

Tugas Kecil 1 IF2211 Strategi Algoritma: IQ Puzzler Pro Solver dengan Pendekatan Algoritma Brute Force

Muhammad Ghifary Komara Putra - 13523066¹

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

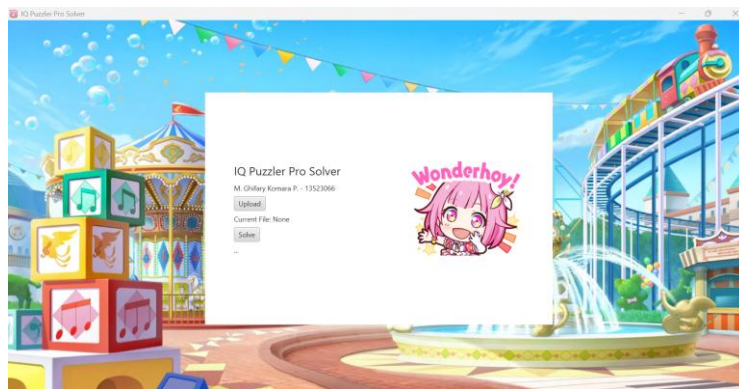
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia

¹m.ghifary.k.p@gmail.com, 13523066@std.stei.itb.ac.id

Abstrak—IQ Puzzler Pro merupakan suatu permainan puzzle yang diproduksi oleh perusahaan Smart Games. Dalam permainan ini, pemain diberikan sebuah papan serta beberapa blok puzzle untuk mengisi seluruh petak dalam papan tersebut dengan seluruh blok puzzle yang tersedia. Penulis mengembangkan alternatif solusi penyelesaian permainan IQ Puzzler Pro dengan pendekatan algoritma brute force, dalam bahasa pemrograman Java. Pengguna dapat memberikan masukan file teks untuk melihat apakah konfigurasi papan dan blok puzzle tertentu memiliki solusi atau tidak, dan seperti apa solusi yang tersedia. Beberapa validasi dan optimalisasi dilakukan untuk mempercepat pencarian solusi, termasuk optimalisasi terhadap dimensi blok puzzle terhadap dimensi papan serta optimalisasi terhadap duplikasi hasil rotasi dan pencerminan. Berdasarkan pengujian, program yang telah dikembangkan mampu menemukan solusi permainan serta melakukan validasi terhadap masukan dengan cukup baik.

Kata Kunci: Brute force, IQ Puzzler Pro, Puzzle

I. PENDAHULUAN



Gambar 1. Home Page Program

IQ Puzzler Pro, diproduksi oleh perusahaan Smart Games, merupakan suatu permainan puzzle di mana pemain diharuskan mengisi seluruh petak (selanjutnya disebut sebagai tile) dalam papan dengan blok puzzle (selanjutnya disebut sebagai piece) yang tersedia. Pada tugas kecil 1 ini, penulis mengembangkan sebuah program dalam bahasa Java yang mampu menyelesaikan permainan IQ Puzzler Pro dengan pendekatan algoritma brute force. Pengguna dapat memberikan data dimensi papan, jumlah piece, serta bentuk setiap piece dalam suatu file dengan ekstensi *.txt untuk mengecek solusi dari konfigurasi permainan tersebut (jika ada). Pengguna pun dapat menyimpan solusi yang disediakan, baik dalam bentuk file teks atau gambar.

II. DESKRIPSI SOLUSI

A. Deskripsi Kelas

Bagian ini akan menjelaskan lebih lanjut beberapa kelas yang relevan dalam pencarian solusi dengan algoritma brute force, mencakup fungsi, atribut, dan method yang tersedia dalam kelas tersebut. Kelas-kelas yang dimaksud antara lain InputData, Board, dan Piece. Program secara keseluruhan tersedia pada repositori yang tertera dalam lampiran.

A.1 InputData

Kelas InputData digunakan untuk menyimpan dan memproses data masukan pengguna. Data tersebut mencakup nilai dimensi papan (N dan M), jumlah piece (P), papan permainan (B), serta matriks berdimensi $P \times 8$ berisi objek piece (pieces). Dalam matriks pieces, baris ke-i menandakan suatu piece, dengan setiap kolom merepresentasikan seluruh konfigurasi rotasi dan pencerminan untuk piece pada baris tersebut.

A.2 Board

Kelas Board adalah kelas yang merepresentasikan papan permainan. Kelas ini menyimpan data dimensi papan permainan (N dan M), matriks berdimensi $N \times M$ yang merepresentasikan papan permainan tersebut (state), status papan (tidak ada solusi, ditemukan solusi, atau belum dilakukan pencarian), serta total kasus yang ditelusuri dalam algoritma brute force yang digunakan. Kelas ini juga menyediakan method place dan pop, yang berturut-turut digunakan untuk menyimpan dan menghapus suatu Piece pada kondisi papan saat ini.

A.3 Piece

Kelas Piece adalah kelas yang merepresentasikan suatu blok puzzle dalam permainan. Kelas ini menyimpan data dimensi piece (width dan height), karakter yang merepresentasikan piece tersebut (id), matriks yang menyimpan data bentuk piece tersebut (shape), dan indeks matriks paling atas kiri yang tidak kosong pada piece tersebut (top left index). Kelas ini juga menyediakan method untuk mencerminkan piece terhadap sumbu-y (mirror) dan merotasikan suatu piece sebesar 90° searah jarum jam (rotate90).

