

LAPORAN TUGAS KECIL 1 IF2211 STRATEGI ALGORITMA

Penyelesaian IQ Puzzler Pro dengan Algoritma Brute Force



Disusun oleh:

Naomi Risaka Sitorus (13523122)

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG**

2025

DAFTAR ISI

DAFTAR ISI	2
BAB I DESKRIPSI MASALAH DAN ALGORITMA	3
1.1 Algoritma <i>Brute Force</i>	3
1.2 IQ Puzzler Pro	3
1.3 Algoritma Penyelesaian IQ Puzzler Pro dengan Pendekatan Brute Force.....	4
BAB II IMPLEMENTASI ALGORITMA	6
2.1 Piece	6
2.2 <i>Puzzle Solver</i>	6
2.3 Program Utama	7
BAB III SOURCE CODE PROGRAM	8
3.1 Piece.java.....	8
3.2 PuzzleSolver.java	10
3.3 Main.java.....	12
BAB IV MASUKAN DAN KELUARAN PROGRAM	18
4.1 Test Case 1	18
4.2 Test Case 2	19
4.3 Test Case 3	20
4.4 Test Case 4	21
4.5 Test Case 5	22
4.6 Test Case 6	22
4.7 Test Case 7	23
4.8 Test Case 8	25
4.9 Test Case 9	25
4.10 Test Case 10 (Bonus)	26
4.11 Test Case 11 (Bonus)	27
DAFTAR PUSTAKA	29
LAMPIRAN	30

BAB I

DESKRIPSI MASALAH DAN ALGORITMA

1.1 Algoritma *Brute Force*

Algoritma *brute force* merupakan algoritma dengan pendekatan secara langsung atau lempang (*straightforward*). Algoritma ini menyelesaikan persoalan secara sederhana melalui cara yang jelas dan mudah dipahami dalam artian cara yang terpikir pertama kali, sesuai dengan prinsipnya, yaitu “*just do it*” atau “*just solve it*”. Perancangan algoritma *brute force* dapat langsung dibuat berdasarkan pernyataan di dalam persoalan (*problem statement*) serta konsep yang dilibatkan dalam persoalan tersebut.

Algoritma *brute force* memiliki berbagai kelebihan dan kekurangan yang dapat menjadi bahan pertimbangan dalam memilih pendekatan ini untuk menyelesaikan suatu permasalahan komputasi. Salah satu kelebihan algoritma ini adalah aplikasinya yang luas. Metode *brute force* dapat digunakan untuk menyelesaikan segala persoalan komputasi karena sifatnya yang umum, bahkan terdapat persoalan tertentu yang hanya dapat diselesaikan dengan *brute force*, seperti mencari nilai minimum atau maksimum dalam suatu senarai (*list*) dan penjumlahan n buah bilangan. Maka itu, jika tidak yakin akan pendekatan yang tepat untuk menyelesaikan suatu persoalan, pendekatan dengan *brute force* dapat dilakukan. Selain itu, algoritma *brute force* sederhana sehingga mudah dipahami serta mudah diaplikasikan.

Akan tetapi, algoritma *brute force* seringkali tidak efisien dan efektif dalam menyelesaikan permasalahan sebab ia harus mencoba segala kemungkinan hingga ditemukan opsi yang tepat. Hal tersebut tidak terlalu berpengaruh ketika ukuran masukan kecil, tetapi akan tampak ketika ukuran masukan besar sebab *brute force* dapat menghasilkan algoritma dengan kompleksitas melebihi $O(n!)$. Pendekatan ini juga sangat bergantung pada performa komputer yang digunakan serta kurang memanfaatkan kreativitas dalam menyusun strategi pemecahan masalah.

1.2 IQ Puzzler Pro

IQ Puzzler Pro merupakan sebuah permainan papan yang dikembangkan oleh SmartGames dengan tujuan permainan mengisi seluruh papan dengan potongan puzzle (*pieces*) yang tersedia. Permainan ini memiliki dua komponen utama, yaitu papan permainan dan potongan puzzle. Papan permainan memiliki lubang-lubang yang dapat diisi dengan potongan puzzle yang sesuai sebab setiap potongan puzzle memiliki bentuk yang unik.

Permainan IQ Puzzler Pro dimulai dengan kondisi awal papan kosong. Pemain kemudian dapat meletakkan potongan puzzle sedemikian rupa hingga seluruh papan terisi tanpa adanya tumpang tindih, kecuali dalam mode 3D, dan semua potongan terpakai. Setiap potongan puzzle dapat dirotasikan maupun dicerminkan untuk membantu penyusunan. Permainan dinyatakan selesai ketika papan terisi penuh dan semua potongan puzzle berhasil diletakkan pada papan.

1.3 Algoritma Penyelesaian IQ Puzzler Pro dengan Pendekatan Brute Force

Penyelesaian permasalahan permainan IQ Puzzler Pro dapat dilakukan dengan pendekatan *brute force* sebagai berikut.

1. Program meminta file masukan permainan IQ Puzzler Pro.

Pengguna diminta untuk memasukkan nama file masukan permainan IQ Puzzler Pro yang berbentuk file teks (.txt) di terminal. Apabila file masukan berhasil dibaca, program akan melakukan validasi terhadap format serta konten dalam file tersebut. Baris pertama harus berisi nilai N, M, dan P dengan $N \times M$ mewakili ukuran papan permainan serta P mewakili jumlah potongan puzzle yang ada. Baris kedua harus berisi tipe konfigurasi permainan, yaitu *default*.

Baris selanjutnya dalam file masukan merupakan potongan puzzle permainan yang harus berjumlah P buah. Setiap baris dengan huruf yang sama dianggap sebagai satu potongan puzzle. Setiap huruf hanya dapat merepresentasikan satu potongan puzzle dan potongan puzzle dijamin terhubung. Bentuk potongan puzzle dalam program ini tidak harus unik seperti yang tertera pada aturan permainan IQ Puzzler Pro yang sesungguhnya. Seluruh potongan puzzle disimpan dalam *list of piece* dengan *piece* yang merepresentasikan setiap potongan.

2. Program menjalankan algoritma *brute force* untuk mencoba segala kemungkinan solusi permainan.

Setelah pembacaan file masukan berhasil, program menyusun permutasi seluruh potongan puzzle dengan bantuan fungsi rekursif sehingga dihasilkan semua kemungkinan urutan pemasangan potongan puzzle dalam bentuk *list of piece*. Kemungkinan urutan pemasangan potongan puzzle dicoba satu per satu dengan setiap potongan dicoba semua orientasinya yang memungkinkan secara satu per satu di setiap koordinat papan yang memungkinkan, dalam artian tidak *out of bounds*. Orientasi yang dimaksud adalah potongan puzzle pada kondisi awal dan kondisi telah dirotasi (90, 180, dan 270° dari posisi awal), maupun dicerminkan secara horizontal.

