

LAPORAN TUGAS KECIL 1
IF2211 STRATEGI ALGORITMA
PENYELESAIAN IQ PUZZLER PRO DENGAN ALGORITMA
BRUTE FORCE



DISUSUN OLEH:

Asybel Bintang Pardomuan Sianipar

15223011

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2025

DAFTAR ISI

DAFTAR ISI.....	1
I. DESKRIPSI MASALAH.....	2
II. ALGORITMA.....	3
1. Brute Force Murni.....	3
2. Penyimpanan Data Bentuk Blok sebagai Pasangan (Row, Column).....	3
3. Transformasi Blok (Rotasi & Pencerminan).....	4
4. Penyimpanan dan Pengelolaan Blok.....	4
5. Penempatan Blok dengan blockAnchor.....	5
6. Backtracking dengan Permutasi Urutan Blok.....	6
7. Validasi Penempatan Blok.....	8
8. Streaming Permutasi Peletakkan Blok dan Early Termination.....	9
III. SOURCE CODE PROGRAM DALAM JAVA.....	10
A. Main.java.....	11
B. Transform.java.....	17
C. PuzzleBoard.java.....	19
IV. HASIL EKSEKUSI PROGRAM.....	24
V. LAMPIRAN.....	31
A. Pranala Repository Github.....	31
B. Tabel Ketercapaian.....	31

I. DESKRIPSI MASALAH

IQ Puzzler Pro adalah permainan teka-teki yang tiap pemainnya harus menyusun potongan-potongan puzzle unik untuk memenuhi papan berukuran $N \times M$. Setiap potongan memiliki bentuk yang berbeda dan dapat diputar atau dicerminkan untuk dicocokkan ke dalam papan.

Dalam tugas ini, program dibuat untuk menyelesaikan IQ Puzzler Pro menggunakan **algoritma Brute Force**, yang mencoba semua kemungkinan susunan blok hingga menemukan solusi yang valid atau menyatakan bahwa tidak ada solusi.

Program akan:

- a. Membaca file input yang berisi ukuran papan, jumlah blok, tipe kasus (**DEFAULT** atau **CUSTOM**), dan bentuk setiap blok.
- b. Mencoba menyusun blok-blok dengan rotasi dan pencerminan.
- c. Menampilkan hasil solusi (jika ada), waktu eksekusi, serta jumlah iterasi yang dilakukan.

Langkah-langkah penyelesaian:

1. Membaca Input
 - a. Program membaca ukuran papan ($N \times M$), jumlah blok (P), serta konfigurasi awal dari file teks.
 - b. Pada mode **CUSTOM**, area papan yang harus diisi ditandai dengan 'X', sedangkan '.' adalah area kosong.
2. Menghasilkan Semua Kemungkinan Transformasi Blok
 - a. Setiap blok dapat dirotasi searah jarum jam (90°) dan setiap variasi rotasi akan dicerminkan secara horizontal/vertikal.
 - b. Hanya menyimpan transformasi unik agar tidak ada duplikasi dalam pencarian solusi.
3. Mencoba Semua Susunan Blok dengan Backtracking
 - a. Program menempatkan blok satu per satu pada papan.
 - b. Jika blok bisa ditempatkan, program melanjutkan ke blok berikutnya.
 - c. Jika tidak, program melakukan backtrack dan mencoba variasi lain.
4. Menampilkan Hasil
 - a. Jika solusi ditemukan, program mencetak papan dalam bentuk visual.
 - b. Jika tidak ada solusi, program mencetak "Solusi tidak ditemukan."
 - c. Program juga menampilkan waktu eksekusi dan jumlah iterasi yang dilakukan.

II. ALGORITMA

1. Brute Force Murni

Brute Force adalah pendekatan dengan mencoba **semua kemungkinan solusi** hingga menemukan hasil yang benar. Dalam kasus ini, program mencoba **semua kemungkinan susunan blok** di papan IQ Puzzler Pro, termasuk semua **rotasi dan pencerminan**.

Dalam bentuk **murni**, algoritma Brute Force tidak menggunakan **pruning (pemotongan jalur pencarian yang buruk)** atau **heuristik**. Setiap kombinasi dicoba secara menyeluruh tanpa optimasi tambahan.

Karena jumlah kemungkinan kombinasi sangat besar, penulis menerapkan backtracking untuk mengurangi eksplorasi jalur yang sudah pasti tidak valid.

2. Penyimpanan Data Bentuk Blok sebagai Pasangan (Row, Column)

Dalam program ini, setiap bentuk blok disimpan sebagai sekumpulan koordinat pasangan (row, column) yang merepresentasikan posisi relatif setiap bagian blok. Metode ini efektif dibandingkan representasi berbasis matriks atau string karena memberikan fleksibilitas lebih dalam transformasi. Selain itu, metode ini juga mudah dipahami dan efisien untuk banyak kasus.

Contoh: Misalkan terdapat blok berbentuk **cerminan L** seperti berikut (huruf A merepresentasikan subbagian blok):

A
A
AA

Blok tersebut akan memiliki *arrangement*: [(0,1), (1,1), (2,0), (2,1)]

3. Transformasi Blok (Rotasi & Pencerminan)

Setiap blok dalam permainan dapat diubah menggunakan:

- **Rotasi:** Diputar 90° searah jarum jam.
- **Pencerminan:** Dibalik secara **horizontal** atau **vertikal**.

Kode berikut menunjukkan bagaimana penulis melakukan **rotasi 90° searah jarum jam**:

```
// clockwise rotation function
public static ArrayList<Main.Pair> rotateCW(ArrayList<Main.Pair> arrangement) {
    ArrayList<Main.Pair> rotatedArr = new ArrayList<>();
    // finding max row as the rotation anchor/reference (because the position (0,0) ->
    // (0, max_row_of_initial_state) after 90 degree rotation CW)
    int max_row = MaxRowColumn(arrangement).row();
    int new_row, new_column;
    for (Main.Pair p : arrangement) {
        if (p.row() > max_row) {
            max_row = p.row();
        }
    }
    for (Main.Pair p : arrangement) {
        new_row = p.column();
        new_column = max_row - p.row();
        rotatedArr.add(new Main.Pair(new_row, new_column));
    }

    return rotatedArr;
}
```

Setiap variasi transformasi (rotasi dan pencerminan) yang dihasilkan disimpan dalam struktur **HashMap**, dengan `block_id` (A, B, C, ...) sebagai *key* dan variasi *block arrangement* sebagai *values*.