B. Algoritma Brute Force

Solusi dengan pendekatan algoritma brute force yang dikembangkan adalah sebagai berikut:

1. Periksa koordinat paling atas kiri dari matriks papan yang belum terisi oleh piece
2. Traversal setiap piece yang tersedia. Jika piece belum diletakkan dalam papan, Coba letakkan dengan top left index diletakkan pada koordinat yang dicek pada tahap (1)
3. Jika piece tidak bisa diletakkan, traversal variasi rotasi dan pencerminan dari piece tersebut, coba letakkan dengan cara serupa dengan tahap (2)
4. Jika tidak ada konfigurasi piece yang dapat diletakkan untuk setiap piece yang belum diletakkan dalam papan, hapus piece yang terakhir kali diletakkan pada papan
5. Ulangi langkah 1-4 hingga seluruh tile dalam papan terisi oleh piece (solusi ditemukan) atau seluruh kemungkinan telah diperiksa (tidak ada solusi).

Implementasi algoritma brute force dalam bahasa Java tertera pada Algoritma 1.

Algoritma 1: Solve.java

```
package Function;

public class Solve {
    public static Board BruteForce(InputData data, int row, int col){
        int i, j;
        int currentRow = row;
        int currentCol = col;
        Board board = data.B;
        Piece[][] pieces = data.pieces;
        int total_pieces = data.P;

        // Menentukan indeks paling atas kiri yang dapat diisi
        boolean found = false;
        while(currentCol < board.width && !found){
            if(board.state[currentRow][currentCol] == ' '){
                found = true;
            } else{
                currentCol++;
            }
        }
        if(!found){
            currentRow++;
        }
    }
}
```

```

while(currentRow < board.height && !found){
    currentCol = 0;
    while(currentCol < board.width && !found){
        if(board.state[currentRow][currentCol] == ' '){
            found = true;
        } else{
            currentCol++;
        }
    }
    if(!found){
        currentRow++;
    }
}
// Basis: solusi ditemukan
if(currentRow == board.height && currentCol == board.width && board.state[currentRow-1][currentCol-1] != ' '){
    board.status = 1;
    return board;
}

// Mencoba meletakkan piece di dalam board
for(i=0; i<total_pieces; i++){
    j = 0;
    while(j<8 && pieces[i][0].isPlaced == false){
        board.total_case++;
        if(pieces[i][0].isPlaced == false){
            board.place(pieces[i][j], currentRow, currentCol);
            // Jika berhasil diletakkan, lanjutkan ke tahap berikutnya
            if(board.state[currentRow][currentCol] == pieces[i][j].id){
                pieces[i][0].isPlaced = true;
                board = BruteForce(data, currentRow, currentCol);
                // Jika solusi sudah ditemukan, keluar dari fungsi
                if(board.status == 1){
                    return board;
                }
                // Jika solusi tidak ditemukan, hapus piece yang baru saja diletakkan, tukar dengan konfigurasi lain
                pieces[i][0].isPlaced = false;
                board.pop(pieces[i][j], currentRow, currentCol);
            }
        }
        j++;
    }
}

// Tidak ada solusi yang ditemukan
data.B.status = -1;
return data.B;
}
}

```

C. Validasi Masukan Pengguna

Beberapa validasi masukan pengguna yang diimplementasikan dalam program ini adalah sebagai berikut:

- Validasi terhadap ketidaktersediaan file pada folder input
- Validasi terhadap empty line pada bagian mana pun pada file masukan
- Nilai N, M, dan P tidak ditulis dengan benar (banyaknya data kurang, berlebih, atau bukan angka)
- Nilai N, M, P ≤ 0 serta nilai P > 26
- Baris 2 tidak bertuliskan DEFAULT
- Validasi terhadap karakter piece (bukan merupakan karakter kapital, terdapat dua karakter berbeda dalam satu baris, atau terdapat karakter yang sudah digunakan oleh piece lain)
- Validasi terhadap jumlah piece (kurang atau lebih dari nilai P)

D. Optimalisasi Program

Dalam program ini, terdapat beberapa validasi dan optimalisasi yang digunakan sebelum pencarian dimulai untuk meningkatkan efisiensi pencarian. Validasi tambahan yang tersedia adalah validasi dimensi setiap piece terhadap dimensi papan serta validasi jumlah petak piece terhadap jumlah petak pada papan. Pada validasi

pertama, program akan mengecek setiap piece yang tersedia apakah memiliki dimensi melebihi ukuran papan, misalnya berdimensi 5×4 pada papan berukuran 3×2 . Pada kasus ini, tidak akan ada konfigurasi piece tersebut yang dapat diletakkan dalam papan, sehingga papan dapat langsung dinyatakan tidak memiliki solusi tanpa perlu melakukan pencarian. Pada validasi kedua, program akan mengecek tile pada setiap piece, menjumlahkannya, kemudian membandingkannya dengan tile kosong pada papan awal. Jika jumlah tile tersebut tidak sama dengan tile kosong pada papan awal, papan dapat langsung dinyatakan tidak memiliki solusi karena seluruh piece tidak mungkin tepat mengisi papan tersebut.

Optimalisasi yang diterapkan dalam program ini antara lain optimasi terhadap rotasi dan pencerminan setiap piece. Pertama, seluruh konfigurasi rotasi dan pencerminan disimpan pada kelas InputData sebelum pencarian dilakukan ketimbang melakukannya saat pencarian berlangsung. Kedua, program akan mengecek apakah terdapat konfigurasi hasil pencerminan dan/atau rotasi yang sama. Jika ada, salah satu konfigurasi akan dianggap sebagai piece kosong (berdimensi 0×0). Selain itu, program juga akan mengeliminasi konfigurasi yang tidak dapat diletakkan dalam papan. Misalnya, untuk suatu piece berdimensi 2×5 pada papan berdimensi 4×10 , seluruh konfigurasi hasil rotasi dan/atau pencerminan berdimensi 5×2 akan dianggap sebagai piece kosong (tidak mungkin diletakkan), cukup perhatikan konfigurasi berdimensi 2×5 . Optimalisasi ini diharapkan akan mengurangi waktu pencarian solusi dengan memangkas perhitungan dan piece yang redundan.

