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

Pemecahan Teka-Teki IQ Puzzler Pro dengan Algoritma Brute-Force

Disusun Oleh :

Muhammad Jibril Ibrahim 13523085

K - 02

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2025

DAFTAR ISI

BAB 1: PENDAHULUAN

1.1. Deskripsi Tugas

Gambar 1 Permainan IQ Puzzler Pro
(Sumber: https://www.smartgamesusa.com)

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.

Permainan dimulai dengan papan yang kosong. Pemain dapat meletakkan blok puzzle sedemikian sehingga tidak ada blok yang bertumpang tindih (kecuali dalam kasus 3D). Setiap blok puzzle dapat dirotasikan maupun dicerminkan. Puzzle dinyatakan selesai jika dan hanya jika papan terisi penuh dan seluruh blok puzzle berhasil diletakkan.

Tujuan dari tugas ini adalah untuk menemukan cukup satu solusi dari permainan IQ Puzzler Pro dengan menggunakan *algoritma Brute Force*, atau menampilkan bahwa solusi tidak ditemukan jika tidak ada solusi yang mungkin dari puzzle.

BAB 2: PENYELESAIAN & ALGORITMA

2.1. Algoritma Bruteforce

Penyelesaian puzzle ini yaitu sebagai berikut

1. Terima input dari file txt dan menginisialisasi *board* dan *piece* sesuai dengan input yang diberikan.
2. Program akan melakukan exhaustive search untuk menemukan kombinasi setiap *piece* beserta rotasi dan balikan-nya yang muat pada *board*.
3. Exhaustive search ini dilakukan secara rekursif dengan meletakkan piece pada board mulai dari ujung kiri atas hingga ujung kanan bawah. Fungsi rekursif ini memiliki basis jika semua kotak pada *board* sudah terisi.
4. Fungsi ini pertama mencoba semua rotasi dan balikkan suatu piece, jika bisa diletakkan maka lanjut ke piece selanjutnya dan pada lokasi kosong selanjutnya. Jika tidak bisa diletakkan maka, pada lokasi yang sama, dicoba piece berikutnya beserta rotasi dan balikkan-nya.
5. Jika setelah mencoba semua piece yang dapat diletakkan namun belum memenuhi *board* maka mundur atau *backtrack* ke piece sebelumnya dengan melepaskan *piece* pada *board* dan mencoba rotasi, balikan, atau *piece* lain.
6. Jika telah melakukan *Exhaustive search* namun *board* tetap tidak terisi maka puzzle yang diberikan tidak memiliki solusi.

2.2. Pseudocode

```
FUNCTION Solve(x, y, used_piece):
   IF x >= n THEN
      RETURN TRUE  // Solution found

   iteration = iteration + 1  // Count iterations

   FOR piece_idx FROM 0 TO p - 1:  // try each piece
      FOR rot FROM 0 TO 3:  // rotated
         FOR flip FROM 0 TO 1:  // flipped
            IF used_piece[piece_idx] IS FALSE THEN  // check if used
               IF placePiece(x, y, piece_idx, rot, flip) IS TRUE THEN  // Check if can be placed
                  used_piece[piece_idx] = TRUE

                  // Find next empty position
                  nextX = x
                  nextY = y
                  WHILE board[nextX][nextY] ≠ 0:
                     nextY = nextY + 1
                     IF nextY >= m THEN
                        nextX = nextX + 1
```

```
                nextY = 0
            IF nextX >= n THEN
                RETURN TRUE  // Board is filled

        IF Solve(nextX, nextY, used_piece) IS TRUE THEN
            RETURN TRUE

        removePiece(x, y, piece_idx, rot, flip)
        used_piece[piece_idx] = FALSE

RETURN FALSE  // No valid configuration found
```

