

IF 2211 Strategi Algoritma

Penyelesaian IQ Puzzler Pro dengan Algoritma Brute Force

Laporan Tugas Kecil

Disusun untuk memenuhi tugas besar mata kuliah IF2211 Strategi Algoritma pada
Semester II Tahun Akademik 2024/2025



Oleh
Stefan Matthew Susanto 13523020

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

DAFTAR ISI

DAFTAR ISI	2
Bab 1 Deskripsi Masalah	3
1.1. Algoritma Brute Force	3
1.2. Permainan IQ Puzzler Pro	3
Bab 2 Penyelesaian	5
2.1. Langkah penyelesaian IQ Puzzler Pro dengan algoritma Brute Force	5
Bab 3 Implementasi Program	7
3.1. Spesifikasi Teknis Program	7
3.1.1. Struktur data	7
3.1.2. Fungsi dan Prosedur	8
3.1.3. Source Code Program	9
Bab 4 Analisis dan Pengujian	16
4.1. Kasus Uji	16
4.1.1. Kasus Uji berdasarkan contoh pada spesifikasi Tugas Kecil 1	16
4.2. Kasus Uji Mandiri	17
4.2.1. Kasus Uji 1	17
4.2.2. Kasus Uji 2	18
4.2.3. Kasus Uji 3	19
4.2.4. Kasus Uji 4	20
4.2.5. Kasus Uji 5	21
4.2.6. Kasus Uji 6	22
4.2.7. Kasus Uji 7	23
Bab 5 Kesimpulan	24
Bab 6 Lampiran	25
6.1. Github	25
6.2. Tabel Pemeriksaan	25

Bab 1 Deskripsi Masalah

1.1. Algoritma Brute Force

Algoritma Brute Force merupakan sebuah metode algoritma dengan pendekatan lurus atau lempang (straightforward) untuk memecahkan suatu persoalan. Algoritma Brute Force biasanya digunakan pada definisi dan konsep yang digunakan serta pernyataan di dalam persoalan(problem statement). Algoritma Brute Force memecahkan persoalan dengan sangat sederhana, langsung, langkah penyelesaian yang jelas, dan dengan prinsip “Just do it! atau Just Solve it!.

Umumnya, algoritma Brute Force bukan merupakan algoritma yang cerdas dan sangkil karena algoritma ini membutuhkan cost komputasi yang besar dan waktu yang lama dalam penyelesaiannya. Kata “force” pada kata Brute Force memiliki arti implisit tenaga lebih dibutuhkan dibandingkan otak. Terkadang, algoritma Brute force disebut juga algoritma naif (naive algorithm).

Algoritma Brute Force lebih sesuai digunakan pada persoalan yang memiliki ukuran masukannya. Pertimbangan untuk menggunakan algoritma Brute Force adalah algoritma ini cukup sederhana dan bentuk implementasinya cukup mudah. Algoritma Brute Force sering digunakan sebagai basis pembanding dengan algoritma lain yang relatif lebih mangkus. Sangat jarang terdapat suatu persoalan yang tidak dapat diselesaikan dengan algoritma ini. Bahkan, terdapat persoalan yang hanya mampu diselesaikan dengan algoritma Brute Force.

Algoritma Brute Force memiliki cakupan luas (wide applicability) dan dapat diterapkan untuk menyelesaikan berbagai jenis permasalahan. Selain itu, algoritma ini mampu menghasilkan solusi yang efektif untuk beberapa masalah penting seperti pencarian, pengurutan, pencocokan string, dan perkalian matriks. Brute force juga digunakan sebagai pendekatan standar dalam berbagai tugas komputasi dasar, seperti operasi penjumlahan atau perkalian sejumlah bilangan serta pencarian elemen minimum atau maksimum dalam sebuah array.

Algoritma Brute Force memiliki beberapa kelemahan, salah satunya algoritma Brute Force jarang menghasilkan solusi yang efisien. Metode ini umumnya lambat ketika digunakan pada masukan yang berukuran besar, sehingga tidak selalu praktis untuk diterapkan. Selain itu, pendekatan Brute Force cenderung kurang konstruktif dan kreatif dibandingkan strategi pemecahan masalah lainnya.

1.2. Permainan IQ Puzzler Pro

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

Pada permainan IQ Puzzler Pro terdapat beberapa komponen penting, yakni Board dan Piece/Blok. Board adalah komponen utama dalam tujuan permainan dimana pemain harus mampu meingisi seluruh area papan menggunakan blok-blok yang telah disediakan. Sedangkan Piece/Blok merupakan komponen yang digunakan pemain untuk mengisi board kosong hingga board terisi penuh. Piece yang terdapat pada permainaan ini memiliki bentuk yang unik dan semua piece wajib digunakan untuk menyelesaikan puzzle.

Permainan dimulai dengan papan kosong, di mana pemain harus menempatkan blok puzzle tanpa menyebabkan tumpang tindih, kecuali untuk kasus 3D. Setiap blok dapat diputar atau dicerminkan sebelum diletakkan. Puzzle dianggap selesai ketika seluruh papan telah terisi sepenuhnya dengan semua blok yang tersedia.



