LAPORAN TUGAS KECIL 3 STRATEGI ALGORITMA

Vincent Christian Siregar

13520136

I. Spesifikasi Program

Bahasa: Java 8

Link Github: https://github.com/Vincent136/Tucil3_15Puzzle.git

No.	Poin	Ya	Tidak
1.	Program berhasil kompilasi	✓	
2.	Program berhasil dijalankan	✓	
3.	Program dapat menerima masukan dan menuliskan keluaran.	✓	
4.	Luaran sudah benar untuk seluruh data uji.	✓	
5.	Bonus dibuat.	✓	

II. Cara Kerja Branch & Bound dalam Penyelasaian Puzzle 15

Pada program ini digunakan struktur data prio queue dan array 2 dimensi (representasi puzzle) dalam penerapan algoritma branch and bound untuk menyelesaikan 15 puzzle.

1. puzzle awal dicek kemustahilannya. Puzzle mustahil diselesaikan jika nilai $\sum kurang(i)$ merupakan bilangan genap.

*kurang(i) merupakan fungsi untuk mencari nilai kotak yang kurang dari i pada kotak setelah i sampai habis

2. jika puzzle mustahil diselesaikan maka proses berhenti. Jika puzzle tidak mustahil, lanjutkan ke langkah 3.

3. masukan puzzle awal (enqueue) ke prioqueue. Prioqueue diurutkan berdasarkan cost.

*cost ditentukan oleh seberapa banyak kotak yang tidak sesuai pada tempatnya.

4. dequeue puzzle dari prioqueue, jika puzzle merupakan solusi, maka proses berhenti. Jika puzzle bukan merupakan solusi, maka lanjutkan ke langkah 5.

5. jika puzzle bukan merupakan solusi, setiap command yang legal dan valid di jalan kan dan hasil puzzle baru dari setiap command dimasukkan ke dalam prioqueue.

*command legal dan valid jika merupakan salah satu dari command (“UP”,”RIGHT,”DOWN”,”LEFT”) dan tidak kembali ke state yang sudah pernah di cek

6. kembali ke langkah 4, dan begitu seterusnya sampai mendapat solusi.

III. Cara menjalankan program

Requirement: Java Development Kit (JDK) 8

1. masukkan puzzle yang ingin diselesaikan dalam file text pada folder test.

Contoh isi file test1.txt:

1 3 4 15

2 16 5 12

7 6 11 14

8 9 10 13

2. Jalankan Puzzle15_13520136.jar pada folder bin. (klik kanan pada .jar -> open with... -> Java(TM) Platform SE Binary)

3. Alternatif kedua untuk menjalankan program adalah dengan menjalankan command `java -jar Puzzle15_13520136.jar` pada command prompt pada folder bin.

4. Masukkan nama file pada textbox yagn tersedia.

5. tekan tombol start.

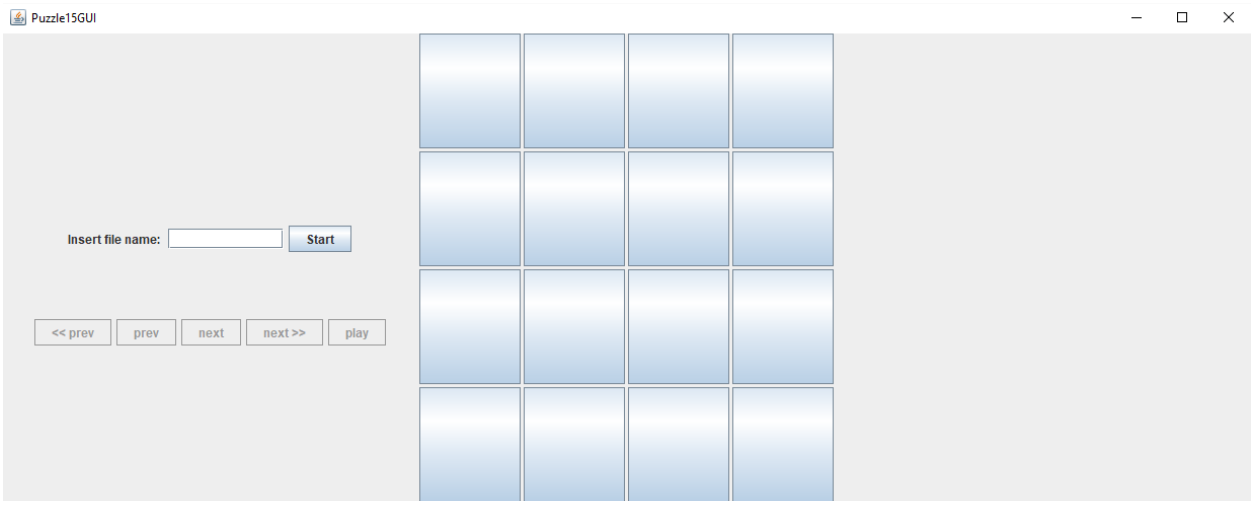
6. Tunggu proses hingga selesai (Waktu yang diperlukan bervariasi sesuai kerumitan puzzle).