BAB 3: IMPLEMENTASI ALGORITMA

3.1. Struktur Repository

Note: name ending in / means its a folder with something (insignificant) in it

```
tucil1_13523085_Iq_Puzzler_Pro_Solver
├── app
│   ├── bin
│   │   ├── main
│   │   │   └── iq_puzzler_solver
│   │   │       ├── gui
│   │   │       │   └── AppGUI.class
│   │   │       ├── Main.class
│   │   │       └── primordials
│   │   │           ├── Board.class
│   │   │           ├── Input.class
│   │   │           └── Piece.class
│   │   └── test/
│   ├── build/
│   ├── build.gradle.kts
│   └── src
│       ├── main
│       │   └── java
│       │       └── iq_puzzler_solver
│       │           ├── gui
│       │           │   └── AppGUI.java
│       │           ├── Main.java
│       │           └── primordials
│       │               ├── Board.java
│       │               ├── Input.java
│       │               └── Piece.java
│       └── test/
├── assets
│   └── puzzle.jpg
├── docs
│   └── Tucil1_K2_13523085_Muhammad Jibril Ibrahim
├── gradle/
├── gradlew
├── gradlew.bat
├── README.md
└── settings.gradle.kts
```

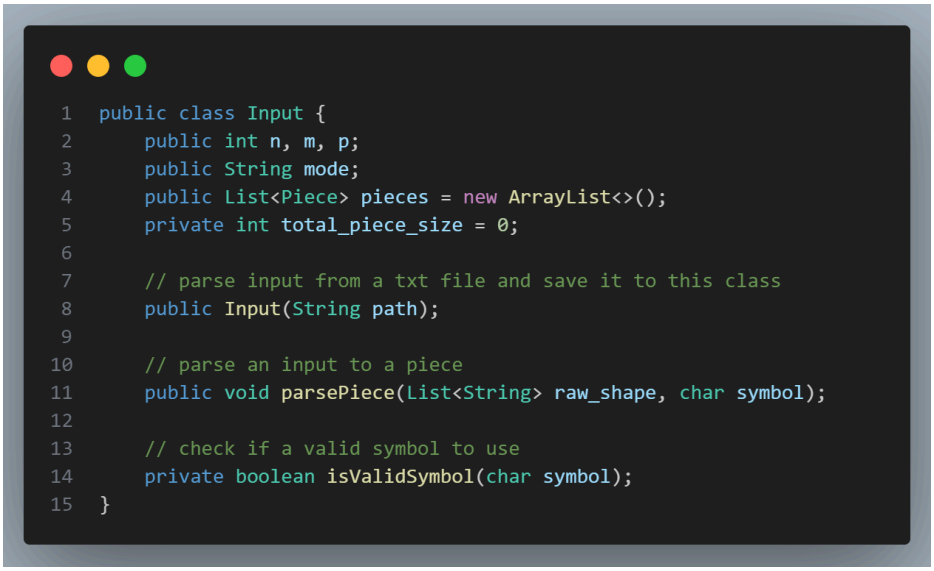
Notes:

1. Folder **app/src** berisi source code program.
2. Folder **app/bin** berisi executable file
3. Folder **doc** berisi laporan.
4. Folder **gradle** berisi wrapper untuk Graphical User Interface (GUI)
5. Folder **assets** berisi file-file lain.

3.2. Source Code

Disini saya hanya memasukkan algoritma dan class utama yaitu parsing input dan solver serta kelas input, piece, dan board. Untuk Source code lengkap dapat dilihat di repository

3.2.1. Input Class



```java
public class Input {
    public int n, m, p;
    public String mode;
    public List<Piece> pieces = new ArrayList<>();
    private int total_piece_size = 0;

    // parse input from a txt file and save it to this class
    public Input(String path);

    // parse an input to a piece
    public void parsePiece(List<String> raw_shape, char symbol);

    // check if a valid symbol to use
    private boolean isValidSymbol(char symbol);
}
```