4. Penyimpanan dan Pengelolaan Blok

Setiap blok diwakili oleh karakter huruf A-Z dan disimpan dalam struktur data berikut:

```
HashMap<Character, ArrayList<ArrayList<Main.Pair>>> blocks;
```

Program akan menghilangkan duplikasi untuk menghindari eksplorasi solusi yang sama berulang kali, yang dapat menyebabkan pemborosan waktu dan sumber daya komputasi. Setiap blok dapat dirotasi dan dicerminkan memiliki potensi terdapat duplikasi, tetapi beberapa transformasi mungkin menghasilkan bentuk yang identik. Jika tidak dihapus, algoritma akan mencoba variasi yang sebenarnya sama berulang kali. Oleh karena itu, program hanya menyimpan transformasi yang unik untuk setiap blok dengan fungsi *helper*: `cleanBlockVariations(blocks)`.

5. Penempatan Blok dengan `blockAnchor`

blockAnchor adalah titik referensi (biasanya titik kiri atas) untuk menempatkan blok di papan. Penempatan blok dilakukan dengan **menyesuaikan koordinat setiap sub-blok** agar sesuai dengan titik **blockAnchor**:

```
boolean validity = true;
for (Main.Pair p : arrangement) {
    int row_position = blockAnchor.row() + p.row();
    int column_position = blockAnchor.column() + p.column();
    if (!isValidPosition(row_position, column_position)) {
        validity = false;
    }
}
```

Jika valid, blok diletakkan pada papan.

Dalam program ini, penulis ingin **memastikan bahwa setidaknya satu bagian dari suatu blok selalu berada di titik ankur (`blockAnchor`) saat ditempatkan di papan**. Untuk mencapai ini, penulis melakukan **normalisasi koordinat blok** sebelum menyimpannya atau menggunakannya dalam proses pencarian solusi.

Normalisasi dilakukan dengan **menggeser semua koordinat blok** sedemikian rupa sehingga bagian paling atas dan paling kiri dari blok selalu berada di **koordinat (0,0)** dalam sistem koordinat relatifnya.

Normalisasi dilakukan dengan **menggeser seluruh koordinat blok** sehingga salah satu bagian blok selalu berada di titik (0,0).

Langkah-langkah normalisasi:

1. **Cari titik (row_minimum, col_minimum)** → Menentukan bagian blok yang paling atas dan paling kiri.
2. **Kurangi setiap koordinat dengan (row_minimum, col_minimum)** → Menggeser semua bagian blok sehingga bagian paling atas dan paling kiri menjadi (0,0).

Sebagai contoh, misal blok “A” berbentuk cerminan L sebelumnya [(0,1), (1,1), (2,0), (2,1)] ingin diletakkan dalam sebuah papan. Setelah normalisasi, blok “A” akan memiliki koordinat relatif [(0,0), (1,0), (2,-1), (2,0)].

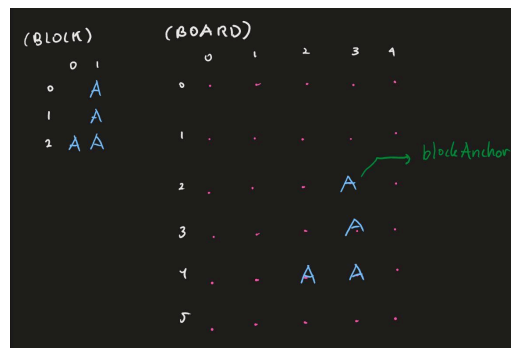
Jika blockAnchor ditetapkan ke (2,3), maka program cukup menjumlahkan setiap koordinat dengan nilai anchor, sehingga didapatkan koordinat absolut:

$$(2+0, 3+0) = (2,3)$$

$$(2+1, 3+0) = (3,3)$$

$$(2+2, 3 + (-1)) = (4,2)$$

$$(2+2, 3+0) = (4,3)$$



Lalu program akan mengecek apakah posisi tersebut valid untuk ditempati (semua subbagian dari blok menempati posisi kosong dalam papan). Jika valid, maka blok ditempatkan di papan. Jika belum valid, program mencoba rotasi, pencerminan, atau backtracking untuk mencari solusi lain.

Catatan: ilustrasi di atas hanyalah contoh. Program yang sebenarnya akan menjadikan titik (0,0) sebagai blockAnchor karena titik tersebut merupakan titik kosong yang pertama (paling atas dan paling kiri).

6. Backtracking dengan Permutasi Urutan Blok

Program menggunakan **rekursi** untuk mencoba semua kemungkinan urutan penempatan blok. Jika suatu jalur tidak menghasilkan solusi, program **kembali (backtrack)** dan mencoba kemungkinan lain.

Ringkasan Alur Kerja Rekursi

1. Memilih blok yang akan ditempatkan.
2. Mencoba setiap bentuk rotasi dan pencerminan blok.
3. Mencari posisi kosong pertama (block anchor) dan menempatkan blok jika memungkinkan (tidak menabrak blok lain dan tidak keluar dari papan).
4. Jika blok berhasil ditempatkan, rekursi dipanggil untuk menempatkan blok berikutnya.
5. Jika seluruh blok telah ditempatkan dan papan penuh, solusi ditemukan, program akan *bubble up* dan mengembalikan **true**.
6. Jika penempatan blok tidak menghasilkan solusi, backtracking dilakukan:
 - Blok yang sudah ditempatkan dihapus.
 - Program kembali ke langkah sebelumnya untuk mencoba alternatif lain.
7. Jika semua kombinasi telah diuji dan tidak ada solusi, program mengembalikan **false**.

Pseudocode *solver*:

```
FUNCTION solvePuzzle(blockIndex, block_ids, placedBlocks, blocks, path)
    path.exploredCases++

    IF gameFinishStatus() AND placedBlocks.SIZE() == block_ids.SIZE() THEN
        RETURN TRUE

    id ← block_ids[blockIndex]

    FOR each variation i IN blocks[id] DO
        arrangement ← blocks[id][i]
        blockPlaced ← placeBlock(id, arrangement, findEmptyBlockAnchor())

        IF blockPlaced THEN
            path.recordPath(id, i)
            placedBlocks.ADD(id)

            IF solvePuzzle(blockIndex + 1, block_ids, placedBlocks, blocks, path) THEN
```



```
RETURN TRUE

REMOVE BLOCK (BACKTRACK)

RETURN FALSE
```

7. Validasi Penempatan Blok

Sebelum meletakkan blok, program memeriksa apakah semua bagian dari blok bisa ditempatkan tanpa menabrak blok lain atau ada bagian dari blok yang keluar dari papan. Dengan metode ini, program menguji semua kemungkinan solusi sampai solusi yang valid ditemukan atau semua kemungkinan habis.

