

# **IF2211 Strategi Algoritma**

## **Laporan Tugas Kecil 1**



Disusun oleh:

Rafael Marchel Darma Wijaya (13523146)

**PROGRAM STUDI TEKNIK INFORMATIKA**  
**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**  
**INSTITUT TEKNOLOGI BANDUNG**  
**JL. GANESA 10, BANDUNG 40132**  
**2025**

## Daftar Isi

<b>Daftar Isi</b>	<b>2</b>
<b>BAB I</b>	
<b>DESKRIPSI MASALAH DAN ALGORITMA</b>	<b>4</b>
1.1 IQ Puzzler Pro	4
1.2 Algoritma Brute Force	4
1.3 Algoritma Brute Force dalam penyelesaian IQ Puzzler Pro	5
<b>BAB II</b>	
<b>IMPLEMENTASI PROGRAM DALAM BAHASA JAVA</b>	<b>7</b>
2.1 Main.java	7
2.2 Board.java	7
2.3 Piece.java	9
2.4 Solver.java	11
2.5 Util.java	12
<b>BAB III</b>	
<b>SOURCE CODE PROGRAM</b>	<b>16</b>
3.1 Repository Program	16
3.2 Source Code	16
<b>BAB IV</b>	
<b>EKSPERIMEN</b>	<b>21</b>
4.1 Test Case 1	21
4.2 Test Case 2	24
4.3 Test Case 3	29
4.4 Test Case 4	32
4.5 Test Case 5	36
4.6 Test Case 6	41
4.7 Test Case 7	43
4.12 Test Case Custom 1	52
4.13 Test Case Custom 2	54
4.14 Test Case Custom 3	56
<b>BAB V</b>	
<b>LAMPIRAN</b>	<b>59</b>



## **BAB I**

### **DESKRIPSI MASALAH DAN ALGORITMA**

#### **1.1 IQ Puzzler Pro**

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

Komponen penting dari permainan IQ Puzzler Pro terdiri dari:

- Board (Papan) – Board merupakan komponen utama yang menjadi tujuan permainan dimana pemain harus mampu mengisi seluruh area papan menggunakan blok-blok yang telah disediakan.
- Blok/Piece – Blok adalah komponen yang digunakan pemain untuk mengisi papan kosong hingga terisi penuh. Setiap blok memiliki bentuk yang unik dan semua blok harus digunakan untuk menyelesaikan puzzle.

Permainan dimulai dengan papan yang kosong. Pemain dapat meletakkan blok puzzle sedemikian sehingga tidak ada blok yang bertumpang tindih. Setiap blok puzzle dapat dirotasikan maupun dicerminkan. Puzzle dinyatakan selesai jika dan hanya jika papan terisi penuh dan seluruh blok puzzle berhasil diletakkan.

#### **1.2 Algoritma Brute Force**

Algoritma Brute force adalah suatu pendekatan yang lurus (straightforward) untuk memecahkan suatu persoalan. Algoritma brute force memecahkan persoalan secara langsung dan sangat sederhana. Algoritma brute force umumnya tidak “cerdas” dan tidak sangkil, karena ia membutuhkan cost komputasi yang besar dan waktu yang lama dalam penyelesaiannya. Sehingga, algoritma brute force disebut juga algoritma naif. Algoritma brute force sangat cocok untuk persoalan yang ukurannya masukannya

(n) kecil. Algoritma brute force juga sering dijadikan sebagai basis pembandingan dengan algoritma lain. Meskipun bukan metode problem solving yang mangkus, hampir semua persoalan dapat diselesaikan dengan algoritma brute force.

### **1.3 Algoritma Brute Force dalam penyelesaian IQ Puzzler Pro**

Dalam implementasi brute force pada program ini, penulis menggunakan metode brute force yang rekursif dan melakukan backtracking jika seluruh kemungkinan sudah dicoba untuk satu kasus. Penulis juga menentukan terlebih dahulu semua kemungkinan bentuk puzzle termasuk rotasi dan pencerminan sebelum melakukan brute force. Berikut adalah langkah-langkah dan penjelasan algoritma brute force yang diimplementasikan:

1. Program awalnya membaca konfigurasi input txt dan menentukan dimensi board. Dalam program, board dibuat dan diisi dengan karakter '.' (titik) untuk menandakan petak kosong.
2. Kemudian program membaca semua bentuk puzzle dan menyimpannya dalam bentuk list.
3. Sebelum melakukan brute force, program terlebih dahulu menentukan segala kemungkinan bentuk yang akan dicoba. setiap puzzle piece akan dirotasi dan dicerminkan untuk mencari bentuk baru yang unik (jika ada bentuk yang sama ketika dirotasi atau dicerminkan, tidak perlu disimpan lagi) dan disimpan dalam bentuk list. Sehingga, tercipta suatu list orientasi dari satu puzzle yang juga disimpan dalam list of puzzle (list of list).
4. Lalu program melakukan rekursi untuk mencoba setiap kemungkinan pemasangan puzzle ke board. Program akan terus memanggil fungsi solve secara rekursif jika puzzle piece berhasil ditaruh di board. Program akan terus melakukan ini sampai solusi ditemukan (seluruh puzzle

piece habis dan tidak ada petak kosong) atau program melakukan backtrack jika menemui “dead-end” (seluruh kemungkinan puzzle piece telah habis). Backtrack dilakukan dengan menghapus puzzle piece yang telah ditaruh dan melanjutkan rekursi.

5. Ketika suatu orientasi dari suatu puzzle piece dapat ditaruh di board, maka seluruh orientasi piece tersebut tidak ikut dicoba lagi (piece remove from the list).
6. Program ini akan selalu mencoba menempatkan puzzle piece di petak kosong pertama pada board yaitu petak kosong paling kiri atas.
7. Penempatan puzzle piece dilakukan dengan offset. Offset bertujuan agar petak kosong pertama di board sesuai tempatnya dengan karakter alfabet pertama pada puzzle piece. Penempatan puzzle ini juga mengecek apakah puzzle piece bertabrakan dengan puzzle lain atau keluar dari board.
8. Program akan berhenti jika seluruh puzzle piece telah habis dan seluruh petak kosong pada board telah terisi atau jika solusi tidak ditemukan setelah seluruh kemungkinan telah dicoba.