III. PANDUAN PENGGUNAAN PROGRAM

1. Lakukan instalasi Java 23 dan JavaFX 23
2. Clone repositori pada lampiran ke dalam perangkat Anda
3. Pada CLI, bergeraklah menuju folder bin
4. Jalankan perintah berikut

```
java --module-path "path menuju folder lib javafx-sdk-23" --add-modules javafx.controls,javafx.fxml -
jar IQPuzzlerProSolver.jar
```

5. Unggah masukan .txt dengan format sebagai berikut

```
N M P
DEFAULT
Piece ke-1
Piece ke-2
.
.
.
Piece ke-P
```

Dengan N dan M merupakan dimensi papan permainan

6. Jika hendak menyimpan solusi, silakan klik button yang bersesuaian. File akan tersimpan pada folder bin/data/output

Jika hendak melakukan kompilasi kembali terhadap file .jar, silakan ikuti langkah berikut:

1. Buka CLI pada root directory
2. Jalankan perintah berikut untuk melakukan kompilasi terhadap file .java

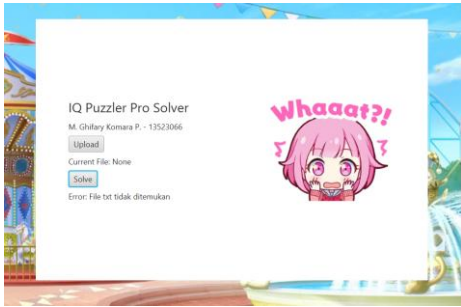
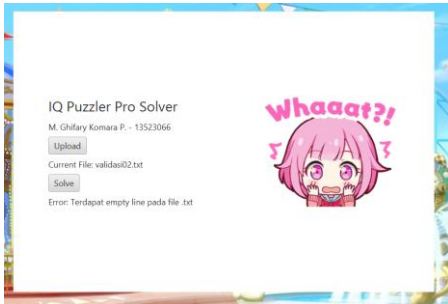
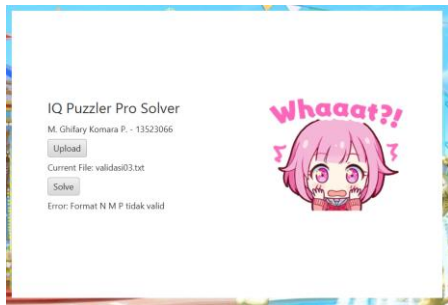
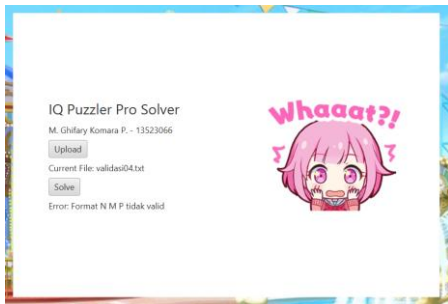
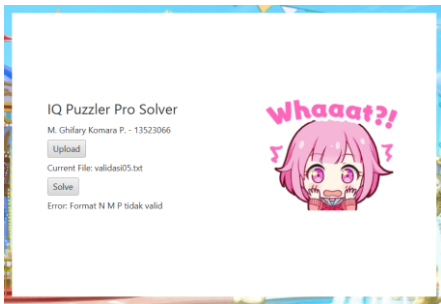
```
javac --module-path "path menuju folder lib javafx-sdk-23" --add-modules javafx.controls,javafx.fxml -
d bin src/Function/*.java src/GUI/*.java
```
3. Untuk melakukan kompilasi kembali menjadi file .jar, jalankan perintah berikut

```
jar cfe bin/IQPuzzlerProSolver.jar GUI.Main -C bin .
```

IV. PENGUJIAN PROGRAM

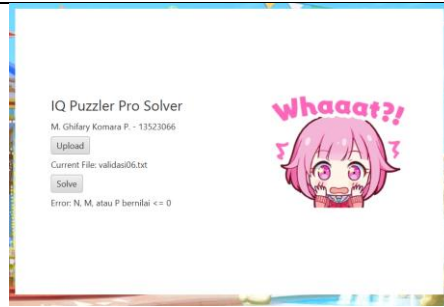
Pengujian program dilakukan untuk memeriksa apakah program yang sudah dikembangkan bekerja sebagaimana mestinya. Hal ini mencakup melakukan validasi terhadap input yang tidak valid serta mengembalikan solusi dari konfigurasi papan dan piece yang diberikan, waktu eksekusi algoritma, serta jumlah kasus yang ditinjau. Keterangan pengujian, masukan, serta keluaran yang diberikan oleh program tertera pada Tabel 1. File pengujian dapat diakses pada repositori yang tersedia pada bagian Lampiran, tepatnya pada folder bin/data/input. File validasiXX.txt merujuk pada pengujian nomor 1-14 dan file tcXX.txt merujuk pada pengujian nomor 15-24 (dimulai dari tc01.txt hingga tc10.txt).

Tabel 1. Pengujian Program

No.	Deskripsi Pengujian	Keluaran Program
1.	Validasi terhadap ketidaktersediaan file pada folder input	
2.	Validasi terhadap empty line pada bagian mana pun pada file masukan 2 3 3 DEFAULT AA A B CC	
3.	Nilai N, M, dan P tidak lengkap 2 3 DEFAULT AA A B CC	
4.	Nilai N, M, dan P berlebih 2 3 3 3 DEFAULT AA A B CC	
5.	Nilai N, M, atau P bukan angka 2 X 3 DEFAULT AA A B CC	