3.2.1.1. Input()

```java
public Input(String path) {
        try {
            File f = new File(path);
            Scanner s = new Scanner(f);
            int i = 0;
            char cur_piece_symbol = '-';
            List<String> cur_piece_shape = new ArrayList<>();

            while (s.hasNextLine()) {
                // get the n, m, and p values from the txt file
                // n and m is the board size (n*m)
                // p is the number of pieces
                if (i == 0) {
                    String line = s.nextLine();
                    String[] vals = line.split(" ");
                    int[] values;
                    if (vals.length != 3) throw new IllegalArgumentException("Error: Invalid N, M, and P Value");
                    try {
                        values = Arrays.stream(vals).mapToInt(Integer::parseInt).toArray();
                    } catch (Exception e) {
                        e.printStackTrace();
                        throw new IllegalArgumentException("Error: Invalid N, M, and P Value");
                    }
                    this.n = values[0];
                    this.m = values[1];
                    this.p = values[2];
                    if (this.p > 26) throw new IllegalArgumentException("Error: Invalid P Value. can't be more than 26");
                    i++;
                }
                // get the mode of the board ("DEFAULT", "CUSTOM", "PYRAMID") i aint doin pyramid tho
                else if (i == 1) {
                    String line = s.nextLine();
                    line = line.strip();

                    if (!"DEFAULT".equals(line) && !"CUSTOM".equals(line)) {
                        throw new IllegalArgumentException("Error: Invalid board mode. valid mode: 'DEFAULT', 'CUSTOM'.");
                    }

                    this.mode = line;
                    i++;
                }
                // get the pieces
                else {
                    String line = s.nextLine();

                    if (line.isBlank()) throw new IllegalArgumentException("Error: Empty line in Piece");

                    if (line.indexOf(cur_piece_symbol) == -1) {
                        // create new piece
                        if (cur_piece_symbol != '-') {
                            parsePiece(cur_piece_shape, cur_piece_symbol);
                            cur_piece_shape.clear(); // clear the temp piece
                        }

                        // change 'cur_piece_symbol' to the new symbol
                        boolean isValidPieceSymbol = false;
                        for (char c : line.toCharArray()) {
                            if (isValidSymbol(c)) {
                                cur_piece_symbol = c;
                                isValidPieceSymbol = true;
                                break;
                            }
                        }

                        // invalid symbol (A-Z only)
                        if (!isValidPieceSymbol) throw new IllegalArgumentException("Error: Invalid piece symbol. (A-Z only)");
                    }

                    char cur_symbol = cur_piece_symbol;
                    if (line.chars().allMatch(c -> c == cur_symbol || c == ' '))  {
                        // add line to current temporary shape
                        cur_piece_shape.add(line.stripTrailing());

                    } else {
                        // Multiple symbols in a single line: AAAB
                        System.out.println(line);
                        throw new IllegalArgumentException("Error: Invalid piece shape. Multiple symbols in a single line");
                    }
                }
            }
            // parse last piece
            parsePiece(cur_piece_shape, cur_piece_symbol);

            // check if total piece size is equal to board size
            if (this.n * this.m != this.total_piece_size) {
                throw new IllegalArgumentException("Error: Total piece size is not equal to board size");
            }

            // check if number of piece is equal to the given value
            if (this.pieces.size() != this.p) {
                throw new IllegalArgumentException("Error: Number of piece is not equal to the given value P");
            }

            if (this.pieces.size() > 26) throw new IllegalArgumentException("Error: Pieces can't be more than 26");

            s.close();
        } catch (IllegalArgumentException e) {
            throw e;
        } catch (Exception e) {
            e.printStackTrace();
            throw new IllegalArgumentException("Error: Something went wrong. (bruhhh TC nya apa cik)");
        }
    }
```

