LAPORAN TUGAS KECIL 1

MATA KULIAH IF 2211 STRATEGI ALGORITMA

PENYELESAIAN PERMAINAN "IQ PUZZLER PRO" MENGGUNAKAN ALGORITMA BRUTE FORCE

Dosen Pengampu

Dr. Nur Ulfa Maulidevi, S.T, M.Sc.

Disusun Oleh:

Adhimas Aryo Bimo 13523052

SEKOLAH TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

FEBRUARI 2025

DAFTAR ISI

Isi

No table of contents entries found.

BAB I

LATAR BELAKANG

I. IQ PUZZLER PRO

Gambar 1.1

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.

Gambar 1.2 Mainan Terselesaikan

II. BRUTE FORCE

```
[80][http-get-form] host: 192.168.100.155   login: admin   password: password
[80][http-get-form] host: 192.168.100.155   login: admin   password: p@ssword
[80][http-get-form] host: 192.168.100.155   login: admin   password: 12345
[80][http-get-form] host: 192.168.100.155   login: admin   password: 1234567890
[80][http-get-form] host: 192.168.100.155   login: admin   password: Password
[80][http-get-form] host: 192.168.100.155   login: admin   password: 123456
[80][http-get-form] host: 192.168.100.155   login: admin   password: 1234567
[80][http-get-form] host: 192.168.100.155   login: admin   password: 12345678
[80][http-get-form] host: 192.168.100.155   login: admin   password: 1q2w3e4r
[80][http-get-form] host: 192.168.100.155   login: admin   password: 123
[80][http-get-form] host: 192.168.100.155   login: admin   password: 1
[80][http-get-form] host: 192.168.100.155   login: admin   password: 12
1 of 1 target successfully completed, 12 valid passwords found
Hydra (http://www.thc.org/thc-hydra) finished at 2017-07-27 15:28:24
```

Gambar 1.3 Contoh Algoritma Brute Force

Dalem dunia komputer sains, brute force merupakan suatu metode pemecahan persoalan yang cukup umum. Brute force memiliki ciri-ciri yakni mengecek semua kemungkinan kejadian tak peduli apakah kejadian tersebut memenuhi persoalan atau tidak. Algoritma brute force akan mencari setiap opsi hingga jawaban dari suatu masalah dapat ditemukan.

Pendekatan bruteforce cukup umum ditemukan, bahkan tidak disadari kita lakukan dalam menyelesaikan berbagai persoalan pemrograman. Namun, karena pencarian solusi yang terlalu mendalam, algoritma ini cenderung memiliki kompleksitas yang cukup tinggi dan tidak efisien untuk menyelesaikan permasalahan yang besar.

BAB II

DESKRIPSI & IMPLEMENTASI ALGORITMA

I. Alur Berpikir

Dalam menyelesaikan permasalahan yang tidak memiliki langkah pasti, algoritma Brute-Force menjadi salah satu alternatif yang dapat digunakan dalam menyelesaikan persoalan tersebut. Seperti yang dipaparkan pada penjelasan mengenai peraturan permainan IQ PUZZLER, kita dapat fokus pada beberapa aspek yang krusial untuk menjadi landasan berpikir dalam merencanakan program ini.

Pertama, blok. Bentuk tiap blok pada permainan ini acak, Penempatan blok tidak bergantung pada posisi mana pun dan akan valid jika tiap satuan blok memenuhi tempat pada papan. Dalam hal ini, perlu untuk mengecek semua kemungkinan posisi yang valid bagi blok untuk ditempatkan pada papan. Semua posisi berpotensi untuk menjadi tempat yang pas bagi blok

Kedua, transformasi blok. Blok tidak hanya ditempatkan, melainkan dapat diubah orientasi dan juga dapat dicerminkan. Dalam hal ini, dalam semua kondisi bagi yang valid ataupun tidak, blok memiliki alternatif cara penempatan. Oleh karena itu, diperlukan untuk mengecek transformasi orientasi tiap blok pada tiap posisi untuk mencari solusi pada permainan ini.

Ketiga, metode penempatan blok. Dalam hal ini ada beberapa alternatif bagaimana cara menyusun blok yang baik. Misal, dengan meletakkan posisi yang rumit pada pojok papan, atau mulai dengan bentuk blok yang paling besar. Namun, karena algoritma yang digunakan pada persoalan ini berfokus pada Brute-Force, metode penempatan yang paling mendekati algoritma itu adalah dengan menggunakan Backtracking. Jika semua posisi yang telah dicoba oleh blok ternyata tidak memenuhi aturan penempatan blok. blok sebelumnya akan diubah cara penempatannya dengan mencari tempat penempatan lain yang valid. Hal ini dapat digunakan dengan menggunakan backtracing pada posisi terakhir blok ditempatkan secara valid.

Ibaratkan ketika manusia mencoba memainkan puzzle, kita akan mencoba memasangkan blok yang pas pada papan dan sesuai dengan aturan permainan. Jika ada blok yang tidak dapat dipasang, maka blok lain akan dilepas dan diubah entah posisi maupun orientasinya agar blok yang tidak dipasang itu dapat dipasang pada papan.