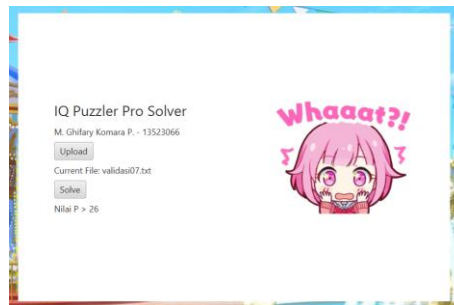
-
6. Nilai N, M, atau P ≤ 0

2 -1 3
DEFAULT
AA
A
B
CC



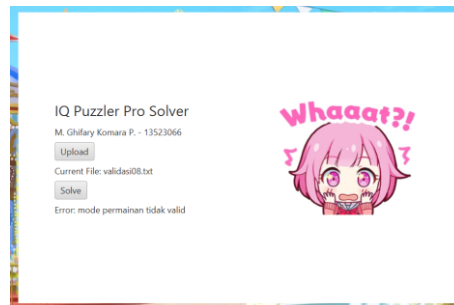
7. Nilai P > 26

2 3 27
DEFAULT
AA
A
B
CC



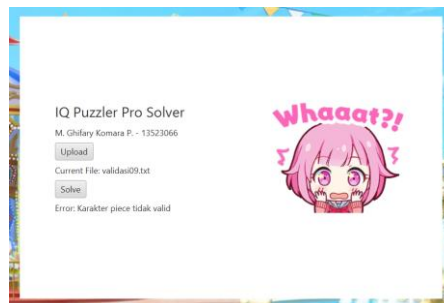
8. Baris 2 tidak bertuliskan
DEFAULT

2 3 3
NORMAL
AA
A
B
CC



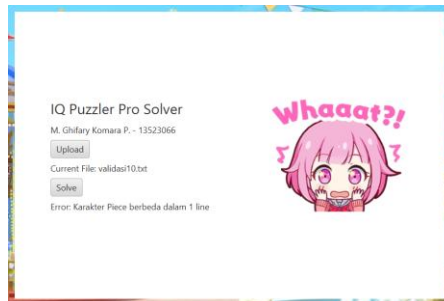
9. Karakter piece bukan
merupakan karakter kapital

2 3 3
DEFAULT
aa
a
B
CC



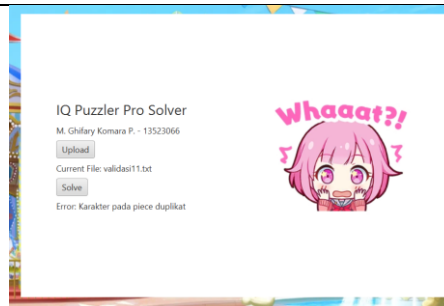
- 10 Terdapat dua karakter berbeda
dalam satu baris pada
pembacaan piece

2 3 3
DEFAULT
AX
A
B
CC



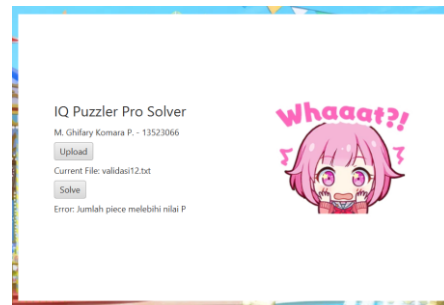
11. Terdapat karakter yang sudah digunakan oleh piece lain

2 3 3
DEFAULT
AA
A
B
AA



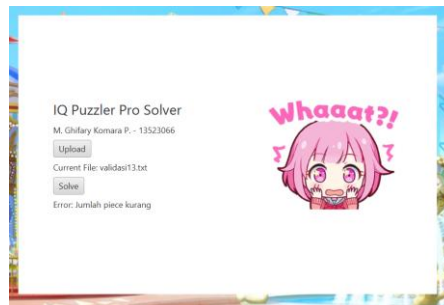
12. Jumlah piece melebihi nilai P

2 3 1
DEFAULT
AA
A
B
CC



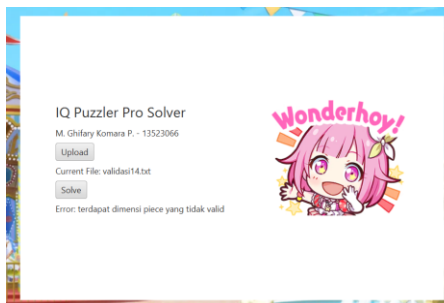
13. Jumlah piece kurang dari nilai P

2 3 5
DEFAULT
AA
A
B
CC



14. Terdapat piece berdimensi lebih besar dari dimensi papan

2 3 3
DEFAULT
AAAA
B
C



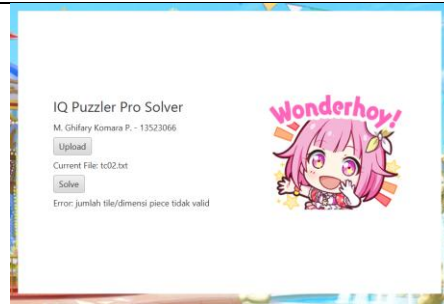
15. Test case 1 (tidak ada solusi)

3 3 2
DEFAULT
AAA
A
A
BB
BB



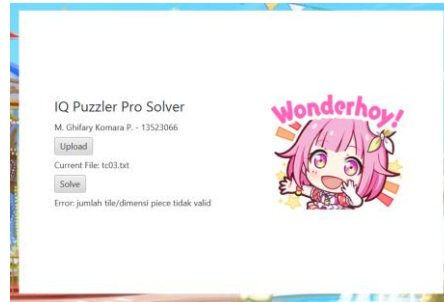
16. Test case 2 (tidak ada solusi)

3 3 2
 DEFAULT
 AAA
 A
 A
 BBB
 BB



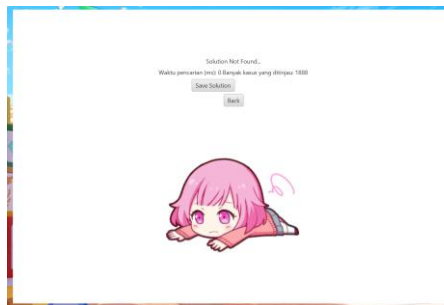
17. Test case 3 (tidak ada solusi)