Apabila orientasi yang dicoba gagal, papan permainan akan dikembalikan ke kondisi awal, yaitu kosong. Setelah itu, orientasi potongan puzzle yang selanjutnya dicoba untuk diletakkan di papan. Di sisi lain, apabila peletakkan potongan puzzle pada orientasi tertentu berhasil, potongan puzzle selanjutnya akan dicoba diletakkan di papan. Apabila seluruh orientasi dari potongan puzzle berdasarkan urutan yang sedang diproses gagal, akan dicoba kemungkinan urutan pemasangan yang selanjutnya. Penambahan iterasi yang dihitung sebagai jumlah percobaan ditandai dengan dilakukannya pengosongan papan permainan.

Algoritma *brute force* tersebut dijalankan sampai ditemukan suatu solusi permainan IQ Puzzler Pro ditemukan, yaitu papan terisi penuh serta seluruh potongan puzzle terpakai. Jika memang tidak ada solusi permainan untuk file masukan tersebut, algoritma *brute force* akan berhenti ketika seluruh kemungkinan telah dicoba.

3. Program menampilkan solusi permainan IQ Puzzler Pro di terminal (jika ada).

Jika ditemukan solusi permainan, program akan menampilkan solusi permainan dalam bentuk susunan papan permainan yang telah terisi dengan potongan-potongan puzzle yang disajikan dalam warna yang berbeda setiap potongannya di terminal. Program akan menampilkan pesan “Tidak ada solusi yang ditemukan” apabila file masukan permainan tidak memiliki solusi. Selain itu, program akan menampilkan

jumlah percobaan yang telah dilakukan serta waktu eksekusi algoritma *brute force*, baik ditemukan atau tidaknya solusi permainan IQ Puzzler Pro.

4. Program memberikan *prompt* untuk menyimpan solusi ke dalam bentuk file teks (.txt) maupun file gambar (.png) jika ditemukan solusi.

Apabila ditemukan solusi permainan dari file masukan, program akan menanyakan apakah pengguna ingin menyimpan solusi dalam bentuk file teks dengan nama file sesuai keinginan pengguna. Selain itu, dari bonus yang dikerjakan, program akan menanyakan apakah pengguna ingin menyimpan solusi dalam bentuk file gambar dengan nama file sesuai keinginan pengguna. Setiap potongan puzzle akan ditampilkan dengan warna yang berbeda dalam file gambar dengan warna serupa dengan solusi yang ditampilkan di CLI (*command line interface*).

BAB II

IMPLEMENTASI ALGORITMA

2.1 Piece

Berisi kelas Piece serta konstruktor, atribut, dan *method* lainnya yang digunakan untuk memanipulasi serta menampilkan potongan puzzle. Terdapat dalam file Piece.java yang memiliki *class* Piece.

Konstruktor	Deskripsi
public Piece(List<String> shapeLines, char letter)	Membuat Piece yang menyimpan huruf yang digunakan serta bentuk potongan puzzle dalam bentuk <i>array</i> dua dimensi

Method	Deskripsi
public List<Piece> getAllOrientations()	Mengembalikan semua orientasi yang mungkin dari sebuah potongan puzzle
private Piece rotatePiece()	Mengembalikan sebuah potongan puzzle yang telah dirotasi sebesar 90° searah jarum jam dari bentuk awalnya
private Piece flipPiece()	Mengembalikan sebuah potongan puzzle yang telah dicerminkan secara horizontal dari bentuk awalnya
private List<String> charArrayToList(char[][] arr)	Mengembalikan <i>array of char</i> yang telah dikonversi menjadi <i>list of strings</i> , digunakan sebagai <i>method</i> bantuan dari <i>method</i> rotatePiece() dan flipPiece()
public void printPiece()	Menampilkan sebuah potongan puzzle ke terminal (CLI)

2.2 Puzzle Solver

Berisi *method* untuk menyelesaikan puzzle secara *brute force*. Terdapat dalam file PuzzleSolver.java yang memiliki *class* PuzzleSolver.

Method	Deskripsi
public static long tryAllConfigurations(char[][] board, List<Piece> pieces, int N, int M, long triesAmt)	Mengembalikan jumlah percobaan yang dilakukan untuk menemukan solusi dari puzzle dengan nilai negatif mengindikasikan bahwa tidak terdapat solusi dari puzzle
public static List<List<Piece>> generatePermutations(List<Piece> pieces)	Mengembalikan segala kemungkinan permutasi, yaitu urutan penempatan potongan puzzle, dari seluruh potongan puzzle
private static void generatePermutationsHelper(List<Piece> pieces, int index, List<List<Piece>> permutations)	Membantu penyusunan permutasi dengan rekursi, digunakan sebagai <i>method</i> bantuan dari <i>method</i> generatePermutations(List<Piece> pieces)

private static void swapPieces(List<Piece> pieces, int i, int j)	Melakukan penukaran posisi dari dua potongan puzzle dalam sebuah <i>list of piece</i>
public static void clearBoard(char[][] board)	Mengembalikan papan permainan ke kondisi awal, yaitu kosong
public static boolean canPlaceAllPieces(char[][] board, List<Piece> pieces, int N, int M)	Mengembalikan bisa atau tidaknya seluruh potongan puzzle diletakkan pada papan permainan
private static boolean canPlacePiece(char[][] board, Piece piece, int x, int y, int N, int M)	Mengembalikan bisa atau tidaknya sebuah potongan puzzle diletakkan pada papan permainan
private static void placePiece(char[][] board, Piece piece, int x, int y)	Meletakkan sebuah potongan puzzle pada papan permainan di posisi (x, y)
private static boolean isBoardFull(char[][] board)	Mengembalikan penuh atau tidaknya papan permainan

2.3 Program Utama

Berisi *method* untuk menjalankan program IQ Puzzle Pro Solver secara keseluruhan dan memproses file masukan serta keluaran, baik dalam bentuk teks maupun gambar (bonus). Terdapat dalam file Main.java yang memiliki *class* Main.