## **BAB II**

### **IMPLEMENTASI PROGRAM DALAM BAHASA JAVA**

#### **2.1 Main.java**

Main.java	
Class Main adalah class utama yang menjalankan program.	
Atribut	
-	
Konstruktor	
-	-
Metode	
main	Metode utama yang menjalankan aplikasi. Berfungsi untuk tampilan terminal dan input pengguna dari terminal.

#### **2.2 Board.java**

Board.java	
Class Board adalah class yang merepresentasikan papan permainan.	
Atribut	
<ol style="list-style-type: none"><li>1. private static final String RESET (untuk mereset warna)</li><li>2. private static final String[] COLORS (list warna)</li></ol>	

3. <code>private final char[][] board</code> (representasi board dengan matriks karakter) 4. <code>private final int row;</code> (banyak baris) 5. <code>private final int col;</code> (banyak kolom)	
Konstruktor	
<code>public Board(int row, int col)</code>	Konstruktor yang membuat papan permainan dengan ukuran <code>rows × cols</code> dan mengisinya dengan tanda '.' (petak kosong).
<code>public Board(char[][] customBoard)</code>	Konstruktor yang membuat papan permainan berdasarkan matriks karakter (custom board).
Metode	
<code>public int getRow()</code>	Mengembalikan jumlah baris papan.
<code>public int getCol()</code>	Mengembalikan jumlah kolom papan.
<code>public char[][] getBoard()</code>	Mengembalikan matriks board.
<code>public int[] placePiece(Piece piece, int boardRow, int boardCol)</code>	Menaruh piece ke matriks board. Dilakukan kalkulasi offset terlebih dahulu dan pengecekan apakah piece overlapped atau keluar dari board. mengembalikan nilai



	offset yang nantinya dipakai ketika menghapus piece dari board.
<code>public void removePiece(Piece piece, int[] offset)</code>	Menghapus piece yang telah ditaruh dengan menggunakan offset berdasarkan method <code>placePiece</code> .
<code>public boolean isBoardFull()</code>	Mengecek apakah seluruh petak kosong telah terisi.
<code>public void printBoard()</code>	print kondisi board ke terminal dan mewarnai puzzle piece.

### 2.3 Piece.java

Piece.java	
Class Piece adalah class yang merepresentasikan puzzle piece.	
Atribut	
<ol style="list-style-type: none"> <li>1. <code>private final char[][] shape</code> (matriks karakter representasi bentuk puzzle piece.)</li> <li>2. <code>private final int row</code> (banyak baris)</li> <li>3. <code>private final int col</code> (banyak kolom)</li> </ol>	
Konstruktor	
<code>public Piece(char[][] shape)</code>	Membuat puzzle piece berdasarkan matriks karakter.

Metode	
<code>public char[][] getShape()</code>	Mengembalikan matriks bentuk puzzle piece
<code>public int getRow()</code>	Mengembalikan jumlah baris matriks puzzle piece.
<code>public int getCol()</code>	Mengembalikan jumlah kolom matriks puzzle piece.
<code>public Piece rotate()</code>	Mengembalikan objek puzzle piece baru yang telah dirotasi 90° secara counter-clockwise
<code>public Piece mirror()</code>	Mengembalikan objek puzzle piece baru yang telah dicerminkan

## 2.4 Solver.java

Solver.java
Class Solver adalah class yang mengimplementasikan algoritma brute force.
Atribut
<ol style="list-style-type: none"> <li>1. <code>private long attempts</code> (banyak kasus yang dicoba)</li> <li>2. <code>private double time</code> (waktu yang dibutuhkan dalam ms)</li> </ol>
Konstruktor

-	-
Metode	
public boolean findSolution(Board board, List<Piece> pieces)	Memulai proses solving dengan menentukan semua kemungkinan bentuk puzzle piece, menghitung waktu, dan mengembalikan true jika menemukan solusi.
private List<Piece> getUniqueOrientations(Piece piece)	Mengembalikan list bentuk unik dari suatu puzzle piece setelah dirotasi dan dicerminkan.
private boolean sameShape(Piece firstPiece, Piece secondPiece)	Mengecek apakah dua buah pieces sama atau tidak (dalam konteks bentuk).
private boolean containShape(List<Piece> listPieces, Piece checkedPiece)	Mengecek apakah suatu bentuk piece telah ada dalam suatu list of piece.
private boolean solve(Board board, List<List<Piece>> piecesShapes)	Algoritma brute force yang diimplementasikan dengan rekursif backtracking.
private int[] findEmptySlot(Board board)	Mengembalikan koordinat (index) dari petak kosong paling kiri atas di board.

## 2.5 Util.java

Util.java	
Class Util adalah class yang menyediakan fungsi helper dalam I/O program ini.	
Atribut	
1. private static final Color[] COLORS (warna untuk file image)	
Konstruktor	
-	-
Metode	
public static Board readBoardFromFile(Scanner scanner, int row, int col)	Membaca file konfigurasi dan mengembalikan objek board berdasarkan konfigurasi yang dibaca.
public static List<Piece> readPieceFromFile(Scanner scanner, int pieceCount)	Membaca file konfigurasi untuk bentuk puzzle piece dan mengembalikan list of puzzle piece.
public static void saveResult(String filename, Solver solver, boolean solved, Board board)	Menyimpan output terminal ke dalam suatu file txt.
public static boolean validateConfigFile(Scanner scanner)	Validasi format file konfigurasi.
public static void	Menyimpan board dalam

saveSolutionAsImage(Board board, String filename)	bentuk image (png).
---	---------------------

## **BAB III**

### **SOURCE CODE PROGRAM**

#### **3.1 Repository Program**

Repository Program dapat diakses melalui tautan Github berikut: [https://github.com/V-Kleio/Tucill\\_13523146](https://github.com/V-Kleio/Tucill_13523146)

