

Laporan Tugas Kecil 1
IF2211 Strategi Algoritma
Penyelesaian IQ Puzzler Pro dengan Algoritma Brute Force



Disusun Oleh
13523095 - Rafif Farras

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2024/2025

DAFTAR ISI

DAFTAR ISI.....	1
BAB 1	
DESKRIPSI MASALAH DAN ALGORITMA.....	2
1.1. IQ Puzzler Pro.....	2
1.2. Algoritma Brute Force.....	3
1.3. Penerapan Algoritma Brute Force pada Penyelesaian IQ Puzzler Pro.....	3
BAB 2	
IMPLEMENTASI PROGRAM.....	5
2.1. File Block.java.....	5
2.2. File Board.java.....	5
2.3. File Coloring.java.....	6
2.4. File Data.java.....	6
2.5. File Point.java.....	6
2.6. File Reader.java.....	6
2.7. File Save.java.....	7
2.8. File Solver.java.....	7
2.9. File Main.java.....	7
BAB 3	
SOURCE CODE PROGRAM.....	8
3.1. Repository Github.....	8
3.2. Source Code.....	8
BAB 4	
UJI COBA PROGRAM.....	15
LAMPIRAN.....	19

BAB 1

DESKRIPSI MASALAH DAN ALGORITMA

1.1. IQ Puzzler Pro



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

Permainan IQ Puzzler Pro adalah permainan papan yang diproduksi oleh perusahaan Smart Games. Tujuan dari permainan ini adalah pemain harus dapat mengisi seluruh board dengan seluruh block puzzle yang telah tersedia.

Pada permainan ini, terdapat 2 komponen penting yaitu::

1. Board (Papan) : Board merupakan komponen utama yang menjadi tujuan permainan dimana pemain harus mampu mengisi seluruh area board menggunakan block-block yang telah disediakan.
2. Block (Blok) : Block adalah komponen yang digunakan pemain untuk mengisi board kosong hingga terisi penuh. Setiap block memiliki bentuk yang unik dan semua block harus digunakan untuk menyelesaikan puzzle.

Permainan dimulai dengan board yang kosong. Pemain dapat meletakkan block puzzle sedemikian sehingga tidak ada block yang tumpang tindih (kecuali dalam kasus 3D). Setiap block puzzle dapat dirotasikan maupun dicerminkan. Board secara umum berbentuk persegi panjang, namun sekarang sudah banyak bentuk board yang lebih rumit dan beragam. Puzzle dinyatakan selesai jika dan hanya jika board terisi penuh dan seluruh block puzzle berhasil diletakkan.

1.2. Algoritma Brute Force

Algoritma Brute Force merupakan pendekatan yang lurus dan sederhana (*straightforward*) dalam memecahkan suatu persoalan. Metode ini menerapkan konsep pemecahan masalah secara langsung dan sangat sederhana, dengan cara yang jelas dan mudah dipahami. Metode ini secara sistematis mengevaluasi setiap kemungkinan solusi untuk mencapai solusi yang diinginkan. Meskipun dianggap sebagai pendekatan yang sederhana dan sering kali tidak efisien, algoritma brute force umumnya dapat memberikan solusi yang akurat untuk permasalahan sederhana sampai rumit. Namun, kelemahannya adalah kinerja yang buruk dimana jumlah solusi yang mungkin sangat besar, karena harus memeriksa setiap kemungkinan secara satu-persatu.

1.3. Penerapan Algoritma Brute Force pada Penyelesaian IQ Puzzler Pro

Pengaplikasian algoritma brute force pada penyelesaian permainan IQ Puzzler Pro dilakukan dengan mencoba setiap kemungkinan penempatan block puzzle pada papan dengan pendekatan rekursif. Untuk mempermudah, disini juga digunakan pendekatan menggunakan koordinat kartesius sebagai papan yang nantinya akan ditempati oleh block.