(a). Default

(b).Custom

(c).Pyramid

Permainan ini memiliki beberapa mode permainan, yakni default, Custom, dan Piramid. Pada gambar (a) menunjukkan hasil akhir permainan IQ Puzzler Pro dengan menggunakan mode default. Pada gambar (b) menunjukkan hasil akhir permainan IQ Puzzler Pro dengan menggunakan mode Custom. Dan yang terakhir, gambar (c) menunjukkan IQ Puzzler Pro dengan mode pyramid.

Bab 2 Penyelesaian

2.1. Langkah penyelesaian IQ Puzzler Pro dengan algoritma Brute Force

Berikut adalah langkah-langkah penyelesaian IQ Puzzler Pro dengan pendekatan algoritma Brute Force:

1. Pengguna meminta user untuk memasukkan nama file (.txt) yang akan dibaca. File tersebut berisi dimensi board (NxM), jumlah piece yang tersedia, jenis kasus yang digunakan, yakni Default, Custom, atau Piramida, serta bentuk dari tiap piece yang terdiri dari karakter A-Z.
2. Program akan membaca file txt yang diberikan. Jika program dapat membaca akan melanjutkan, namun jika tidak dapat dibaca maka akan mengembalikan false. Program akan membaca baris pertama lalu akan didapatkan info baris, kolom, dan banyaknya piece. Program membaca jenis kasus permainan, antara default, custom, atau piramida.
3. Program membuat board kosong berukuran baris dan kolom yang sesuai. Program akan mengecek apakah jumlah piece sesuai dengan jumlah dari tiap piece tersebut.
4. Program akan membaca tiap line sampai line kosong. Setiap baris dicek karakter pertamanya untuk mengelompokkan data. Jika masih dalam kelompok yang sama, barisnya disimpan. Jika karakter berubah, kelompok sebelumnya dikonversi jadi matriks dan disimpan. Proses akan diulang sampai selesai. Terakhir, jumlah blok dicek, dan jika terjadi error saat membaca file, program menampilkan pesan dan mengembalikan false.
5. Program akan mengambil waktu mulai real-time sebelum algoritma Brute Force dimulai.
6. Program akan menjalankan fungsi solve, pertama program akan mengecek apakah semua piece sudah ditempatkan, jika sudah maka akan return true.
7. Potongan yang akan ditempatkan diambil berdasarkan pieceIndex, lalu dibuat berbagai variasi hasil rotasi dan pencerminan menggunakan createVariation (piece). Program akan melakukan rotasi 90 derajat lalu mirror sebanyak 4 kali, dan jika belum terdapat pada uniquePiece, maka akan menambah variasi baru.
8. Setiap variasi kemudian dicoba untuk ditempatkan pada papan dengan mengecek semua kemungkinan posisi (row, col).
9. Jika potongan dapat ditempatkan, program menentukan simbol piece (pieceSymbol) berdasarkan karakter pertama yang bukan spasi. Potongan tersebut kemudian ditempatkan di papan, jumlah iterasi ditingkatkan, dan keadaan papan dicetak. Setelah itu, program melanjutkan ke potongan berikutnya dengan solve(pieceIndex + 1). Jika berhasil, metode mengembalikan true.
10. Jika solusi gagal ditemukan, potongan yang terakhir ditempatkan akan dihapus dengan removePiece(variant, row, col), lalu mencoba posisi lain. Jika semua kemungkinan telah dicoba dan tidak ada solusi yang cocok, metode mengembalikan false, menandakan bahwa susunan saat ini tidak berhasil.

11. Program akan mengecek waktu akhir real-time setelah algoritma solve dijalankan.
12. Program menampilkan berapa lama waktu program ini berjalan dan jumlah iterasi yang dilakukan.
13. Program dapat menyimpan hasil solusi (.txt) dan solusi dalam bentuk gambar(.png)

Bab 3 Implementasi Program

3.1. Spesifikasi Teknis Program

3.1.1. Struktur data

```
└── Tucil1_13523020
    └── src
        └── IQPuzzlePro.java

    └── doc
        └── Tucil1_K1_13523020_Stefan Matthew Susanto.pdf

    └── bin
        └── IQPuzzlePro.class

    └── test
        ├── test1.txt
        ├── test2.txt
        ├── test3.txt
        ├── test4.txt
        ├── test5.txt
        ├── test6.txt
        ├── test7.txt
        ├── test8.txt
        ├── Solusi gambar test1.txt.png
        ├── Solusi gambar test2.txt.png
        ├── Solusi gambar test3.txt.png
        ├── Solusi gambar test4.txt.png
        ├── Solusi gambar test5.txt.png
        ├── Solusi gambar test6.txt.png
        ├── Solusi gambar test7.txt.png
        ├── Solusi gambar test8.txt.png
        ├── Solusi test1.txt
        ├── Solusi test2.txt
        ├── Solusi test3.txt
        ├── Solusi test4.txt
        ├── Solusi test5.txt
        ├── Solusi test6.txt
        └── Solusi test7.txt
```