<i>Method</i>	Deskripsi
public static void main(String[] args)	Menjalankan program secara keseluruhan dengan memanggil <i>method</i> untuk membaca file masukan, mencari solusi dari puzzle, dan memberikan <i>prompt</i> menyimpan solusi ke dalam bentuk file teks serta gambar
private static List<Piece> readInputFile(Scanner fileScanner)	Membaca serta memvalidasi file masukan dan menyimpan potongan puzzle ke dalam <i>list of piece</i> yang kemudian dikembalikan
private static void assignColors(List<Piece> pieces)	Mengatur warna dari setiap potongan puzzle berdasarkan huruf yang digunakan
private static void printBoard(char[][] board)	Menampilkan papan permainan yang telah terisi ke terminal (CLI)
private static void saveSolution(char[][] board, Scanner scanner)	Menampilkan <i>prompt</i> untuk menyimpan solusi ke dalam bentuk file teks (.txt)
private static void saveSolutionAsImage(char[][] board, Scanner scanner)	Menampilkan <i>prompt</i> untuk menyimpan solusi ke dalam bentuk file gambar (.png), merupakan spesifikasi bonus tugas

BAB III

SOURCE CODE PROGRAM

3.1 Piece.java

```
1  import java.util.ArrayList;
2  import java.util.List;
3
4  class Piece {
5      char letter;
6      char[][] shape;
7
8      // constructor
9      public Piece(List<String> shapelines, char letter) {
10         this.letter = letter;
11         int height = shapelines.size();
12         int width = 0;
13
14         // piece width based on the longest line
15         for (String line : shapelines) {
16             width = Math.max(width, line.length());
17         }
18
19         // matrix initialization for piece shape
20         shape = new char[height][width];
21         for (int i = 0; i < height; i++) {
22             String line = shapelines.get(i);
23             for (int j = 0; j < width; j++) {
24                 if (j < line.length()) {
25                     shape[i][j] = line.charAt(j);
26                 } else {
27                     shape[i][j] = ' ';
28                 }
29             }
30         }
31     }
32
33     // gets all possible orientations of a piece
34     public List<Piece> getAllOrientations() {
35         List<Piece> orientations = new ArrayList<>();
36         orientations.add(this); // og orientation
37
38         // rotations
39         Piece rotated1 = rotatePiece();
40         Piece rotated2 = rotated1.rotatePiece();
41         Piece rotated3 = rotated2.rotatePiece();
42
43         // flips
44         Piece flipped = flipPiece();
45         Piece flipped1 = flipped.rotatePiece();
46         Piece flipped2 = flipped1.rotatePiece();
47         Piece flipped3 = flipped2.rotatePiece();
48
49         orientations.add(rotated1);
50         orientations.add(rotated2);
51         orientations.add(rotated3);
52         orientations.add(flipped);
53         orientations.add(flipped1);
54         orientations.add(flipped2);
55         orientations.add(flipped3);
56
57         return orientations;
58     }
}
```



```

60 // rotates a piece 90 degrees clockwise
61 private Piece rotatePiece() {
62     int h = shape.length;
63     int w = shape[0].length;
64     char[][] rotated = new char[w][h];
65
66     // piece adjustment after rotating
67     for (int i = 0; i < w; i++) {
68         for (int j = 0; j < h; j++) {
69             rotated[i][j] = ' ';
70         }
71     }
72
73     for (int i = 0; i < h; i++) {
74         for (int j = 0; j < w; j++) {
75             if (shape[i][j] != ' ') {
76                 rotated[j][h - 1 - i] = shape[i][j];
77             }
78         }
79     }
80
81     return new Piece(charArrayToList(rotated), letter);
82 }

```

```

84 // flips a piece horizontally
85 private Piece flipPiece() {
86     int h = shape.length;
87     int w = shape[0].length;
88     char[][] flipped = new char[h][w];
89
90     // piece adjustment after flipping
91     for (int i = 0; i < h; i++) {
92         for (int j = 0; j < w; j++) {
93             flipped[i][j] = shape[i][w - 1 - j];
94         }
95     }
96
97     return new Piece(charArrayToList(flipped), letter);
98 }
99
100 // converts a char matrix to a list of strings
101 private List<String> charArrayToList(char[][] arr) {
102     List<String> list = new ArrayList<>();
103     for (char[] row : arr) {
104         list.add(new String(row));
105     }
106     return list;
107 }

```

```

109 // prints a piece
110 public void printPiece() {
111     for (char[] row : shape) {
112         System.out.println(new String(row));
113     }
114     System.out.println();
115 }
116 }

```

3.2 PuzzleSolver.java

```
1  import java.util.*;
2
3  public class PuzzleSolver {
4      // tries all possible configuration of the pieces
5      public static long tryAllConfigurations(char[][] board, List<Piece> pieces, int N, int M, long triesAmt) {
6          List<List<Piece>> allPermutations = generatePermutations(pieces);
7
8          for (List<Piece> permutation : allPermutations) {
9              triesAmt++; // incremented per iteration (permutation)
10             clearBoard(board); // reset board for each iteration
11
12             if (canPlaceAllPieces(board, permutation, N, M)) {
13                 return triesAmt; // all pieces are succesfully placed on the board
14             }
15         }
16         return -triesAmt; // no solution found
17     }
18
19     // generates all possible permutations of the pieces
20     private static List<List<Piece>> generatePermutations(List<Piece> pieces) {
21         List<List<Piece>> permutations = new ArrayList<>();
22         generatePermutationsHelper(pieces, index:0, permutations);
23         return permutations;
24     }
25
26     // helper function for generating permutations
27     private static void generatePermutationsHelper(List<Piece> pieces, int index, List<List<Piece>> permutations) {
28         if (index == pieces.size()) {
29             permutations.add(new ArrayList<>(pieces));
30             return;
31         }
32
33         for (int i = index; i < pieces.size(); i++) {
34             swapPieces(pieces, index, i);
35             generatePermutationsHelper(pieces, index + 1, permutations);
36             swapPieces(pieces, index, i);
37         }
38     }
39
40     // swaps two pieces in a list of pieces
41     private static void swapPieces(List<Piece> pieces, int i, int j) {
42         Piece temp = pieces.get(i);
43         pieces.set(i, pieces.get(j));
44         pieces.set(j, temp);
45     }
46
47     // resets the working board to empty
48     private static void clearBoard(char[][] board) {
49         for (int i = 0; i < board.length; i++) {
50             Arrays.fill(board[i], val: ' ');
51         }
52     }
53 }
```

```

54 // tries placing all pieces on the board
55 private static boolean canPlaceAllPieces(char[][] board, List<Piece> pieces, int N, int M) {
56     for (Piece piece : pieces) {
57         boolean piecePlaced = false;
58         List<Piece> allOrientations = piece.getAllOrientations();
59
60         // tries all orientation of a piece
61         for (Piece orientation : allOrientations) {
62             // tries all position for placing a piece on the board
63             for (int i = 0; i < N; i++) {
64                 for (int j = 0; j < M; j++) {
65                     if (canPlacePiece(board, orientation, i, j, N, M)) {
66                         placePiece(board, orientation, i, j);
67                         piecePlaced = true;
68                         break;
69                     }
70                 }
71                 if (piecePlaced) {
72                     break;
73                 }
74             }
75             if (piecePlaced) {
76                 break;
77             }
78         }
79         if (!piecePlaced) {
80             return false; // piece cannot be placed on the board
81         }
82     }
83     return true; // all pieces are succesfully placed on the board
84 }