#### **3.2 Source Code**

Main.java:

```
package iq_puzzler_pro;

import java.io.File;
import java.io.FileNotFoundException;
import java.util.List;
import java.util.Scanner;

public class Main {
    @SuppressWarnings("ConvertToTryWithResources")
    // ! Suppress so that the IDE doesnt tell me to
    put the input scanner inside a try block
    public static void main(String[] args) {
        Board board;
        List<Piece> pieces;
        Scanner input = new Scanner(System.in);

        System.out.println("=====
        =====");
        System.out.println("Selamat datang di IQ
        Puzzler Pro Solver");

        System.out.println("=====
        =====");
        System.out.println();

        System.out.print("Masukkan nama file
        konfigurasi (txt file): ");
        String file = input.nextLine().trim();

        try {
```

```

        Scanner fileScanner = new Scanner(new
File(file));
        if
(!Util.validateConfigFile(fileScanner)) {
System.out.println("\u001B[1;91mKonfigurasi file
tidak valid.");
            input.close();
            return;
        }

        fileScanner.close();
        fileScanner = new Scanner(new
File(file));

        int row = fileScanner.nextInt();
        int col = fileScanner.nextInt();
        int pieceCount = fileScanner.nextInt();
        fileScanner.nextLine(); // * go to next
line after the reading of N M P
        board =
Util.readBoardFromFile(fileScanner, row, col);
        pieces =
Util.readPieceFromFile(fileScanner, pieceCount);
        System.out.println();
        System.out.println("File berhasil
diproses.");
        System.out.println();

System.out.println("=====");
        System.out.println();
    } catch (FileNotFoundException e) {
        System.out.println("\u001B[1;91mFile
tidak ditemukan, tulis nama file lengkap dengan
ekstensi txt dan pastikan file ada.\n" +
e.getMessage());
        input.close();
        return;
    }

    if (pieces == null) {

```

```

        input.close();
        return;
    }

    Solver solver = new Solver();
    boolean solved = solver.findSolution(board,
pieces);
    if (solved) {
        board.printBoard();
        System.out.println("Solusi berhasil
ditemukan!");
    } else {
        System.out.println("Tidak ada solusi yang
ditemukan berdasarkan konfigurasi yang diberikan.");
    }

    System.out.println();
    System.out.print("Apakah anda ingin menyimpan
solusi? (ya/tidak) ");
    String save = input.nextLine().trim();
    while (!("ya".equalsIgnoreCase(save) ||
"tidak".equalsIgnoreCase(save))) {
        System.out.println("\u001B[1;91mPerintah
invalid\u001B[0m");
        System.out.print("Apakah anda ingin
menyimpan solusi? (ya/tidak) ");
        save = input.nextLine().trim();
    }
    if ("ya".equalsIgnoreCase(save)) {
        System.out.print("Masukkan nama file
(tanpa ekstensi file): ");
        save = input.nextLine().trim();
        Util.saveResult(save, solver, solved,
board);
    }

    if (solved) {
        System.out.println();
        System.out.print("Apakah anda ingin
menyimpan solusi sebagai gambar? (ya/tidak) ");
        save = input.nextLine().trim();
        while (!("ya".equalsIgnoreCase(save) ||

```



```

"tidak".equalsIgnoreCase(save))) {

System.out.println("\u001B[1;91mPerintah
invalid\u001B[0m");
        System.out.print("Apakah anda ingin
menyimpan solusi sebagai gambar? (ya/tidak) ");
        save = input.nextLine().trim();
    }
    if ("ya".equalsIgnoreCase(save)) {
        System.out.print("Masukkan nama file
(tanpa ekstensi file): ");
        save = input.nextLine().trim();
        Util.saveSolutionAsImage(board,
save);
    }
}

input.close();
}
}

```

#### Board.java:

```

package iq_puzzler_pro;

public class Board {
    private static final String RESET = "\u001B[0m";

    private static final String[] COLORS = {
        "\u001B[101m",
        "\u001B[102m",
        "\u001B[103m",
        "\u001B[104m",
        "\u001B[105m",
        "\u001B[106m",
        "\u001B[41m",
        "\u001B[42m",
        "\u001B[43m",
        "\u001B[44m",
        "\u001B[45m",
        "\u001B[46m",
        "\u001B[91m",
    }
}

```

```

        "\u001B[92m",
        "\u001B[93m",
        "\u001B[94m",
        "\u001B[95m",
        "\u001B[96m",
        "\u001B[1;91m",
        "\u001B[1;92m",
        "\u001B[1;93m",
        "\u001B[1;94m",
        "\u001B[1;95m",
        "\u001B[1;96m",
        "\u001B[4;31m",
        "\u001B[4;32m"
    };

    private final char[][] board;
    private final int row;
    private final int col;

    public Board(int row, int col) {
        this.row = row;
        this.col = col;
        this.board = new char[row][col];

        // * Board is empty (filled with dots)
        for (int i = 0; i < row; i++) {
            for (int j = 0; j < col; j++) {
                board[i][j] = '.';
            }
        }
    }

    public Board(char[][] customBoard) {
        this.row = customBoard.length;
        this.col = customBoard[0].length;
        this.board = customBoard;
    }

    public int getRow() {
        return row;
    }

```

```

    public int getCol() {
        return col;
    }

    public char[][] getBoard() {
        return board;
    }

    public int[] placePiece(Piece piece, int
boardRow, int boardCol) {
        char[][] shape = piece.getShape();
        int pieceRows = piece.getRow();
        int pieceCols = piece.getCol();

        for (int charRow = 0; charRow < pieceRows;
charRow++) {
            for (int charCol = 0; charCol <
pieceCols; charCol++) {
                // * Find coordinates of char in the
piece
                if (shape[charRow][charCol] == '.') {
                    continue;
                }

                // * Find the offset
                // * Offset is the board coordinates
that align with the piece [0,0]
                int offsetRow = boardRow - charRow;
                int offsetCol = boardCol - charCol;
                boolean canPlace = true;

                // * Iterate through the board
starting from the offset coordinates
                // * Compare it with the piece from
[0,0] to check for validity
                // * Valid -> within the board and
not overlapped (dots represent empty slot)
                for (int i = 0; i < pieceRows &&
canPlace; i++) {
                    for (int j = 0; j < pieceCols;
j++) {
                        if (shape[i][j] != '.') {