└── Solusi test8.txt

└── README.md

3.1.2. Fungsi dan Prosedur

Fungsi/ Prosedur

```
public static boolean cekReadFile(String fileName) {  
  
    public static char[][] convertToMatriks(List<String> shapeLine, char letter){  
  
    public static char[][] copyMatriks ( char[][] matriks){  
  
    public static void printMatriks(char[][] matriks){  
  
    public static List<char[][]> createVariation(char[][] piece){  
        public static boolean solve (int pieceIndex){  
            if (pieceIndex == 0) {  
                return true;  
            }  
            for (int r = 0; r < 4; r++) {  
                for (int c = 0; c < 4; c++) {  
                    if (canPlace(piece, r, c)) {  
                        placePiece(piece, r, c, 'X');  
                        if (solve(pieceIndex + 1)) {  
                            return true;  
                        }  
                        removePiece(piece, r, c);  
                    }  
                }  
            }  
            return false;  
        }  
        public static char[][] rotate (char[][] piece){  
            char[][] rotated = new char[4][4];  
            for (int r = 0; r < 4; r++) {  
                for (int c = 0; c < 4; c++) {  
                    rotated[c][3 - r] = piece[r][c];  
                }  
            }  
            return rotated;  
        }  
        public static char[][] mirror (char[][] piece){  
            char[][] mirrored = new char[4][4];  
            for (int r = 0; r < 4; r++) {  
                for (int c = 0; c < 4; c++) {  
                    mirrored[c][3 - r] = piece[r][c];  
                }  
            }  
            return mirrored;  
        }  
        public static boolean canPlace(char[][] piece, int r, int c){  
            for (int i = 0; i < 4; i++) {  
                for (int j = 0; j < 4; j++) {  
                    if (piece[i][j] == 'X' && (i == r || j == c)) {  
                        return false;  
                    }  
                }  
            }  
            return true;  
        }  
        public static void placePiece(char[][] piece, int r, int c, char var ){  
            for (int i = 0; i < 4; i++) {  
                for (int j = 0; j < 4; j++) {  
                    if (i == r && j == c) {  
                        piece[i][j] = var;  
                    }  
                }  
            }  
        }  
        public static void removePiece (char[][] piece, int r , int c){  
            for (int i = 0; i < 4; i++) {  
                for (int j = 0; j < 4; j++) {  
                    if (i == r && j == c) {  
                        piece[i][j] = ' ';  
                    }  
                }  
            }  
        }  
        public static void saveFile(String fileOutput){  
            String output = "";  
            for (int r = 0; r < 4; r++) {  
                for (int c = 0; c < 4; c++) {  
                    output += piece[r][c];  
                }  
                output += "\n";  
            }  
            File file = new File(fileOutput);  
            try {  
                PrintWriter writer = new PrintWriter(file);  
                writer.print(output);  
                writer.close();  
            } catch (FileNotFoundException e) {  
                e.printStackTrace();  
            }  
        }  
        public static void saveImage(String fileName){  
            BufferedImage image = new BufferedImage(400, 400, BufferedImage.TYPE_INT_RGB);  
            for (int r = 0; r < 4; r++) {  
                for (int c = 0; c < 4; c++) {  
                    if (piece[r][c] == 'X') {  
                        image.setRGB(c * 100, r * 100, Color.BLACK.getRGB());  
                    }  
                }  
            }  
            File file = new File(fileName);  
            try {  
                ImageIO.write(image, "png", file);  
            } catch (IOException e) {  
                e.printStackTrace();  
            }  
        }  
    }  
}
```

