA
AA
AA
AA
AA
```

```
AA
AA
AA
AA
BBBBBBBB
BBBBBBBB
CC
CC
CC
CC
CC
CC
CC
DD
DD
EE
EE
XXXXXX
XXXXXX
XX  XX
XX  XX
XXXXXX
XX  XX
XX  XX
```

Solusi :


PS C:\Users\juans\work\project\IQPP-Solver> java -cp bin Main
IQPP Solver.
Enter input file name (.txt) : testcase_6.txt
A|A|A|A|A|A|A|A|A|
A|A|A|A|A|A|A|A|A|
B|B|C|C|C|C|C|C|C|
B|B|C|C|C|C|C|C|C|
B|B|X|X|X|X|X|X|X|
B|B|X|X|X|X|X|X|X|
B|B|D|D|X|E|E|X|X|
B|B|D|D|X|E|E|X|X|
B|B|X|X|X|X|X|X|X|
B|B|X|X|X|X|X|X|X|
Image saved as: test\solutions\testcase_6.png
Execution time: 221 ms
Combinations tried : 567359



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/2021-2022/Algoritma-Brute-Force-(2022)-Bag1.pdf

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag2.pdf

https://core.ac.uk/download/pdf/288088999.pdf