3 3 2
 DEFAULT
 AAA
 A
 A
 BB
 B



18. Test case 4 (tidak ada solusi)

4 5 4
 DEFAULT
 A
 AAA
 A
 BBB
 BB
 CCC
 C C
 DD
 D
 DD



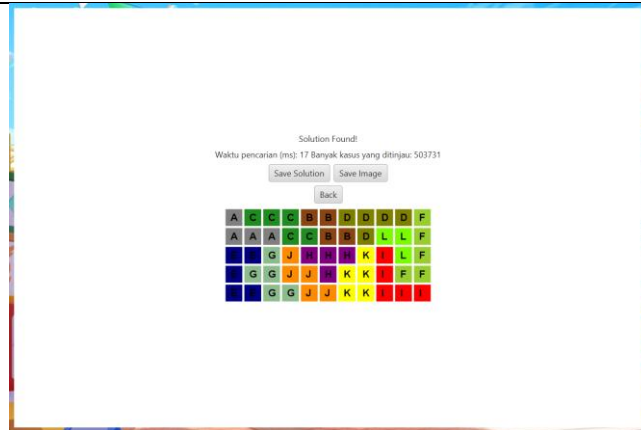
19. Test case 5 (ada solusi)

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



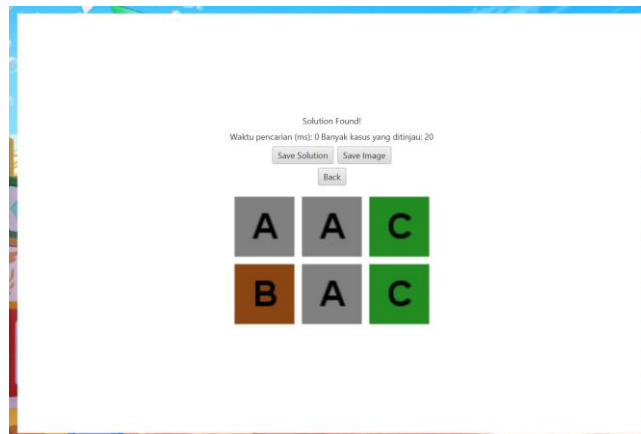
20. Test case 6 (ada solusi)

5 11 12
 DEFAULT
 A
 AAA
 BB
 BB
 CCC
 CC
 DDDD
 D
 EE
 E
 EE
 F
 FFFF
 G
 GG
 GG
 HHH
 H
 I
 I
 III
 J
 JJ
 JJ
 KK
 KK
 K
 L
 LL



21. Test case 7 (ada solusi)

2 3 3
 DEFAULT
 AA
 A
 B
 CC



22. Uji coba penyimpanan file output (tidak ada solusi)

4 5 5
 DEFAULT
 AAA
 AA
 AA
 B
 BB
 CCC

```
bin > data > output > tc08-solution.txt
1 Solusi untuk file tc08.txt
2
3 Tidak ada solusi yang ditemukan
4
5 Waktu pencarian (ms) : 0
6 Banyak kasus ditinjau: 7152
7
```

C
DD
D
EEE

23. Uji coba penyimpanan file output (ada solusi)

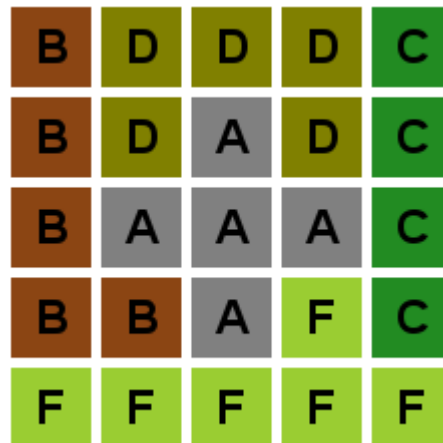
4 5 5
DEFAULT
AAA
AA
AA
B
BB
CCC
C
DD
D
EEE

bin > data > output > tc09-solution.txt

```
1 Solusi untuk file tc09.txt
2
3 AAABB
4 CCAAB
5 CDDAA
6 CDEEE
7
8 Waktu pencarian (ms) : 0
9 Banyak kasus ditinjau: 26
10
```

24. Uji coba penyimpanan file image

5 5 5
DEFAULT
A
AAA
A
BBBB
B
CCCC
FFFFF
F
DDD
D D



V. POTONGAN SOURCE CODE

Berikut merupakan potongan kode program yang telah dikembangkan. Kode keseluruhan dapat diakses melalui repositori yang tersedia pada lampiran

InputData.java

```
1 package Function;
2
3 public class InputData {
4     public int N, M, P;
5     public String S;
6
7     public Board B;
8     public Piece[][] pieces;
9
10    public void printPieces(){
11        for(int i = 0; i < P; i++){
12            pieces[i][0].printShape();
13            System.out.println();
14        }
15    }
16 }
17
```

Board.java

```
package Function;

public class Board {
    public int width = -1;
    public int height = -1;
    public int status = 0; // -1 : no solution, 0 : still searching, 1 : found

    public int total_case = 0;
    public char[][] state;

    // Menempatkan piece P dengan tli pada posisi [row][col]
    public void place(Piece P, int row, int col){
        int i, j;

        // validasi dimensi awal
        if(row + P.height > height || col - P.top_left_index < 0 || col + P.width - P.top_left_index - 1 >= width){
            // total_case++;
            return;
        }

        // validasi apakah seluruh petak dapat ditempati
        for(i=0;i<P.height;i++){
            for(j=0;j<P.width;j++){
                if(P.shape[i][j] != ' ' && state[row + i][col + j - P.top_left_index] != ' '){
                    // total_case ++;
                    return;
                }
            }
        }

        // Simpan Piece
        for(i=0;i<P.height;i++){
            for(j=0;j<P.width;j++){
                if(P.shape[i][j] != ' ' && state[row + i][col + j - P.top_left_index] == ' '){
                    state[row + i][col + j - P.top_left_index] = P.id;
                }
            }
        }
    }

    public void pop(Piece P, int row, int col){
        int i, j;
        for(i=0;i<P.height;i++){
            for(j=0;j<P.width;j++){
                if(P.shape[i][j] != ' ' && state[row + i][col + j - P.top_left_index] == P.id){
                    state[row + i][col + j - P.top_left_index] = ' ';
                }
            }
        }
    }
}
```