3.1.3. Source Code Program

1. File IQ Puzzler Pro

```
import java.awt.Graphics2D;
import java.awt.image.BufferedImage;
import java.io.*;
import java.util.*;

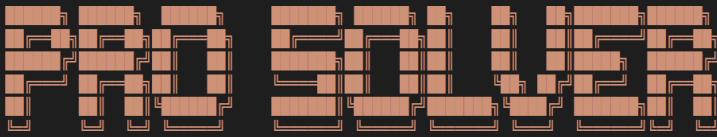
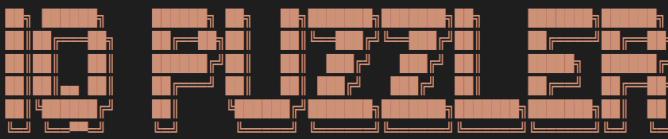
import javax.imageio.ImageIO;
import java.awt.Color;

public class IQPuzzlePro {
    private static int rows, cols, count_piece;
    private static String type_board;
    private static char[][] board;
    private static List<char[][]> pieces = new ArrayList<>();
    private static long iterationCount;

    private static final String[] color = {
        "\u001B[31m", // Red
        "\u001B[32m", // Green
        "\u001B[33m", // Yellow
        "\u001B[34m", // Blue
        "\u001B[35m", // Magenta
        "\u001B[36m", // Cyan
        "\u001B[38;5;208m", // Orange
        "\u001B[38;5;214m", // Light Orange
        "\u001B[38;5;165m", // Purple
        "\u001B[38;5;200m", // Pink
        "\u001B[38;5;118m", // Bright Lime Green
        "\u001B[38;5;75m", // Sky Blue
        "\u001B[38;5;220m", // Gold
        "\u001B[38;5;130m", // Brown
        "\u001B[38;5;255m", // Light Gray
        "\u001B[38;5;21m", // Deep Blue
        "\u001B[91m", // Bright Red
        "\u001B[92m", // Bright Green
        "\u001B[93m", // Bright Yellow
        "\u001B[94m", // Bright Blue
        "\u001B[95m", // Bright Magenta
        "\u001B[96m", // Bright Cyan
        "\u001B[90m", // Dark Gray
        "\u001B[97m", // White
        "\u001B[38;5;196m", // Dark Red
        "\u001B[38;5;46m", // Dark Green
    };

    private static final String reset_color = "\u001B[0m";
```

```
private static String title= """
```



```
""";
```

```
Run | Debug
public static void main(String[] args) {
    System.out.println(title);
    Scanner scanner = new Scanner(System.in);
    System.out.print(s:"Masukkan path file(.txt): ");
    String fileName = scanner.nextLine();
```

```
if (!cekReadFile(fileName)) {
    System.out.println(x:"File tidak valid!");
    scanner.close();
    return;
}

long start = System.currentTimeMillis();

boolean solved = solve(pieceIndex:0);

long end = System.currentTimeMillis();

System.out.println(title);
if (solved) {
    printBoard();
} else {
    System.out.println(x:"Tidak ditemukan solusi.\n");
}
System.out.println("Waktu pencarian: " + (end - start) + " ms\n");
System.out.println("Banyak kasus yang ditinjau: " + iterationCount +"\n");

System.out.print(s:"Apakah anda ingin menyimpan solusi? (ya/tidak): ");
if (scanner.nextLine().equalsIgnoreCase(anotherString:"ya")) {
    saveFile("Solusi "+fileName );
}
```

```
System.out.print(s:"Apakah anda ingin menyimpan solusi gambar? (ya/tidak): ");
if (scanner.nextLine().equalsIgnoreCase(anotherString:"ya")) {
    saveImage("Solusi gambar "+fileName );
}
scanner.close();
```

```

public static boolean cekReadFile(String fileName) {
    try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {
        String[] info = reader.readLine().split(regex:" ");
        if (info.length < 3) return false;

        rows = Integer.parseInt(info[0]);
        cols = Integer.parseInt(info[1]);
        count_piece = Integer.parseInt(info[2]);

        type_board = reader.readLine().trim();
        if (!type_board.equals(anObject:"DEFAULT") && !type_board.equals(anObject:"PIRAMID") && !type_board.equals(anObject:"CUSTOM"))
            return false;
    }

    board =new char[rows][cols];
    for (char[] row : board){
        Arrays.fill(row,val:'.');
    }
}

int tes_count_piece = 0;
String temp = "";           //huruf sementara yang disimpan
String line;
List<String> shapeLine = new ArrayList<>();

while ((line = reader.readLine()) != null) {
    line = line.stripTrailing();      // delete spasi after piece
    if( line.isEmpty()){
        continue;                  //skip baris kosong
    }

    char firstChar = line.trim().charAt(index:0);      //mengambil char pertama (bukan spasi)

    if ( temp.isEmpty() || firstChar== temp.charAt(index:0) ){ // jika karakter masih sama atau pertama
        shapeLine.add(line);
    }else{
        pieces.add(convertToMatriks(shapeLine, temp.charAt(index:0))); //karakter baru,karakter lama diubah ke matriks
        tes_count_piece++;

        shapeLine.clear(); // reset shapeline
        shapeLine.add(line);
    }

    temp = String.valueOf(firstChar);
}

if (!shapeLine.isEmpty()) { // last piece
    pieces.add(convertToMatriks(shapeLine, temp.charAt(index:0)));
    tes_count_piece++;
}

return tes_count_piece == count_piece;
} catch (IOException e) {
    System.out.println(x:"Error membaca file!");
    return false;
}
}

```

```

public static char[][] convertToMatriks(List<String> shapeLine, char letter){
    int tinggi, lebar ;
    tinggi = shapeLine.size();
    lebar = shapeLine.stream().mapToInt(String::length).max().orElse(0); // find max long

    char[][] shape = new char[tinggi][lebar];

    for (int i = 0; i < tinggi; i++) {
        Arrays.fill(shape[i], val:' '); // spasi
        String line = shapeLine.get(i); // ambil baris

        for (int j = 0; j < line.length(); j++) {
            if (line.charAt(j) == letter) {
                shape[i][j] = letter;
            }
        }
    }
    return shape;
}

```

```

public static char[][] copyMatriks (char[][] matriks){
    int r = matriks.length;
    int c = matriks[0].length;
    char[][] copy = new char[r][c];

    for (int i = 0; i < r; i++) {
        for (int j = 0; j < c; j++) {
            copy[i][j] = matriks[i][j];
        }
    }
    return copy;
}

public static void printMatriks(char[][] matriks){
    for (char[] row : matriks){
        for (char isi :row){
            System.out.print(isi == ' ' ? '.' : isi); // jika spasi maka jadi titik
        }
        System.out.println();
    }
}