II. Kode Program

BruteForce.java

```java
package solver;

import pkg.*;
import java.util.*;
import java.io.*;

public class BruteForce {
  private Board board;
  private List<Block> listBlocks;
  private int totalCase = 0;
  private long startTime;
  private long endTime;

  public BruteForce(Board board, List<Block> listBlocks) {
    this.board = board;
    this.listBlocks = listBlocks;
  }

  // Getter
  public int getTotalCase() {
    return totalCase;
  }

  public double getTime() {
    return endTime - startTime;
  }

  public Board solve() {
    System.out.println("Menggunakan algoritma Brute Force...");
    startTime = System.currentTimeMillis();
    int currentBlockIndex = 0;

    if(!checkBoxAndArea()){
      System.out.println("Total kotak tiap Block tidak sama dengan area Board");
      return null;
    }

    boolean result =  tryAllPossibility(currentBlockIndex);
    endTime = System.currentTimeMillis();
    System.out.println("Waktu pencarian : " + (endTime - startTime) + " ms");
    System.out.println("Banyak kasus yang ditinjau : "+ totalCase);

    if (result) {
      System.out.println("Solusi ditemukan!");
      board.showBoard();
      return board;
    } else {
      System.out.println("Solusi tidak ditemukan!");
      board.showBoard();
      return null;
    }
  }
```


Gambar 2.1 Constructor, Getter, dan Metode Pemanggilan BruteForce

BruteForce.java

```java
public boolean tryAllPossibility(int blockIndex){
    // Jika semua block sudah ditempatkan
    if(board.isBoardSolved()){
      return true;
    }

    if(blockIndex ≥ listBlocks.size()){
      return false;
    }

        System.out.println("Test ke: " + totalCase);
        board.showBoard();
        System.out.println();

    Block currentBlock = listBlocks.get(blockIndex);
    Block originalState = currentBlock;
    for(int t = 0; t < 8 ; t++){
      for(int i = 0; i ≤ board.getRowsCount() - currentBlock.getRows(); i++){
        for(int j = 0; j ≤ board.getColsCount() - currentBlock.getCols(); j++){
          // Cek 8 transformasi 4 rotasi biasa dan 4 rotasi yang dicerminkan
            if(isValidPlace(currentBlock, i,j)){
              totalCase++;
              placeBlock(currentBlock, i, j);
              // Secara rekursif tempatkan blok lagi
              if(tryAllPossibility(blockIndex+1)){
                return true;
              }
              // Jika salah, hapus block sebelumnya
              removeBlock(currentBlock, i, j);
            }
          }
        }
      if(t == 3){
        currentBlock.mirrorBlock();
      }else {
        currentBlock.rotateBlock();
      }
    }

    currentBlock.setAllData(originalState.getAllData());
    listBlocks.set(blockIndex, currentBlock);
    return false;
  }
```

snappify.com

Gambar 2.2 Rekursif dan Backtracking untuk Mencari Semua Solusi

Pada kedua gambar diatas, dituliskan program dari hasil pemikiran pada bagian Alur Berpikir. Berikut merupakan rincian penjelasan program.

1. Inisiasi
   Untuk melalukan pemnggilan kelas Brute Force, perlu diinisasi dengan koleksi dari semua blok (listBlocks) dan papan (Board). Hal ini diperlukan sebagai wahana untuk melakukan percobaan untuk mencari solsui pada permainan

2. Cek Tile Block dan Area Board
   Sebelum melakukan operasi, kita dapat mengecek terlebih dahulu apakah solusi yang ada pada input valid. Setidaknya, dengan mengecek total kepingan blok dengan area papan, kita dapat mengetahui bahwa jika keduanya cocok, kemungkinan akan ada solusi yang bisa dicari. Jika terdapat perbedaan, maka tidak mungkin akan ada solusi bagi permainan.

3. Rekursif
   Program akan mencari solusi secara rekursi melalui metode tryAllPossibility. Basis pada metode ini ada 2 kemungkinan, yakni. Jika semua kotak pada papan sudah terisi, dan jika semua blok sudah dicoba. Jika semua kotak pada papan sudah terisi, maka akan menghasilkan solusi yang valid, dan kebalikannya jika semua blok sudah dicoba dan papan belum terisi, maka tidak mungkin ada solusi.

   Lalu, program dilanjurkan dengan mengecek posisi yang valid pada tiap kolom dan alternatif orientasi blok. Untuk mengecek apakah posisi pada blok valid, dapat dilakukan dengan mengecek index yang valid untuk blok. Dalam hal ini, mengurangi baris dan kolom pada papan dengan baris dan kolom pada blok akan menghasilkan posisi yang valid bagi blok. Jika terdapat blok dengan ukuran 2 x 2 pada papan berukuran 4 x4, maka kita hanyak perlu untuk memosisikan indeks blok tersebut hingga baris dan kolom 2 karena pada posisi itulah blok akan valid ditempatkan.

   Selanjutnya, blok akan dicek apakah pada posisi tersebut blok tidak bertabrakan dengan blok lain melalui metode isValidPlace. Jika valid blok akan diletakkan dan dilanjutkan dengan rekursif untuk mencari posisi lain yang valid.