7. Program akan menampilkan informasi pengecekan kemustahilan puzzle pada panel sebelah kanan pada program.

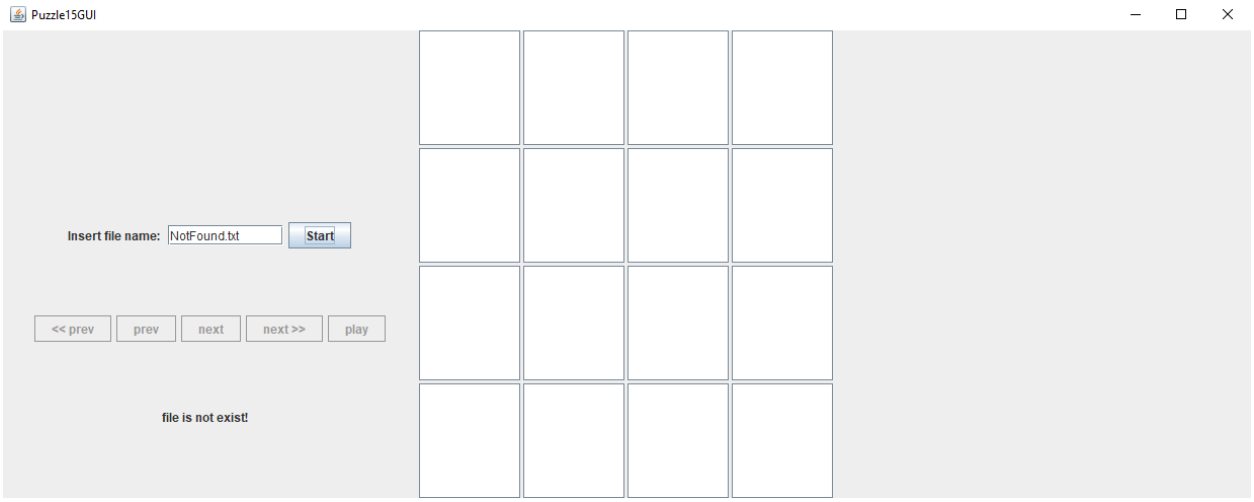
8. Tombol prev digunakan untuk melihat 1 step sebelum, tombol next digunakan untuk melihat 1 step setelah, dan tombol play digunakan untuk autoplay penyelesaian puzzle.

IV. Tes Program

Tampilan awal



Contoh File tidak ditemukan

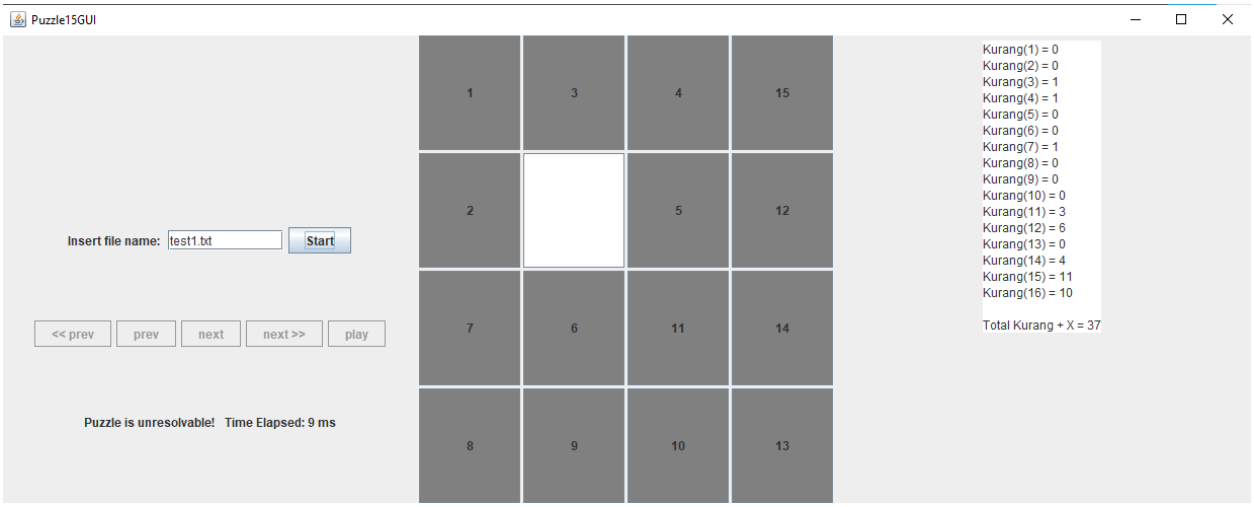


Test Case 1 (Mustahil)