```

```

public static List<char[][]> createVariation(char[][] piece){
    Set<String> uniquePiece = new HashSet<>();
    List<char[][]> variations = new ArrayList<>();

    for (int i = 0; i < 4; i++) {
        piece = rotate(piece); //rotasi dicek 4 kali

        if (uniquePiece.add(Arrays.deepToString(piece))) {
            variations.add(copyMatriks(piece)); // tambahkan variasi dari rotate
        }

        char[][] mirroredPiece = mirror(piece); // mirror hasil rotasi
        if (uniquePiece.add(Arrays.deepToString(mirroredPiece))) {
            variations.add(copyMatriks(mirroredPiece)); // tambah hasil pencerminan
        }
    }
    return variations;
}

```

```

public static boolean solve (int pieceIndex){
    if (pieceIndex >= pieces.size()){
        return true;
    }

    char[][] piece = pieces.get(pieceIndex);

    List<char[][]> variations = createVariation(piece);

    for (char[][] variant : variations) {
        for (int row = 0; row <= rows - variant.length; row++) {
            for (int col = 0; col <= cols - variant[0].length; col++) {
                if (canPlace(variant, row, col)) {

                    // Menentukan simbol potongan (pieceSymbol), menghindari spasi
                    char pieceSymbol = piece[0][0];
                    int index = 1;
                    while (pieceSymbol == ' ' && index < piece[0].length) {
                        pieceSymbol = piece[0][index++];
                    }

                    System.out.println("Place piece" + pieceSymbol + " : " + row + " , " + col );
                    placePiece(variant, row, col, pieceSymbol);
                    iterationCount++;

                    printBoard();
                    System.out.println("Count: " + iterationCount);

                    // jika solusi ditemukan
                    if (solve(pieceIndex + 1)) return true;

                    // jika tidak berhasil
                    System.out.println("Remove piece " + pieceSymbol + " : " + row + " , " + col );
                    removePiece(variant, row, col);
                }
            }
        }
    }
    return false;
}

public static char[][] rotate (char[][] piece){ //rotate 90 degree
    int h = piece.length;
    int w = piece[0].length;
    char[][] rotated = new char[w][h];
    for (int i=0;i<h; i++){
        for (int j=0 ; j<w ; j++){
            rotated[j][h-1-i]= piece[i][j];
        }
    }
    return rotated;
}

public static char[][] mirror (char[][] piece){ //mirror
    int h = piece.length;
    int w = piece[0].length;
    char[][] mirrored = new char[h][w];
    for (int i=0;i<h; i++){
        for (int j=0 ; j<w ; j++){
            mirrored[i][w-1-j]= piece[i][j];
        }
    }
    return mirrored;
}

```

```

public static boolean canPlace(char[][] piece, int r, int c){
    int h = piece.length;

    for (int i = 0 ; i < h ; i++){
        for (int j=0; j < piece[i].length ; j++){
            if (piece[i][j] != ' ' && board [r + i][c+j] != '.'){
                return false;
            }
        }
    }
    return true;
}

public static void placePiece(char[][] piece, int r, int c, char var ){

    for (int i = 0; i < piece.length; i++) {
        for (int j = 0; j < piece[i].length; j++) {
            if (piece[i][j] != ' '){
                board[r + i][c + j] = var;
            }
        }
    }
}

public static void removePiece (char[][] piece, int r , int c){
    for (int i = 0; i < piece.length; i++) {
        for (int j = 0; j < piece[i].length; j++) {
            if (piece[i][j] != ' '){
                board[r + i][c + j] = '.';
            }
        }
    }
}

public static void printBoard() {
    Map<Character, String> colorMap = new HashMap<>();

    for (char c = 'A'; c <= 'Z'; c++) {
        colorMap.put(c, color[(c - 'A') % color.length]);
    }
    for (char[] row : board) {
        for (char isi : row) {
            if (isi == '.') {
                System.out.print(isi + " ");
            } else {
                System.out.print(colorMap.get(isi) + isi + reset_color + " ");
            }
        }
        System.out.println();
    }
    System.out.println();
}

public static void saveFile(String fileOutput){
    try (PrintWriter writer = new PrintWriter(fileOutput)) {
        for (char[] row : board) {
            writer.println(new String(row));
        }
        System.out.println("File telah berhasil disimpan di " + fileOutput);

    } catch (IOException e) {
        System.out.println("Gagal menyimpan solusi.");
    }
}