4. Transformasi blok
   Jika pada penempatan blok gagal, kita dapat mencari alternatif penempatan blok dengan mengubah orientasi blok. Orientasi dilakukan sebesar 90^o pada sekali iterasi (berarti akan ada 4 kali rotasi, 0^o , 90^o , 180^o , 270^o). Selain itu, dilakukan juga pencerminan pada blok. Kedua metode tersebut jika ditotal akan memberikan 8 alternatif kemungkinan posisi valid blok pada satu tempat.

Untuk megeksekusi kode diatas, terdapat kode utilitas untuk membantu menjalankan kode. Berikut merupakan kode utilitas yang digunakan.

```java
  public boolean isValidPlace(Block block, int startRow, int startCol) {
    for(int i = 0; i < block.getRows(); i++){
      for(int j = 0; j < block.getCols(); j++){
        if (block.getData(i,j) == '.') continue;

        if (startRow + i >= board.getRowsCount() || startCol + j >= board.getColsCount()) return false;

        if (board.getData(startRow + i, startCol + j) != '.') return false;
      }
    }
    return true;
  }

  public void placeBlock(Block block, int startRow, int startCol) {
    char value = block.getId();
    for(int i = 0; i < block.getRows(); i++){
      for(int j = 0; j < block.getCols(); j++){
        if (block.getData(i,j) != '.') {
          board.setData(i + startRow, j + startCol, value);
        }
      }
    }
  }

  public void removeBlock(Block block, int startRow, int startCol) {
    for(int i = 0; i < block.getRows(); i++){
      for(int j = 0; j < block.getCols(); j++){
        if (block.getData(i,j) != '.') {
          board.setData(i + startRow, j + startCol, '.');
        }
      }
    }

  }

  public boolean checkBoxAndArea(){
    int boardArea = board.getColsCount() * board.getRowsCount();
    int blockTiles = 0;
    for(int i = 0; i < listBlocks.size(); i++){
      Block currentBlock = listBlocks.get(i);
      for(int k = 0; k < currentBlock.getCols(); k++){
        for(int j = 0; j < currentBlock.getRows(); j++){
          if(currentBlock.getData(j,k) != '.'){
            blockTiles++;
          }
        }
      }
    }
    System.out.println("Board Area: " + boardArea);
    System.out.println("Block Tiles: " + blockTiles);
    return blockTiles == boardArea;
  }
}
```


Gambar 2.3 Utilitas pada Kelas Brute Force

BAB III

SEKILAS KODE PROGRAM

I. Kelas Blok

Block.java

```java
package pkg;

import java.io.*;
import java.util.*;
import static util.Const.*;

public class Block {
  private char id;
  private char [][] data;
  private int rows;
  private int cols;

  //  Constructor
  public Block (int rows, int cols, char id) {
    this.rows = rows;
    this.cols = cols;
    this.id = id;
    this.data = new char[rows][cols];

    for (int i = 0; i < rows; i++) {
      for (int j = 0; j < cols; j++) {
        data[i][j] = '.';
      }
    }
  }

  public Block(Block other) {
    this.id = other.id;
    this.rows = other.rows;
    this.cols = other.cols;
    this.data = new char[rows][cols];

    for (int i = 0; i < rows; i++) {
      System.arraycopy(other.data[i], 0, this.data[i], 0, cols);
    }
  }


  //  Getter
  public char getId() { return id;}
  public char getData(int i , int j) { return data[i][j]; }
  public char[][] getAllData() { return data; }
  public int getRows() { return rows; }
  public int getCols() { return cols; }

  //  Setter
  public void setId(char id) { this.id = id;}
  public void setAllData(char[][] data) {
    if(data.length == 0 || data[0].length == 0){
      throw new IllegalArgumentException("Block() : Invalid cols and rows!");
    }
    this.rows = data.length;
    this.data = data;
    int maxCols = 0;
    for (char[] row : data) {
      maxCols = Math.max(maxCols, row.length);
    }
    this.cols = maxCols;

  }
```

Gambar 3.1 Constructor dan Getter pada Blok

```java
//  Setter
public void setId(char id) { this.id = id;}
public void setAllData(char[][] data) {
  if(data.length == 0 || data[0].length == 0){
    throw new IllegalArgumentException("Block() : Invalid cols and rows!");
  }
  this.rows = data.length;
  this.data = data;
  int maxCols = 0;
  for (char[] row : data) {
    maxCols = Math.max(maxCols, row.length);
  }
  this.cols = maxCols;

}

public void setData(int rows, int cols, char value) { this.data[rows][cols] = value; }

//  Function
public void rotateBlock(){
  char[][] newData = new char[cols][rows];
  for(int i = 0; i < rows; i++){
    for(int j = 0; j < cols; j++){
      newData[j][rows - 1 - i] = data[i][j];
    }
  }
  this.data = newData;
  this.rows = newData.length;
  this.cols = newData[0].length;
}

public Block getRotateBlock(){
  char[][] newData = new char[cols][rows];
  for(int i = 0; i < rows; i++){
    for(int j = 0; j < cols; j++){
      newData[j][rows - 1 - i] = data[i][j];
    }
  }
  Block newBlock = new Block(rows, cols, id);
  newBlock.setAllData(newData);
  return newBlock;
}
```