```

```

        int offsetRowPointer =
offsetRow + i;
        int offsetColPointer =
offsetCol + j;
        if (offsetRowPointer < 0
|| offsetRowPointer >= this.row || offsetColPointer <
0 || offsetColPointer >= this.col ||
board[offsetRowPointer][offsetColPointer] != '.') {
            canPlace = false;
            break;
        }
    }
}

// * Iterate again to replace the
board with the piece
    if (canPlace) {
        for (int i = 0; i < pieceRows;
i++) {
            for (int j = 0; j <
pieceCols; j++) {
                if (shape[i][j] != '.') {
                    board[offsetRow +
i][offsetCol + j] = shape[i][j];
                }
            }
        }
        return new int[]{offsetRow,
offsetCol}; // ! Offset is also needed for removing
the piece
    }
}

return null; // ! No valid placement
}

public void removePiece(Piece piece, int[]
offset) {
    if (offset == null) {
        return;
    }
}

```

```

    }

    int offsetRow = offset[0];
    int offsetCol = offset[1];
    char[][] shape = piece.getShape();
    int pieceRows = piece.getRow();
    int pieceCols = piece.getCol();

    // * We already know the offset so just
replace the alphabet back to dots
    for (int i = 0; i < pieceRows; i++) {
        for (int j = 0; j < pieceCols; j++) {
            if (shape[i][j] != '.') {
                board[offsetRow + i][offsetCol +
j] = '.';
            }
        }
    }
}

public boolean isBoardFull() {
    for (int i = 0; i < row; i++) {
        for (int j = 0; j < col; j++) {
            if (board[i][j] == '.') {
                return false;
            }
        }
    }

    return true;
}

public void printBoard() {
    for (int i = 0; i < row; i++) {
        for (int j = 0; j < col; j++) {
            char character = board[i][j];

            int characterValue = character - 'A';
            String color;
            if (characterValue >= 0 &&
characterValue < COLORS.length) {
                color = COLORS[characterValue];

```

```

        } else {
            color = "";
        }

        System.out.print(color + character +
RESET);
    }
    System.out.println();
}
}
}

```

### Piece.java:

```

package iq_puzzler_pro;

public class Piece {
    private final char[][] shape;
    private final int row;
    private final int col;

    public Piece(char[][] shape) {
        this.shape = shape;
        this.row = shape.length;
        this.col = shape[0].length;
    }

    public char[][] getShape() {
        return shape;
    }

    public int getRow() {
        return row;
    }

    public int getCol() {
        return col;
    }

    public Piece rotate() {
        // * This will rotate the piece ccw
        char[][] rotatedPiece = new char[col][row];
    }
}

```

```

        for (int i = 0; i < row; i++) {
            for (int j = 0; j < col; j++) {
                rotatedPiece[col - 1 - j][i] =
shape[i][j];
            }
        }

        return new Piece(rotatedPiece);
    }

    public Piece mirror() {
        char[][] mirroredPiece = new char[row][col];
        for (int i = 0; i < row; i++) {
            for (int j = 0; j < col; j++) {
                mirroredPiece[i][col - 1 - j] =
shape[i][j];
            }
        }

        return new Piece(mirroredPiece);
    }
}

```

#### Solver.java:

```

package iq_puzzler_pro;

import java.util.ArrayList;
import java.util.List;
public class Solver {
    private long attempts = 0;
    private double time = 0.0;

    public long getAttempts() {
        return attempts;
    }

    public double getTime() {
        return time;
    }

    public boolean findSolution(Board board,

```

```

List<Piece> pieces) {
    // * Store every piece in a list
    // * for each piece also store all the
possible shapes (rotated & mirrored)
    // * Making it list inside a list
    List<List<Piece>> piecesShapes = new
ArrayList<>();
    for (Piece p : pieces) {

piecesShapes.add(getUniqueOrientations(p));
    }

    long startTime = System.nanoTime();
    boolean solved = solve(board, piecesShapes);
    long endTime = System.nanoTime();

    time = (endTime - startTime) / 1000000.0;

    System.out.println("Waktu pencarian: " + time
+ " ms");
    System.out.println("Banyak kasus yang
ditinjau: " + attempts);

    return solved;
}

private List<Piece> getUniqueOrientations(Piece
piece) {
    List<Piece> orientations = new ArrayList<>();

    Piece currentPiece = piece;
    Piece mirroredPiece = piece.mirror();

    // * Rotate both original and mirrored piece
    for (int i = 0; i < 4; i++) {
        // ! Don't add existing shape
        if (!containShape(orientations,
currentPiece)) {
            orientations.add(currentPiece);
        }

        if (!containShape(orientations,

```



```

        mirroredPiece)) {
            orientations.add(mirroredPiece);
        }

        currentPiece = currentPiece.rotate();
        mirroredPiece = mirroredPiece.rotate();
    }

    return orientations;
}

private boolean sameShape(Piece firstPiece, Piece
secondPiece) {
    // * Helper to check if two pieces are the
    same
    if (firstPiece.getRow() !=
secondPiece.getRow() || firstPiece.getCol() !=
secondPiece.getCol()) {
        return false;
    }

    char[][] firstShape = firstPiece.getShape();
    char[][] secondShape =
secondPiece.getShape();

    for (int i = 0; i < firstPiece.getRow(); i++)
    {
        for (int j = 0; j < firstPiece.getCol();
j++) {
            if (firstShape[i][j] !=
secondShape[i][j]) {
                return false;
            }
        }
    }

    return true;
}

private boolean containShape(List<Piece>
listPieces, Piece checkedPiece) {
    // * Helper to check if a piece shape already

```

```

exist in a list of piece
    for (Piece p : listPieces) {
        if (sameShape(p, checkedPiece)) {
            return true;
        }
    }

    return false;
}

private boolean solve(Board board,
List<List<Piece>> piecesShapes) {
    // * If piecesShapes is empty, then there is
no more piece to place
    if (piecesShapes.isEmpty()) {
        return (findEmptySlot(board) == null);
    }

    // * Find empty slot
    int[] emptySlot = findEmptySlot(board);
    if (emptySlot == null) {
        return false;
    }

    int emptyRow = emptySlot[0];
    int emptyCol = emptySlot[1];

    for (int i = 0; i < piecesShapes.size(); i++)
    {
        List<Piece> orientations =
piecesShapes.get(i);

        for (Piece p : orientations) {
            attempts++;
            int[] offset = board.placePiece(p,
emptyRow, emptyCol);
            if (offset != null) {
                List<List<Piece>> remaining = new
ArrayList<>(piecesShapes);
                remaining.remove(i);

                if (solve(board, remaining)) {