test1.txt

```
1  3  4  15
2  16 5  12
7  6  11 14
8  9  10 13
```

Hasil Run

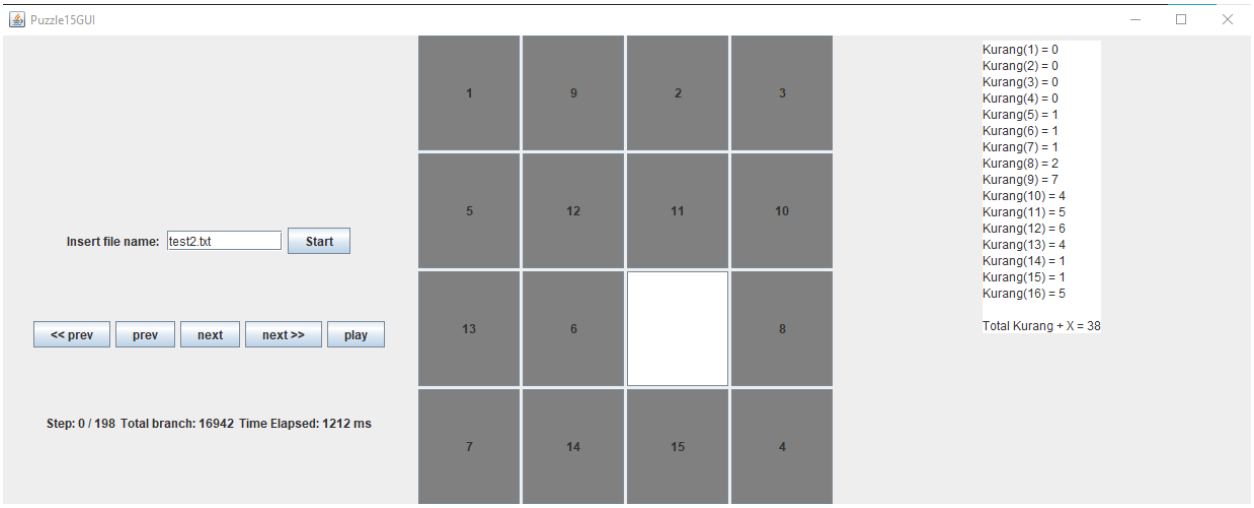


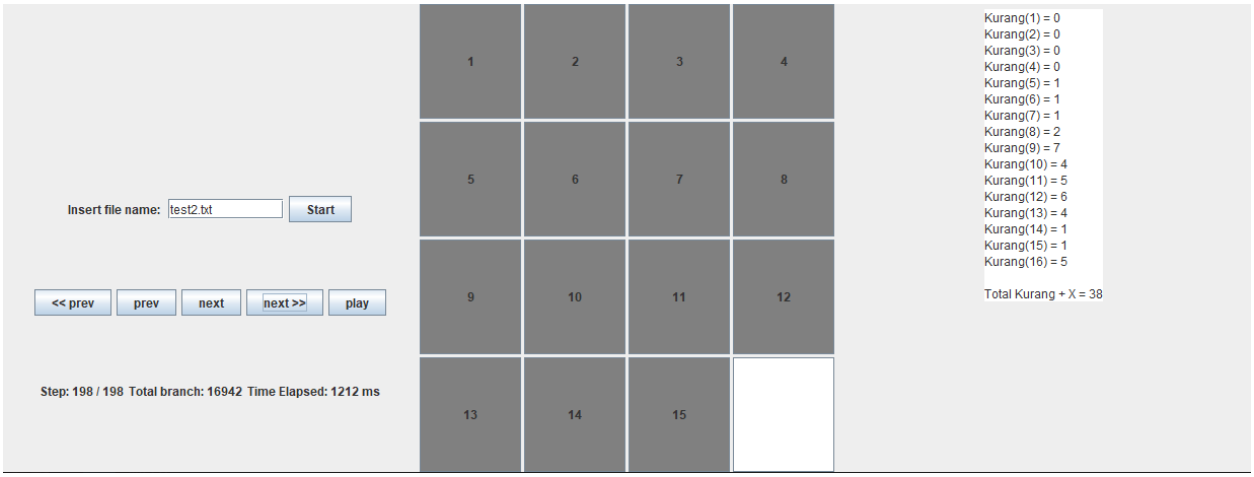
Test Case 2

test2.txt

```
1  9  2  3
5  12  11  10
13  6  16  8
7  14  15  4
```