```

```
public static void saveImage(String fileName){
    int cellSize =50;
    int l = cols *cellSize;
    int t = rows *cellSize;

    BufferedImage image = new BufferedImage (l,t,BufferedImage.TYPE_INT_ARGB);
    Graphics2D g2d = image.createGraphics();

    Map<Character, Color> colorMap = new HashMap<>();
    Random rand = new Random();

    for (char c = 'A'; c <= 'Z'; c++) {
        colorMap.put(c, new Color(rand.nextInt(bound:256), rand.nextInt(bound:256), rand.nextInt(bound:256)));
    }

    for (int r = 0; r < rows; r++) [
        for (int c = 0; c < cols; c++) {
            char block = board[r][c];
            if (block != '.') {
                g2d.setColor(colorMap.getOrDefault(block, Color.GRAY));
                g2d.fillRect(c * cellSize, r * cellSize, cellSize, cellSize);
            }
            g2d.setColor(Color.BLACK);
            g2d.drawRect(c * cellSize, r * cellSize, cellSize, cellSize);
        }
    ]

    g2d.dispose();

    try {
        ImageIO.write(image, formatName:"png", new File(fileName + ".png"));
        System.out.println("Solusi telah disimpan sebagai gambar: " + fileName + ".png");
    } catch (IOException e) {
        System.out.println("Gagal menyimpan gambar solusi.");
    }
}
```

Bab 4 Analisis dan Pengujian

4.1. Kasus Uji

4.1.1. Kasus Uji berdasarkan contoh pada spesifikasi Tugas Kecil 1

Input:

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

Hasil output:

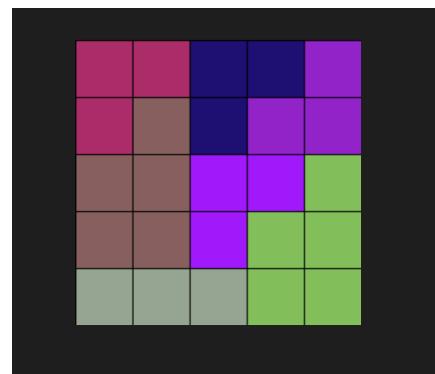
**IQ PUZZLER
PRO SOLVER**

```
A A B B D
A E B D D
E E C C F
E E C F F
G G G F F

Waktu pencarian: 2358 ms

Banyak kasus yang ditinjau: 862

Apakah anda ingin menyimpan solusi? (ya/tidak): ya
File telah berhasil disimpan di Solusi test1.txt
Apakah anda ingin menyimpan solusi gambar? (ya/tidak): ya
Solusi telah disimpan sebagai gambar: Solusi gambar test1.txt.png
```



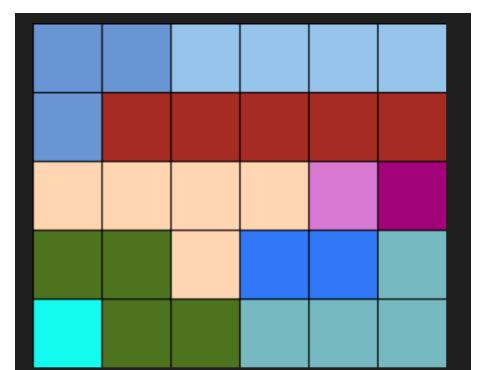
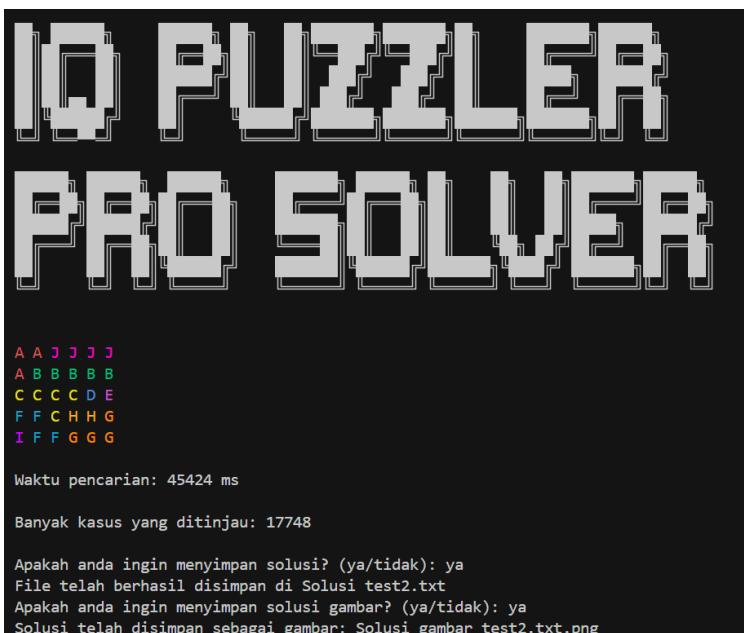
4.2. Kasus Uji Mandiri

4.2.1. Kasus Uji 1

Input:

```
1 5 6 10
2 DEFAULT
3 A
4 AA
5 B
6 B
7 B
8 B
9 B
10 C
11 CCCC
12 D
13 E
14 FF
15 | FF
16 GGG
17 G
18 H
19 H
20 I
21 JJJJ
```

Output:



```
TUCIL STIMA 1 > Tucil1_13523020 > src > Solusi test2.txt
1 AAJJJJ
2 ABBBBB
3 CCCDDE
4 FFCHHG
5 IFFGGG
```