```

```

        return true;
    }

    board.removePiece(p, offset);
}

}

return false;
}

private int[] findEmptySlot(Board board) {
    // * Find the top-leftmost empty slot
    for (int i = 0; i < board.getRow(); i++) {
        for (int j = 0; j < board.getCol(); j++)
        {
            if (board.getBoard()[i][j] == '.') {
                return new int[]{i, j};
            }
        }
    }

    return null;
}
}

```

#### Util.java:

```

package iq_puzzler_pro;

import java.awt.Color;
import java.awt.Graphics2D;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
import javax.imageio.ImageIO;

```

```

public class Util {
    // * Color generated randomly using
    https://mokole.com/palette.html
    private static final Color[] COLORS = {
        new Color(211, 211, 211),
        new Color(47, 79, 79),
        new Color(139, 69, 19),
        new Color(34, 139, 24),
        new Color(128, 128, 0),
        new Color(72, 61, 139),
        new Color(0, 0, 128),
        new Color(154, 205, 50),
        new Color(102, 205, 170),
        new Color(255, 0, 0),
        new Color(255, 165, 0),
        new Color(255, 255, 0),
        new Color(124, 252, 0),
        new Color(0, 250, 154),
        new Color(138, 43, 226),
        new Color(139, 0, 139),
        new Color(220, 20, 60),
        new Color(0, 191, 255),
        new Color(0, 0, 255),
        new Color(255, 127, 80),
        new Color(255, 0, 255),
        new Color(30, 144, 255),
        new Color(219, 112, 147),
        new Color(240, 230, 140),
        new Color(255, 20, 147),
        new Color(238, 130, 238)
    };

    public static Board readBoardFromFile(Scanner
scanner, int row, int col) {
        String boardType = "";
        if(scanner.hasNextLine()) {
            boardType = scanner.nextLine().trim();
        }

        if("CUSTOM".equalsIgnoreCase(boardType)) {
            char[][] customLayout = new

```

```

char[row][col];
    for (int i = 0; i < row; i++) {
        String line =
scanner.nextLine().trim();
        for (int j = 0; j < col; j++) {
            char character = line.charAt(j);
            if (character == 'X') {
                customLayout[i][j] = '.';
            } else {
                customLayout[i][j] = ' ';
            }
        }
    }

    System.out.println("Membuat tipe board
custom.");
    return new Board(customLayout);
} else if
("PYRAMID".equalsIgnoreCase(boardType)) {
    System.out.println("Fitur pyramid tidak
diimplementasikan.");
    System.out.println("Membuat tipe board
default.");
    return new Board(row, col);
} else {
    System.out.println("Membuat tipe board
default.");
    return new Board(row, col);
}

}

public static List<Piece>
readPieceFromFile(Scanner scanner, int pieceCount) {
    List<String> allLines = new ArrayList<>();
    while (scanner.hasNextLine()) {
        String line = scanner.nextLine();
        line = line.replaceAll("\\s+$", ""); // !
preserve leading whitespaces and remove trailing
whitespaces
        if (!line.isEmpty()) {
            allLines.add(line);
        }
    }
}

```

```

        }
    }

    List<Piece> pieces = new ArrayList<>();
    int i = 0;

    while (i < allLines.size() && pieces.size() <
pieceCount) {
        List<String> shapeLines = new
ArrayList<>();
        char letter =
allLines.get(i).trim().charAt(0); // * Find the
alphabet representing a piece
        shapeLines.add(allLines.get(i));
        i++;

        while (i < allLines.size() &&
allLines.get(i).trim().charAt(0) == letter) {
            shapeLines.add(allLines.get(i));
            i++;
        }

        // * In lines with the same alphabet,
find the line with the most characters (most column)
        int pieceRow = shapeLines.size();
        int pieceCol = 0;
        for (String line : shapeLines) {
            if (line.length() > pieceCol) {
                pieceCol = line.length();
            }
        }

        char[][] shape = new
char[pieceRow][pieceCol];
        for (int j = 0; j < pieceRow; j++) {
            String currentLine =
shapeLines.get(j);
            for (int k = 0; k < pieceCol; k++) {
                if (k < currentLine.length()) {
                    shape[j][k] =
(currentLine.charAt(k) == ' ') ? '.' :
currentLine.charAt(k); // ! Turn the leading

```

```

whitespaces into dots
                } else {
                    shape[j][k] = '.';
                }
            }
        }
        pieces.add(new Piece(shape));
    }

    if (pieces.size() != pieceCount) {
        System.out.println("\u001B[1;91mBanyak
blok puzzle tidak valid.");
        return null;
    }

    return pieces;
}

public static void saveResult(String filename,
Solver solver, boolean solved, Board board) {
    try (PrintWriter writer = new PrintWriter(new
FileWriter("test/" + filename + ".txt"))) {
        writer.println("Waktu pencarian: " +
solver.getTime() + " ms");
        writer.println("Banyak kasus yang
ditinjau: " + solver.getAttempts() + " kasus");
        writer.println();

        if (solved) {
            writer.println("Solusi:");
            for (int i = 0; i < board.getRow();
i++) {
                for (int j = 0; j <
board.getCol(); j++) {
                    writer.print(board.getBoard()[i][j]);
                }
                writer.println();
            }
        } else {
            writer.println("Tidak ada solusi yang
ditemukan.");
        }
    }
}

```

```

    }
    } catch (IOException e) {
        System.out.println("\u001B[1;91mTerjadi
kesalahan saat menyimpan file: " + e.getMessage());
    }
}

    public static boolean validateConfigFile(Scanner
scanner) {
        try {
            // ! Check for dimension and piece count
            int row = scanner.nextInt();
            int col = scanner.nextInt();
            int pieceCount = scanner.nextInt();

            if (row <= 0 || col <= 0 || pieceCount <=
0 || pieceCount > 26) {
                System.out.println("\u001B[1;91mDimensi atau banyak
blok puzzle tidak valid.");
                return false;
            }
            scanner.nextLine();

            // ! Check for board type
            if (!scanner.hasNextLine()) {
                System.out.println("\u001B[1;91mJenis
Kasus tidak ditemukan.");
                return false;
            }
            String boardType =
scanner.nextLine().trim();
            if ("CUSTOM".equalsIgnoreCase(boardType))
{
                // ! Check for custom board
configuration
                for (int i = 0; i < row; i++) {
                    if (!scanner.hasNextLine()) {
                        System.out.println("\u001B[1;91mJumlah baris tidak
sesuai dengan dimensi.");
                        return false;
                    }
                }
            }
        }
    }
}