Hasil Run



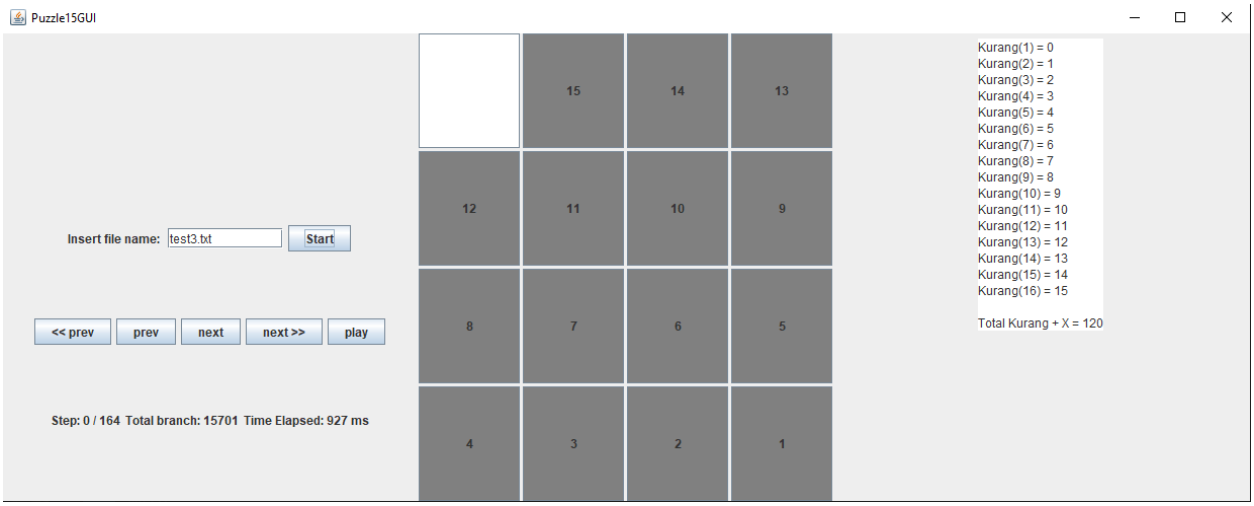


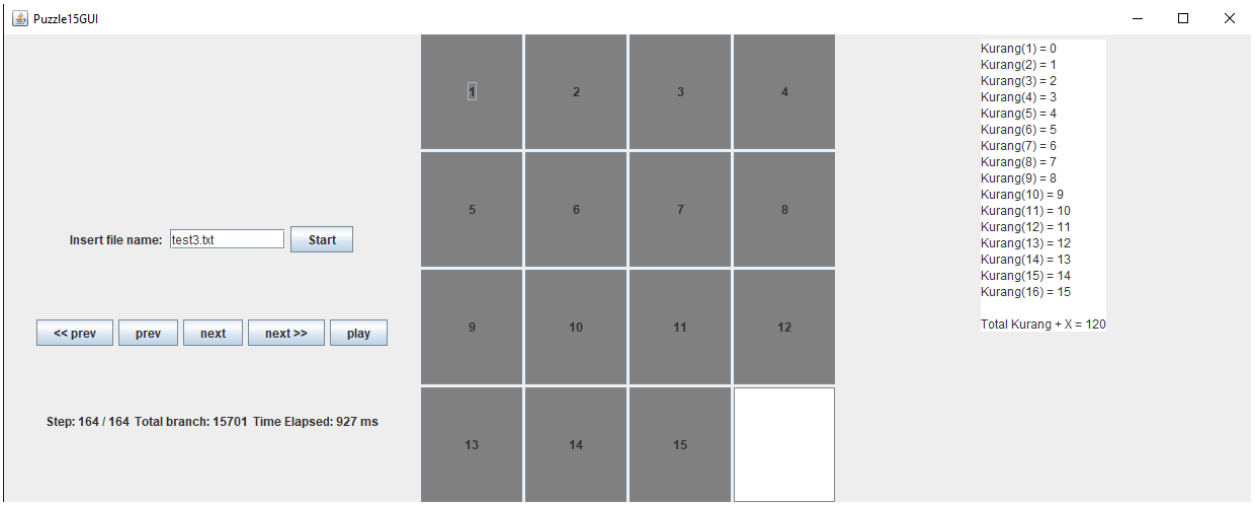
Test Case 3

test3.txt

```
16 15 14 13
12 11 10 9
8 7 6 5
4 3 2 1
```

Hasil Run



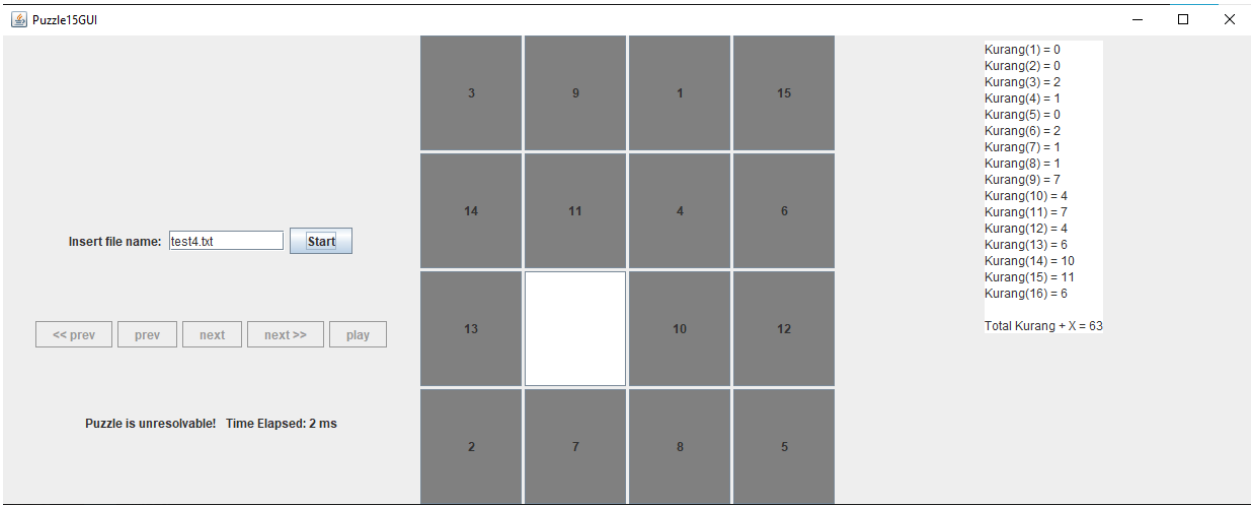


Test Case 4 (Mustahil)