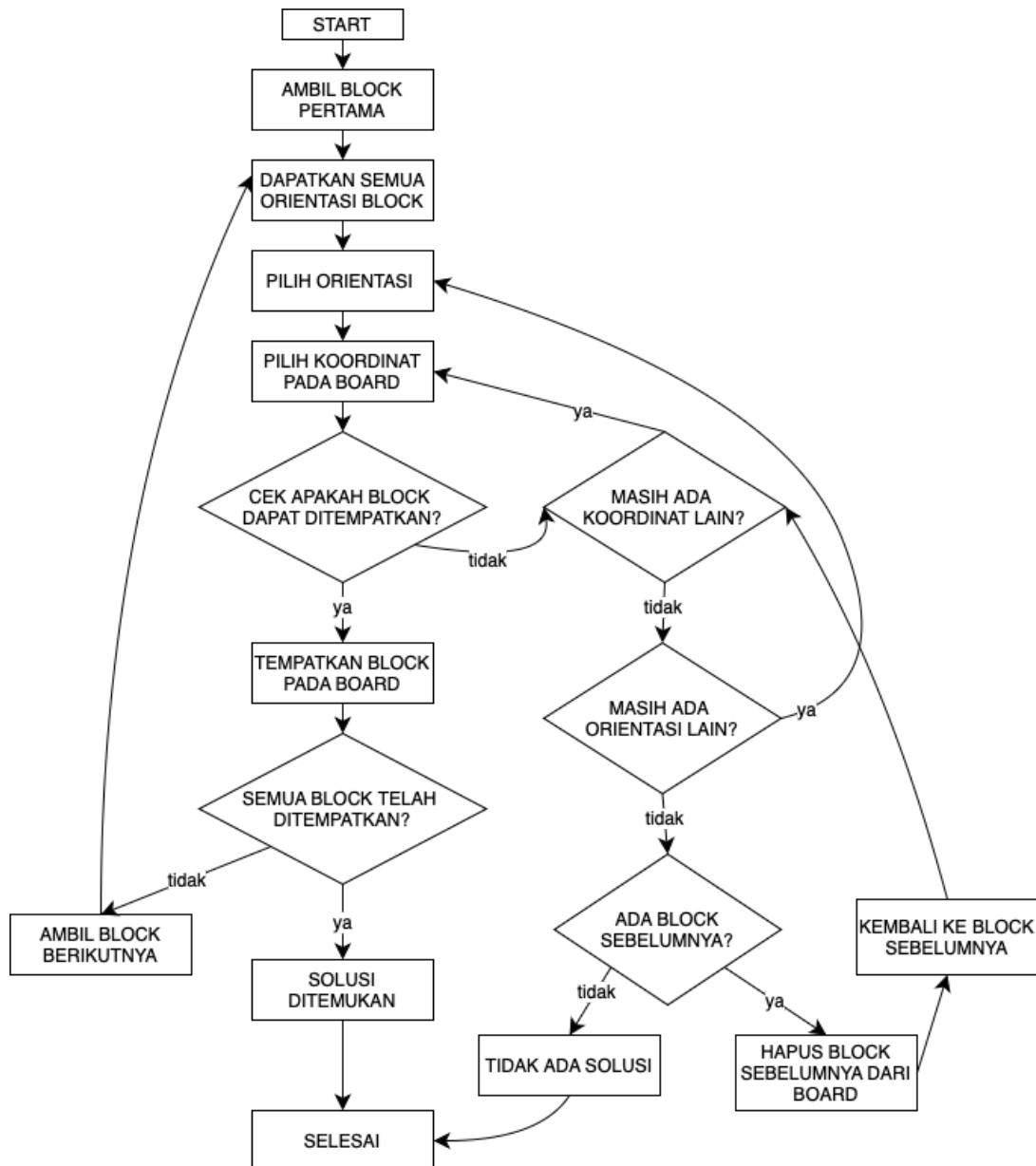
Proses dimulai dengan mengambil block pertama yang telah dibaca dari file input dan mencoba menempatkannya di setiap posisi pada papan permainan (Dilakukan loop untuk setiap titik). Untuk setiap block, algoritma pertama-tama menghasilkan semua orientasi yang mungkin melalui metode `getAllOrientations()`, yang mencakup rotasi (0°, 90°, 180°, 270°) dan flip (normal dan terbalik), menghasilkan hingga 8 orientasi untuk setiap blok. Kemudian, untuk setiap orientasi, algoritma mencoba menempatkan blok pada setiap koordinat papan yang dimulai dari kanan bawah ke kiri atas..

Jadi, akan dilakukan loop bertingkat 3 dimulai dari orientasi block, lebar board, dan tinggi board seperti berikut:

```
● ● ●  
1 for (Block orientation : orientations) {  
2     for (int x = board.width - 1; x >= 0; x--) {  
3         for (int y = 0; y < board.height; y++) {
```

Selanjutnya, pendekatan rekursif digunakan melalui metode `solveRecursive()`, di mana setelah berhasil menempatkan satu blok, algoritma akan memanggil dirinya sendiri untuk mencoba menempatkan blok berikutnya. Jika semua blok berhasil ditempatkan, maka solusi telah ditemukan. Namun, jika penempatan blok berikutnya tidak memungkinkan, algoritma akan membatalkan/menghapus penempatan blok saat ini pada board (*backtracking*) dan melanjutkan proses loop pada block saat ini hingga dapat ditempatkan pada board dengan posisi atau orientasi lain. Jika semua orientasi block saat ini juga sudah tidak bisa ditempatkan, maka mundur lagi ke block sebelumnya dan melakukan *backtracking* dengan cara yang sama.

Proses ini berlanjut hingga semua kemungkinan telah dicoba atau solusi ditemukan. Untuk membantu melacak kompleksitas pencarian, algoritma menyimpan jumlah kasus yang diperiksa dalam variabel casesChecked.



Gambar 2. Flow Chart Algoritma Brute Force pada Penyelesaian IQ Puzzler Pro

BAB 2

IMPLEMENTASI PROGRAM

2.1. File Block.java

Nama Fungsi	Deskripsi
Block(char,List<Point>)	Konstruktor utama yang membuat objek Block
Block(Block other)	Deep copy konstruktor
void rotate()	Memutar block 90 derajat searah jarum jam
void flip()	Mencerminkan block secara horizontal
void normalizeToOrigin()	Normalisasi setiap titik agar berada di kuadran 1
String toString()	Konversi informasi block menjadi string

2.2. File Board.java

Nama Fungsi	Deskripsi
Board(int width, int height)	Konstruktor utama yang membuat objek Board dengan dimensi width x height
Board(Board other)	Deep copy konstruktor
void clear()	Mengembalikan elemen pada board menjadi #
boolean canPlace(Block block, int x, int y)	Memeriksa apakah sebuah block dapat ditempatkan pada board atau tidak
void placeBlock(Block block, int x, int y)	Menempatkan sebuah block pada koordinat tertentu
void removeBlock(Block block, int x, int y)	Menghapus sebuah block dari koordinat tertentu
String toString()	Konversi informasi board menjadi string
String toColoredString()	Konversi board menjadi string berwarna

2.3. File Coloring.java

Nama Fungsi	Deskripsi
String[] ANSI_COLORS	Menyimpan 26 warna untuk CLI
Color[] COLORS	Menyimpan 26 warna untuk output berupa img
Color getColorForChar(char c)	Mendapatkan warna untuk setiap alfabet yang akan digunakan untuk output img
String getAnsiColorForChar(char c)	Mendapatkan warna untuk setiap alfabet yang akan digunakan untuk output CLI
String colorize(String boardString)	Memberi warna pada setiap karakter alfabet