Pseudocode untuk menempatkan blok dan validasi penempatan blok:

```
FUNCTION placeBlock(id, arrangement, blockAnchor)
    first_row ← arrangement[0].row
    first_col ← arrangement[0].column

    NORMALIZE arrangement USING first_row, first_col

    FOR each p IN arrangement DO
        row_position ← blockAnchor.row + p.row
        col_position ← blockAnchor.column + p.column
        IF NOT isValidPosition(row_position, col_position) THEN
            RETURN FALSE

    PLACE BLOCK ON BOARD
    RETURN TRUE

FUNCTION isValidPosition(row, column)
    TRY
        RETURN board[row][column] == '0' // Hanya posisi kosong ('0') yang valid
    CATCH EXCEPTION
        RETURN FALSE // Jika out of bounds (subbagian blok keluar papan), tidak valid
```

8. Streaming Permutasi Peletakkan Blok dan Early Termination

Alih-alih menghasilkan semua permutasi sekaligus dan menyimpannya dalam daftar (yang tidak efisien), program menggunakan pendekatan streaming dengan **Lexicographic Permutation**.

Program menguji setiap permutasi urutan penempatan blok langsung saat permutasi tersebut telah dibentuk. Jika salah satu permutasi berhasil menyelesaikan puzzle, program langsung berhenti.

Kode berikut menggambarkan bagaimana program mencoba setiap permutasi sambil berjalan:

```
// Permutasi pertama dari block_ids
if (board.solvePuzzle(blockIndex:0, singleBlockPermutation, path.placedBlocks, blocks, path)) {
    answerFound = true;
}
// Menghasilkan permutasi baru di tiap loop sembari mencoba apakah permutasi tersebut dapat menyelesaikan IQ Puzzler Pro
while (!answerFound && nextPermutation(singleBlockPermutation)) {
    if (board.solvePuzzle(blockIndex:0, singleBlockPermutation, path.placedBlocks, blocks, path)) {
        answerFound = true;
    }
}
```

Pada kode di atas:

- **solvePuzzle** langsung mengembalikan *true* jika solusi ditemukan.
- **Loop berhenti segera setelah solusi ditemukan**, sehingga tidak perlu mengeksplorasi permutasi lainnya.

Lexicographic permutation streaming diperlukan sehingga program tidak harus menyimpan terlebih dahulu seluruh kombinasi permutasi yang bisa berjumlah sangat besar (berpotensi menyebabkan *error out of memory*), seperti 12! kombinasi untuk P = 12.

III. ***SOURCE CODE PROGRAM DALAM JAVA***

Program ini dituliskan dalam bahasa pemrograman Java. Struktur file dan fungsinya:

- |— src/ (Folder berisi source code utama)
 - |— Main.java → Entry point program. Mengatur eksekusi utama dan pemanggilan algoritma.
 - |— ReadFile.java → Membaca file input yang berisi konfigurasi permainan.
 - |— PuzzleBoard.java → Mengelola papan permainan, penempatan blok, dan algoritma brute force (backtracking).
 - |— Transform.java → Mengatur rotasi dan pencerminan blok agar semua kemungkinan bentuk diuji.
 - |— Path.java → Melacak jalur yang sudah dicoba selama pencarian solusi.
 - |— PrettyOutput.java → Mencetak hasil permainan dalam format warna ke terminal atau menyimpan gambar hasil.

Dari struktur ini, file utama yang berhubungan langsung dengan algoritma brute force adalah Main.java, Transform.java, PuzzleBoard.java, dan Path.Java.

Catatan: file test input “.txt” terdapat dalam folder “test_input/”; output test terdapat dalam folder “test/”.

Screenshot source code:

A. Main.java

```
1 import java.io.BufferedReader;
2 import java.io.FileWriter;
3 import java.io.InputStreamReader;
4 import java.nio.file.Paths;
5 import java.util.*;
6
7 public class Main {
8     public static void main(String[] args) throws Exception {
9         // MEMBACA FILE "*.TXT"
10        // Scanner object untuk user input
11        BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
12        System.out.print(s:"Enter filename (including .txt): "); // e.g.: ./test/test1.txt
13        String filename = reader.readLine();
14
15        String fileContent = ReadFile.readFile(filename);
16        String[] parsedFile = parseFile(fileContent);
17
18        // MEMPROSES PARAMETER UTAMA
19        // menyortir parameter utama N M P dan S
20        // Split baris pertama parsedFile berdasarkan whitespace
21        String[] tokens = parsedFile[0].trim().split(regex:"\\s+");
22
23        // membaca parameter: N, M, P
24        int N = Integer.parseInt(tokens[0]);
25        int M = Integer.parseInt(tokens[1]);
26        int P = Integer.parseInt(tokens[2]);
27
28        // membaca kategori puzzle (DEFAULT, CUSTOM, or PYRAMID)
29        String S = parsedFile[1];
30        int startParsingBlocks = 0;
31        if (S.equalsIgnoreCase(anotherString:"DEFAULT")) {
32            startParsingBlocks = 2;
33        }
34        else if (S.equalsIgnoreCase(anotherString:"CUSTOM")) {
35            startParsingBlocks = 2 + N; // input block dalam .txt
36            // berada di bawah konfigurasi papan CUSTOM yang setinggi N.
37        }
38        else {
39            throw new java.lang.Error(message:"Board category invalid. Board category seharusnya
40            DEFAULT atau CUSTOM.");
41        }
42    }
43 }
```

```

41 // MEMBACA BLOK PUZZLE
42 // menyortir blok dan menyetor data arrangement pertama tiap blok
43 HashMap<Character, ArrayList<ArrayList<Pair>>> blocks = new HashMap<>
44 ();
45 ArrayList<Character> block_ids = new ArrayList<>();
46 char id = ' '; // initialize id
47 Set<Character> seen_ids = new HashSet<>();
48 int row = 0;
49 int column = 0;
50 for (int i = startParsingBlocks; i < parsedFile.length; i++) {
51     boolean foundAlphabet = false;
52     Character current_id = ' ';
53     for (char ch: parsedFile[i].toCharArray()) {
54         if (Character.isAlphabetic(ch)) {
55             current_id = ch;
56             foundAlphabet = true;
57             break;
58         }
59     }
60     // Jika tidak ada karakter alfabet yang ditemukan dalam suatu
61     // line, input invalid.
62     if (!foundAlphabet) {
63         throw new IllegalArgumentException("Invalid input at line " +
64             (i + 1) + ": Tidak ada karakter alfabet yang ditemukan.");
65     }
66     // Jika id blok ini sudah pernah diinput sebelumnya, tetapi bukan
67     // id terakhir yang diinput -> blok diskontinu
68     if (seen_ids.contains(current_id) && !current_id.equals(id)) {
69         throw new IllegalArgumentException("Invalid input at line " +
70             (i + 1) + ": Blok dengan id '" + current_id + "' diskontinu.");
71     }
72     // Menyetor data blok jika telah dibuktikan valid
73     seen_ids.add(current_id);
74     id = current_id;
75     if (!block_ids.contains(id)) {
76         row = 0;
77         column = 0;
78         block_ids.add(id);
79         blocks.putIfAbsent(id, new ArrayList<>());
80         blocks.get(id).add(new ArrayList<>());
81     }
82     for (char ch: parsedFile[i].toCharArray()) {
83         if (!Character.isWhitespace(ch)) {
84             blocks.get(id).get(index:0).add(new Pair(row,
85                 column)); // Menyimpan koordinat blok pertama
86         }
87         column++;
88     }
89     column = 0;
90     row++;
91 }