test4.txt

```
3  9  1  15
14  11  4  6
13  16  10  12
2  7  8  5
```

Hasil Run

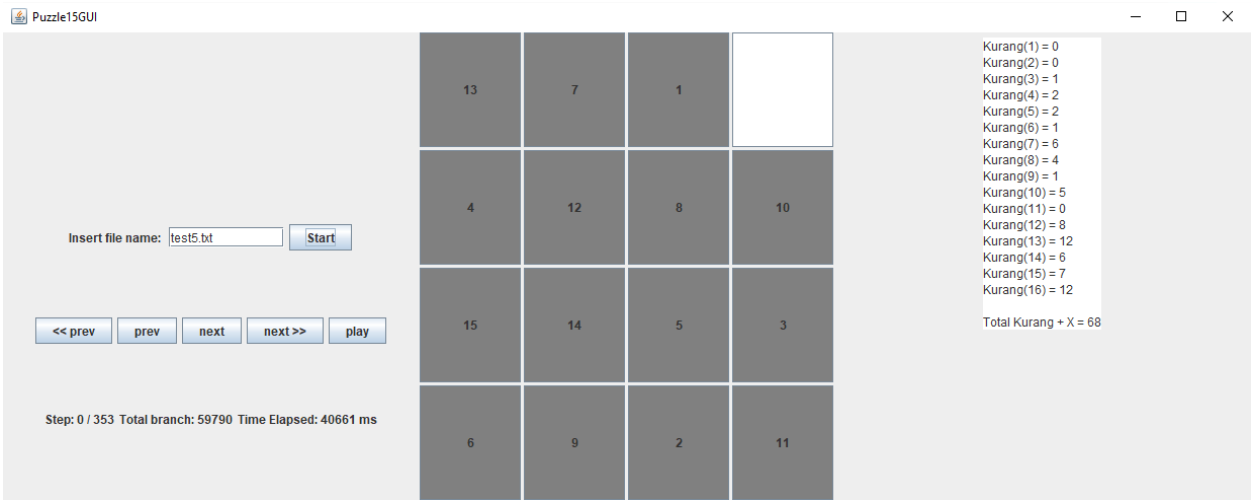


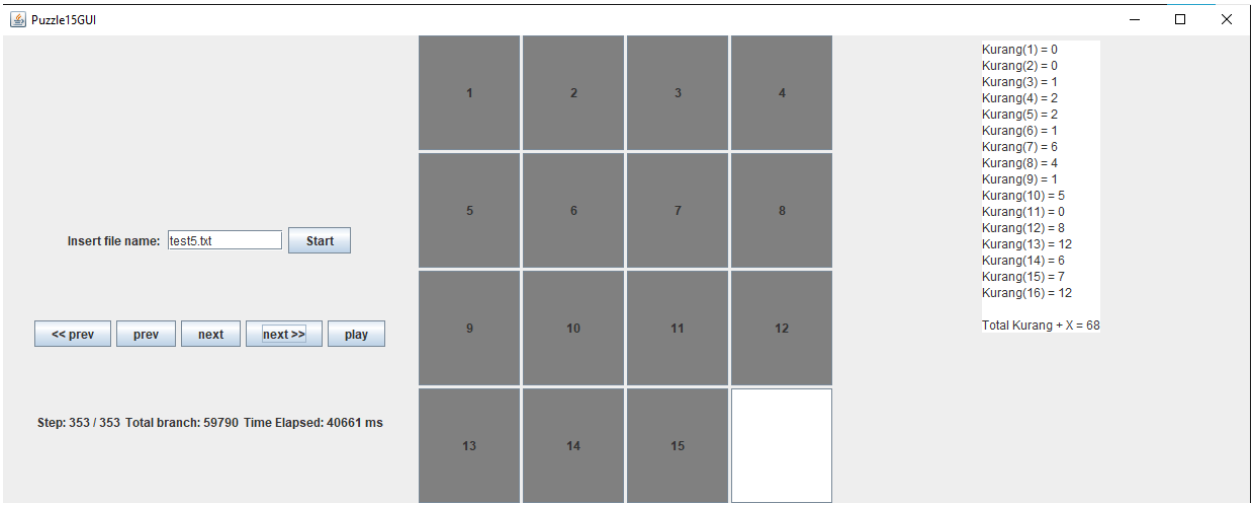
Test Case 5

test5.txt

```
13  7  1  16
4  12  8  10
15  14  5  3
6  9  2  11
```

Hasil Run





Kode Program

Puzzle.java

```
import java.io.*;
import java.util.*;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.stream.Collectors;

public class Puzzle {

    // atribut Puzzle
    public long id;
    public int[][] matrix = new int[4][4];
    private int row = 4;
    private int col = 4;
    public int row16;
    public int col16;
    private int cost;
    private int level;
    public String prevCommand;
    public Puzzle parent;
    public boolean isEmpty = false;

    // ctor Puzzle dengan input string filename dan counter (counter merupakan
variable global untuk menghitung jumlah living node)
    public Puzzle(String filename, long counter){

        String content = null;
        try {
            content = Files.lines(Paths.get("../test/"+ filename))
                    .collect(Collectors.joining(System.lineSeparator()));
            String[] splitLine = content.split("\r\n");

            for (int i = 0; i < 4; i++) {
                String[] spaceSplit = splitLine[i].split(" ");
                for (int j = 0; j < 4; j++) {
                    int number =Integer.parseInt(spaceSplit[j]);
                    if (number == 16) {
                        this.row16 = i;
                        this.col16 = j;
                    }
                    this.matrix[i][j] = number;
                }
```

```
            }

            this.level = 0;
            int GX = getGX();
            this.cost = level + GX;

            this.prevCommand = "-";
            this.id = counter;

            System.out.println("kelar");
        } catch (IOException e) {
            this.isEmpty = true;
        }
    }

    // ctor Puzzle dengan input puzzle parent dan command
("up","down","right","left") dan counter (counter merupakan variable global untuk
menghitung jumlah living node)
    public Puzzle(Puzzle p, String command, Long counter){
        if (command.equals("up")) {
            for(int i = 0; i < row; i++) {
                for (int j = 0; j < col; j++) {
                    this.matrix[i][j] = p.matrix[i][j];
                }
            }
            this.row16 = p.row16;
            this.col16 = p.col16;

            int temp = this.matrix[row16][col16];
            this.matrix[row16][col16] = this.matrix[row16-1][col16];
            this.matrix[row16-1][col16] = temp;

            this.row16--;
            this.prevCommand = "up";

        } else if (command.equals("right")) {
            for(int i = 0; i < row; i++) {
                for (int j = 0; j < col; j++) {
                    this.matrix[i][j] = p.matrix[i][j];
                }
            }
            this.row16 = p.row16;
            this.col16 = p.col16;

            int temp = this.matrix[row16][col16];
```

```
            this.matrix[row16][col16] = this.matrix[row16][col16+1];
            this.matrix[row16][col16+1] = temp;

            this.col16++;
            this.prevCommand = "right";

        } else if (command.equals("down")) {
            for(int i = 0; i < row; i++) {
                for (int j = 0; j < col; j++) {
                    this.matrix[i][j] = p.matrix[i][j];
                }
            }
            this.row16 = p.row16;
            this.col16 = p.col16;

            int temp = this.matrix[row16][col16];
            this.matrix[row16][col16] = this.matrix[row16+1][col16];
            this.matrix[row16+1][col16] = temp;

            this.row16++;
            this.prevCommand = "down";

        } else if (command.equals("left")) {
            for(int i = 0; i < row; i++) {
                for (int j = 0; j < col; j++) {
                    this.matrix[i][j] = p.matrix[i][j];
                }
            }
            this.row16 = p.row16;
            this.col16 = p.col16;