2.4. File Data.java

Nama Fungsi	Deskripsi
Data(int width, int height, int blockCount, CaseType caseType, List<Block> blocks)	Konstruktor utama pembuat objek data yang menyimpan informasi input

2.5. File Point.java

Nama Fungsi	Deskripsi
Point(int x, int y)	Konstruktor utama pembuat objek point
Point add(Point other)	Menambahkan point dengan suatu point lain

2.6. File Reader.java

Nama Fungsi	Deskripsi
Data readFromFile(String filename)	Membaca input file txt dan menyimpan datanya
List<Block> readBlocks(BufferedReader reader, int blockCount)	Membaca blocks yang ada pada file txt

Block parseBlock(char type, List<String> shapeLines)	Konversi dari string menjadi koordinat untuk setiap block
List<Point> normalizeCoordinates(List<Point> points)	Normalisasi koordinat dan mengembalikan dalam bentuk List of Point
Board readCustomBoard(BufferedReader reader, int width, int height)	Membaca custom board pada file txt

2.7. File Save.java

Nama Fungsi	Deskripsi
void saveToFile(String inputFileName, boolean hasSolution, Board solution, long totalCases, long timeTaken)	Menyimpan hasil ke file txt
void saveToImg(String inputFileName, Board board)	Menyimpan hasil dalam format gambar (jpg)

2.8. File Solver.java

Nama Fungsi	Deskripsi
Solver(Data data, Board board)	Konstruktor pembuat solver dengan data dan board yang diberikan
boolean solve()	Fungsi pemanggil solveRecursive() dan menampilkan jumlah kasus
boolean solveRecursive(int blockIndex, Board board)	Fungsi utama yang melakukan algoritma brute force untuk menempatkan block pada board dengan backtracking untuk mencoba semua kemungkinan
List<Block> getAllOrientations(Block block)	Menghasilkan 8 kemungkinan orientasi block setelah rotate dan flip

2.9. File Main.java

File ini akan memanggil fungsi-fungsi pada file lain dan akan menangani interaksi pengguna, mulai dari meminta input file, membaca konfigurasi file, menjalankan algoritma brute force, hingga menampilkan hasil dan opsi penyimpanan (txt atau gambar).

BAB 3

SOURCE CODE PROGRAM

3.1. Repository Github

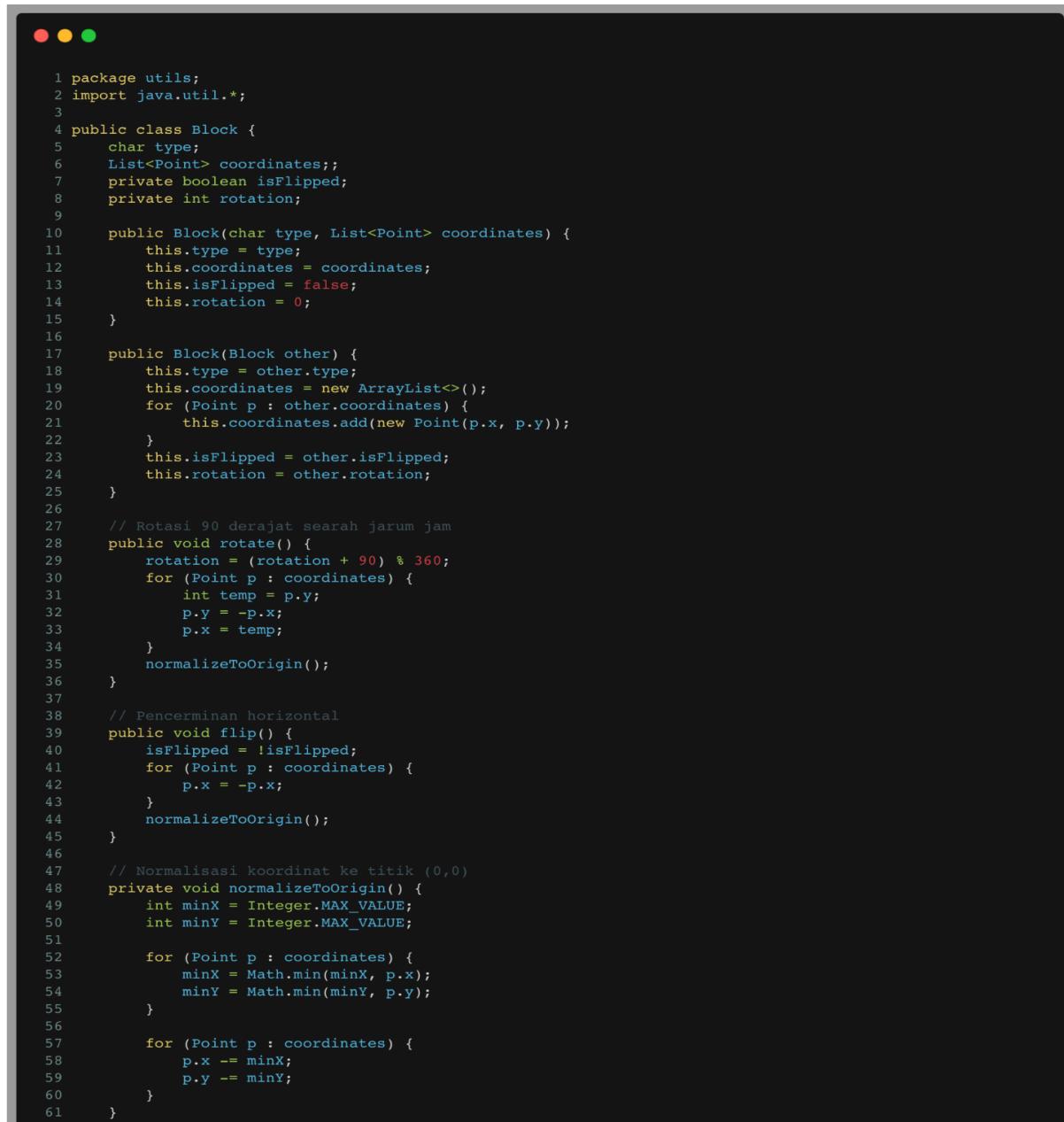
Source Code lengkap dapat diakses melalui link github berikut:

https://github.com/rafifrs/Tucil1_13523095

3.2. Source Code

Pada bagian ini, akan ditampilkan source code program yang berkaitan dengan algoritma brute force yang digunakan untuk menyelesaikan IQ Puzzler Pro (Coloring.java dan Save.java tidak dimasukkan karena tidak berkaitan langsung dengan algoritma brute force)

1. Block.java

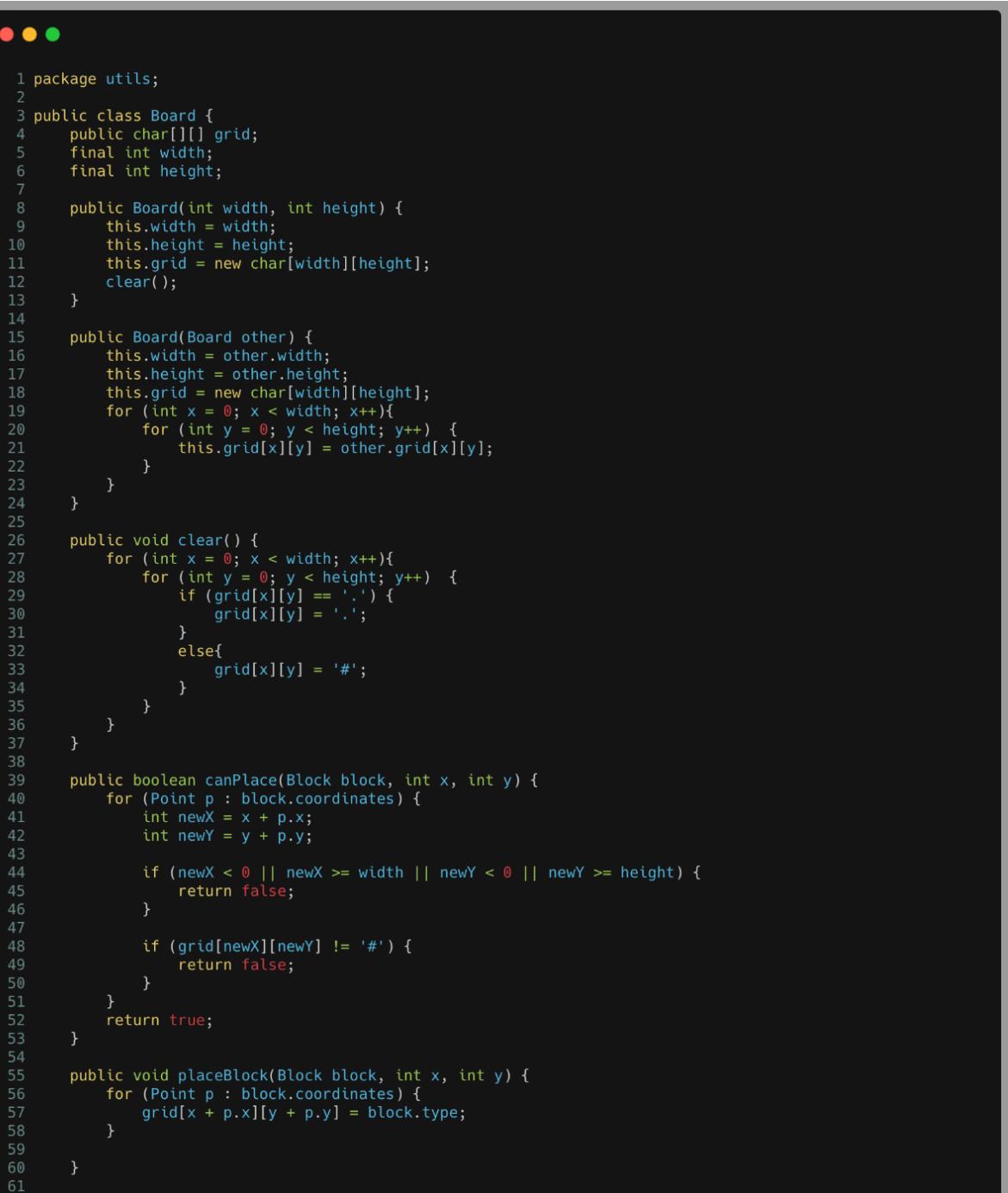


```
1 package utils;
2 import java.util.*;
3
4 public class Block {
5     char type;
6     List<Point> coordinates;
7     private boolean isFlipped;
8     private int rotation;
9
10    public Block(char type, List<Point> coordinates) {
11        this.type = type;
12        this.coordinates = coordinates;
13        this.isFlipped = false;
14        this.rotation = 0;
15    }
16
17    public Block(Block other) {
18        this.type = other.type;
19        this.coordinates = new ArrayList<>();
20        for (Point p : other.coordinates) {
21            this.coordinates.add(new Point(p.x, p.y));
22        }
23        this.isFlipped = other.isFlipped;
24        this.rotation = other.rotation;
25    }
26
27    // Rotasi 90 derajat searah jarum jam
28    public void rotate() {
29        rotation = (rotation + 90) % 360;
30        for (Point p : coordinates) {
31            int temp = p.y;
32            p.y = -p.x;
33            p.x = temp;
34        }
35        normalizeToOrigin();
36    }
37
38    // Pencerminan horizontal
39    public void flip() {
40        isFlipped = !isFlipped;
41        for (Point p : coordinates) {
42            p.x = -p.x;
43        }
44        normalizeToOrigin();
45    }
46
47    // Normalisasi koordinat ke titik (0,0)
48    private void normalizeToOrigin() {
49        int minX = Integer.MAX_VALUE;
50        int minY = Integer.MAX_VALUE;
51
52        for (Point p : coordinates) {
53            minX = Math.min(minX, p.x);
54            minY = Math.min(minY, p.y);
55        }
56
57        for (Point p : coordinates) {
58            p.x -= minX;
59            p.y -= minY;
60        }
61    }
}
```

```

62     public String toString() {
63         StringBuilder sb = new StringBuilder();
64         sb.append("Block ").append(type).append(": \n");
65         sb.append("Rotation: ").append(rotation).append("°\n");
66         sb.append("Flipped: ").append(isFlipped).append("\n");
67         sb.append("Coordinates:\n");
68         for (Point p : coordinates) {
69             sb.append("(").append(p.x).append(", ").append(p.y).append(")\n");
70         }
71     }
72     return sb.toString();
73 }
74 }
```