```

```

86 // checks if a piece can be placed on the board
87 private static boolean canPlacePiece(char[][] board, Piece piece, int x, int y, int N, int M) {
88     char[][] shape = piece.shape;
89     int h = shape.length;
90     int w = shape[0].length;
91
92     // out of board bounds validation
93     if (x + h > N || y + w > M) {
94         return false;
95     }
96
97     for (int i = 0; i < h; i++) {
98         for (int j = 0; j < w; j++) {
99             if (shape[i][j] != ' ' && board[x + i][y + j] != ' ') {
100                 return false;
101             }
102         }
103     }
104     return true;
105 }

```

```

107 // places a piece in the board
108 private static void placePiece(char[][] board, Piece piece, int x, int y) {
109     char[][] shape = piece.shape;
110     int h = shape.length;
111     int w = shape[0].length;
112
113     for (int i = 0; i < h; i++) {
114         for (int j = 0; j < w; j++) {
115             if (shape[i][j] != ' ') {
116                 board[x + i][y + j] = piece.letter;
117             }
118         }
119     }
120 }
121
122 // checks whether the board is fully filled with pieces
123 public static boolean isBoardFull(char[][] board) {
124     for (char[] row : board) {
125         for (char cell : row) {
126             if (cell == ' ') {
127                 return false;
128             }
129         }
130     }
131     return true;
132 }
133 }

```

3.3 Main.java

```
14 // ANSI color codes for CLI output
15 private static Map<Character, String> pieceColors = new HashMap<>();
16 private static final String[] ANSI_COLORS = {
17     "\u001B[31m", // red
18     "\u001B[32m", // green
19     "\u001B[33m", // yellow
20     "\u001B[34m", // blue
21     "\u001B[35m", // magenta
22     "\u001B[36m", // cyan
23     "\u001B[91m", // bright red
24     "\u001B[92m", // bright green
25     "\u001B[93m", // bright yellow
26     "\u001B[94m", // bright blue
27     "\u001B[95m", // bright magenta
28     "\u001B[96m", // bright cyan
29     "\u001B[103m", // bright background yellow
30     "\u001B[90m", // dark gray
31     "\u001B[97m", // white
32     "\u001B[30m", // black
33     "\u001B[41m", // background red
34     "\u001B[42m", // background green
35     "\u001B[43m", // background yellow
36     "\u001B[44m", // background blue
37     "\u001B[45m", // background magenta
38     "\u001B[46m", // background cyan
39     "\u001B[104m", // bright background blue
40     "\u001B[100m", // bright background black
41     "\u001B[101m", // bright background red
42     "\u001B[102m" // bright background green
43 };
44 private static final String ANSI_RESET = "\u001B[0m";

46 // RGB color codes for image output
47 private static final Map<Character, Color> COLOR_MAP = new HashMap<>();
48 static {
49     // closest colors to the ANSI ones
50     COLOR_MAP.put(key:'A', new Color(r:255, g:0, b:0)); // red
51     COLOR_MAP.put(key:'B', new Color(r:0, g:255, b:0)); // green
52     COLOR_MAP.put(key:'C', new Color(r:255, g:255, b:0)); // yellow
53     COLOR_MAP.put(key:'D', new Color(r:0, g:0, b:255)); // blue
54     COLOR_MAP.put(key:'E', new Color(r:255, g:0, b:255)); // magenta
55     COLOR_MAP.put(key:'F', new Color(r:0, g:255, b:255)); // cyan
56     COLOR_MAP.put(key:'G', new Color(r:250, g:85, b:85)); // bright red
57     COLOR_MAP.put(key:'H', new Color(r:85, g:255, b:85)); // bright green
58     COLOR_MAP.put(key:'I', new Color(r:255, g:255, b:85)); // bright yellow
59     COLOR_MAP.put(key:'J', new Color(r:85, g:85, b:255)); // bright blue
60     COLOR_MAP.put(key:'K', new Color(r:255, g:85, b:255)); // bright magenta
61     COLOR_MAP.put(key:'L', new Color(r:85, g:255, b:255)); // bright cyan
62     COLOR_MAP.put(key:'M', new Color(r:255, g:255, b:85)); // bright background yellow
63     COLOR_MAP.put(key:'N', new Color(r:85, g:85, b:85)); // dark gray
64     COLOR_MAP.put(key:'O', new Color(r:200, g:200, b:200)); // white
65     COLOR_MAP.put(key:'P', new Color(r:0, g:0, b:0)); // black
66     COLOR_MAP.put(key:'Q', new Color(r:255, g:0, b:0)); // background red
67     COLOR_MAP.put(key:'R', new Color(r:0, g:255, b:0)); // background green
68     COLOR_MAP.put(key:'S', new Color(r:255, g:255, b:0)); // background yellow
69     COLOR_MAP.put(key:'T', new Color(r:0, g:0, b:255)); // background blue
70     COLOR_MAP.put(key:'U', new Color(r:255, g:0, b:255)); // background magenta
71     COLOR_MAP.put(key:'V', new Color(r:0, g:255, b:255)); // background cyan
72     COLOR_MAP.put(key:'W', new Color(r:0, g:0, b:128)); // bright background blue
73     COLOR_MAP.put(key:'Q', new Color(r:85, g:85, b:85)); // bright background black
74     COLOR_MAP.put(key:'Y', new Color(r:255, g:85, b:85)); // bright background red
75     COLOR_MAP.put(key:'Z', new Color(r:85, g:255, b:85)); // bright background green
76 }
```

```

77 public static void main(String[] args) {
78     System.out.println(x:"=====");
79     System.out.println("\r\n" +
80         "
81         "
82         "
83         "
84         "
85         "
86         """);
87     System.out.println(x:"=====");
88     System.out.println();
89
90     Scanner scanner = new Scanner(System.in);
91
92     System.out.println(x:"Selamat datang di program IQ Puzzle Pro Solver!");
93     System.out.println(x:"Program ini akan membantu Anda menyelesaikan IQ Pro Puzzle.");
94     System.out.println();
95     System.out.print(s:"Masukkan nama file puzzle yang ingin diselesaikan: ");
96     String inputFileName = scanner.nextLine();
97
98     // empty file name validation
99     while (inputFileName.isEmpty()) {
100         System.out.println(x:"Nama file tidak boleh kosong");
101         System.out.println();
102         System.out.print(s:"Masukkan nama file puzzle yang ingin diselesaikan: ");
103         inputFileName = scanner.nextLine();
104     }
105
106     // file extension validation
107     while (!inputFileName.endsWith(suffix:".txt")) {
108         System.out.println(x:"File harus berekstensi .txt");
109         System.out.println();
110         System.out.print(s:"Masukkan nama file puzzle yang ingin diselesaikan: ");
111         inputFileName = scanner.nextLine();
112     }
113
114     try {
115         File file = new File(inputFileName);
116         Scanner fileScanner = new Scanner(file);
117         try {
118             List<Piece> pieces = readInputFile(fileScanner);
119             assignColors(pieces);
120
121             char[][] board = new char[N][M];
122             for (int i = 0; i < N; i++) {
123                 Arrays.fill(board[i], val: ' ');
124             }
125
126             long triesAmt = 0;
127             long startTime = System.nanoTime();
128             triesAmt = PuzzleSolver.tryAllConfigurations(board, pieces, N, M, triesAmt);
129             long endTime = System.nanoTime();
130