            int temp = this.matrix[row16][col16];
            this.matrix[row16][col16] = this.matrix[row16][col16-1];
            this.matrix[row16][col16-1] = temp;

            this.col16--;
            this.prevCommand = "left";
        }
        this.parent = p;
        this.id = counter;
        this.level = p.level+1;
        int GX = getGX();
        this.cost = GX;
    }
```

```java
    // fungsi untuk mengetahui apakah puzzle bisa diselesaikan parameter
outputkurang merupakan list dari kurang(i)
    public boolean checkPossible(List<Integer> outputKurang) {
        boolean Possible = true;
        int sigma = 0;
        for(int i = 0; i < row; i++) {
            for (int j = 0; j < col; j++) {
                int numberNow = matrix[i][j];
                int k = i;
                int l = j + 1;
                int kurang = 0;
                while (k < row) {
                    if (l == col) {
                        l = 0;
                        k++;
                    }
                    if (k < row) {
                        if (matrix[k][l] < numberNow) {
                            kurang += 1;
                        }
                        l++;
                    }
                }
                outputKurang.set(numberNow-1, kurang);
                sigma += kurang;
            }
        }
        int X = (row16 + col16)%2;
        Possible = ((sigma + X)%2) == 0;
        outputKurang.set(16, sigma + X);
        return Possible;
    }

    // fungsi mengecek apakah kotak kosong bisa digerakan ke atas
    public boolean checkUp() {
        return row16 != 0 && !prevCommand.equals("down");
    }

    // fungsi mengecek apakah kotak kosong bisa digerakan ke kanan
    public boolean checkRight() {
        return col16 != 3 && !prevCommand.equals("left");
    }

    // fungsi mengecek apakah kotak kosong bisa digerakan ke bawah
    public boolean checkDown() {
```

```
        return row16 != 3 && !prevCommand.equals("up");
    }

    // fungsi mengecek apakah kotak kosong bisa digerakan ke kiri
    public boolean checkLeft() {
        return col16 != 0 && !prevCommand.equals("right");
    }

    // Fungsi untuk mengecek apakah bentuk matrix sudah pernah di check
    public boolean checkState(List<Puzzle> state) {
        for (Puzzle p : state) {
            boolean isSame = true;
            for (int i = 0; i < 4; i++) {
                for (int j = 0; j < 4; j++) {
                    if (p.matrix[i][j] != this.matrix[i][j]) {
                        isSame = false;
                    }
                }
            }
            if (isSame) {
                return false;
            }
        }
        return true;
    }

    // fungsi untuk mendapatkan g(x) saat menentukan cost
    public int getGX() {
        int GX = 0;
        for (int i = 0; i < row; i++) {
            for (int j = 0; j < col; j++) {
                if (matrix[i][j] != 16) {
                    if (matrix[i][j] != i*4 + j + 1) {
                        GX++;
                    }
                }
            }
        }
        return GX;
    }

    // fungsi untuk mengecek apakah matrix sudah merupakan goal
    public boolean checkGoal() {
        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 4; j++) {
```

```
                if (matrix[i][j] != i*4 + j + 1) {
                    return false;
                }
            }
        }
        return true;
    }

    // getter untuk mendapatkan cost dari puzzle
    public int getCost() {
        return this.cost;
    }

    // fungsi untuk print puzzle pada cli (tidak digunakan dalam GUI)
    public void printPuzzle() {
        for (int i = 0; i < row; i++) {
            for (int j = 0; j < col; j++) {
                if (matrix[i][j] != 16) {
                    System.out.print(matrix[i][j] + " ");
                } else {
                    System.out.print("- ");
                }
            }
            System.out.println();
        }
    }
}
```