2. Board.java



```

1 package utils;
2
3 public class Board {
4     public char[][] grid;
5     final int width;
6     final int height;
7
8     public Board(int width, int height) {
9         this.width = width;
10        this.height = height;
11        this.grid = new char[width][height];
12        clear();
13    }
14
15    public Board(Board other) {
16        this.width = other.width;
17        this.height = other.height;
18        this.grid = new char[width][height];
19        for (int x = 0; x < width; x++) {
20            for (int y = 0; y < height; y++) {
21                this.grid[x][y] = other.grid[x][y];
22            }
23        }
24    }
25
26    public void clear() {
27        for (int x = 0; x < width; x++) {
28            for (int y = 0; y < height; y++) {
29                if (grid[x][y] == '.') {
30                    grid[x][y] = '.';
31                } else {
32                    grid[x][y] = '#';
33                }
34            }
35        }
36    }
37
38    public boolean canPlace(Block block, int x, int y) {
39        for (Point p : block.coordinates) {
40            int newX = x + p.x;
41            int newY = y + p.y;
42
43            if (newX < 0 || newX >= width || newY < 0 || newY >= height) {
44                return false;
45            }
46
47            if (grid[newX][newY] != '#') {
48                return false;
49            }
50        }
51        return true;
52    }
53
54    public void placeBlock(Block block, int x, int y) {
55        for (Point p : block.coordinates) {
56            grid[x + p.x][y + p.y] = block.type;
57        }
58    }
59
60 }
```

```

62     public void removeBlock(Block block, int x, int y) {
63
64         for (Point p : block.coordinates) {
65             if (grid[x + p.x][y + p.y] == '.') {
66                 grid[x + p.x][y + p.y] = '.';
67             }
68             else{
69                 grid[x + p.x][y + p.y] = '#';
70             }
71             grid[x + p.x][y + p.y] = '#';
72         }
73     }
74
75     public String toString() {
76         StringBuilder sb = new StringBuilder();
77         for (int y = 0; y < height; y++) {
78             for (int x = 0; x < width; x++) {
79                 sb.append(grid[x][y]);
80             }
81             sb.append('\n');
82         }
83         return sb.toString();
84     }
85
86     public String toColoredString() {
87         return Coloring.colorize(toString());
88     }
89 }
```

3. Data.java

```

1 package utils;
2 import java.util.*;
3
4 enum CaseType {
5     DEFAULT,
6     PYRAMID,
7     CUSTOM
8 }
9
10 class Data {
11     public final int width;
12     public final int height;
13     public final int blockCount;
14     public final CaseType caseType;
15     public final List<Block> blocks;
16
17     public Data(int width, int height, int blockCount, CaseType caseType, List<Block> blocks) {
18         this.width = width;
19         this.height = height;
20         this.blockCount = blockCount;
21         this.caseType = caseType;
22         this.blocks = blocks;
23     }
24 }
```

4. Point.java

```

1 package utils;
2 public class Point {
3     int x, y;
4
5     public Point(int x, int y) {
6         this.x = x;
7         this.y = y;
8     }
9
10    public Point add(Point other) {
11        return new Point(this.x + other.x, this.y + other.y);
12    }
13 }
14
```