```

```

132 ~         if (triesAmt > 0 && PuzzleSolver.isBoardFull(board)) {
133             printBoard(board);
134             long duration = (endTime - startTime) / 1000000; // duration in ms
135             System.out.println("Jumlah kasus yang ditinjau: " + triesAmt);
136             System.out.println("Waktu eksekusi: " + duration + " ms");
137             saveSolution(board, scanner);
138             saveSolutionAsImage(board, scanner);
139 ~         } else {
140             long duration = (endTime - startTime) / 1000000; // duration in ms
141             System.out.println();
142             System.out.println(x:"Tidak ada solusi yang ditemukan.");
143             System.out.println();
144             System.out.println("Jumlah kasus yang ditinjau: " + -triesAmt);
145             System.out.println("Waktu eksekusi: " + duration + " ms");
146         }
147 ~     } catch (Exception e) {
148         System.out.println("Error: " + e.getMessage());
149         return;
150     }
151     fileScanner.close();
152
153 ~ } catch (FileNotFoundException e) {
154     System.out.println(x:"File tidak ditemukan.");
155 }
156
157 scanner.close();
158 }

```

```

160 // reads input file and validate its contents
161 private static List<Piece> readInputFile(Scanner fileScanner) throws Exception {
162     // N (number of rows) validation
163     if (!fileScanner.hasNextInt()) {
164         throw new Exception(message:"File tidak memiliki nilai N.");
165     }
166     N = fileScanner.nextInt();
167     if (N <= 0) {
168         throw new Exception(message:"N harus bernilai lebih besar dari 0.");
169     }
170
171     // M (number of cols) validation
172     if (!fileScanner.hasNextInt()) {
173         throw new Exception(message:"File tidak memiliki nilai M.");
174     }
175     M = fileScanner.nextInt();
176     if (M <= 0) {
177         throw new Exception(message:"M harus bernilai lebih besar dari 0.");
178     }
179
180     // P (number of pieces) validation
181     if (!fileScanner.hasNextInt()) {
182         throw new Exception(message:"File tidak memiliki nilai P.");
183     }
184     int P = fileScanner.nextInt(); // number of pieces
185     if (P <= 0) {
186         throw new Exception(message:"P harus bernilai lebih besar dari 0.");
187     }

```

```

189 // configuration type validation
190 fileScanner.nextLine();
191 if (!fileScanner.hasNextLine()) {
192     throw new Exception(message:"File tidak memiliki tipe konfigurasi.");
193 }
194 String configType = fileScanner.nextLine().trim(); // configuration type
195 if (configType.isEmpty()) {
196     throw new Exception(message:"Tipe konfigurasi tidak boleh kosong.");
197 }
198 if (!configType.equalsIgnoreCase(anotherString:"default")) {
199     throw new Exception(message:"Tipe konfigurasi yang tersedia adalah 'default'.");
200 }
201
202 // pieces reading and validation
203 Map<Character, List<String>> pieceShapes = new LinkedHashMap<>();
204 Set<Character> usedLetters = new HashSet<>();
205 Character currPiece = null;
206
207 while (fileScanner.hasNextLine()) {
208     String pieceLine = fileScanner.nextLine(); // Keep leading spaces
209     if (pieceLine.trim().isEmpty()) {
210         currPiece = null;
211         continue;
212     }

```

```

218 String upperPieceline = pieceline.trim().toUpperCase(); // set all letters to uppercase
219 char firstLetter = upperPieceline.charAt(index:0);
220 for (char c : upperPieceline.toCharArray()) {
221     // alphabet validation
222     if (!Character.isLetter(c) && c != ' '){
223         throw new Exception(message:"Terdapat baris yang mengandung karakter bukan alphabet.");
224     }
225     if (c != firstLetter && c != ' ') {
226         throw new Exception(message:"Terdapat baris yang mengandung lebih dari satu huruf.");
227     }
228 }
229
230 // unique letter validation
231 if (currPiece != null && firstLetter != currPiece && usedLetters.contains(firstLetter)) {
232     throw new Exception("Huruf '" + firstLetter + "' digunakan oleh lebih dari satu piece yang terpisah.");
233 }
234
235 usedLetters.add(firstLetter);
236 pieceShapes.putIfAbsent(firstLetter, new ArrayList<>());
237 pieceShapes.get(firstLetter).add(pieceline); // Add the whole line including spaces
238 currPiece = firstLetter;
239
240
241 // P validation according to the number of pieces read
242 if (pieceShapes.size() != P) {
243     throw new Exception(message:"Jumlah piece tidak sesuai dengan nilai P.");
244 }

```

```

241 List<Piece> pieces = new ArrayList<>();
242 for (Map.Entry<Character, List<String>> entry : pieceShapes.entrySet()) {
243     pieces.add(new Piece(entry.getValue(), entry.getKey()));
244 }
245
246 return pieces;
247 }
248
249 // assigns colors to pieces based on their letters
250 private static void assignColors(List<Piece> pieces) {
251     for (Piece piece : pieces) {
252         char letter = piece.letter;
253         if (!pieceColors.containsKey(letter)) {
254             int index = letter - 'A'; // letter to index conversion for color assignment
255             if (index >= 0 && index < ANSI_COLORS.length) {
256                 pieceColors.put(letter, ANSI_COLORS[index]);
257             } else {
258                 pieceColors.put(letter, ANSI_RESET);
259             }
260         }
261     }
262 }

```

```

264 // displays solution board in CLI
265 private static void printBoard(char[][] board) {
266     System.out.println();
267     System.out.println(x:"Solusi ditemukan.");
268     for (char[] row : board) {
269         for (char cell : row) {
270             if (cell == ' ') {
271                 System.out.print(cell);
272             } else {
273                 System.out.print(pieceColors.get(cell) + cell + ANSI_RESET);
274             }
275         }
276         System.out.println();
277     }
278     System.out.println();
279 }
280
281 // saves solution board as a text file
282 private static void saveSolution(char[][] board, Scanner scanner) {
283     System.out.println();
284     System.out.print(s:"Apakah Anda ingin menyimpan solusi dalam bentuk teks? (ya/tidak) ");
285     String saveOpt = scanner.nextLine();
286
287     // save option validation
288     while (!saveOpt.equalsIgnoreCase(anotherString:"ya") || saveOpt.equalsIgnoreCase(anotherString:"tidak") || saveOpt.isEmpty()) {
289         System.out.println();
290         System.out.print(s:"Apakah Anda ingin menyimpan solusi dalam bentuk teks? (ya/tidak) ");
291         saveOpt = scanner.nextLine();
292     }