Prioqueue.java

```
import java.util.*;

// kelas prioqueue digunakan untuk menyimpan living node pada pencarian branch & bound
public class PrioQueue {
    // attribute
    public List<Puzzle> liveNode = new LinkedList<Puzzle>();

    // fungsi untuk memasukan puzzle ke dalam Prioqueue sesuai dengan prioritas (cost terkecil akan masuk pada index terendah)
    public void enQueue(Puzzle puzzle) {
        if (liveNode.isEmpty()) {
            liveNode.add(puzzle);
        } else {
            int i = 0;
```

```
            boolean found = false;
            for (Puzzle elmt : liveNode) {
                if (elmt.getCost() > puzzle.getCost()) {
                    found = true;
                    break;
                } else {
                    i++;
                }
            }
            if (found) {
                liveNode.add(i, puzzle);
            } else {
                liveNode.add(puzzle);
            }
        }
    }

    // fungsi untuk mengambil puzzle pada index terendah dari prioqueue
    public Puzzle deQueue() {
        Puzzle out = liveNode.get(0);
        liveNode.remove(0);
        return out;
    }
}
```

GUI.java

```
import java.util.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class GUI implements ActionListener {
    // attribut object yang digunakan pada GUI
    JFrame frame = new JFrame("Puzzle15GUI");
    JButton s1 = new JButton("");
    JButton s2 = new JButton("");
    JButton s3 = new JButton("");
    JButton s4 = new JButton("");
    JButton s5 = new JButton("");
    JButton s6 = new JButton("");
    JButton s7 = new JButton("");
    JButton s8 = new JButton("");
    JButton s9 = new JButton("");
    JButton s10 = new JButton("");
```

```
    JButton s11 = new JButton("");
    JButton s12 = new JButton("");
    JButton s13 = new JButton("");
    JButton s14 = new JButton("");
    JButton s15 = new JButton("");
    JButton s16 = new JButton("");

    JButton StartBtn =  new JButton("Start");
    JButton zeroButton;
    JButton prevButton;
    JButton nextButton;
    JButton lastButton;
    JButton playButton;

    JTextArea kurang = new JTextArea();
    JTextField FileName = new JTextField(10);

    JLabel label = new JLabel("Insert file name: ");
    JLabel Step = new JLabel();
    JLabel TotalBranch = new JLabel();
    JLabel TimeElapsed = new JLabel();

    JPanel PuzzlePanel = new JPanel();
    JPanel InputPanel = new JPanel();
    JPanel MainPanel = new JPanel();
    JPanel LeftPanel = new JPanel();
    JPanel RightPanel = new JPanel();
    JPanel PrevNextPanel = new JPanel();
    JPanel InfoPanel = new JPanel();

    java.util.List<Puzzle> output;
    java.util.List<Integer> outputKurang;
    int idxNow = 0;

    // ctor GUI untuk menyipakan objek objek dan menampilkan window GUI ke layar
    public GUI() {
        MainPanel.setLayout(new GridLayout(0,3));
        LeftPanel.setLayout(new GridLayout(5,0));
        InputPanel.add(label);
        InputPanel.add(FileName);
        InputPanel.add(StartBtn);

        LeftPanel.add(new JPanel());
        LeftPanel.add(new JPanel());
        LeftPanel.add(InputPanel);
```

```
        zeroButton = new JButton( new AbstractAction("<< prev") {
            @Override
            public void actionPerformed( ActionEvent e ) {
                idxNow = 0;
                Step.setText("Step: " + idxNow + " / " + (output.size()-1));
                refreshPuzzle();
            }
        });

        prevButton = new JButton( new AbstractAction("prev") {
            @Override
            public void actionPerformed( ActionEvent e ) {
                if (idxNow != 0) {
                    idxNow--;
                    Step.setText("Step: " + idxNow + " / " + (output.size()-1));
                }
                refreshPuzzle();
            }
        });

        nextButton = new JButton( new AbstractAction("next") {
            @Override
            public void actionPerformed( ActionEvent e ) {
                if (idxNow != output.size()-1) {
                    idxNow++;
                    Step.setText("Step: " + idxNow + " / " + (output.size()-1));
                }
                refreshPuzzle();
            }
        });

        lastButton = new JButton( new AbstractAction("next >>") {
            @Override
            public void actionPerformed( ActionEvent e ) {
                idxNow = output.size()-1;
                Step.setText("Step: " + idxNow + " / " + (output.size()-1));
                refreshPuzzle();
            }
        });

        playButton = new JButton( new AbstractAction("play") {
            @Override
            public void actionPerformed( ActionEvent e ) {
                new Thread(new Runnable() {
```

```
                    public void run() {
                        zeroButton.setEnabled(false);
                        prevButton.setEnabled(false);
                        nextButton.setEnabled(false);
                        lastButton.setEnabled(false);
                        playButton.setEnabled(false);

                        for (int i = idxNow + 1; i < output.size(); i++) {
                            idxNow = i;

                            // Runs inside of the Swing UI thread
                            SwingUtilities.invokeLater(new Runnable() {
                                public void run() {
                                    refreshPuzzle();
                                    Step.setText("Step: " + idxNow + " / " +
(output.size()-1));
                                }
                            });

                            try {
                                java.lang.Thread.sleep(100);
                            }
                            catch(Exception e) {

                            }
                        }
                        idxNow = output.size() -1;
                        zeroButton.setEnabled(true);
                        prevButton.setEnabled(true);
                        nextButton.setEnabled(true);
                        lastButton.setEnabled(true);
                        playButton.setEnabled(true);
                    }
                }).start();
            }

        });

        PrevNextPanel.add(zeroButton);
        PrevNextPanel.add(prevButton);
        PrevNextPanel.add(nextButton);
        PrevNextPanel.add(lastButton);
        PrevNextPanel.add(playButton);

        LeftPanel.add(PrevNextPanel);
```

```
        InfoPanel.add(Step);
        InfoPanel.add(TotalBranch);
        InfoPanel.add(TimeElapsed);

        LeftPanel.add(InfoPanel);

        PuzzlePanel.setLayout(new GridLayout(4,4,3,3));
        PuzzlePanel.add(s1);
        PuzzlePanel.add(s2);
        PuzzlePanel.add(s3);
        PuzzlePanel.add(s4);
        PuzzlePanel.add(s5);
        PuzzlePanel.add(s6);
        PuzzlePanel.add(s7);
        PuzzlePanel.add(s8);
        PuzzlePanel.add(s9);
        PuzzlePanel.add(s10);
        PuzzlePanel.add(s11);
        PuzzlePanel.add(s12);
        PuzzlePanel.add(s13);
        PuzzlePanel.add(s14);
        PuzzlePanel.add(s15);
        PuzzlePanel.add(s16);

        RightPanel.add(kurang);

        MainPanel.add(LeftPanel);
        MainPanel.add(PuzzlePanel);
        MainPanel.add(RightPanel);

        frame.add(MainPanel);

        zeroButton.setEnabled(false);
        prevButton.setEnabled(false);
        nextButton.setEnabled(false);
        lastButton.setEnabled(false);