5. Reader.java

```
 1 package utils;
 2 import java.io.*;
 3 import java.util.*;
 4
 5 class Reader {
 6
 7     public static Data readFromFile(String filename) throws IOException {
 8         try (BufferedReader reader = new BufferedReader(new FileReader(filename))) {
 9             // Baca dimensi dan tipe case
10             String[] dimensions = reader.readLine().trim().split("\\s+");
11             if (dimensions.length != 3) {
12                 throw new IOException("Invalid dimension format. Required format: width height blockCount");
13             }
14
15             int height = Integer.parseInt(dimensions[0]);
16             int width = Integer.parseInt(dimensions[1]);
17             int blockCount = Integer.parseInt(dimensions[2]);
18
19             String caseType = reader.readLine().trim().toUpperCase();
20             CaseType puzzleCase;
21             try {
22                 puzzleCase = CaseType.valueOf(caseType);
23             } catch (IllegalArgumentException e) {
24                 throw new IOException("Invalid case type. Options: DEFAULT, PYRAMID, CUSTOM");
25             }
26
27             List<Block> blocks;
28
29             if (puzzleCase == CaseType.DEFAULT) {
30                 blocks = readBlocks(reader, blockCount);
31             } else{
32                 for (int i = 0; i < height; i++) {
33                     reader.readLine();
34                 }
35                 blocks = readBlocks(reader, blockCount);
36             }
37
38             if (blocks.size() != blockCount) {
39                 throw new IOException("The number of blocks does not match the blockCount");
40             }
41
42
43             int totalCoordinates = 0;
44             for (Block block : blocks) {
45                 totalCoordinates += block.coordinates.size();
46             }
47
48             if (totalCoordinates != width * height) {
49                 throw new IOException("No solution because the number of block coordinates does not match width * height");
50             }
51
52             return new Data(width, height, blockCount, puzzleCase, blocks);
53         }
54     }
55
56     public static List<Block> readBlocks(BufferedReader reader, int blockCount) throws IOException {
57         List<Block> blocks = new ArrayList<>();
58         String line;
59         List<String> shapeLines = new ArrayList<>();
60         char currentType = '\0';
61
62         while ((line = reader.readLine()) != null) {
63             if (line.isEmpty()) continue;
64
65             char type = line.trim().charAt(0);
66
67             if (type != currentType) {
68                 if (!shapeLines.isEmpty()) {
69                     blocks.add(parseBlock(currentType, shapeLines));
70                 }
71                 shapeLines.clear();
72                 currentType = type;
73             }
74             shapeLines.add(line);
75         }
76     }
}
```

```

77     if (!shapeLines.isEmpty()) {
78         blocks.add(parseBlock(currentType, shapeLines));
79     }
80     return blocks;
81 }
82
83
84
85     public static Block parseBlock(char type, List<String> shapeLines) {
86         List<Point> points = new ArrayList<>();
87
88         int height = shapeLines.size();
89
90         for (int row = 0; row < height; row++) {
91             String line = shapeLines.get(row);
92             for (int col = 0; col < line.length(); col++) {
93                 if (line.charAt(col) != ' ') {
94                     points.add(new Point(col, height - 1 - row));
95                 }
96             }
97         }
98
99         return new Block(type, normalizeCoordinates(points));
100    }
101
102
103    public static List<Point> normalizeCoordinates(List<Point> points) {
104        if (points.isEmpty()) return points;
105
106        int minX = Integer.MAX_VALUE;
107        int minY = Integer.MAX_VALUE;
108
109        for (Point p : points) {
110            minX = Math.min(minX, p.x);
111            minY = Math.min(minY, p.y);
112        }
113
114        List<Point> normalized = new ArrayList<>();
115        for (Point p : points) {
116            normalized.add(new Point(p.x - minX, p.y - minY));
117        }
118
119        return normalized;
120    }
121
122    public static Board readCustomBoard(BufferedReader reader, int width, int height) throws
123    IOException {
124        Board board = new Board(width, height);
125        for (int y = height - 1; y >= 0; y--) {
126            String line = reader.readLine();
127            for (int x = 0; x < width; x++) {
128                board.grid[x][y] = line.charAt(x);
129            }
130        }
131    }
132 }

```

6. Solver.java



```

1 package utils;
2
3 import java.util.*;
4
5 public class Solver {
6     public final Board board;
7     private final List<Block> blocks;
8     public int casesChecked = 0;
9
10    public Solver(Data data, Board board) {
11        this.board = board;
12        this.blocks = data.blocks;
13    }
14
15    public boolean solve() {
16        boolean foundSolution = solveRecursive(0, board);
17        System.out.println();
18        System.out.println("\u033[32mTotal cases checked: " + casesChecked + "\u033[0m");
19        return foundSolution;
20    }

```

```

21
22     private boolean solveRecursive(int blockIndex, Board board) {
23         if (blockIndex >= blocks.size()) {
24             System.out.println();
25             System.out.println("\033[32mSolution found: \033[0m");
26             System.out.println();
27             System.out.println(board.toColoredString());
28             return true;
29         }
30
31         Block currentBlock = blocks.get(blockIndex);
32
33         List<Block> orientations = getAllOrientations(currentBlock);
34
35         for (Block orientation : orientations) {
36             for (int x = board.width - 1; x >= 0; x--) {
37                 for (int y = 0; y < board.height; y++) {
38                     if (board.canPlace(orientation, x, y)) {
39                         board.placeBlock(orientation, x, y);
40                         if (solveRecursive(blockIndex + 1, board)) {
41                             return true;
42                         }
43                         board.removeBlock(orientation, x, y);
44                     }
45                 }
46             }
47         }
48         casesChecked++;
49         return false;
50     }
51
52     private List<Block> getAllOrientations(Block block) {
53         List<Block> orientations = new ArrayList<>();
54
55         Block current = new Block(block);
56
57         for (int flip = 0; flip < 2; flip++) {
58             for (int rot = 0; rot < 4; rot++) {
59                 orientations.add(new Block(current));
60                 current.rotate();
61             }
62             current.flip();
63         }
64     }
65 }
66 }
```

7. Main.java

```

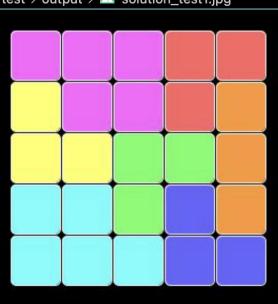
1 package utils;
2 import java.io.*;
3 import java.util.*;
4
5 public class Main {
6
7     public static void main(String[] args) {
8         Data puzzleData;
9         try{
10             Scanner scanner = new Scanner(System.in);
11             System.out.print("\033[33mEnter the file name: \033[0m");
12             String fileName = scanner.nextLine();
13             puzzleData = Reader.readFromFile("test/input/" + fileName);
14
15             Board board;
16
17             if (puzzleData.caseType == CaseType.DEFAULT) {
18                 board = new Board(puzzleData.width, puzzleData.height);
19             } else {
20                 try (BufferedReader reader = new BufferedReader(new FileReader("test/input/" +
21                     fileName))) {
22                     reader.readLine();
23                     reader.readLine();
24                     board = Reader.readCustomBoard(reader, puzzleData.width, puzzleData.height);
25                 }
26             }
27         }
```