Piece.java

```
package Function;

public class Piece {
    public int width = 0;
    public int height = 0;

    public char id = '?';
    public boolean isPlaced = false;

    public int top_left_index = 0;
    public char[][] shape;
    // clockwise rotation
    public Piece rotate90(){
        int i, j;
        Piece NP = new Piece();
        NP.width = height;
        NP.height = width;
        NP.id = id;

        NP.shape = new char[NP.height][NP.width];
        for(i=0; i<NP.width; i++){
            for(j=0; j<NP.height; j++){
                NP.shape[j][NP.width - 1 - i] = shape[i][j];
            }
        }

        for(i=0; i<NP.width; i++){
            if(NP.shape[0][i] == NP.id){
                NP.top_left_index = i;
                break;
            }
        }

        return NP;
    }

    public Piece mirror(){
        int i, j;
        Piece NP = new Piece();
        NP.width = width;
        NP.height = height;
        NP.id = id;

        NP.shape = new char[NP.height][NP.width];
        for(i=0; i<NP.height; i++){
            for(j=0; j<NP.width; j++){
                NP.shape[i][NP.width - 1 - j] = shape[i][j];
            }
        }

        for(i=0; i<NP.width; i++){
            if(NP.shape[0][i] == NP.id){
                NP.top_left_index = i;
                break;
            }
        }

        return NP;
    }
}
```

Solve.java

```
package Function;

public class Solve {
    public static Board BruteForce(InputData data, int row, int col){
        int i, j;
        int currentRow = row;
        int currentCol = col;
        Board board = data.B;
        Piece[][] pieces = data.pieces;
        int total_pieces = data.P;

        // Menentukan indeks paling atas kiri yang dapat diisi
        boolean found = false;
        while(currentCol < board.width && !found){
            if(board.state[currentRow][currentCol] == ' '){
                found = true;
            } else{
                currentCol++;
            }
        }
        if(!found){
            currentRow++;
        }

        while(currentRow < board.height && !found){
            currentCol = 0;
            while(currentCol < board.width && !found){
                if(board.state[currentRow][currentCol] == ' '){
                    found = true;
                } else{
                    currentCol++;
                }
            }
            if(!found){
                currentRow++;
            }
        }

        // Basis: solusi ditemukan
        if(currentRow == board.height && currentCol == board.width && board.state[currentRow-1][currentCol-1] != ' '){
            board.status = 1;
            return board;
        }

        // Mencoba meletakkan piece di dalam board
        for(i=0; i<total_pieces; i++){
            j = 0;
            while(j<8 && pieces[i][0].isPlaced == false){
                board.total_case++;
                if(pieces[i][0].isPlaced == false){
                    board.place(pieces[i][j], currentRow, currentCol);
                    // Jika berhasil diletakkan, lanjutkan ke tahap berikutnya
                    if(board.state[currentRow][currentCol] == pieces[i][j].id){
                        pieces[i][0].isPlaced = true;
                        board = BruteForce(data, currentRow, currentCol);
                        // Jika solusi sudah ditemukan, keluar dari fungsi
                        if(board.status == 1){
                            return board;
                        }
                        // Jika solusi tidak ditemukan, hapus piece yang baru saja diletakkan, tukar dengan konfigurasi lain
                        pieces[i][0].isPlaced = false;
                        board.pop(pieces[i][j], currentRow, currentCol);
                    }
                }
                j++;
            }
        }

        // Tidak ada solusi yang ditemukan
        data.B.status = -1;
        return data.B;
    }
}
```