```

```

90 // Mengecek validitas input (CONSTRAINTS)
91 // Mengecek apakah parameter P (jumlah blok) sesuai dengan jumlah id terdaftar
92 if (block_ids.size() != P) {
93     throw new java.lang.Error("Parameter input P (jumlah blok) tidak sesuai dengan
94         jumlah id blok unik yang terdaftar." + "\n\nJumlah id terdaftar: " + block_ids.size
95         () + "\nP input: " + P + "\n");
96 }
97
98 // Tidak ada dua blok yang direpresentasikan alfabet yang sama.
99 Set<Character> uniqueBlock_ids = new HashSet<>();
100 for (Character block_id : block_ids) {
101     if (!uniqueBlock_ids.add(block_id)) { // Jika gagal ditambahkan, berarti sudah ada
102         duplikat.
103         throw new java.lang.Error(message:"Terdapat blok yang direpresentasikan alfabet
104         yang sama.");
105     }
106 }
107
108 // MEMPROSES BLOK PUZZLE
109 // menciptakan semua kemungkinan variasi tiap blok (rotasi dan cermin)
110 for (Map.Entry<Character, ArrayList<ArrayList<Pair>>> entry : blocks.entrySet()) {
111     // take id and entry
112     id = entry.getKey();
113     ArrayList<ArrayList<Pair>> arrangements = entry.getValue();
114     ArrayList<ArrayList<Pair>> newVariations = new ArrayList<>();
115     // ROTATION
116     int num_of_rotation = 3; // variasi rotasi <= 3
117     ArrayList<Pair> arrangement = arrangements.get(index:0); // starting arrangement
118     for (int i = 0; i < num_of_rotation; i++) {
119         arrangement = Transform.rotateCW(arrangement);
120         newVariations.add(arrangement);
121     }
122     // // END ROTATION
123     blocks.get(id).addAll(newVariations); // add rotation
124     variations to the arrangements (before they are mirrored.)
125     newVariations = new ArrayList<>();
126     // MIRROR
127     for (ArrayList<Pair> arr : arrangements) {
128         arrangement = Transform.mirrorV(arr);
129         newVariations.add(arrangement);
130         arrangement = Transform.mirrorH(arr);
131         newVariations.add(arrangement);
132     }
133     // // END MIRROR
134     blocks.get(id).addAll(newVariations); // add mirror
135     variations to the arrangements.
136 }
137
138 // membersihkan variasi tiap blok (agar tidak ada variasi duplikat)
139 blocks = Transform.cleanBlockVariations(blocks);

```

```

135 // MENYELESAIKAN PUZZLE
136 PuzzleBoard board = new PuzzleBoard(N, M, S);
137 if (S.equalsIgnoreCase(anotherString:"CUSTOM")) {
138     board.insertCustomConfiguration(Arrays.copyOfRange(parsedFile, from:2, 2 + N));
139 }
140 board.buildBoard();
141 Path path = new Path(block_ids);
142 boolean answerFound = false;
143 ArrayList<Character> singleBlockPermutation = block_ids;
144 Collections.sort(singleBlockPermutation);
145
146 long startTime = System.currentTimeMillis(); // mengukur waktu
147     awal algoritma
148     // Permutasi pertama dari block_ids
149     if (board.solvePuzzle(blockIndex:0, singleBlockPermutation, path.placedBlocks,
150         blocks, path)) {
151         answerFound = true;
152         // Menghasilkan permutasi baru di tiap loop sembari mencoba apakah permutasi tersebut
153         // dapat menyelesaikan IQ Puzzler Pro
154         while (!answerFound && nextPermutation(singleBlockPermutation)) {
155             if (board.solvePuzzle(blockIndex:0, singleBlockPermutation, path.placedBlocks,
156                 blocks, path)) {
157                 answerFound = true;
158             }
159         }
160     }
161     long endTime = System.currentTimeMillis(); // mengukur waktu
162     akhir algoritma
163     long executionTime = endTime - startTime; // Waktu eksekusi
164     dalam milisecond

```

```

160 // OUTPUT PRINT
161 if (answerFound == true) {
162     board.printBoard();
163 }
164 else {
165     System.out.println(x: "\nSolusi tidak ditemukan.");
166 }
167 System.out.println("\n" + "Waktu pencarian: " + executionTime + " ms");
168 System.out.println("\n" + "Banyak kasus yang ditinjau: " + path.exploredCases);
169 System.out.println("\n" + "Apakah Anda ingin menyimpan solusi? (ya/tidak) ");
170 String menyimpanSolusi = reader.readLine();
171
172 // MENYIMPAN SOLUSI
173 String inputFilename = Paths.get(filename).getFileName().toString().replaceAll
(regex: "\\..txt$", replacement: ""); // Get file name
174 if (menyimpanSolusi.equalsIgnoreCase(anotherString: "ya")){
175     try (FileWriter writer = new FileWriter(String.format(format: "test/%s_output.
txt", inputFilename))) {
176         writer.write(board.boardToText());
177         System.out.println(x: "File saved successfully as 'output.txt'.");
178     }
179     PrettyOutput.generatePuzzleImage(board.board, String.format(format: "test/%s.png",
inputFilename));
180 }
181 }
182

```



```

184 // HELPER FUNCTIONS
185 // function to parse input file
186 public static String[] parseFile(String args) {
187     String[] texts = args.split(regex:"\\n"); // Split by new line
188     return texts;
189 }
190
191 You, 24 hours ago | 1 author (You)
192 // function for (row, column) pairs
193 public record Pair(int row, int column) {
194     @Override
195     public String toString() {
196         return "(" + row + "," + column + ")"; // for better formatting
197     }
198 }
199 // permutation function
200 private static boolean nextPermutation(ArrayList<Character> arr) {
201     int n = arr.size();
202     // Find largest index k where arr[k] < arr[k+1].
203     int k = n - 2;
204     while (k >= 0 && arr.get(k) >= arr.get(k + 1)) {
205         k--;
206     }
207     if (k < 0) {
208         return false; // no more permutations
209     }
210     // Find largest index l > k such that arr[k] < arr[l].
211     int l = n - 1;
212     while (arr.get(k) >= arr.get(l)) {
213         l--;
214     }
215     // Swap arr[k] and arr[l].
216     Collections.swap(arr, k, l);
217     // Reverse the sub-list from k+1 to the end.
218     reverse(arr, k + 1, n - 1);
219     return true;
220 }
221
222 private static void reverse(List<Character> arr, int start, int end) {
223     while (start < end) {
224         Collections.swap(arr, start, end);
225         start++;
226         end--;
227     }
228 }
229 }
230
231