4.2.2. Kasus Uji 2

Input:

```
1   6 5 12
2   DEFAULT
3   A
4   BBB
5   |   BBB
6   CC
7   C
8   C
9   C
10  D
11  E
12  F
13
14  G
15  HH
16  | H
17  |
18  HH
19  I
20  J
21  JJ
22
23  J
24  J
25  K
26  K
27  L
```

Output:

**IQ PUZZLER
PRO SOLVER**

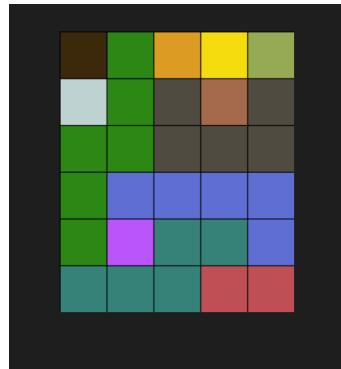
```
A B D E F
G B H I H
B B H H H
B C C C C
B L J J C
J J J K K

Waktu pencarian: 99 ms
Banyak kasus yang ditinjau: 12

Apakah anda ingin menyimpan solusi? (ya/tidak): ya
File telah berhasil disimpan di Solusi test3.txt
Apakah anda ingin menyimpan solusi gambar? (ya/tidak): ya
Solusi telah disimpan sebagai gambar: Solusi gambar test3.txt.png
```

TUCIL STIMA 1 > Tucil1_13523020 > src > Solusi test3.txt

```
1   ABDEF
2   GBHIH
3   BBHHH
4   BCCCC
5   BLJJC
6   JJJKK
```



4.2.3. Kasus Uji 3

Input:

```

1   7 9 20
2   DEFAULT
3   AA
4   A
5   AA
6   AAAA
7   AA
8   AAA
9   A
10  A
11  A
12  B
13  B
14  CCC
15  D
16  E
17  F
18  G
19  H
20  I
21  III
22  II
23  I
24  JJ
25  K
26  K
27  KKKK
28  L
29  L

```

30	~ M
31	NN
32	NN
33	N
34	N
35	O
36	OO
37	P
38	Q
39	R
40	SS
41	S
42	TT

Output:

**PUZZLER
PRO SOLVER**

```

BBCDEFGHA
JTCLLAAAAA
JTCMAAAANK
OOIIAAPNK
OIIAQANNK
IIIASNNRK
AAAASSKKK

```

Waktu pencarian: 21908 ms

Banyak kasus yang ditinjau: 5253

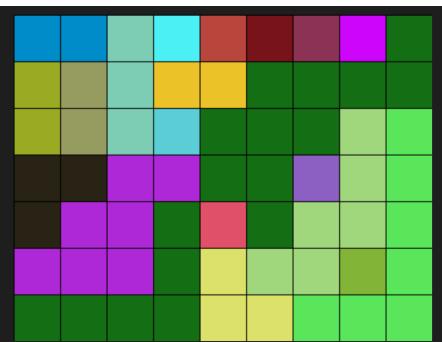
Apakah anda ingin menyimpan solusi? (ya/tidak): ya
File telah berhasil disimpan di Solusi test4.txt
Apakah anda ingin menyimpan solusi gambar? (ya/tidak): ya
Solusi telah disimpan sebagai gambar: Solusi gambar test4.txt.png

TUCIL STIMA 1 > Tucl1_13523020 > src > Solusi test4.txt

```

1 BBCDEFGHA
2 JTCLLAAAAA
3 JTCMAAAANK
4 OOIIAAPNK
5 OIIAQANNK
6 IIIASNNRK
7 AAAASSKKK
8

```



4.2.4. Kasus Uji 4

Input:

```
1   6 5 26
2   DEFAULT
3   A
4   B
5   C
6   D
7   EE
8   F
9   G
10  HH
11  I
12  J
13  K
14  LL
15  M
16  N
17  O
18  P
19  QQ
20  R
21  S
22  T
23  U
24  V
25  W
26  X
27  Y
28  Z
```

Output:

```
A B C D E
F G H I E
J K H L M
N O P L Q
R S T U Q
V W X Y Z

Waktu pencarian: 150 ms

Banyak kasus yang ditinjau: 26

Apakah anda ingin menyimpan solusi? (ya/tidak): ya
File telah berhasil disimpan di Solusi test5.txt
Apakah anda ingin menyimpan solusi gambar? (ya/tidak): ya
Solusi telah disimpan sebagai gambar: Solusi gambar test5.txt.png
```

1	ABCDE
2	FGHIE
3	JKHLM
4	NOPLQ
5	RSTUQ
6	VWXYZ

4.2.5. Kasus Uji 5

Input:

```
1  5 6 7
2  DEFAULT
3  AA
4  |
5  A
6  BB
7  | B B
8  BBB
9  | CC
10 CCCC
11 D
12 DD
13 EEE
14 |
15 FFFF
16 |
17 G
```

Output:

**10 PUZZLER
PRO SOLVER**

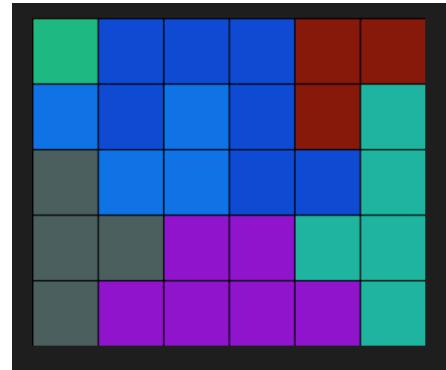
```
G B B B D D
A B A B D F
E A A B B F
E E C C F F
E C C C C F

Waktu pencarian: 10087 ms

Banyak kasus yang ditinjau: 3765

Apakah anda ingin menyimpan solusi? (ya/tidak): ya
File telah berhasil disimpan di Solusi test6.txt
Apakah anda ingin menyimpan solusii gambar? (ya/tidak): ya
Solusi telah disimpan sebagai gambar: Solusi gambar test6.txt.png
```

```
1  GBBBDD
2  ABABDF
3  EAABBF
4  EECCFF
5  ECCCCF
6
```



4.2.6. Kasus Uji 6

Input:

```
1 10 10 7
2 DEFAULT
3 G
4 B
5 BBB
6 | B
7 | B
8 BB
9 C
10 AA
11 A
12 AAA
13 KKK
14 LL
15 M M
16 M M
17 MMM
18
19
```

Output:

```
G C . B B A A A K L
B B B B . A . A K L
B . . B . A . . K .
M M M . . . . .
M . . . . .
M M M . . . .
. . . . .
. . . . .
. . . . .
. . . . .

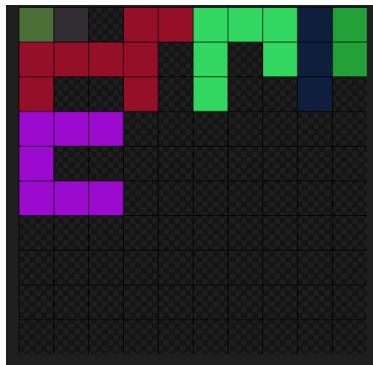
Waktu pencarian: 120 ms

Banyak kasus yang ditinjau: 7

Apakah anda ingin menyimpan solusi? (ya/tidak): ya
File telah berhasil disimpan di Solusi test7.txt
Apakah anda ingin menyimpan solusi gambar? (ya/tidak): ya
```

```
1 GC.BBAAKL
2 BBBB.A.AKL
3 B..B.A..K.
4 MMM.....
5 M.....
6 MMM.....
7 .....
8 .....
9 .....
10 .....
11 .....
```

Kasus ketika board tersisa banyak ruang kosong



4.2.7. Kasus Uji 7

Input:

```
1 4 5 6
2 DEFAULT
3 AA
4 B
5 BB
6 B
7 V
8 GG
9 |
10 G
11 TT
12 T
13 R
14 R
15 RR
16 R
```

Output:

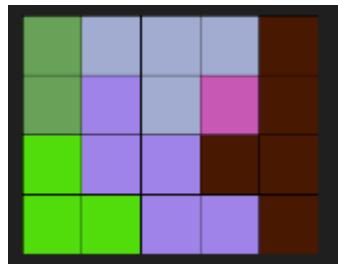
PRO SOLVER

A B B B R
A G B V R
T G G R R
T T G G R

Waktu pencarian: 256 ms
Banyak kasus yang ditinjau: 66

Apakah anda ingin menyimpan solusi? (ya/tidak): ya
File telah berhasil disimpan di Solusi test8.txt
Apakah anda ingin menyimpan solusi gambar? (ya/tidak): ya
Solusi telah disimpan sebagai gambar: Solusi gambar test8.txt.png

```
1 ABBBR
2 AGBVR
3 TGGRR
4 TTGGR
5
```



Bab 5 Kesimpulan

Berdasarkan analisis yang telah dilakukan, dapat disimpulkan bahwa algoritma Brute Force dapat digunakan dalam penyelesaian IQ Puzzler Pro. Namun, algoritma Brute Force memiliki kelemahan, yaitu waktu eksekusi yang dapat menjadi sangat lama terutama pada kasus dengan jumlah blok/piece yang cukup banyak atau konfigurasi papan yang lebih kompleks.

Meskipun algoritma Brute Force kurang efisien untuk ukuran yang relatif besar, namun algoritma Brute Force sangat cocok dalam pemecahan masalah dengan cukup sederhana dan langsung. Untuk penyelesaian masalah yang relatif kompleks membutuhkan suatu algoritma khusus atau algoritma yang cukup canggih sehingga pencarian solusi akan lebih efisien dan efektif. Dengan demikian, memahami keunggulan dan keterbatasan algoritma Brute Force dapat membantu dalam menentukan serta mengaplikasikan algoritma ini secara tepat untuk menyelesaikan berbagai permasalahan.

Bab 6 Lampiran

6.1. Github

https://github.com/StefanMatthew/Tucill_13523020.git

6.2. Tabel Pemeriksaan

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 (GUI)</i>		
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		