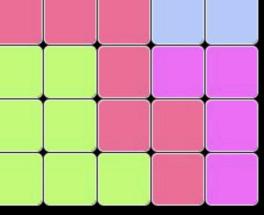
```

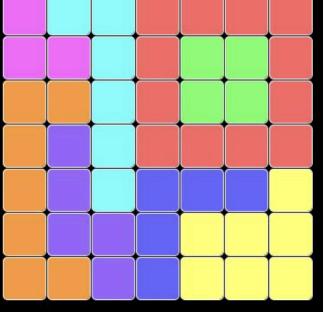
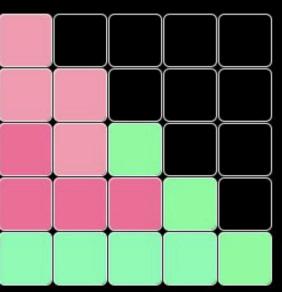
28         Solver solver = new Solver(puzzleData, board);
29
30         long startTime = System.currentTimeMillis();
31         boolean hasSolution = solver.solve();
32         long endTime = System.currentTimeMillis();
33         System.out.println("\033[32mTime taken: " + (endTime - startTime) + " ms\033[0m");
34
35         if (!hasSolution) {
36             System.err.println("\033[31mNo solution found!\033[0m");
37         }
38         System.out.println();
39         System.out.println("\033[34mWhat format do you want to save the solution?\033[0m");
40         System.out.println("1. txt format");
41         System.out.println("2. jpg format");
42         System.out.println("3. Dont save the solution");
43         System.out.print("\033[33mChoose the format: \033[0m");
44
45         String answer = scanner.nextLine();
46
47         // System.out.println(solver.board);
48         if (answer.equalsIgnoreCase("1")) {
49
50             Save.saveToFile(fileName, hasSolution, solver.board, solver.casesChecked, (endTime -
51             startTime));
52             System.out.println("\033[32mSolution saved as text file.\033[0m");
53         } else if (answer.equalsIgnoreCase("2")) {
54             Save.saveToImg(fileName, solver.board);
55             System.out.println("\033[32mSolution saved as image.\033[0m");
56         } else {
57             System.out.println("\033[31mSolution not saved.\033[0m");
58         }
59     } catch (IOException e) {
60         System.err.println("\033[31m" + e.getMessage() + "\033[0m");
61     }
62 }
63
64 }
65 }
66 }
```

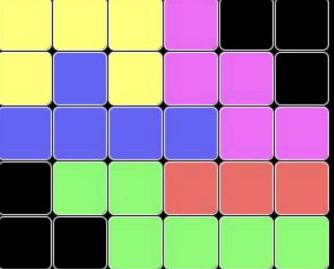
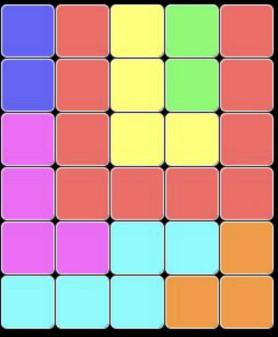
BAB 4

UJI COBA PROGRAM

Test Case #1		
File Input + Output CLI	<pre>test > input > test1.txt 1 5 5 7 2 DEFAULT 3 A 4 AA 5 B 6 BB 7 C 8 CC 9 D 10 DD 11 EE 12 EE 13 E 14 FF 15 FF 16 F 17 GGG</pre> <p>Enter the file name: test1.txt Solution found: EEEA DEEAG DDBBG FFBCG FFFCC Total cases checked: 27016 Time taken: 134 ms What format do you want to save the solution? 1. txt format 2. jpg format 3. Dont save the solution Choose the format: 1 Solution saved as text file.</p>	
Output file .txt dan Output image	<pre>test > output > solution_test1.txt 1 Solution found: 2 EEEAA 3 DEEAG 4 DDBBG 5 FFBCG 6 FFFCC 7 Total cases checked: 27016 8 Time taken: 134 milliseconds</pre> 	
Test Case #2		
File Input + Output CLI	<pre>test > input > test2.txt 1 4 5 4 2 DEFAULT 3 ZZ 4 EE 5 E 6 E 7 QQQ 8 Q 9 QQ 10 Q 11 OO 12 OO 13 000</pre> <p>Enter the file name: test2.txt Solution found: QQQZZ 000EE 000QE 000QE Total cases checked: 29 Time taken: 11 ms What format do you want to save the solution? 1. txt format 2. jpg format 3. Dont save the solution Choose the format: 2 Solution saved as image.</p>	

Output file .txt dan Output image	<pre>test > output > solution_test2.txt</pre> <pre> 1 Solution found: 2 QQZZ 3 OOQE 4 OQQE 5 OOOQE 6 Total cases checked: 29 7 Time taken: 12 milliseconds </pre>	
Test Case #3		
File Input + Output CLI	<pre>test > input > test3.txt</pre> <pre> 1 5 7 5 2 CUSTOM 3 ...#... 4 .#####. 5 ###### 6 .#####. 7 ...#... 8 A 9 AAA 10 BB 11 BBB 12 CCCC 13 C 14 D 15 EEE 16 E </pre>	<pre>Enter the file name: test3.txt</pre> <pre> Solution found: ...E... .EEECD. BBBCCCC .BBAAA.A... </pre> <pre> Total cases checked: 2268 Time taken: 34 ms </pre> <pre> What format do you want to save the solution? 1. txt format 2. jpg format 3. Dont save the solution Choose the format: 2 Solution saved as image. </pre>
Test Case #4		
File Input + Output CLI	<pre>test > input > test4.txt</pre> <pre> 1 7 7 8 2 DEFAULT 3 AAAA 4 A A 5 A A 6 AAAA 7 BB 8 BB 9 C 10 C 11 CCC 12 DDD 13 DDD 14 D 15 EE 16 E 17 FFFFF 18 F 19 GGGGG 20 G G 21 HH 22 HHH </pre>	<pre>Enter the file name: test4.txt</pre> <pre> Solution found: EFFAAAAA EEFABBA GGFABBA GHFAAAA GHFCCCD GHHCDDD GGHCDDD </pre> <pre> Total cases checked: 217535 Time taken: 726 ms </pre> <pre> What format do you want to save the solution? 1. txt format 2. jpg format 3. Dont save the solution Choose the format: 2 Solution saved as image. </pre>