3.2.1.2. parsePiece()

```java
public void parsePiece(List<String> raw_shape, char symbol) {
        int n, m = 0;
        boolean[][] shape;
        boolean isValid = true;

        // find n and m
        n = raw_shape.size();
        for (String s : raw_shape) {
            m = (m < s.length()) ? s.length() : m;
        }

        // if piece is bigger than board
        if (n > this.n || m > this.m) throw new IllegalArgumentException("Error: Invalid piece shape. Piece is bigger than the board");

        shape = new boolean[n][m];
        int startX = -1, startY = -1;

        // assign shape
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < m; j++) {
                if (raw_shape.get(i).length() > j && raw_shape.get(i).charAt(j) != ' ') {
                    shape[i][j] = true;
                    this.total_piece_size++;
                    startX = i;
                    startY = j;
                }
                else shape[i][j] = false;
            }
        }

        boolean[][] connected = new boolean[n][m];

        // check if shape is connected
        Queue<Point> q = new LinkedList<>();
        q.add(new Point(startX, startY));

        while (!q.isEmpty()) {
            Point cur_point = q.poll();
            int x = cur_point.x;
            int y = cur_point.y;

            // System.err.println("Symbol: " + symbol + ", x: " + x + ", y: " + y);
            connected[x][y] = true;

            // horizontal vertical
            if (x > 0 && shape[x - 1][y] && !connected[x - 1][y]) q.add(new Point(x - 1, y));
            if (x < n - 1 && shape[x + 1][y] && !connected[x + 1][y]) q.add(new Point(x + 1, y));
            if (y > 0 && shape[x][y - 1] && !connected[x][y - 1]) q.add(new Point(x, y - 1));
            if (y < m - 1 && shape[x][y + 1] && !connected[x][y + 1]) q.add(new Point(x, y + 1));

        }

        for (int i = 0; i < n; i++) {
            for (int j = 0; j < m; j++) {
                if (shape[i][j] != connected[i][j]) {
                    throw new IllegalArgumentException("Error: Invalid piece shape. Disconnected shape");
                }
            }
        }

        this.pieces.add(new Piece(n, m, shape, symbol));
    }
```

3.2.2. Piece Class

```java
public class Piece {
    int n, m;
    public char symbol;
    boolean[][] shape;
    static int numOfPieces = 0;

    public Piece(int n, int m, boolean shape[][], char symbol);

    public Piece flip();

    public Piece rotate();

}
```

3.2.2.1. Piece()

```java
public Piece(int n, int m, boolean shape[][], char symbol) {
        this.n = n;
        this.m = m;
        this.shape = new boolean[n][m];
        this.symbol = symbol;

        for (int i = 0; i < n; i++) {
            System.arraycopy(shape[i], 0, this.shape[i], 0, m);
        }

        Piece.numOfPieces++;
    }
```

3.2.2.2. flip()

```java
public Piece flip() {
        boolean newShape[][] = new boolean[this.n][this.m];

        for (int i = 0; i < this.n; i++) {
            for (int j = 0; j < this.m; j++) {
                newShape[i][j] = this.shape[i][this.m - j - 1];
            }
        }

        return new Piece(this.n, this.m, newShape, this.symbol);
    }
```

3.2.2.3. rotate()

```java
public Piece rotate() {
        boolean newShape[][] = new boolean[this.m][this.n];

        for (int i = 0; i < this.m; i++) {
            for (int j = 0; j < this.n; j++) {
                newShape[i][j] = this.shape[this.n - j - 1][i];
            }
        }

        return new Piece(this.m, this.n, newShape, this.symbol);
    }
```

3.2.3. Board Class

```java
public class Board {
    public int n, m, p;
    public int[][] board;
    public List<Piece> pieces;
    public List<Color> colors;
    public boolean isSolved = false;
    public int iteration = 0;
    public long exec_time;

    // create board with pieces
    public Board(int n, int m, List<Piece> pieces);

    public Board(String pathToFile);

    // place a piece to the board
    private boolean placePiece(int x, int y, int piece_idx, int rot, int flip);

    // remove piece from the board (assuming you dont need to check cuz if placed means its valid, no?)
    private void removePiece(int x, int y, int piece_idx, int rot, int flip);

    private boolean Solve(int x, int y, boolean[] used_piece);

    public boolean solve();

    // prints the board with the symbols and distinct colors
    public void printBoard();

    // save the solution to txt
    public void saveToTxt(String path);

    // save solution to png image
    public void saveToImg(String path);

}