Validation.java & GUIValidation.java (logika keduanya serupa)

```
package Function;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class Validation {
    public static int max2(int a, int b){
        if(a>b){return a;}
        return b;
    }

    public static void msg(String s){
        System.out.println(s);
    }

    public static boolean txt(String directory, String txt){
        String line;
        String[] currentLine;

        int i, j, k;
        char currentChar, c, firstOcc;

        txt = directory + txt;

        try (BufferedReader br = new BufferedReader(new FileReader(txt))){
            InputData data = new InputData();

            // Validasi baris pertama
            line = br.readLine();
            if(line == null){
                msg(s:"Error: File .txt tidak valid");
                return false;
            }
            line.replaceAll(regex:"\\s+$", replacement:"");

            if(line.length() == 0){
                msg(s:"Error: Terdapat empty line pada file .txt");
                return false;
            }
        }
    }
}
```

```
// Data N M P kurang/lebih/bukan angka
currentLine = line.split(regex: " ");
if(currentLine.length != 3){
    msg(s:"Error: Format N M P tidak valid");
    return false;
}
try{
    data.N = Integer.parseInt(currentLine[0]);
    data.M = Integer.parseInt(currentLine[1]);
    data.P = Integer.parseInt(currentLine[2]);

    if(data.N <= 0 || data.M <= 0 || data.P <= 0){
        msg(s:"Error: N, M, atau P bernilai <= 0");
        return false;
    }
    if(data.P > 26){
        msg(s:"Nilai P > 26");
        return false;
    }
} catch(Exception as){
    msg(s:"Error: Format N M P tidak valid");
    return false;
}

// Baris 2
data.S = br.readLine();
if(data.S == null){
    msg(s:"Error: File .txt tidak valid");
    return false;
}
line.replaceAll(regex:"\\s+$", replacement:"");

if(line.length() ==0){
    msg(s:"Error: Terdapat empty line pada file .txt");
    return false;
}

if(data.S.equals(anObject:"DEFAULT") == false){
    msg(s:"Error: mode permainan tidak valid");
    return false;
}
```

```

// Baris 3-selesai
// Cek height setiap piece
line = br.readLine();
data.pieces = new Piece[data.P][8];
data.pieces[0][0] = new Piece();
i = 0;
currentChar = '?';
try{
    while(line != null){
        line.replaceAll(regex:"\\s+$", replacement:"");

        if(line.length() ==0){
            msg(s:"Error: Terdapat empty line pada file .txt");
            return false;
        }

        firstOcc = '?';

        // Validasi karakter dalam 1 baris
        for(j=0; j<line.length(); j++){
            c = line.charAt(j);
            if(c != ' '){
                currentChar = c;
                // Validasi alfabet kapital
                if(!isUpperCaseAlphabet(c)){
                    msg(s:"Error: Karakter piece tidak valid");
                    return false;
                }
            }

            if(currentChar != data.pieces[i][0].id){
                // Inisialisasi id piece
                if(data.pieces[i][0].id == '?'){
                    data.pieces[i][0].id = currentChar;
                }
                if(firstOcc != currentChar){
                    if(firstOcc == '?'){
                        firstOcc = currentChar;
                    } else{
                        msg(s:"Error: Format piece tidak valid (Karakter berbeda dalam 1 line)");
                        return false;
                    }
                }
            }
        }
    }
}

```

```

        // Cek apakah baris saat ini merupakan piece baru atau sambungan baris sebelumnya;
        if(data.pieces[i][0].id != '?' && data.pieces[i][0].height != 0){
            if(currentChar != data.pieces[i][0].id){
                i++;
                data.pieces[i][0] = new Piece();
                data.pieces[i][0].id = currentChar;
            }
        }
        data.pieces[i][0].height++;
        data.pieces[i][0].width = max2(data.pieces[i][0].width, line.length());

        line = br.readLine();
    }

    if(i < data.P-1){
        msg(s:"Error: Jumlah piece kurang");
        return false;
    }
} catch(Exception as){
    if(i > data.P-1){
        msg(s:"Error: Jumlah piece melebihi nilai P");
        return false;
    }
    msg(s:"Error: Format piece tidak valid");
    return false;
}

// Periksa apakah id duplikat
for(i=0;i<data.P;i++){
    for(k=i+1;k<data.P;k++){
        if(data.pieces[i][0].id == data.pieces[k][0].id){
            msg(s:"Error: Karakter pada piece duplikat");
            return false;
        }
    }
}
} catch (IOException e){
    msg(s:"Error: File txt tidak ditemukan");
    return false;
}

return true;
}

```

FileIO.java

```

package Function;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

public class FileIO {

```

```
public static InputData readInputTxt(String directory, String txt){
    InputData data = new InputData();

    String line;
    String[] currentLine;

    int i = 0;
    char currentChar = '?';
    int j, k;

    boolean isValid = Validation.txt(directory, txt);
    if(!isValid){
        System.out.println(x:"Input teks file tidak valid!");
        return data;
    }

    txt = directory + txt;

    try (BufferedReader br = new BufferedReader(new FileReader(txt))){
        line = br.readLine();
        if(line == null){
            data.N = -1;
            return data;
        }
        currentLine = line.split(regex:" ");

        data.N = Integer.parseInt(currentLine[0]);
        data.M = Integer.parseInt(currentLine[1]);
        data.P = Integer.parseInt(currentLine[2]);
        data.pieces = new Piece[data.P][8];
        data.S = br.readLine();
        data.B = new Board();
        data.B.width = data.M;
        data.B.height = data.N;
        data.B.initiate();

        if(data.S.equals(anObject:"DEFAULT") == false){
            data.N = -1;
            return data;
        }
    }
```

```

// Cek height setiap piece
line = br.readLine();
data.pieces[0][0] = new Piece();

while(line != null){
    for(j=0; j<line.length(); j++){
        char c = line.charAt(j);
        if(c != ' '){
            currentChar = c;
            break;
        }
    }

    if(currentChar != data.pieces[i][0].id){
        if(data.pieces[i][0].id != '?'){i++;}
        data.pieces[i][0] = new Piece();
        data.pieces[i][0].id = currentChar;
    }
    data.pieces[i][0].height++;
    data.pieces[i][0].width = max2(data.pieces[i][0].width, line.length());

    line = br.readLine();
}
} catch (IOException e){
    e.printStackTrace();
}

```

```

// Pembacaan bentuk setiap piece
try (BufferedReader br = new BufferedReader(new FileReader(txt))){
    line = br.readLine();
    line = br.readLine();

    for(i=0; i<data.P; i++){
        data.pieces[i][0].shape = new char[data.pieces[i][0].height][data.pieces[i][0].width];
        for(j=0; j<data.pieces[i][0].height; j++){
            line = br.readLine();

            for(k=0; k<line.length(); k++){
                data.pieces[i][0].shape[j][k] = line.charAt(k);
            }
            for(k=line.length(); k<data.pieces[i][0].width; k++){
                data.pieces[i][0].shape[j][k] = ' ';
            }
        }

        // cek bagian paling "atas kiri" dari piece
        if(j==0){
            for(k=0; k<line.length(); k++){
                if(line.charAt(k) == data.pieces[i][0].id){
                    data.pieces[i][0].top_left_index = k;
                    break;
                }
            }
        }
    }
} catch (IOException e){
    e.printStackTrace();
}