Output file .txt dan Output image	<pre>test > output > solution_test4.txt</pre> <pre> 1 Solution found: 2 EFFAAAA 3 EEFABBA 4 GGFABBA 5 GHFAAAA 6 GHFCCCD 7 GHHCDDD 8 GHHCDDD 9 Total cases checked: 217535 10 Time taken: 734 milliseconds </pre>	
Test Case #5		
File Input + Output CLI	<pre>test > input > test5.txt</pre> <pre> 1 5 5 4 2 CUSTOM 3 ##### 4 ####. 5 ##.. 6 #... 7 #.... 8 PPPP 9 QQQ 10 Q 11 R 12 RR 13 R 14 S 15 S 16 S </pre>	<pre>Enter the file name: test5.txt</pre> <pre>Solution found:</pre> <pre>R....</pre> <pre>RR...:</pre> <pre>QRS..</pre> <pre>QQS.</pre> <pre>PPPS</pre> <pre>Total cases checked: 36</pre> <pre>Time taken: 6 ms</pre> <pre>What format do you want to save the solution?</pre> <ol style="list-style-type: none"> 1. txt format 2. jpg format 3. Dont save the solution <pre>Choose the format: 1</pre> <pre>Solution saved as text file.</pre>
Output file .txt dan Output image	<pre>test > output > solution_test5.txt</pre> <pre> 1 Solution found: 2 R.... 3 RR... 4 QRS.. 5 QQS. 6 PPPS 7 Total cases checked: 36 8 Time taken: 6 milliseconds </pre>	
Test Case #6		
File Input + Output CLI	<pre>test > input > test6.txt</pre> <pre> 1 5 6 5 2 CUSTOM 3 ..##### 4 #####. 5 ###### 6 #####. 7 #####.. 8 AAA 9 BB 10 BBBB 11 CCCC 12 C 13 DDD 14 D D 15 E 16 EE 17 EE </pre>	<pre>Enter the file name: test6.txt</pre> <pre>Solution found:</pre> <pre>DDDE..</pre> <pre>DCDDEE..</pre> <pre>CCCCEE</pre> <pre>.BBAA</pre> <pre>..BBBB</pre> <pre>Total cases checked: 170</pre> <pre>Time taken: 17 ms</pre> <pre>What format do you want to save the solution?</pre> <ol style="list-style-type: none"> 1. txt format 2. jpg format 3. Dont save the solution <pre>Choose the format: 1</pre> <pre>Solution saved as text file.</pre>

Output file .txt dan Output image	<pre>test > output > solution_test6.txt</pre> <pre> 1 Solution found: 2 DDDE.. 3 DCDEE. 4 CCCCEE 5 .BBAAA 6 ..BBBBB 7 Total cases checked: 170 8 Time taken: 17 milliseconds 9 </pre>	
Test Case #7		
File Input + Output CLI	<pre>test > input > test7.txt</pre> <pre> 1 6 5 7 2 DEFAULT 3 AAAA 4 A A 5 A A 6 A A 7 BB 8 CC 9 DDD 10 D 11 EEE 12 E 13 FFF 14 FF 15 GG 16 G </pre>	<pre>Enter the file name: test7.txt</pre> <pre>Solution found:</pre> <pre>CADBA CADBA EADDA EAAAA EEFFG FFFGG</pre> <pre>Total cases checked: 28447 Time taken: 166 ms</pre> <pre>What format do you want to save the solution? 1. txt format 2. jpg format 3. Dont Save the solution Choose the format: 1 Solution saved as text file.</pre>
Output file .txt dan Output image	<pre>test > output > solution_test7.txt</pre> <pre> 1 Solution found: 2 CADBA 3 CADBA 4 EADDA 5 EAAAA 6 EEFFG 7 FFFGG 8 Total cases checked: 28447 9 Time taken: 166 milliseconds </pre>	

LAMPIRAN

Tautan Repository: https://github.com/rafifrs/Tucil1_13523095

NO	Poin	Ya	Tidak
1	Program berhasil dikompilasi tanpa kesalahan	<input checked="" type="checkbox"/>	
2	Program berhasil dijalankan	<input checked="" type="checkbox"/>	
3	Solusi yang diberikan program benar dan mematuhi aturan permainan	<input checked="" type="checkbox"/>	
4	Program dapat membaca masukan berkas .txt serta menyimpan solusi dalam berkas .txt	<input checked="" type="checkbox"/>	
5	Program memiliki <i>Graphical User Interface</i> (GUI)		<input checked="" type="checkbox"/>
6	Program dapat menyimpan solusi dalam bentuk file gambar	<input checked="" type="checkbox"/>	
7	Program dapat menyelesaikan kasus konfigurasi <i>custom</i>	<input checked="" type="checkbox"/>	
8	Program dapat menyelesaikan kasus konfigurasi Piramida (3D)		<input checked="" type="checkbox"/>
9	Program dibuat oleh saya sendiri	<input checked="" type="checkbox"/>	