```

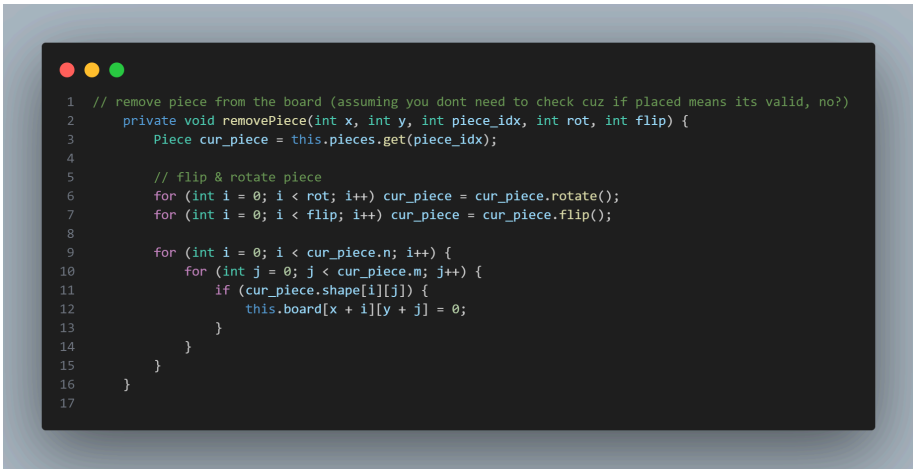
3.2.3.1. Board()

```java
// create board with path to txt file
public Board(String pathToFile) {
    // String currentDir = Paths.get("").toAbsolutePath().toString();
    System.out.println(pathToFile);
    Input inp = new Input(pathToFile);
    this.n = inp.n;
    this.m = inp.m;
    this.p = inp.p;
    this.board = new int[n][m];
    this.pieces = inp.pieces;
    this.colors = new ArrayList<>();

    // generate color
    Random random = new Random();
    float hue = random.nextFloat();
    for (int i = 0; i < 26; i++) {
        hue += 0.618033988749895f;
        hue %= 1;
        Color color = Color.getHSBColor(hue, 0.9f, 0.9f);

        this.colors.add(color);
    }
}

```

3.2.3.2. placePiece()

```java
// place a piece to the board
    private boolean placePiece(int x, int y, int piece_idx, int rot, int flip) {
        Piece cur_piece = this.pieces.get(piece_idx);

        // flip & rotate piece
        for (int i = 0; i < rot; i++) cur_piece = cur_piece.rotate();
        for (int i = 0; i < flip; i++) cur_piece = cur_piece.flip();

        // check if can be place
        for (int i = 0; i < cur_piece.n; i++) {
            for (int j = 0; j < cur_piece.m; j++) {
                if (cur_piece.shape[i][j]) {
                    if (x + i < 0 || x + i >= n || y + j < 0 || y + j >= m) return false;
                    if (this.board[x + i][y + j] != 0) return false;
                }
            }
        }

        // if valid, put the piece
        for (int i = 0; i < cur_piece.n; i++) {
            for (int j = 0; j < cur_piece.m; j++) {
                if (cur_piece.shape[i][j]) {
                    this.board[x + i][y + j] = piece_idx + 1;
                }
            }
        }

        return true;
    }

```

3.2.3.3. removePiece()



```java
// remove piece from the board (assuming you dont need to check cuz if placed means its valid, no?)
    private void removePiece(int x, int y, int piece_idx, int rot, int flip) {
        Piece cur_piece = this.pieces.get(piece_idx);

        // flip & rotate piece
        for (int i = 0; i < rot; i++) cur_piece = cur_piece.rotate();
        for (int i = 0; i < flip; i++) cur_piece = cur_piece.flip();

        for (int i = 0; i < cur_piece.n; i++) {
            for (int j = 0; j < cur_piece.m; j++) {
                if (cur_piece.shape[i][j]) {
                    this.board[x + i][y + j] = 0;
                }
            }
        }
    }

```

3.2.3.4. Solve()

```java

private boolean Solve(int x, int y, boolean[] used_piece) {
    if (x >= this.n) return true;
    this.iteration++;
    for (int piece_idx = 0; piece_idx < this.p; piece_idx++) {      // every piece,
        for (int rot = 0; rot < 4; rot++) {                         // rotated,
            for (int flip = 0; flip < 2; flip++) {                  // flipped,
                if (!used_piece[piece_idx]) {                       // check if used,
                    if (placePiece(x, y, piece_idx, rot, flip)) {   // placed if can.
                        used_piece[piece_idx] = true;

                        int nextX = x;
                        int nextY = y;
                        // find next point
                        while (this.board[nextX][nextY] != 0) {
                            nextY++;
                            if (nextY >= this.m) {
                                nextX++;
                                nextY = 0;
                            }

                            if (nextX >= this.n) return true;
                        }

                        // System.out.println("currently iterating x: " + nextX + ", y: " + nextY);
                        if (this.Solve(nextX, nextY, used_piece)) return true;
                        removePiece(x, y, piece_idx, rot, flip);
                        used_piece[piece_idx] = false;
                    }
                }
            }
        }
    }

    return false;
}