```

B. Transform.java

```
1  import java.util.ArrayList;
2  import java.util.Comparator;
3  import java.util.HashMap;
4  import java.util.Map;
5
6  You, 19 hours ago | 1 author (You)
7  public class Transform {
8      // 2D GEOMETRY TRANSFORMATION FUNCTIONS
9      // clockwise rotation function
10     public static ArrayList<Main.Pair> rotateCW(ArrayList<Main.Pair> arrangement) {
11         ArrayList<Main.Pair> rotatedArr = new ArrayList<>();
12         // finding max row as the rotation anchor/reference (because the position (0,0) -> (0,
13         // max_row_of_initial_state) after 90 degree rotation CW)
14         int max_row = MaxRowColumn(arrangement).row();
15         int new_row, new_column;
16         for (Main.Pair p : arrangement) {
17             if (p.row() > max_row) {
18                 max_row = p.row();
19             }
20         }
21         for (Main.Pair p : arrangement) {
22             new_row = p.column();
23             new_column = max_row - p.row();
24             rotatedArr.add(new Main.Pair(new_row, new_column));
25         }
26         return rotatedArr;
27     }
28
29     // mirror functions You, 2 days ago • feat(geometry transformation): hardcoded transf...
30     public static ArrayList<Main.Pair> mirrorV(ArrayList<Main.Pair> arrangement) {
31         ArrayList<Main.Pair> mirroredArr = new ArrayList<>();
32         // vertical mirror (cerminkan terhadap kolom -> posisi kolom berubah; baris tetap)
33         int max_column = MaxRowColumn(arrangement).column();
34         for (Main.Pair p : arrangement) {
35             mirroredArr.add(new Main.Pair(p.row(), max_column - p.column()));
36         }
37         return mirroredArr;
38     }
39
40     public static ArrayList<Main.Pair> mirrorH(ArrayList<Main.Pair> arrangement) {
41         ArrayList<Main.Pair> mirroredArr = new ArrayList<>();
42         // horizontal mirror (cerminkan terhadap baris -> posisi baris berubah; kolom tetap)
43         int max_row = MaxRowColumn(arrangement).row();
44         for (Main.Pair p : arrangement) {
45             mirroredArr.add(new Main.Pair(max_row - p.row(), p.column()));
46         }
47         return mirroredArr;
48     }
49 }
```

```

46 // finding max row and max column in "list of Main.Pairs"
47 public static Main.Pair MaxRowColumn(ArrayList<Main.Pair> arrangement) {
48     int max_row = -1, max_column = -1;
49     for (Main.Pair p : arrangement) {
50         if (p.row() > max_row) {
51             max_row = p.row();
52         }
53         if (p.column() > max_column) {
54             max_column = p.column();
55         }
56     }
57     return new Main.Pair(max_row, max_column);
58 }
59
60 // ADDITIONAL FUNCTION
61 public static HashMap<Character, ArrayList<ArrayList<Main.Pair>>> cleanBlockVariations(HashMap<Character,
62     ArrayList<ArrayList<Main.Pair>>> blocks) {
63     for (Map.Entry<Character, ArrayList<ArrayList<Main.Pair>>> entry : blocks.entrySet()) {
64         // take id and entry
65         char id = entry.getKey();
66         ArrayList<ArrayList<Main.Pair>> arrangements = entry.getValue();
67
68         ArrayList<ArrayList<Main.Pair>> uniqueArrangements = new ArrayList<>();
69         for (ArrayList<Main.Pair> arr : arrangements) {
70             arr.sort(Comparator.comparingInt(Main.Pair::row).thenComparingInt(Main.Pair::column));
71             if (!uniqueArrangements.contains(arr)) {
72                 uniqueArrangements.add(arr);
73             }
74         }
75         blocks.put(id, uniqueArrangements); // overwrite arrangements dengan uniqueArrangements
76     }
77     return blocks;
78 }

```

C. PuzzleBoard.java

```
1  import java.util.*;
   You, 17 hours ago | 1 author (You)
2  public class PuzzleBoard {
3      private int rows;
4      private int columns;
5      private String category;
6      private String[] customConfiguration;
7      public char[][] board;      You, 17 hours ago • feat(output): add text output, CLI print colore...
8
9      // object instantiation
10     public PuzzleBoard(int rows, int columns, String category) {
11         this.rows = rows;
12         this.columns = columns;
13         this.category = category;
14         this.board = new char[rows][columns];
15     }
16
17     public void buildBoard() {
18         // initialize board with '0' (empty space)
19         // DEFAULT (RECTANGULAR)
20         if (this.category.equalsIgnoreCase(anotherString:"DEFAULT")) {
21             for (int r = 0; r < rows; r++) {
22                 for (int c = 0; c < columns; c++) {
23                     board[r][c] = '0';
24                 }
25             }
26         }
27         // CUSTOM
28         else if (this.category.equalsIgnoreCase(anotherString:"CUSTOM")) {
29             for (int r = 0; r < rows; r++) {
30                 String customRow = customConfiguration[r];
31                 for (int c = 0; c < columns; c++) {
32                     char customColumn = customRow.charAt(c);
33                     if (customColumn == 'X' || customColumn == 'x') {
34                         board[r][c] = '0';
35                     }
36                     else if (customColumn == '.') {
37                         board[r][c] = '.';
38                     } else {
39                         throw new java.lang.Error(message:"custom board configuration format tidak valid.");
40                     }
41                 }
42             }
43         }
44     }
```

```

46 public void insertCustomConfiguration(String[] customConfiguration) {
47     this.customConfiguration = customConfiguration;
48 }
49
50 public boolean placeBlock(char id, ArrayList<Main.Pair> arrangement, Main.Pair blockAnchor){
51     // memperbaiki/normalisasi block yang tidak start dari (0,0) (agar tiap arrangement dipastikan memiliki 1
    subblock dengan Pair(0,0) yang menempati posisi blockAnchor)
52     int first_row = arrangement.get(index:0).row();
53     int first_column = arrangement.get(index:0).column();
54     if (first_row != 0 || first_column != 0) {
55         for (int i = 0; i < arrangement.size(); i++) {
56             arrangement.set(i, new Main.Pair(arrangement.get(i).row() - first_row, arrangement.get(i).column() -
                first_column));
57         }
58     }
59     // meletakkan block dalam bidang jika memenuhi syarat.
60     boolean validity = true;
61     for (Main.Pair p : arrangement) {
62         int row_position = blockAnchor.row() + p.row();
63         int column_position = blockAnchor.column() + p.column();
64         if (!isValidPosition(row_position, column_position)) {
65             validity = false;
66         }
67     }
68     if (validity) {
69         for (Main.Pair p : arrangement) {
70             int row_position = blockAnchor.row() + p.row();
71             int column_position = blockAnchor.column() + p.column();
72             board[row_position][column_position] = id;
73         }
74     }
75     return validity;
76 }
77