        playButton.setEnabled(false);

        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        frame.setSize(1250, 500);
        frame.setVisible(true);

        StartBtn.addActionListener(this);
    }
```

```java
// fungsi utama ketika tombol start dijalankan
@Override
public void actionPerformed(ActionEvent e) {
    BranchnBound(FileName.getText());
}

// algoritma Branch n Bound untuk menyelesaikan puzzle
public void BranchnBound(String filename) {
    output =  new java.util.ArrayList<Puzzle>();
    Integer[] integers = new Integer[17];
    Arrays.fill(integers, 0);
    outputKurang = Arrays.asList(integers);

    clearPuzzle();

    kurang.setText(null);
    TimeElapsed.setText(null);
    Step.setText(null);
    TotalBranch.setText(null);

    idxNow = 0;
    long counter = 1;
    Puzzle p = new Puzzle(filename, counter);

    long start = System.currentTimeMillis();

    if(!p.isEmpty) {
        if (p.checkPossible(outputKurang)) {
            PrioQueue PQ = new PrioQueue();
            java.util.List<Puzzle> state = new ArrayList<Puzzle>();

            PQ.enQueue(p);

            boolean notFound = true;

            while (notFound) {
                Puzzle check = PQ.deQueue();
                state.add(check);
                if (check.checkGoal()) {
                    notFound = false;
                    getSolution(check);
                } else {
                    if (check.checkUp()) {
                        counter++;
```

```
                        Puzzle child = new Puzzle(check, "up", counter);
                        if (child.checkState(state)) {
                            PQ.enQueue(child);
                        } else {
                            counter--;
                        }
                    }
                    if (check.checkRight()) {
                        counter++;
                        Puzzle child = new Puzzle(check, "right", counter);
                        if (child.checkState(state)) {
                            PQ.enQueue(child);
                        } else {
                            counter--;
                        }
                    }
                    if (check.checkDown()) {
                        counter++;
                        Puzzle child = new Puzzle(check, "down", counter);
                        if (child.checkState(state)) {
                            PQ.enQueue(child);
                        } else {
                            counter--;
                        }
                    }
                    if (check.checkLeft()) {
                        counter++;
                        Puzzle child = new Puzzle(check, "left", counter);
                        if (child.checkState(state)) {
                            PQ.enQueue(child);
                        } else {
                            counter--;
                        }
                    }
                }
            }

            zeroButton.setEnabled(true);
            prevButton.setEnabled(true);
            nextButton.setEnabled(true);
            lastButton.setEnabled(true);
            playButton.setEnabled(true);
            TotalBranch.setText("Total branch: " + counter);
            Step.setText("Step: " + idxNow + " / " + (output.size() - 1) +
"");
```

```
                refreshPuzzle();

            } else {
                output.add(p);

                zeroButton.setEnabled(false);
                prevButton.setEnabled(false);
                nextButton.setEnabled(false);
                lastButton.setEnabled(false);
                playButton.setEnabled(false);

                Step.setText("Puzzle is unresolvable!");
                TotalBranch.setText(null);
                refreshPuzzle();
            }
            for (int i = 0; i < 16; i++) {
                kurang.append("Kurang(" + (i+1) + ") = " + outputKurang.get(i) +
"\r\n");
            }
            kurang.append("\r\n");
            kurang.append("Total Kurang + X = " + outputKurang.get(16));

            TimeElapsed.setText("Time Elapsed: " + (System.currentTimeMillis() -
start) + " ms");
        } else {
            zeroButton.setEnabled(false);
            prevButton.setEnabled(false);
            nextButton.setEnabled(false);
            lastButton.setEnabled(false);
            playButton.setEnabled(false);

            Step.setText("file is not exist!");
            TotalBranch.setText(null);
        }
    }

    // fungsi untuk menambahkan solusi pada list output (digunakan pada fungsi
BranchnBound)
    public void getSolution(Puzzle check) {
        if (check.parent == null) {
            output.add(check);
        } else {
            getSolution(check.parent);
            output.add(check);
        }
```

```
    }

    // fungsi untuk mereset tampilan puzzle
    public void clearPuzzle() {
        s1.setText("");
        s1.setBackground(Color.white);
        s2.setText("");
        s2.setBackground(Color.white);
        s3.setText("");
        s3.setBackground(Color.white);
        s4.setText("");
        s4.setBackground(Color.white);
        s5.setText("");
        s5.setBackground(Color.white);
        s6.setText("");
        s6.setBackground(Color.white);
        s7.setText("");
        s7.setBackground(Color.white);
        s8.setText("");
        s8.setBackground(Color.white);
        s9.setText("");
        s9.setBackground(Color.white);
        s10.setText("");
        s10.setBackground(Color.white);
        s11.setText("");
        s11.setBackground(Color.white);
        s12.setText("");
        s12.setBackground(Color.white);
        s13.setText("");
        s13.setBackground(Color.white);
        s14.setText("");
        s14.setBackground(Color.white);
        s15.setText("");
        s15.setBackground(Color.white);
        s16.setText("");
        s16.setBackground(Color.white);
    }

    // fungsi untuk refresh tampilan puzzle sesuai dengan step
    public void refreshPuzzle() {
        refreshSlot(s1, output.get(idxNow).matrix[0][0]);
        refreshSlot(s2, output.get(idxNow).matrix[0][1]);
        refreshSlot(s3, output.get(idxNow).matrix[0][2]);
        refreshSlot(s4, output.get(idxNow).matrix[0][3]);
        refreshSlot(s5, output.get(idxNow).matrix[1][0]);
```

```
        refreshSlot(s6, output.get(idxNow).matrix[1][1]);
        refreshSlot(s7, output.get(idxNow).matrix[1][2]);
        refreshSlot(s8, output.get(idxNow).matrix[1][3]);
        refreshSlot(s9, output.get(idxNow).matrix[2][0]);
        refreshSlot(s10, output.get(idxNow).matrix[2][1]);
        refreshSlot(s11, output.get(idxNow).matrix[2][2]);
        refreshSlot(s12, output.get(idxNow).matrix[2][3]);
        refreshSlot(s13, output.get(idxNow).matrix[3][0]);
        refreshSlot(s14, output.get(idxNow).matrix[3][1]);
        refreshSlot(s15, output.get(idxNow).matrix[3][2]);
        refreshSlot(s16, output.get(idxNow).matrix[3][3]);
    }

    // fungsi pembantu pada refresh puzzle
    public void refreshSlot(JButton btn,int input) {
        if (input != 16) {
            btn.setText(Integer.toString(input));
            btn.setBackground(Color.gray);
        } else {
            btn.setText("");
            btn.setBackground(Color.white);
        }
    }

    // fungsi main yang dijalankan ketika program di run
    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                new GUI();
            }
        });
    }
}
```