```

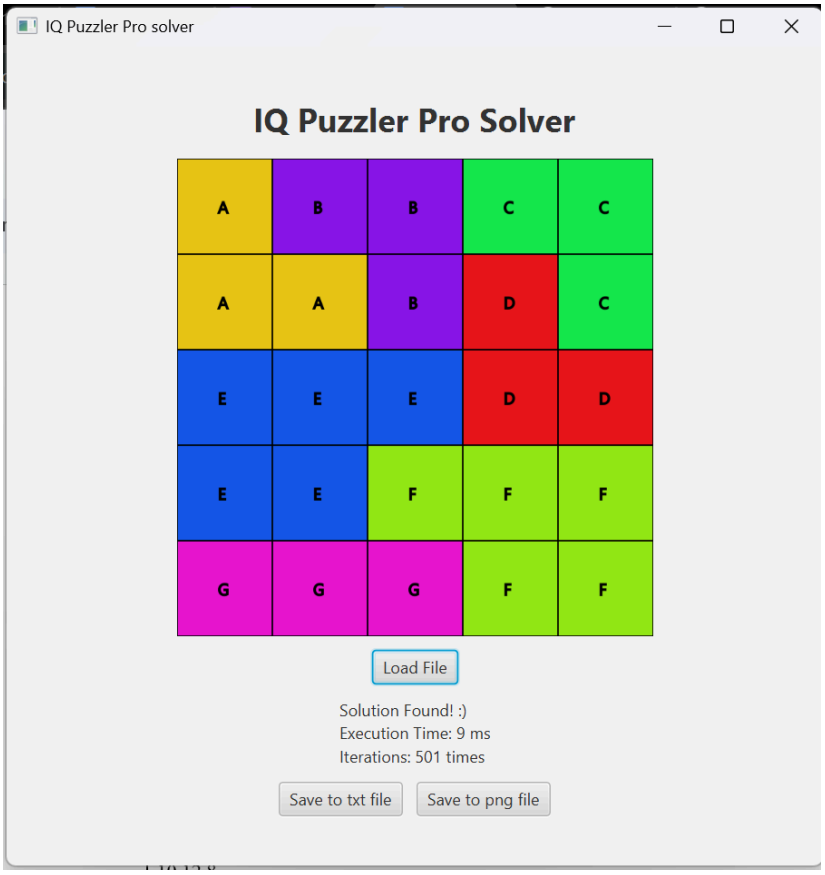
BAB 4: TESTING

4.1. Test Case 1

Input :

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

Output :