Gambar 3.2 Setter pada Blok

```java
//  Function
public void rotateBlock(){
  char[][] newData = new char[cols][rows];
  for(int i = 0; i < rows; i++){
    for(int j = 0; j < cols; j++){
      newData[j][rows - 1 - i] = data[i][j];
    }
  }
  this.data = newData;
  this.rows = newData.length;
  this.cols = newData[0].length;
}

public Block getRotateBlock(){
  char[][] newData = new char[cols][rows];
  for(int i = 0; i < rows; i++){
    for(int j = 0; j < cols; j++){
      newData[j][rows - 1 - i] = data[i][j];
    }
  }
  Block newBlock = new Block(rows, cols, id);
  newBlock.setAllData(newData);
  return newBlock;
}

// Prototype
public void rotateBlock45() {
  int newSize = rows + cols - 1;

  double oldCenterX = (cols - 1) / 2.0;
  double oldCenterY = (rows - 1) / 2.0;

  double offsetX = (newSize - 1) / 2.0;
  double offsetY = (newSize - 1) / 2.0;

  char[][] newData =  new char[newSize][newSize];

  // Init matrix
  for(int i = 0; i < newSize; i++){
    for(int j = 0; j < newSize; j++){
      newData[i][j] = '.';
    }
  }

  for(int i = 0 ; i < rows; i++ ){
    for(int j = 0 ; j < cols; j++){
      // Kali titik dengan matrix rotasi
      if(data[i][j] == '.') continue;

      // Ubah index menjadi koordinat
      double X = j - oldCenterX;
      double Y = oldCenterY - i;

      // Ubah koordinat menjadi index
      int tempX = (int) Math.round(MatrixRotation45[0][0] * X + MatrixRotation45[0][1] * Y + offsetX);
      int tempY = (int) Math.round(MatrixRotation45[1][0] * X + MatrixRotation45[1][1] * Y + offsetY);
      newData[tempX][tempY] = data[i][j];
    }
  }
  this.cols = newSize;
  this.rows = newSize;
  this.data = newData;
}
```

Gambar 3.3 Fungsi Rotasi pada Blok

```java
  public void mirrorBlock(){
    for(int i = 0; i < rows; i++){
      for(int j = 0; j < cols/2; j++){
        char temp = data[i][j];
        data[i][j] = data[i][cols -j -1];
        data[i][cols-j-1] = temp;
      }
    }
  }

  public Block getMirrorBlock(){
    for(int i = 0; i < rows; i++){
      for(int j = 0; j < cols/2; j++){
        char temp = data[i][j];
        data[i][j] = data[i][cols -j -1];
        data[i][cols-j-1] = temp;
      }
    }
    Block newBlock = new Block(rows, cols, id);
    newBlock.setAllData(data);
    return newBlock;
  }

  //  Utilities
  public void printBlock(){
    System.out.println("Block ID: " + id);
    for (int i = 0; i < rows; i++) {
      for (int j = 0; j < cols; j++) {
        System.out.print(data[i][j] + " ");
      }
      System.out.println();
    }
  }

  public String getBlockString() {
    StringBuilder output = new StringBuilder();

    output.append("Block ID: ").append(id).append("\n");
    for (int i = 0; i < rows; i++) {
      for (int j = 0; j < cols; j++) {
        output.append(data[i][j]).append(" ");
      }
      output.append("\n");
    }

    return output.toString();
  }
}
```


Gambar 3.4 Fungsi Mirror dan Utilitas pada Blok

II. Kelas Papan

Board.java

```java
package pkg;

public class Board {
  private char [][] data;
  private int rows;
  private int cols;
  private String caseType;

  //  Constructor
  public Board(int rows, int cols, String caseType) {
    if (rows <= 0 || cols <= 0) {
      throw new IllegalArgumentException("Board(): Invalid board rows and cols!");
    }

    if (caseType.equals("CUSTOM")) {
      System.out.println("On progress");
//      data = new char[rows][cols];

    } else if (caseType.equals("DEFAULT")) {
      this.rows = rows;
      this.cols = cols;
      this.data = new char [rows][cols];
      for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
          this.data[i][j] = '.';
        }
      }
    }
  }

  //  Getter
  public int getRowsCount() { return rows; }
  public int getColsCount() { return cols; }
  public String getCaseType() { return caseType; }
  public char [] getRow(final int row) {return data[row];}
  public char [] getCol(final int col) {
    char [] colData = new char [this.rows];
    for(int i = 0; i < this.rows; i++) {
      colData[i] = data[i][col];
    }
    return colData;
  }
  public char [][] getAllData() { return data; }
  public char getData(int row, int col) { return data[row][col]; }
```