```

```

78 public boolean solvePuzzle(Integer blockIndex, ArrayList<Character> block_ids, ArrayList<Character> placedBlocks,
79 HashMap<Character, ArrayList<ArrayList<Main.Pair>>> blocks, Path path) {
80     path.exploredCases++; // menghitung jumlah iterasi ditinjau dari algoritma rekursif
81
82     if (this.gameFinishStatus() && placedBlocks.size() == block_ids.size()) {
83         return true;
84     } else {
85         char id = block_ids.get(blockIndex);
86
87         // note: default path.getPath(id) = -1
88         for (int i = 0; i < blocks.get(id).size(); i++) {
89             ArrayList<Main.Pair> arrangement = blocks.get(id).get(i);
90             boolean blockPlaced = this.placeBlock(id, arrangement, this.findEmptyBlockAnchor());
91             if (blockPlaced) {
92                 path.recordPath(id, i);
93                 placedBlocks.add(id);
94                 // recursion
95                 if (solvePuzzle(blockIndex + 1, block_ids, placedBlocks, blocks, path)) {
96                     return true;
97                 }
98                 // backtrack
99                 path.exploredCases++; // menghitung jumlah iterasi ditinjau dari algoritma backtrack
100
101                 // reset blok yang telah diletakkan menjadi "0" kembali
102                 for (int r = 0; r < rows; r++) {
103                     for (int c = 0; c < columns; c++) {
104                         if (board[r][c] == id) {
105                             board[r][c] = '0';
106                         }
107                     }
108                 }
109                 placedBlocks.remove(placedBlocks.size() - 1); // unplace/undo block
110             }
111         }
112         return false;
113     }
114 }
115 public boolean gameFinishStatus() {

```

```

115     public boolean gameFinishStatus() {
116         // check if there are no empty spaces left
117         for (int r = 0; r < rows; r++) {
118             for (int c = 0; c < columns; c++) {
119                 if (board[r][c] == '0') {
120                     return false;
121                 }
122             }
123         }
124         return true;
125     }
126
127     public Main.Pair findEmptyBlockAnchor() {
128         // mencari blockAnchor (urut dari kiri ke kanan lalu dari atas ke bawah, untuk mencari posisi yang masih
        kosong)
129         for (int r = 0; r < rows; r++) {
130             for (int c = 0; c < columns; c++) {
131                 if (board[r][c] == '0') {
132                     return new Main.Pair(r, c);
133                 }
134             }
135         }
136         return new Main.Pair(-1, -1);
137     }
138

```

```

139     public void printBoard() {
140         for (char[] row : board) {
141             StringBuilder coloredRow = new StringBuilder();
142             for (char column : row) {
143                 coloredRow.append(PrettyOutput.getColoredChar(column)).append(str: " "); // Space for better
                readability
144             }
145             System.out.println(coloredRow);
146         }
147     }
148
149     public String boardToText() {
150         List<String> boardText = new ArrayList<>();
151         for (char[] row : board) {
152             StringBuilder coloredRow = new StringBuilder();
153             for (char column : row) {
154                 char plainChar = column;
155                 coloredRow.append(plainChar);
156             }
157             boardText.add(coloredRow.toString());
158         }
159         return String.join(delimiter: "\n", boardText); // Join rows with newline characters
160     }
161
162     private boolean isValidPosition(int row, int column) {
163         try {
164             return board[row][column] == '0';
165         } catch (Exception e) {
166             return false;
167         }
168     }
169 }
170

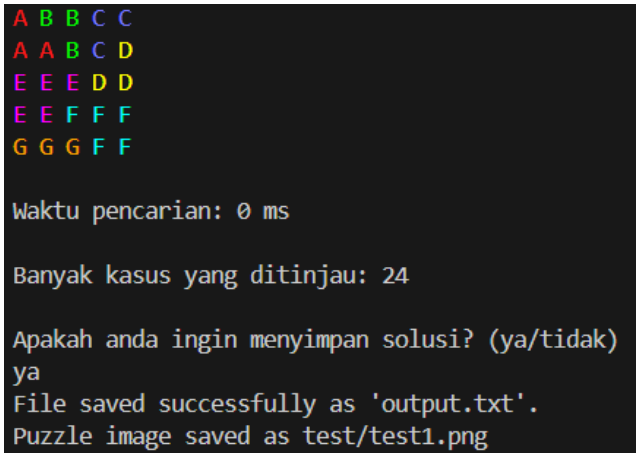
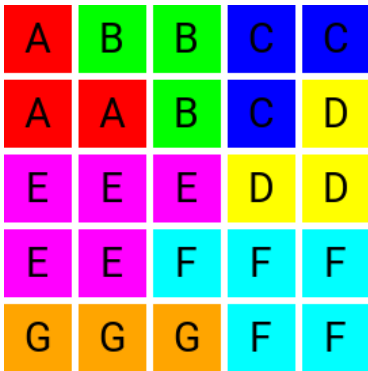
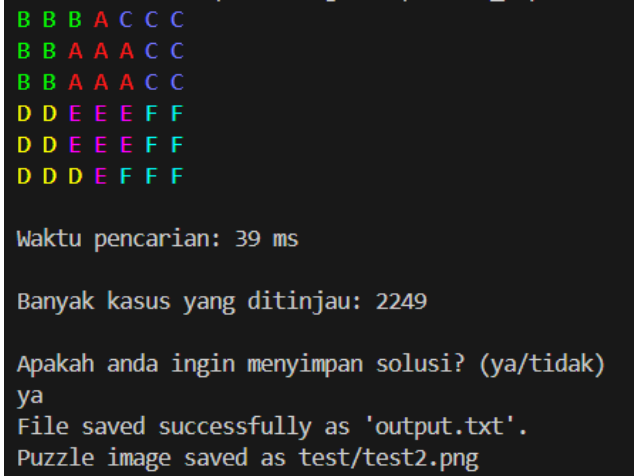
```