```

```

294 if (saveOpt.equalsIgnoreCase(anotherString:"ya")){
295     System.out.print(s:"Masukkan nama file output: ");
296     String outputFileName = scanner.nextLine();
297
298     File directory = new File(pathname:"../test");
299     if (!directory.exists()) {
300         directory.mkdirs();
301     }
302
303     // empty file name validation
304     while (outputFileName.isEmpty()) {
305         System.out.println(x:"Nama file tidak boleh kosong");
306         System.out.println();
307         System.out.print(s:"Masukkan nama file output: ");
308         outputFileName = scanner.nextLine();
309     }
310
311     // file extension validation
312     while (!outputFileName.endsWith(suffix:".txt")) {
313         System.out.println(x:"File harus berekstensi .txt");
314         System.out.println();
315         System.out.print(s:"Masukkan nama file output: ");
316         outputFileName = scanner.nextLine();
317     }
318
319     outputFileName = "../test/" + outputFileName;

```

```

321 // writes solution to a text file
322 try (PrintWriter writer = new PrintWriter(new File(outputFileName))) {
323     for (int i = 0; i < board.length; i++) {
324         writer.print(new String(board[i]));
325         if (i < board.length - 1) {
326             writer.println();
327         }
328     }
329     System.out.println("Solusi disimpan ke " + outputFileName);
330 } catch (FileNotFoundException e) {
331     System.out.println("Error: " + e.getMessage());
332 }
333
334 }
335
336 // saves solution board as an image file
337 private static void saveSolutionAsImage(char[][] board, Scanner scanner) {
338     System.out.println();
339     System.out.print(s:"Apakah Anda ingin menyimpan solusi dalam bentuk gambar? (ya/tidak) ");
340     String saveOpt = scanner.nextLine();
341
342     // save option validation
343     while (!(saveOpt.equalsIgnoreCase(anotherString:"ya") || saveOpt.equalsIgnoreCase(anotherString:"tidak") || saveOpt.isEmpty())) {
344         System.out.println();
345         System.out.print(s:"Apakah Anda ingin menyimpan solusi dalam bentuk gambar? (ya/tidak) ");
346         saveOpt = scanner.nextLine();
347     }

```

```

349 if (saveOpt.equalsIgnoreCase(anotherString:"ya")) {
350     System.out.print(s:"Masukkan nama file output gambar: ");
351     String outputFileName = scanner.nextLine();
352
353     // empty file name validation
354     while (outputFileName.isEmpty()) {
355         System.out.println(x:"Nama file tidak boleh kosong");
356         System.out.println();
357         System.out.print(s:"Masukkan nama file output gambar: ");
358         outputFileName = scanner.nextLine();
359     }
360
361     // file extension validation
362     while (!outputFileName.endsWith(suffix:".png")) {
363         System.out.println(x:"File harus berekstensi .png");
364         System.out.println();
365         System.out.print(s:"Masukkan nama file output gambar: ");
366         outputFileName = scanner.nextLine();
367     }
368
369     outputFileName = "../test/" + outputFileName;
370
371     // creates image file for the solution board
372     int cellSize = 50;
373     int width = board[0].length * cellSize;
374     int height = board.length * cellSize;
375
376     BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_ARGB);
377     Graphics2D graphics = image.createGraphics();

```



```

379     graphics.setColor(Color.WHITE);
380     graphics.fillRect(x:0, y:0, width, height);
381     graphics.setFont(new Font(name:"Arial", Font.BOLD, cellSize / 2));
382     FontMetrics fm = graphics.getFontMetrics();
383
384     for (int i = 0; i < board.length; i++) {
385         for (int j = 0; j < board[0].length; j++) {
386             char cell = board[i][j];
387             // cell background color
388             if (cell != ' ') {
389                 graphics.setColor(COLOR_MAP.getOrDefault(cell, Color.BLACK));
390                 graphics.fillRect(j * cellSize, i * cellSize, cellSize, cellSize);
391             }
392
393             // cell border
394             graphics.setColor(Color.BLACK);
395             graphics.drawRect(j * cellSize, i * cellSize, cellSize, cellSize);
396
397             // cell text (piece letter)
398             if (cell != ' ') {
399                 String text = String.valueOf(cell);
400                 int textWidth = fm.stringWidth(text);
401                 int textHeight = fm.getAscent();
402
403                 // text color based on cell background color
404                 if (graphics.getColor().getRed() + graphics.getColor().getGreen() + graphics.getColor().getBlue() < 382) {
405                     graphics.setColor(Color.WHITE);
406                 } else {
407                     graphics.setColor(Color.BLACK);
408                 }
409
410                 // text positioning
411                 int textX = j * cellSize + (cellSize - textWidth) / 2;
412                 int textY = i * cellSize + (cellSize + textHeight) / 2 - 5;
413                 graphics.drawString(text, textX, textY);
414             }
415         }
416     }
417     graphics.dispose();
418
419     try {
420         File directory = new File(pathname:"../test");
421         if (!directory.exists()) {
422             directory.mkdirs();
423         }
424         ImageIO.write(image, formatName:"png", new File(outputFileName));
425         System.out.println("Gambar solusi disimpan ke " + outputFileName);
426     } catch (IOException e) {
427         System.out.println("Error: " + e.getMessage());
428     }
429 }
430 }
431 }

```

BAB IV

MASUKAN DAN KELUARAN PROGRAM

4.1 Test Case 1

File masukan valid dan puzzle memiliki solusi.

Masukan (CLI)		
Masukan (File Teks)		
Keluaran (CLI)		
Keluaran (File Teks)		

File masukan valid dan puzzle memiliki solusi.