Gambar 3.5 Constructor dan Getter pada kelas Papan

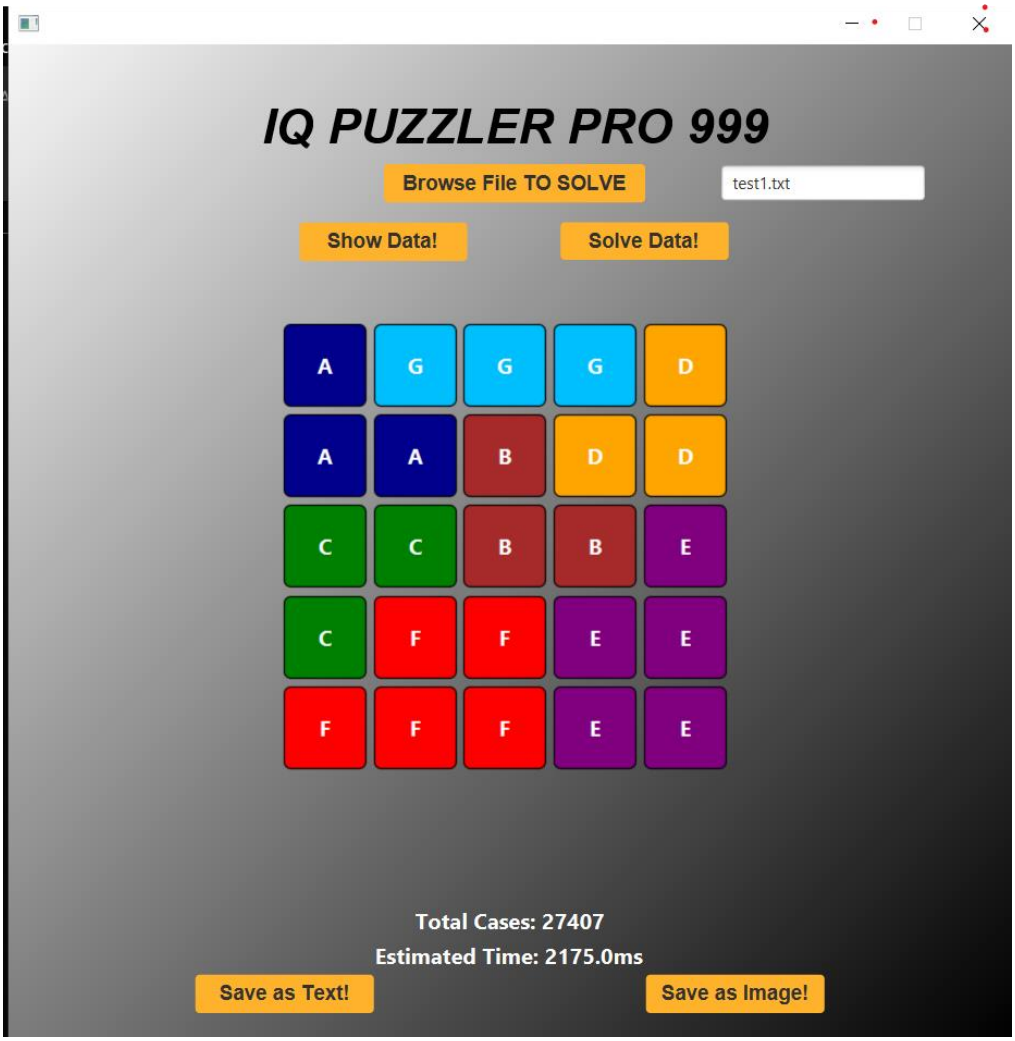
Gambar 3.6 Setter dan Utilitas pada kelas Papan