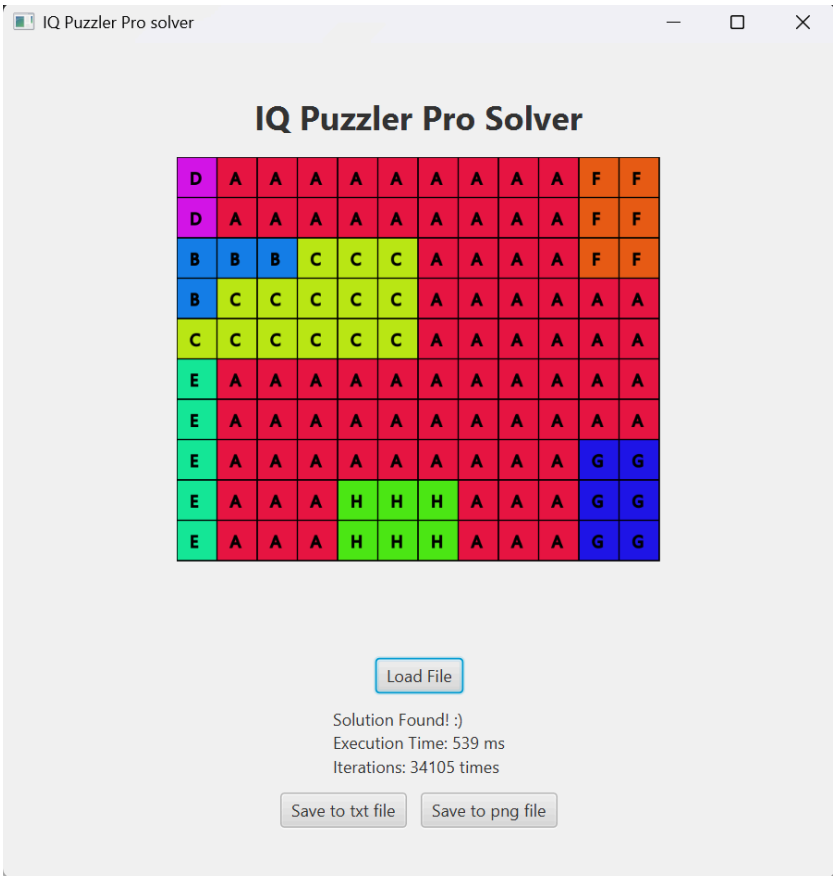
4.2. Test Case 2

Input :

```
10 12 8
DEFAULT
 AAAAAAAAA
 AAAAAAAAA
    AAAA
    AAAAAA
    AAAAAA
 AAAAAAAAAAA
 AAAAAAAAAAA
 AAAAAAAAA
 AAA  AAA
 AAA  AAA
BBB
B
  CCC
 CCCCC
```

```
CCCCCC
DD
EEEEE
FFF
FFF
GGG
GGG
HHH
HHH
```

Output :