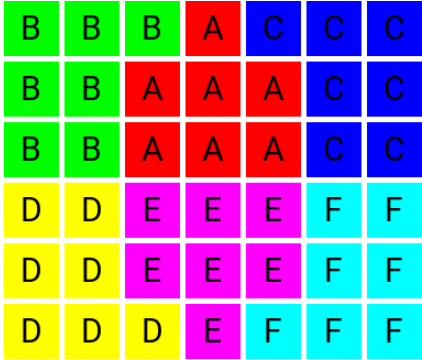
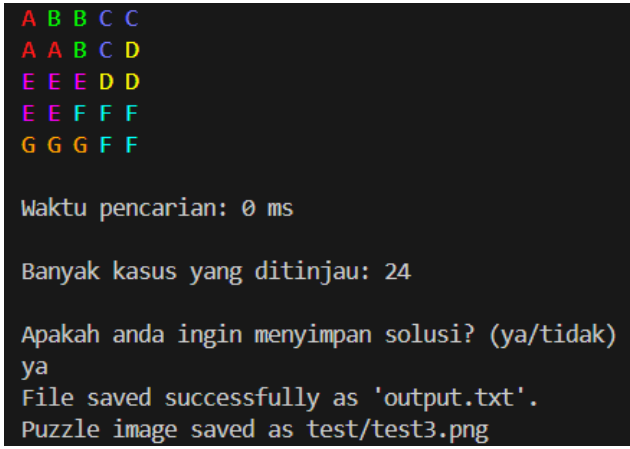

D. Path.java

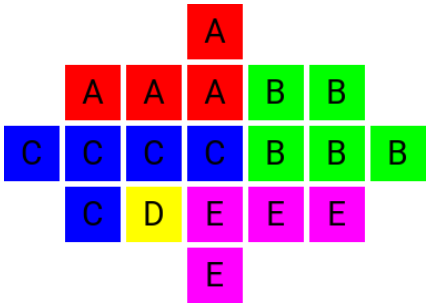
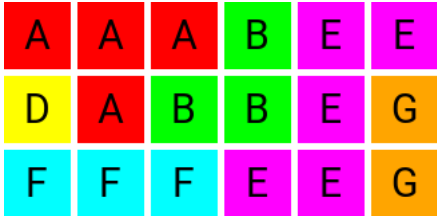
```
1  import java.util.ArrayList;
2  import java.util.HashMap;
3
4  You, 17 hours ago | 1 author (You)
5  public class Path {
6      private HashMap<Character, Integer> path;
7      public ArrayList<Character> placedBlocks;
8      public long exploredCases = 0;
9
10     // Instantiation
11     public Path(ArrayList<Character> ids) {
12         this.path = new HashMap<>();
13         this.placedBlocks = new ArrayList<>();
14         for (char id : ids) {
15             path.put(id, -1);
16         }
17
18     public void recordPath(char id, int variation) {
19         if (path.containsKey(id)) {
20             path.put(id, variation);
21         } else {
22             System.out.println("Error: ID" + id + "not found in path.");
23         }
24     }
25
26     public Integer getPath(char id) {
27         return path.getOrDefault(id, -1);
28     }
29
30     public void printPaths() {
31         for (char id : path.keySet()) {
32             System.out.println("Path for " + id + ": " + path.get(id));
33         }
34     }
35 }
36
```

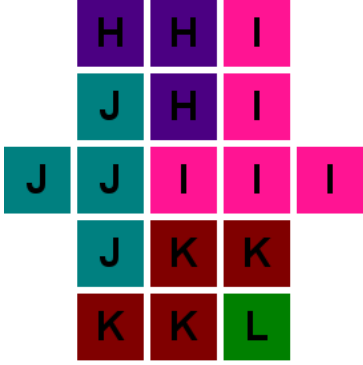
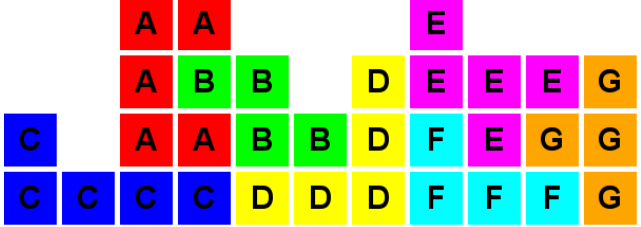

IV. HASIL EKSEKUSI PROGRAM

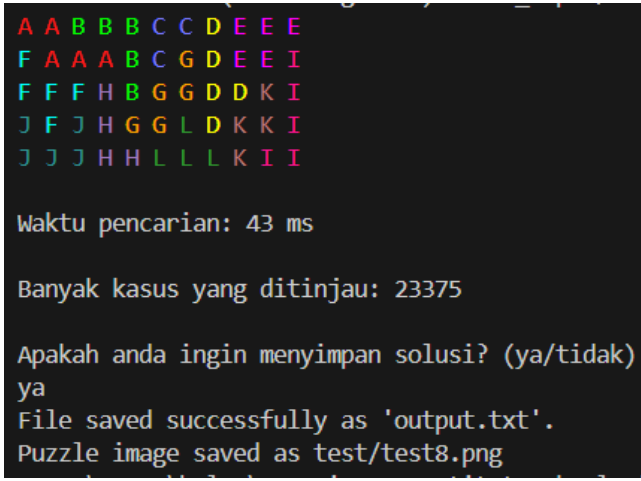
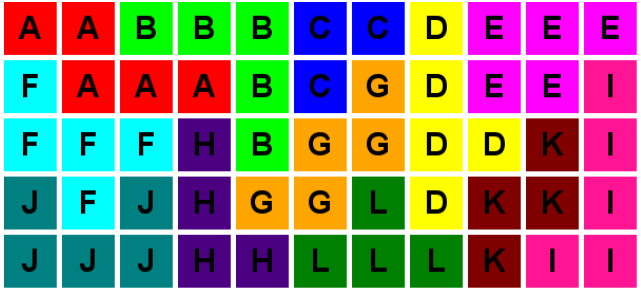
Tangkapan layar yang memperlihatkan *input* dan *output*:

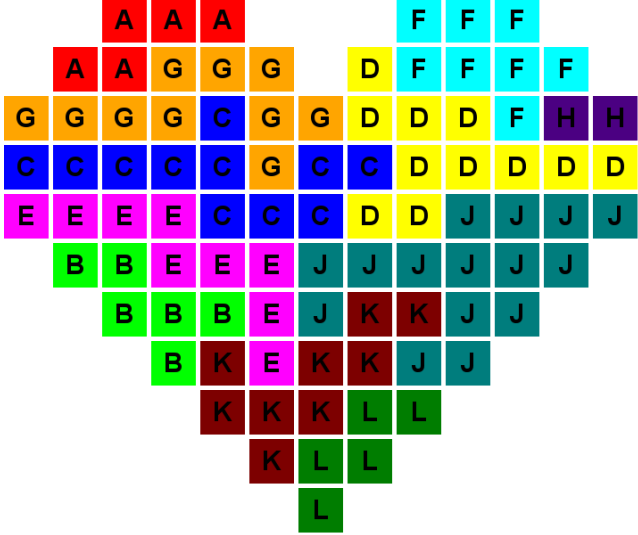
Test File	Input	Ouptut
test1.txt	5 5 7 DEFAULT A AA B BB C CC D DD EE EE E FF FF F GGG	 <pre> A B B C C A A B C D E E E D D E E F F F G G G F F Waktu pencarian: 0 ms Banyak kasus yang ditinjau: 24 Apakah anda ingin menyimpan solusi? (ya/tidak) ya File saved successfully as 'output.txt'. Puzzle image saved as test/test1.png </pre> 
test2.txt	6 7 6 DEFAULT AAA AAA A BBB BB BB CCC CCC C DDD DD DD EEE	 <pre> B B B A C C C B B A A A C C B B A A A C C D D E E E F F D D E E E F F D D E E F F F Waktu pencarian: 39 ms Banyak kasus yang ditinjau: 2249 Apakah anda ingin menyimpan solusi? (ya/tidak) ya File saved successfully as 'output.txt'. Puzzle image saved as test/test2.png </pre>