BAB IV

PERCOBAAN PROGRAM

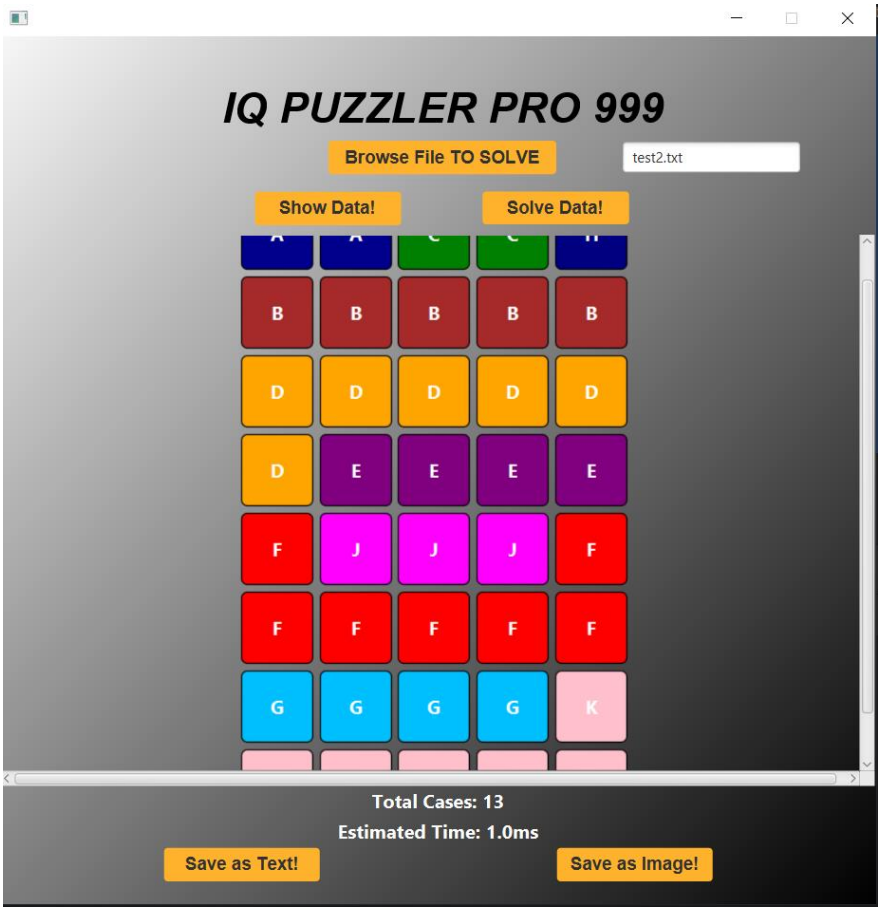
I. Eksperimen test1.txt

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



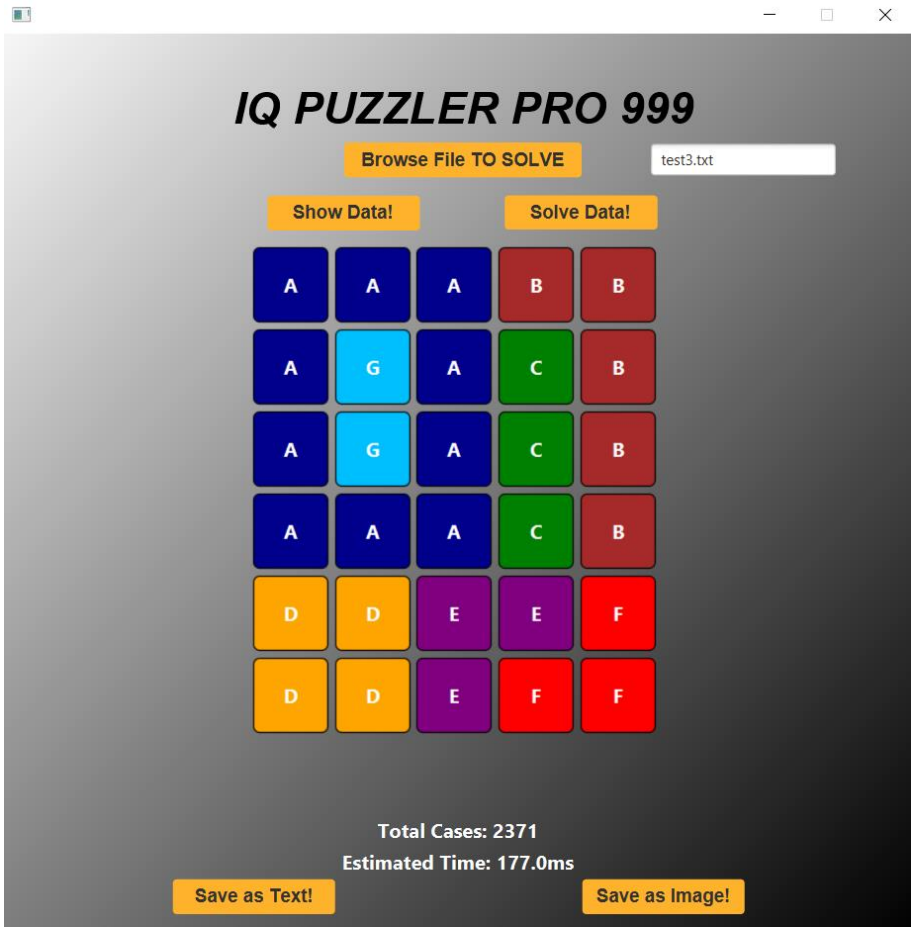
II. Eksperimen test2.txt

```
8 5 10
DEFAULT
AA
BBBBB
CC
DDDDD
D
EEEE
 FF
 F
 F
 F
FF
GGGG
H
JJJ
 K
KKKKK
```