Masukan (CLI)

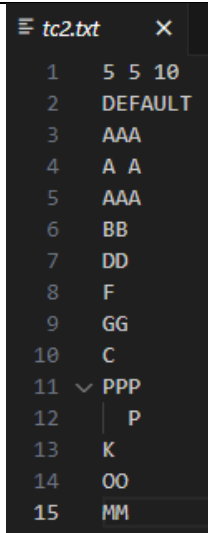
```
=====
IQ Puzzle Pro Solver
=====

Selamat datang di program IQ Puzzle Pro Solver!
Program ini akan membantu Anda menyelesaikan IQ Pro Puzzle.

Masukkan nama file puzzle yang ingin diselesaikan: ../test/tc2
File harus berekstensi .txt

Masukkan nama file puzzle yang ingin diselesaikan: ../test/tc2.txt
```

Masukan (File Teks)



Keluaran (CLI)

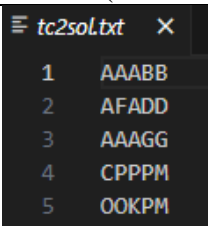
```
Solusi ditemukan.  
AAABBB  
AFADD  
AAAGG  
CPPPM  
OOKPM
```

Jumlah kasus yang ditinjau: 4
Waktu eksekusi: 1704 ms

Apakah Anda ingin menyimpan solusi dalam bentuk teks? (ya/tidak) ya
Masukkan nama file output: tc2sol.txt
Solusi disimpan ke ../test/tc2sol.txt

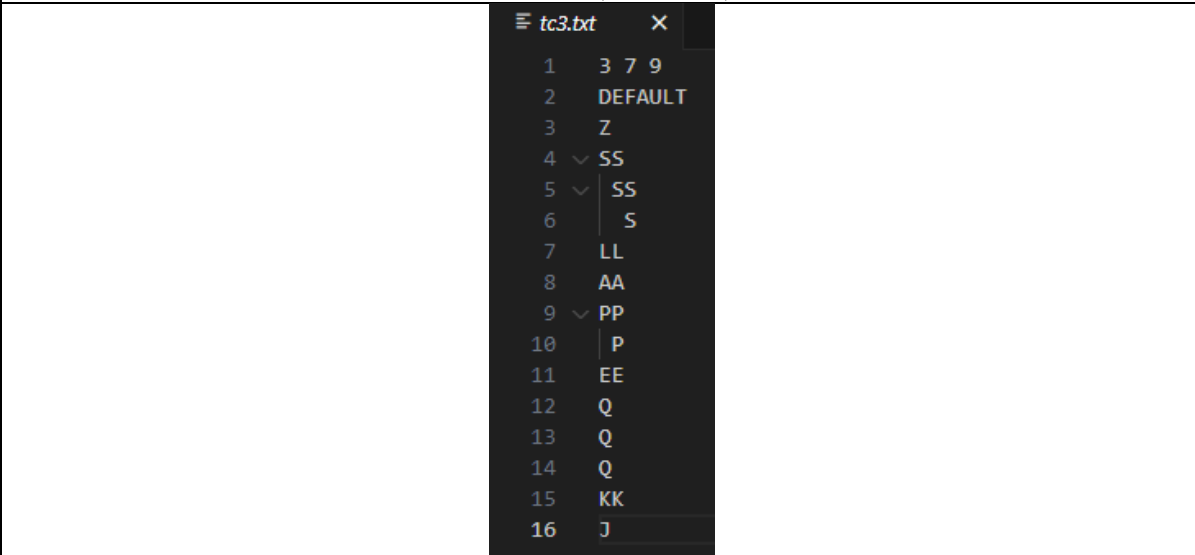
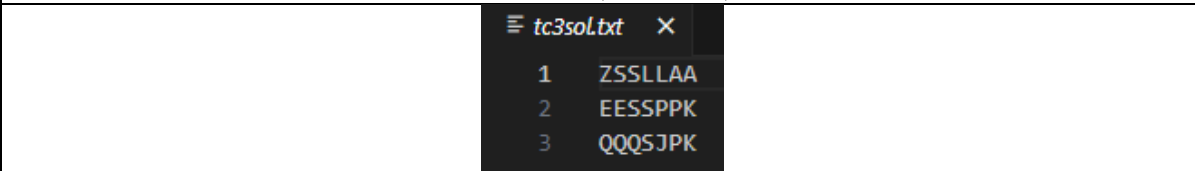
Apakah Anda ingin menyimpan solusi dalam bentuk gambar? (ya/tidak) tidak

Keluaran (File Teks)



4.3 Test Case 3

File masukan valid dan puzzle memiliki solusi.

Masukan (CLI)		
		
Masukan (File Teks)		
		
Keluaran (CLI)		
		
Keluaran (File Teks)		
		

4.4 Test Case 4

File masukan valid dan puzzle memiliki solusi.

Masukan (CLI)		
<pre>===== IQ Puzzle Pro Solver ===== Selamat datang di program IQ Puzzle Pro Solver! Program ini akan membantu Anda menyelesaikan IQ Pro Puzzle. Masukkan nama file puzzle yang ingin diselesaikan: ../test/tc4.txt</pre>		
Masukan (File Teks)		
	<pre>tc4.txt 1 5 5 10 2 DEFAULT 3 I 4 III 5 I 6 AA 7 BB 8 BB 9 O 10 DD 11 D 12 EE 13 F 14 FF 15 GG 16 H 17 JJ</pre>	
Keluaran (CLI)		
	<pre>Solusi ditemukan. OIAAF IIIFF HIEEJ DDBBJ DGGBB Jumlah kasus yang ditinjau: 7321 Waktu eksekusi: 2297 ms Apakah Anda ingin menyimpan solusi dalam bentuk teks? (ya/tidak) ya Masukkan nama file output: tc4sol.txt Solusi disimpan ke ../test/tc4sol.txt Apakah Anda ingin menyimpan solusi dalam bentuk gambar? (ya/tidak) tidak</pre>	
Keluaran (File Teks)		
	<pre>tc4sol.txt 1 OIAAF 2 IIIFF 3 HIEEJ 4 DDBBJ 5 DGGBB</pre>	

4.5 Test Case 5

File masukan valid dan puzzle tidak memiliki solusi.

Masukan (CLI)	
Masukan (File Teks)	
Keluaran (CLI)	

4.6 Test Case 6

File masukan valid dan puzzle memiliki solusi.

Masukan (CLI)	

Masukan (File Teks)		
	<pre> tc6.txt 1 4 8 10 2 DEFAULT 3 AAA 4 A A 5 AAA 6 LL 7 YY 8 XX 9 EEE 10 E 11 U 12 DDD 13 JJJJ 14 J JJ 15 GG 16 H </pre>	
Keluaran (CLI)		
	<pre> Solusi ditemukan. AAALLYU AHAEEXX AAJJJG DDJJJG Jumlah kasus yang ditinjau: 721 Waktu eksekusi: 1953 ms Apakah Anda ingin menyimpan solusi dalam bentuk teks? (ya/tidak) ya Masukkan nama file output: tc6sol.txt Solusi disimpan ke ../test/tc6sol.txt Apakah Anda ingin menyimpan solusi dalam bentuk gambar? (ya/tidak) tidak </pre>	
Keluaran (File Teks)		
	<pre> tc6sol.txt 1 AAALLYU 2 AHAEEXX 3 AAJJJG 4 DDDJJG </pre>	

4.7 Test Case 7

File masukan valid dan puzzle memiliki solusi.