	EEE E FFF FF FF	
test3.txt	5 5 7 DEFAULT EE EE E FF FF F B BB C CC D DD GGG A AA	 

test4.txt	<pre> 5 7 5 CUSTOM ...X... .XXXXX. XXXXXXX .XXXXX. ...X... A AAA BB BBB CCCC C D EEE E </pre>	<pre> . . . A A A A B B . C C C C B B B . C D E E E E . . . Waktu pencarian: 0 ms Banyak kasus yang ditinjau: 36 Apakah anda ingin menyimpan solusi? (ya/tidak) ya File saved successfully as 'output.txt'. Puzzle image saved as test/test4.png </pre> 
test5.txt	<pre> 3 6 6 DEFAULT A AA A D EE E EE B BB GG FFF </pre>	<pre> A A A B E E D A B B E G F F F E E G Waktu pencarian: 0 ms Banyak kasus yang ditinjau: 266 Apakah anda ingin menyimpan solusi? (ya/tidak) ya File saved successfully as 'output.txt'. </pre> 

test6.txt	<pre> 5 5 5 CUSTOM .XXX. .XXX. XXXXX .XXX. .XXX. HH H III I I JJJ J KK KK L </pre>	<pre> . H H I . . J H I . J J I I I . J K K . . K K L . </pre> <p>Waktu pencarian: 0 ms</p> <p>Banyak kasus yang ditinjau: 10</p> <p>Apakah anda ingin menyimpan solusi? (ya/tidak) ya</p> <p>File saved successfully as 'output.txt'. Puzzle image saved as test/test6.png</p> 
test7.txt	<pre> 5 11 7 CUSTOMXX...X... ..XXX.XXXXX X.XXXXXXXXXX XXXXXXXXXXXX AA A AA BB BB C CCCC D D DDD E EEE </pre>	<pre> A A . . . E A B B . D E E E G C . A A B B D F E G G C C C C D D D F F F G </pre> <p>Waktu pencarian: 30 ms</p> <p>Banyak kasus yang ditinjau: 7806</p> <p>Apakah anda ingin menyimpan solusi? (ya/tidak) ya</p> <p>File saved successfully as 'output.txt'. Puzzle image saved as test/test7.png</p> 

	E FFF F G GG G	
test8.txt	5 11 12 DEFAULT AAA AA BBB B B C CC DDDD D EE EE E F FFF F GG GG G HH H H II I I I J J JJJ KK KK LLL L	 <p>Waktu pencarian: 43 ms</p> <p>Banyak kasus yang ditinjau: 23375</p> <p>Apakah anda ingin menyimpan solusi? (ya/tidak) ya File saved successfully as 'output.txt'. Puzzle image saved as test/test8.png</p> 

<p>test9.txt</p>	<pre> 11 13 11 CUSTOM ..XXX...XXX.. .XXXXX.XXXXX. XXXXXXXXXXXXXX XXXXXXXXXXXXXX XXXXXXXXXXXXXX .XXXXXXXXXX. ..XXXXXXXXXX. ...XXXXXXX...XXXXX....XXX.....X..... KK K KK KKK K AAA AA JJJJ JJJJJ J JJ JJ D DDD DDDD DD FFF FFFF F EEEE EEE E E GGG GGGG GG G C CCCCC CC CCC BB </pre>	<pre> Enter filename (including .txt): test_input/test9.txt ..AAA...FFF.. .AAGGG.DFFFF. GGGGCGGGDDDFHH CCCCCGCCDDDDDD EEEECCCDJJJJJ .BBEEJJJJJJJ. ..BBBEJJKJJJ.. ...BKEKJJJ...KKKLL....KLL.....L..... Waktu pencarian: 44177 ms Banyak kasus yang ditinjau: 237535526 Apakah Anda ingin menyimpan solusi? (ya/tidak) ya Apakah Anda ingin menyimpan solusi dalam bentuk gambar (.png)? (ya/tidak) ya File saved successfully as 'output.txt'. Puzzle image saved as test/test9.png </pre> 
------------------	---	--

	BBB B HH LL LL L	
test10.txt	5 7 6 CUSTOM ...X... ..XXX.. XXXXXXX ..XXX.. .XX.XX. E EE EEE A AA CCC C DD GG F	<div> Enter filename (including .txt): test_input/test10.txt <pre> . . . E E E A . . E E E C A A F . . C C C . . . D D . G G . </pre> Waktu pencarian: 42 ms Banyak kasus yang ditinjau: 5447 Apakah Anda ingin menyimpan solusi? (ya/tidak) ya </div>
test11.txt	5 5 7 DEFAULT A AA BB BBB C CC D DD EE EE E FF	<div> Solusi tidak ditemukan. Waktu pencarian: 233 ms Banyak kasus yang ditinjau: 1208376 (solusi tidak ditemukan) </div>

	FF F GGG	
test12.txt	5 5 6 DEFAULT A AA B BB C CC D DD EE EE E FF FF F GGG	<pre>Exception in thread "main" java.lang.Error: Parameter input P (jumlah blok) tidak sesuai dengan jumlah id blok unik yang terdaftar.</pre> <pre>Jumlah id terdaftar: 7</pre> <pre>P input: 6</pre> <p>(jumlah blok terdaftar tidak sesuai parameter P)</p>
test13.txt	5 0 5 DEFAULT A AA B BB C CC D DD EE EE	<pre>Enter filename (including .txt): test_input/test13.txt</pre> <pre>Exception in thread "main" java.lang.Error: Ukuran papan (N dan M) minimal 1</pre> <p>(terdapat ukuran papan yang kurang dari 1)</p>
test14.txt	5 5 7 DEFOLT A AA B BB	<pre>Enter filename (including .txt): test_input/test14.txt</pre> <pre>Exception in thread "main" java.lang.Error: Board category invalid. Board category seharusnya DEFAULT atau CUSTOM.</pre> <p>(kategori board invalid)</p>

	C CC D DD EE EE E FF FF F GGG	
--	---	--

V. LAMPIRAN

A. Pranala *Repository* Github

https://github.com/KalengBalsem/Tucil1_15223011

B. Tabel Ketercapaian

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	√	