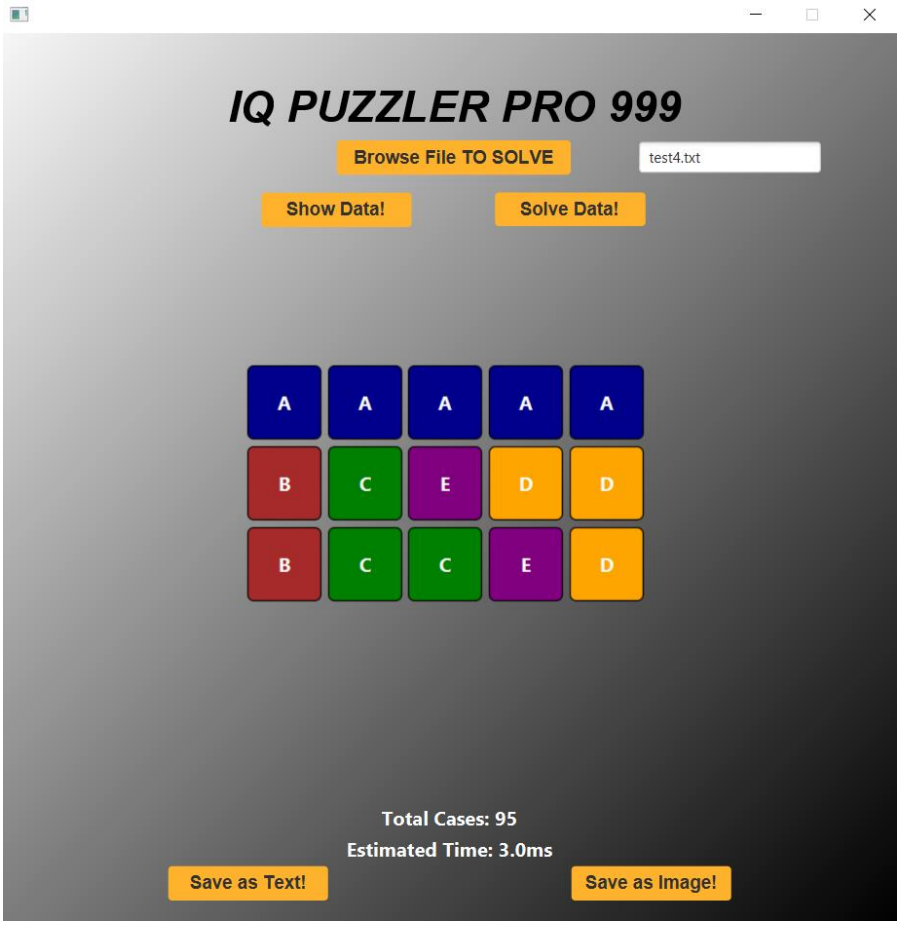
III. Eksperimen test3.txt

```
6 5 7
DEFAULT
AAA
A A
A A
AAA
BBBB
B
CCC
DD
DD
EE
E
FF
F
GG
```



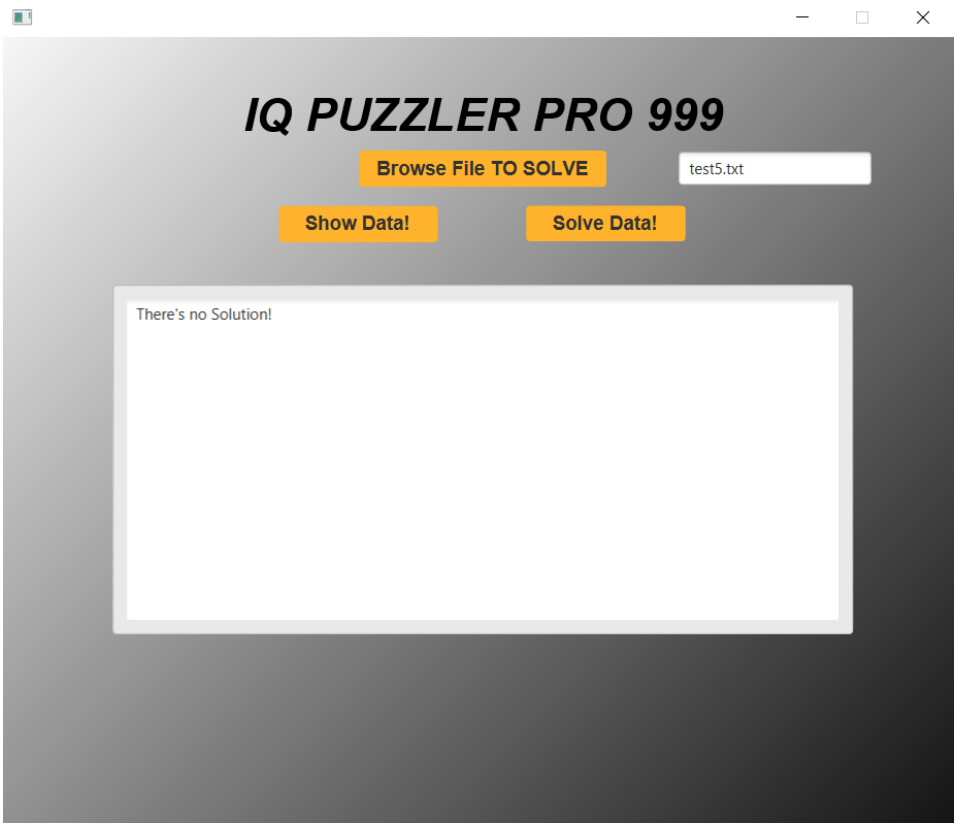
IV. Eksperimen test4.txt

```
3 5 5
DEFAULT
AAAAA
BB
CC
 C
DD
D
E
 E
```