```



```

        }
        String line =
scanner.nextLine().trim();
        if (line.length() < col) {

System.out.println("\u001B[1;91mJumlah kolom tidak
sesuai dengan dimensi.");
            return false;
        }
    }
    } else if
("DEFAULT".equalsIgnoreCase(boardType)) {
        System.out.print("");
    } else if
("PYRAMID".equalsIgnoreCase(boardType)) {
        System.out.println("Jenis kasus
pyramid tidak diimplementasikan. Board menjadi
default.");
    } else {
        System.out.println("\u001B[1;91mJenis
kasus tidak dikenali");
        return false;
    }

    if (!scanner.hasNextLine()) {

System.out.println("\u001B[1;91mKonfigurasi blok
puzzle tidak ditemukan.");
        return false;
    }
    } catch (Exception e) {
        System.out.println("\u001B[1;91mFormat
file tidak valid.");
        return false;
    }
    return true;
}

    public static void saveSolutionAsImage(Board
board, String filename) {
        int size = 40;
        int row = board.getRow();

```

```

        int col = board.getCol();
        int width = col * size;
        int height = row * size;

        BufferedImage image = new
BufferedImage(width, height,
BufferedImage.TYPE_INT_ARGB);
        Graphics2D graphic = image.createGraphics();

        graphic.setColor(Color.WHITE);
        graphic.fillRect(0, 0, width, height);

        for (int i = 0; i < row; i++) {
            for (int j = 0; j < col; j++) {
                int x = j * size;
                int y = i * size;

                // * Square outline
                graphic.setColor(Color.LIGHT_GRAY);
                graphic.drawRect(x, y, size, size);

                char character =
board.getBoard()[i][j];
                int characterValue = character - 'A';
                if (characterValue >= 0 &&
characterValue < COLORS.length) {
                    graphic.setColor(COLORS[characterValue]);
                } else {
                    graphic.setColor(Color.BLACK);
                }

                graphic.fillRect(x, y, size, size);
            }
        }

        graphic.dispose();

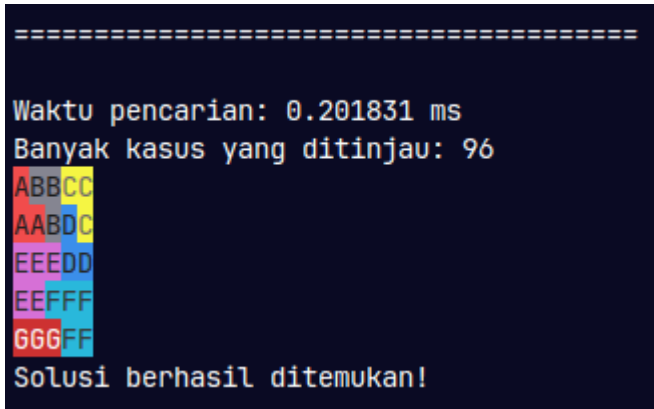
        try {
            ImageIO.write(image, "png", new
File("test/image/" + filename + ".png"));
            System.err.println("Berhasil menyimpan

```

```
solusi sebagai gambar: " + filename + ".png");  
    } catch (IOException e) {  
        System.out.println("Gagal menyimpan  
solusi sebagai gambar.\n" + e.getMessage());  
    }  
}  
}
```

## BAB IV EKSPERIMEN

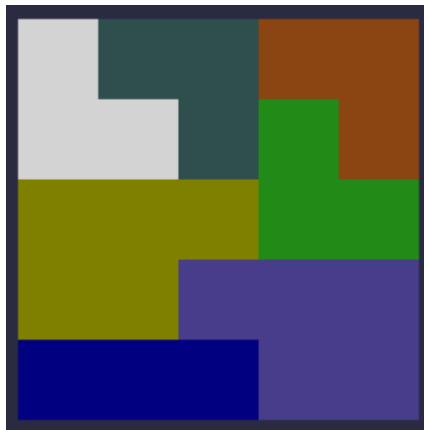
### 4.1 Test Case 1

Masukan
5 5 7 DEFAULT A AA B BB C CC D DD EE EE E FF FF F GGG
Keluaran
Terminal:


File Teks(.txt)

```
1 Waktu pencarian: 0.201831 ms
2 Banyak kasus yang ditinjau: 96 kasus
1
2 Solusi:
3 ABBCC
4 AABDC
5 EEEDD
6 EEFFF
7 GGGFF
8
```

File Gambar(.png)



## 4.2 Test Case 2

Masukan

```
5 11 12
DEFAULT
A
A
AA
A
```

```
B B
BBB
CCCC
C
D
DD
  DD
    E
  EEE
    E
      F
FFFF
GGG
  G
HHH
HH
  I
II
  I
JJ
  J
  J
KK
  K
    L
    L
LLL
```

Keluaran

Terminal:

```

=====
Waktu pencarian: 60.587321 ms
Banyak kasus yang ditinjau: 213756
ABBBCCCCDHH
ABGBCIIDDHH
AAGGIIDDEHL
KAGFJJJEEEL
KKFFFFJELLL
Solusi berhasil ditemukan!