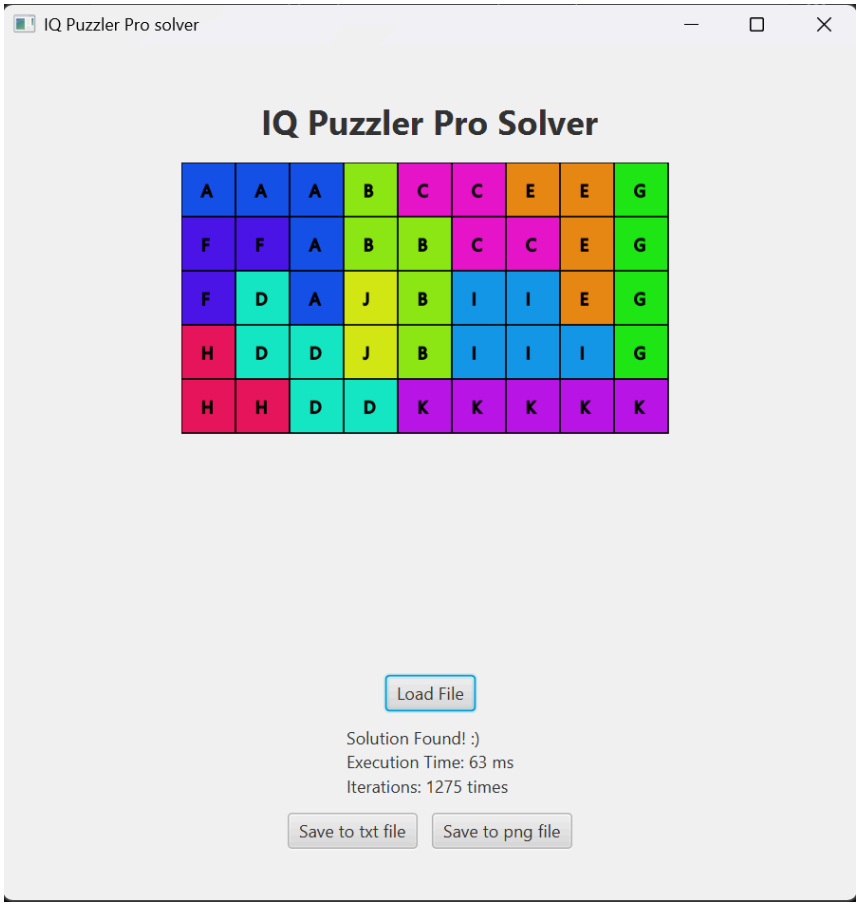
4.3. Test Case 3

Input :

```
5 9 11
DEFAULT
AAA
```

```
 A
  A
 B
BB
B
B
CC
 CC
DD
 DD
  D
EE
E
E
FF
F
G
G
G
G
HH
H
I
II
II
JJ
KKKKK
```

Output :



4.4. Test Case 4

Input :

```
13 6 26
DEFAULT
A
AA
B
BB
C
CC
D
DD
E
EE
F
FF
G
GG
```

```
H
HH
I
II
J
JJ
K
KK
L
LL
M
MM
N
NN
O
OO
P
PP
Q
QQ
R
RR
S
SS
T
TT
U
UU
V
VV
W
WW
X
XX
Y
YY
Z
ZZ
```

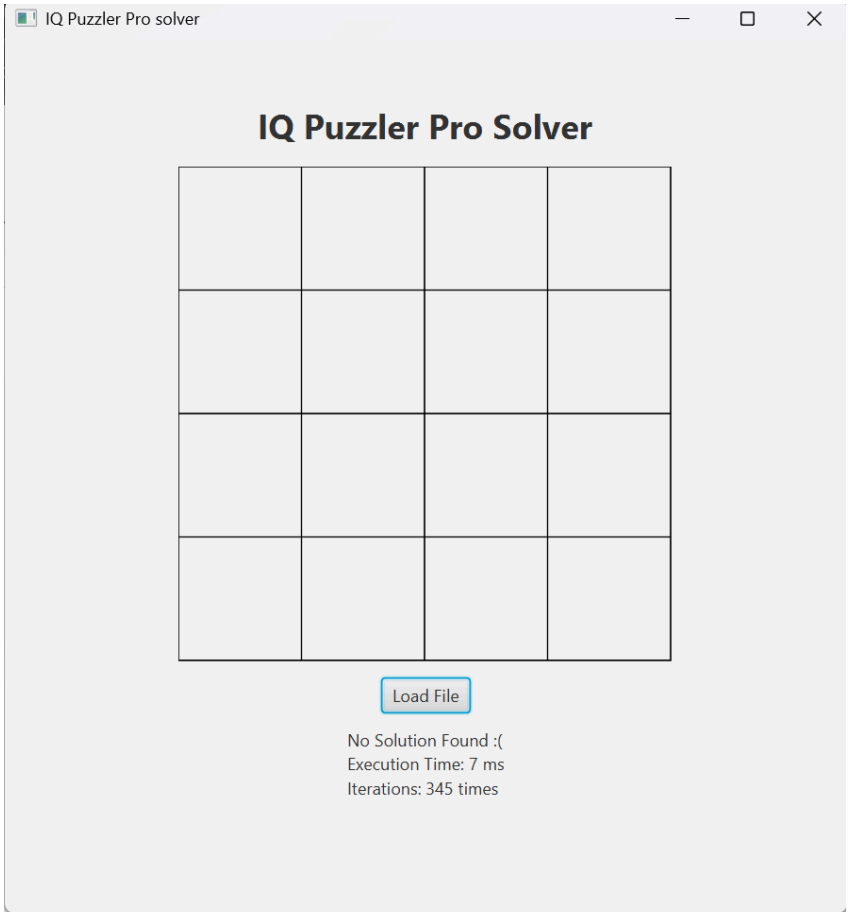
Output :

4.5. Test Case 5

Input :

```
4 4 4
DEFAULT
AAAA
BB
 BB
CC
 CC
DD
 DD
```

Output :


IQ Puzzler Pro solver
IQ Puzzler Pro Solver
Load File
No Solution Found :(
Execution Time: 7 ms
Iterations: 345 times

LAMPIRAN

Github Repository

Program dapat diakses pada https://github.com/BoredAngel/tucil1_13523085_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	✓	