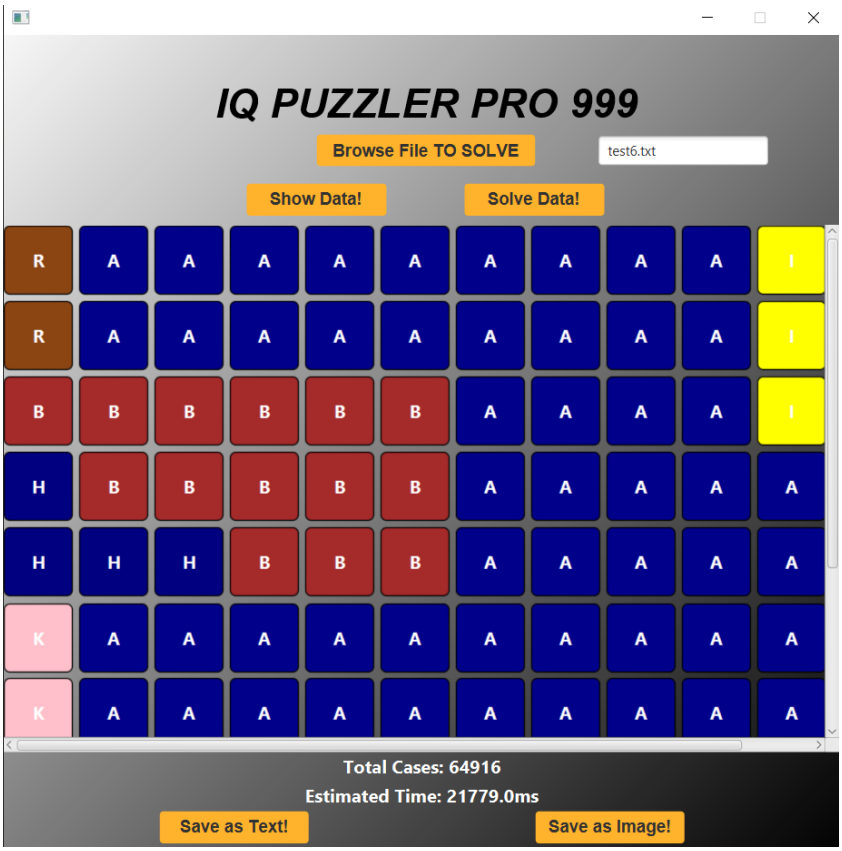
V. Eksperimen test5.txt

```
12 1 3
DEFAULT
A
AA
B
BBBB
C
CC
C
```



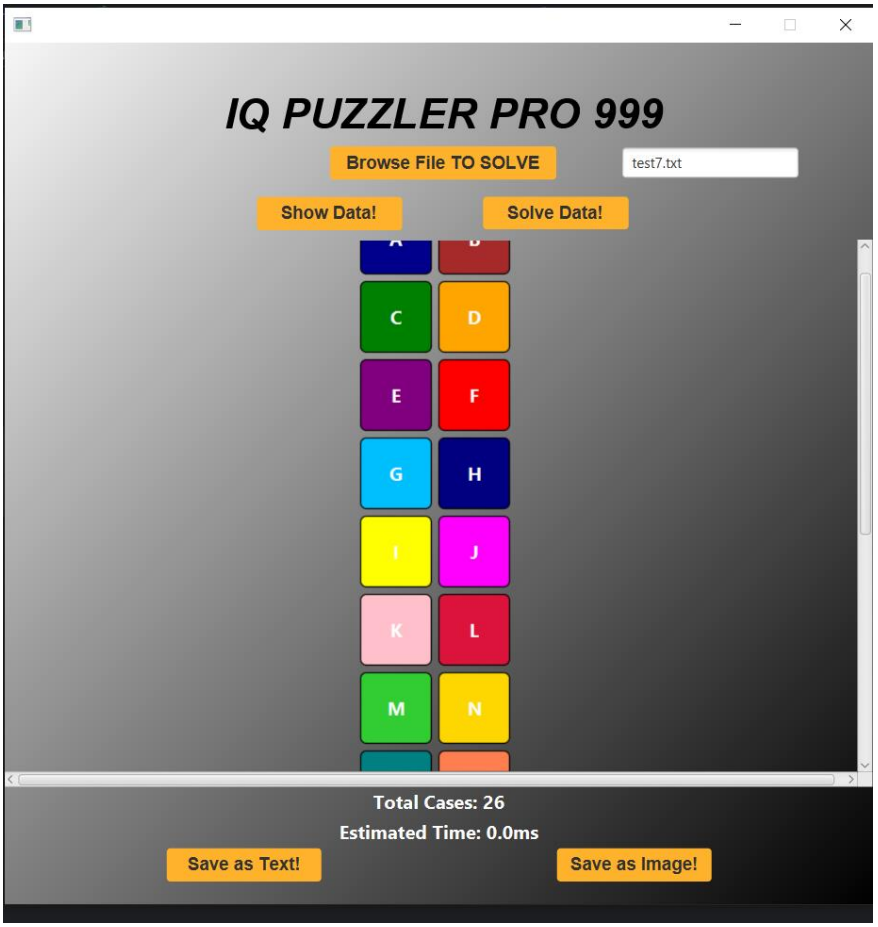
VI. Eksperimen test6.txt

```
10 12 8
DEFAULT
AAAAAAAAA
AAAAAAAAA
    AAAA
    AAAAAA
    AAAAAA
AAAAAAAAAAA
AAAAAAAAAAA
AAAAAAAAA
AAA  AAA
AAA  AAA
HHH
H
  BBB
 BBBBB
BBBBBB
RR
KKKKK
VVV
VVV
III
III
NNN
NNN
```



VII. Eksperimen test7.txt

```
13 2 26
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
```



BAB V

TAUTAN PROGRAM

Program dapat diakses pada :

https://github.com/ryonlunar/tucil1_13523052

BAB VI

LAMPIRAN

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	✓	