```

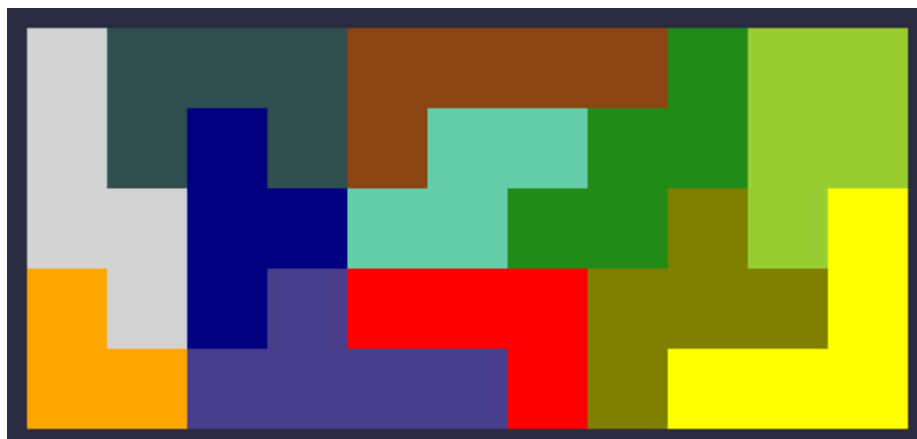
File Teks(.txt)

```

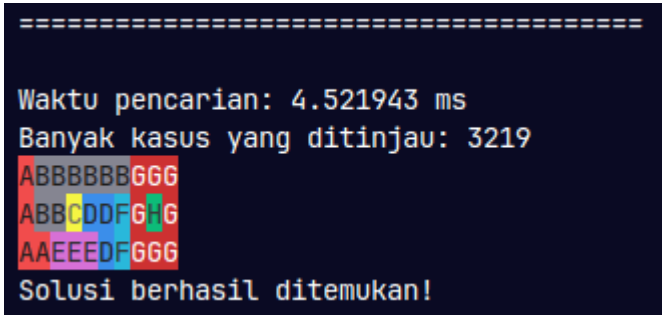
1 Waktu pencarian: 60.587321 ms
1 Banyak kasus yang ditinjau: 213756 kasus
2
3 Solusi:
4 ABBBCCCCDHH
5 ABGBCIIDDHH
6 AAGGIIDDEHL
7 KAGFJJJEEEL
8 KKFFFFJELLL
9

```

File Gambar(.png)



### 4.3 Test Case 3

Masukan
3 10 8 DEFAULT A A AA BB BB B B B B C DD D E E E FF GGG G  G GGG H
Keluaran
Terminal:




File Teks(.txt)

```
1 Waktu pencarian: 4.521943 ms
1 Banyak kasus yang ditinjau: 3219 kasus
2
3 Solusi:
4 ABBBBBBGGG
5 ABBCCDFGHG
6 AAEEEDFGGG
7
```

File Gambar(.png)



#### 4.4 Test Case 4

Masukan

```
2 2 2
DEFAULT
A
A
AA
BB
```

Keluaran

Terminal:

<pre>===== Waktu pencarian: 0.058897 ms Banyak kasus yang ditinjau: 26 Tidak ada solusi yang ditemukan berdasarkan konfigurasi yang diberikan.</pre>
File Teks(.txt)
<pre>1 Waktu pencarian: 0.058897 ms 1 Banyak kasus yang ditinjau: 26 kasus 2 3 Tidak ada solusi yang ditemukan. 4</pre>
File Gambar(.png)
-

#### 4.5 Test Case 5

Masukan
<pre>3 5 6 DEFAULT ZZ Z Z F FF I L LL L A A B</pre>
Keluaran

Terminal:

```
=====
Waktu pencarian: 0.237593 ms
Banyak kasus yang ditinjau: 92
ZZFIL
ZFFLL
ZAALB
Solusi berhasil ditemukan!
```

File Teks(.txt)

```
1 Waktu pencarian: 0.237593 ms
1 Banyak kasus yang ditinjau: 92 kasus
2
3 Solusi:
4 ZZFIL
5 ZFFLL
6 ZAALB
7
```

File Gambar(.png)



#### 4.6 Test Case 6

Masukan

```
5 11 12
DEFAULT
CC
C
CC
W
WW
  WW
L
LLLL
Z
ZZ
  Z
  Z
A
A
AAA
TTT
  T
  M
MMMM
  G
GG
  GG
  QQ
QQ
BB
  B
  PP
PPP
  N
NNN
```

Keluaran

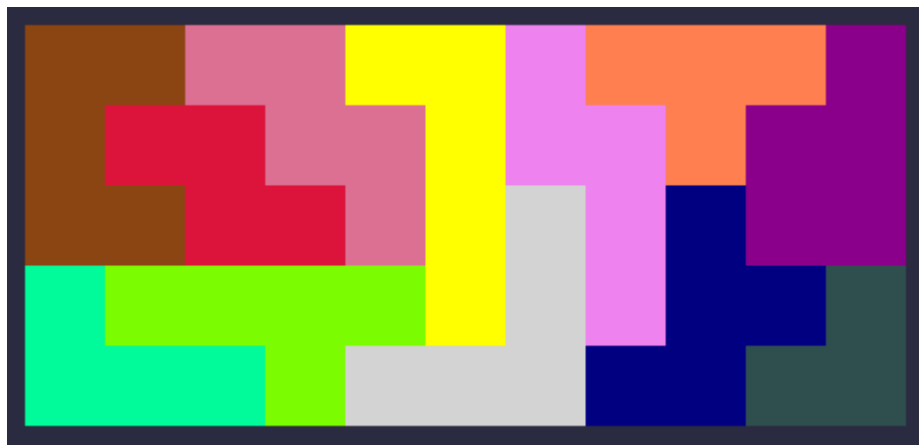
Terminal:

```
=====
Waktu pencarian: 131.110449 ms
Banyak kasus yang ditinjau: 1317712
CCWWLLZTTTP
CQQWWLZZTPP
CCQQWLAZGPP
NNMMMLAZGGB
NNNMAAAGGBB
Solusi berhasil ditemukan!
```

File Teks(.txt)

```
1 Waktu pencarian: 131.110449 ms
1 Banyak kasus yang ditinjau: 1317712 kasus
2
3 Solusi:
4 CCWWLLZTTTP
5 CQQWWLZZTPP
6 CCQQWLAZGPP
7 NNNMMLAZGGB
8 NNNMAAAGGBB
9
```

File Gambar(.png)