Masukan (CLI)
<pre> ===== \ / \ / \ / \ / \ / \ / \ / \ / \ / \ / \ \ / \ / \ / \ / \ / \ / \ / \ / \ / \ / \ ===== Selamat datang di program IQ Puzzle Pro Solver! Program ini akan membantu Anda menyelesaikan IQ Pro Puzzle. Masukkan nama file puzzle yang ingin diselesaikan: ../test/tc7.txt </pre>

Masukan (File Teks)

```
tc7.txt
1 7 3 10
2 DEFAULT
3 H
4 B
5 X
6 SS
7 O
8 OO
9 O
10 I
11 Y
12 W
13 WWW
14 W
15 F
16 FF
17 F
18 A
```

Keluaran (CLI)

```
Solusi ditemukan.
HBX
SSW
WWW
IFW
FFO
FOO
YAO

Jumlah kasus yang ditinjau: 373
Waktu eksekusi: 1219 ms

Apakah Anda ingin menyimpan solusi dalam bentuk teks? (ya/tidak) ya
Masukkan nama file output: tc7sol.txt
Solusi disimpan ke ../test/tc7sol.txt


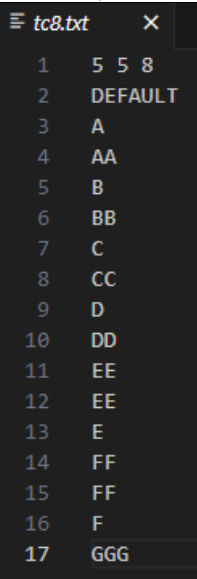
Apakah Anda ingin menyimpan solusi dalam bentuk gambar? (ya/tidak) tidak
```

Keluaran (File Teks)

```
tc7sol.txt
1 HBX
2 SSW
3 WWW
4 IFW
5 FFO
6 FOO
7 YAO
```


4.8 Test Case 8

File masukan tidak valid.

Masukan (CLI)	
	
Masukan (File Teks)	
	
Keluaran (CLI)	
Error: Jumlah piece tidak sesuai dengan nilai P.	

4.9 Test Case 9


File masukan tidak valid.

Masukan (CLI)


Masukan (File Teks)	
	<pre> ≡ tc9.txt X 1 3 7 9 2 DEFAULT 3 Z 4 SS 5 SS 6 S 7 LL 8 AA 9 PP 10 P 11 EE 12 Q 13 Q 14 Q 15 KK 16 E </pre>
Keluaran (CLI)	
Error: Huruf 'E' digunakan oleh lebih dari satu piece yang terpisah.	

4.10 Test Case 10 (Bonus)

File masukan valid dan puzzle memiliki solusi, file keluaran disimpan dalam format .png.

Masukan (CLI)	
 <pre> ===== IQ Puzzle Pro Solver ===== Selamat datang di program IQ Puzzle Pro Solver! Program ini akan membantu Anda menyelesaikan IQ Pro Puzzle. Masukkan nama file puzzle yang ingin diselesaikan: ../test/tc10.txt </pre>	
Masukan (File Teks)	
	<pre> ≡ tc10.txt X 1 4 8 10 2 DEFAULT 3 BB 4 AAA 5 A A 6 AAA 7 WW 8 K 9 XX 10 EEE 11 E 12 U 13 DDD 14 JJJJ 15 J JJ 16 GG </pre>

Keluaran (CLI)	
	<pre> Solusi ditemukan. BBAAAKJJ XXAUWJ GGAAAEJJ DDDEEEJJ Jumlah kasus yang ditinjau: 8673 Waktu eksekusi: 2131 ms Apakah Anda ingin menyimpan solusi dalam bentuk teks? (ya/tidak) tidak Apakah Anda ingin menyimpan solusi dalam bentuk gambar? (ya/tidak) ya Masukkan nama file output gambar: tc10sol File harus berekstensi .png Masukkan nama file output gambar: tc10sol.png Gambar solusi disimpan ke ../test/tc10sol.png </pre>
Keluaran (File Gambar)	
	

4.11 Test Case 11 (Bonus)

File masukan valid dan puzzle memiliki solusi, file keluaran disimpan dalam format .png.

Masukan (CLI)	
	 <pre> ===== Q U N D L E T T E S O L V E R ===== Selamat datang di program IQ Puzzle Pro Solver! Program ini akan membantu Anda menyelesaikan IQ Pro Puzzle. Masukkan nama file puzzle yang ingin diselesaikan: ../test/tc11.txt </pre>

Masukan (File Teks)

```
tc11.txt X
1 3 7 10
2 DEFAULT
3 H
4 B
5 X
6 SS
7 O
8 OO
9 O
10 I
11 Y
12 W
13 WWW
14 W
15 F
16 FF
17 F
18 A
```

Keluaran (CLI)

```
Solusi ditemukan.
HBXOINS
FFYOONS
AFFOWWW

Jumlah kasus yang ditinjau: 901
Waktu eksekusi: 1061 ms

Apakah Anda ingin menyimpan solusi dalam bentuk teks? (ya/tidak) tidak

Apakah Anda ingin menyimpan solusi dalam bentuk gambar? (ya/tidak) ya
Masukkan nama file output gambar: tc11sol.png
Gambar solusi disimpan ke ../test/tc11sol.png
```

Keluaran (File Gambar)

H	B	X	O	I	W	S
F	F	Y	O	O	W	S
A	F	F	O	W	W	W

DAFTAR PUSTAKA

- GeeksforGeeks. Tanpa Tahun. “*Brute force approach and its pros and cons*”. [Online]. Tersedia: <https://www.geeksforgeeks.org/brute-force-approach-and-its-pros-and-cons/>. [23 Februari 2025].
- Munir, R. 2025. “Algoritma Brute Force (Bagian 1). [Online]. Tersedia: [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/02-Algoritma-Brute-Force-\(2025\)-Bag1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/02-Algoritma-Brute-Force-(2025)-Bag1.pdf). [21 Februari 2025].
- Munir, R. 2025. “Tugas Kecil 1: Penyelesaian IQ Puzzler Pro dengan Algoritma Brute Force”. [Online]. Tersedia: <https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/Tucil1-Stima-2025.pdf/>. [18 Februari 2025].
- SmartGames. Tanpa Tahun. “*IQ Puzzler Pro*”. [Online]. Tersedia: <https://www.smartgames.eu/uk/one-player-games/iq-puzzler-pro-0>. [23 Februari 2025].

LAMPIRAN

Lampiran 1. Tautan *Repository*

https://github.com/naomirisaka/Tucil1_13523122

Lampiran 2. Tabel Checklist Program

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	✓	