```

```

// Cek apakah jumlah tile pada piece mungkin mengisi board
isValid = Validation.totalTile(data);
if(!isValid){
    data.B.status = -1;
    return data;
}

// Simpan semua variasi rotasi dan pencerminan pada pieces
for(i=0; i<data.P; i++){
    for(j=1; j<4; j++){
        data.pieces[i][j] = data.pieces[i][j-1].rotate90();
    }
    data.pieces[i][4] = data.pieces[i][j-1].mirror();
    for(j=5; j<8; j++){
        data.pieces[i][j] = data.pieces[i][j-1].rotate90();
    }
}

// Optimalisasi simetri
data = Validation.symmetryOptimization(data);

return data;
}

```

```

public static void writeOutputTxt(String filename, InputData data, long searchTime){
    try {
        int i, j;
        Board B = data.B;

        FileWriter writer = new FileWriter("data/output/" + filename.replace(target:".txt", replacement:"-solution.txt"));
        writer.write("Solusi untuk file " + filename.replace(target:"-solution.txt", replacement:".txt") + "\n");
        writer.write(str+"\n");

        if(B.status == -1){
            writer.write(str:"Tidak ada solusi yang ditemukan\n");
        } else{
            for(i=0; i<B.height; i++){
                for(j=0; j<B.width; j++){
                    writer.write(B.state[i][j]);
                }
                writer.write(str+"\n");
            }

            writer.write(str+"\n");
            writer.write("Waktu pencarian (ms) : " + searchTime + "\n");
            writer.write("Banyak kasus ditinjau: " + B.total_case + "\n");

            writer.close();
            System.out.println("Solusi berhasil disimpan dalam file " + filename);
        } catch (IOException e) {
            System.out.println(x:"Terjadi error saat penulisan file");
        }
    }
}

```

```
package Function;

import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

import javax.imageio.ImageIO;

public class ImageSave {
    public static void create(Board B, String filename){
        int i, j, a, b;
        int width, height;
        int cellSize = 40;
        int borderSize = 5;

        width = B.width;
        height = B.height;

        String[] colors = {
            "#808080", "#8b4513", "#228b22", "#808000", "#000080", "#9acd32",
            "#8fbc8f", "#800080", "#ff0000", "#ff8c00", "#ffff00", "#7cfc00",
            "#8a2be2", "#00ff7f", "#4169e1", "#dc143c", "#00ffff", "#0000ff",
            "#ff00ff", "#db7093", "#f0e68c", "#ff1493", "#ffa07a", "#ee82ee",
            "#87cefa", "#ffe4e1"
        };
    }
}
```

```

// Dictionary
Map<Character, String> colorMap = new HashMap<>();
for (i=0; i<26; i++){
    colorMap.put((char) ('A' + i), colors[i]);
}

int imgWidth = width*cellSize + (width+1)*borderSize;
int imgHeight = height*cellSize + (height+1)*borderSize;
BufferedImage image = new BufferedImage(imgWidth, imgHeight, BufferedImage.TYPE_INT_RGB);

Graphics2D g2d = image.createGraphics();
g2d.setColor(Color.WHITE);
g2d.fillRect(x:0, y:0, imgWidth, imgHeight);

// Generating the image
a = 0;
g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);
for(i=borderSize; i<imgHeight; i+= cellSize+borderSize){
    b = 0;
    for(j=borderSize; j<imgWidth; j+= cellSize+borderSize){
        g2d.setColor(Color.decode(colorMap.get(B.state[a][b])));
        g2d.fillRect(j, i, cellSize, cellSize);

        g2d.setColor(Color.BLACK);
        g2d.setFont(new Font(name:"Arial", Font.BOLD, size:24));
        g2d.drawString(Character.toString(B.state[a][b]), j+ cellSize/2 - 8, i + cellSize/2 + 8);
        b++;
    }
    a++;
}

// Save
File outputFile = new File("data/output/" + filename.replace(target:".txt", replacement:"") + ".png");
outputFile.getParentFile().mkdirs();
try {
    ImageIO.write(image, formatName:"png", outputFile);
    System.out.println("Image disimpan sebagai " + filename.replace(target:".txt", replacement:"") + ".png");
} catch (IOException e) {
    System.out.println("Error: " + e.getMessage());
}
}
}

```

Main.java (hanya potongan kode)

```

StackPane root = new StackPane(grid, secondScreen);
root.setAlignment(grid, Pos.CENTER);
root.setStyle(
    "-fx-background-image: url('file:img/background.png');" +
    "-fx-background-size: cover;"
);

Scene scene = new Scene(root, 1200, 800);

primaryStage.setTitle("IQ Puzzler Pro Solver");
primaryStage.setScene(scene);
primaryStage.setMaximized(true);
primaryStage.getIcons().add(new Image("img/wonderhoy.png"));
primaryStage.show();
}

```

Controller.java (hanya potongan kode)

```
public class Controller {  
    String myFile = "";  
    boolean isImageSaved = false;  
    InputData global;  
    long searchTime;  
  
    public void uploadFile(ImageView imageView, Label currentFile, Label errorMessage){  
  
    public void fullSolve(ImageView imageView, Label errorMessage, GridPane grid,  
        VBox secondScreen, Label solution_state, Label solveInfo, Button saveImageButton,  
        ImageView solutionImg){  
  
    public void deleteImage(){  
  
    public void saveTxt(){  
}
```

LAMPIRAN

Tautan menuju repositori: https://github.com/Sanesasaha/Tucil1_13523066

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 <i>Graphical User Interface</i> (GUI)	✓	
6	Program dapat menyimpan solusi dalam bentuk file gambar	✓	
7	Program dapat menyelesaikan kasus konfigurasi <i>custom</i>		✓
8	Program dapat menyelesaikan kasus konfigurasi Piramida (3D)		✓

9	Program dibuat oleh saya sendiri	✓	
---	----------------------------------	---	--