#### 4.7 Test Case 7

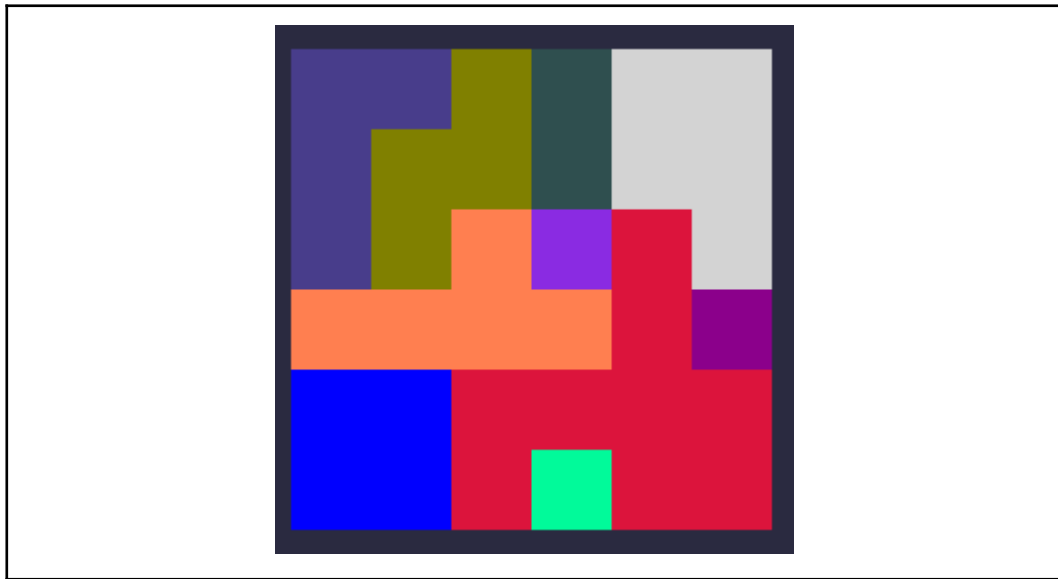
Masukan
6 6 10 DEFAULT FF F F E EE E B B AA AA A O P T TT T T Q Q QQQQ QQ Q SS SS N
Keluaran
Terminal:

```
=====
Waktu pencarian: 3.970647 ms
Banyak kasus yang ditinjau: 4763
FFEBAA
FEEBAA
FETOQA
TTTTQP
SSQQQQ
SSQNQQ
Solusi berhasil ditemukan!
```

File Teks(.txt)

```
1 Waktu pencarian: 3.970647 ms
1 Banyak kasus yang ditinjau: 4763 kasus
2
3 Solusi:
4 FFEBA
5 FEEBA
6 FETOQA
7 TTTTQP
8 SSQQQQ
9 SSQNQQ
10
```

File Gambar(.png)



#### 4.12 Test Case Custom 1

Masukan
<pre> 5 7 5 CUSTOM ...X... .XXXXX. XXXXXXXX .XXXXX. ...X... A AAA BB BBB CCCC C D EEE E </pre>
Keluaran



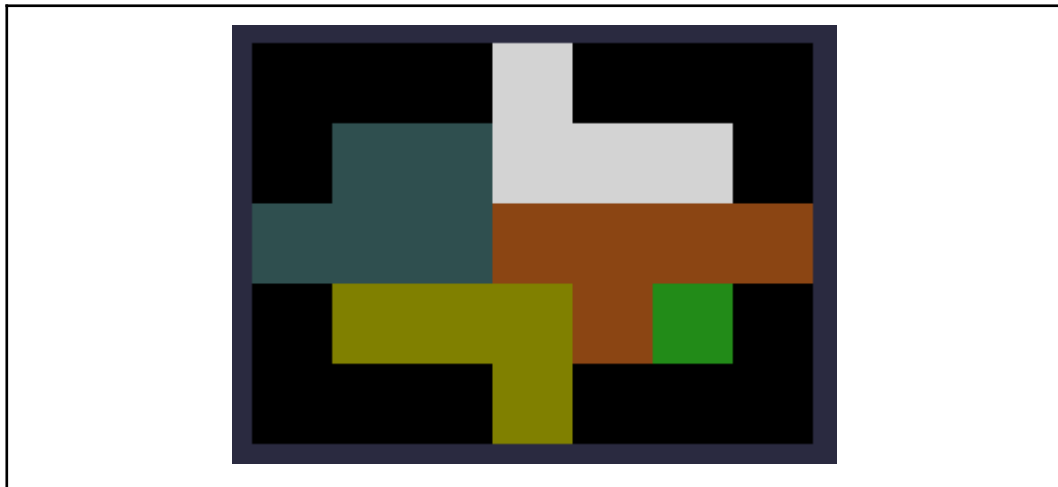
Terminal:

```
=====
Waktu pencarian: 0.205965 ms
Banyak kasus yang ditinjau: 73
  A
BBAAA
BBBCCCC
EEECD
  E
Solusi berhasil ditemukan!
```

File Teks(.txt)

```
1 Waktu pencarian: 0.205965 ms
1 Banyak kasus yang ditinjau: 73 kasus
2
3 Solusi:
4 |..|A|..|
5 |BBAAA|
6 |BBBCCCC|
7 |EEECD|
8 |..|E|..|
9
```

File Gambar(.png)



#### 4.13 Test Case Custom 2

Masukan
<pre> 3 3 2 CUSTOM ..X .XX XXX  K K D D DD </pre>
Keluaran
Terminal:


File Teks(.txt)

File Gambar(.png)


#### 4.14 Test Case Custom 3

Masukan
3 3 2 CUSTOM

<pre> ..X .XX XXX  K K DD D DD </pre>
Keluaran
Terminal:
<pre> ===== Waktu pencarian: 0.048868 ms Banyak kasus yang ditinjau: 10 Tidak ada solusi yang ditemukan berdasarkan konfigurasi yang diberikan. </pre>
File Teks(.txt)
<pre> 4 Waktu pencarian: 0.048868 ms 3 Banyak kasus yang ditinjau: 10 kasus 2 1 Tidak ada solusi yang ditemukan. 5 </pre>
File Gambar(.png)
-

## BAB V LAMPIRAN

No	Poin	Ya	Tidak
1	Program berhasil dikompilasi tanpa kesalahan	✓	
2	Program berhasil dijalankan	✓	
3	Solusi yang diberikan program benar dan mematuhi aturan permainan	✓	
4	Program dapat membaca masukan berkas .txt serta menyimpan solusi dalam berkas .txt	✓	
5	Program memiliki <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	✓	

Repository Program: [https://github.com/V-Kleio/Tucill\\_13523146](https://github.com/V-Kleio/Tucill_13523146)

### **Pustaka**

Munir, R. (2025). *Algoritma brute force bagian 1*. Retrieved from [informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/02-Algoritma-Brute-Force-\(2025\)-Bag1.pdf](http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/02-Algoritma-Brute-Force-(2025)-Bag1.pdf)