LAPORAN TUGAS KECIL 3

IF2211 STRATEGI ALGORITMA

SEMESTER II TAHUN 2021/2022

Persoalan 15-Puzzle dengan Algoritma Branch and Bound

Raka Wirabuana Ninagan || 13520134 || K-02

Tugas ini merupakan tugas untuk mengimplementasikan algoritma *branch & bound*, berupa pembuatan permainan 15-puzzle solver.

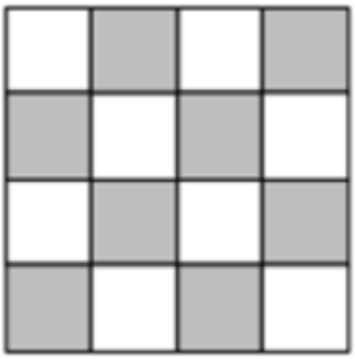
1. Algoritma Branch & Bound pada 15-puzzle

15-puzzle merupakan permainan yang direpresentasikan oleh 15 angka dari 1-15 dan disusun dalam matriks 4x4 seperti di bawah ini.

Gambar contoh 15-puzzle

Penerapan algoritma Branch & Bound pada 15-puzzle dilakukan dengan cara membaginya menjadi 2 permasalahan, yaitu bisa/tidaknya puzzle diselesaikan dan algoritma penentuan langkah puzzle (proses penyelesaian).

Pada permasalahan pertama, digunakan teorema yang mengandung pernyataan “Status tujuan hanya dapat dicapai dari status awal jika $\sum_{i=1}^{16} KURANG(i) + X$ bernilai genap”. Fungsi KURANG(i) adalah fungsi yang menyatakan banyaknya angka yang lebih besar dari i tetapi berada di posisi di atas i (penafsiran pribadi). Nilai X adalah nilai yang dihasilkan berdasarkan posisi petak kosong berada. Nilai yang dihasilkan mengikuti aturan pada gambar ini.



Gambar pemetaan nilai-X

1 untuk abu, 0 untuk putih

Hasil penjumlahan dari total penjumlahan KURANG(i) untuk i = 1, 2, 3,…,16 dengan X akan memberikan nilai di antara genap atau ganjil. Apabila hasilnya ganjil, status tujuan tidak dapat dicapai. Sebaliknya, apabila genap, status tujuan bisa dicapai dan proses bisa dilanjutkan ke tahap penyelesaian puzzle.

Langkah pertama yang dilakukan adalah melakukan gerakan ke semua arah yang memungkinkan di posisi saat ini (di pojok hanya 2 arah, di sisi hanya 3 arah, di 4 petak tengah bisa 4 arah). Semua kemungkinan posisi setelah pergerakan dihitung berapa costnya dan disimpan ke list status yang dibangkitkan. Cost merupakan nilai yang didapatkan dengan cara menghitung seberapa jauh petak kosong sudah bergerak dan seberapa jauh lagi perkiraan status tujuan bisa dicapai. Perkiraan status tujuan dihitung dengan melihat banyaknya posisi angka yang tidak pada tempatnya. Makin besar angka perkiraan ini, makin jauh juga posisi saat ini untuk bisa mencapai status tujuan.

Status berikutnya yang dipilih adalah status dengan cost paling kecil dan pergerakan petak kosong paling banyak (cost diprioritaskan). Proses ini terus dilakukan ke setiap status yang dipilih sampai status tujuan berhasil dicapai.

2. *Source Program*

Implementasi Algoritma (kelas puzzle15)

```
// import java.util.Scanner;
import java.util.*;
class puzzle15{
    private int[][] puzzle;
    private char[] moveDirection;
    private boolean usingFile;
    private int Xabsis;
    private int Xordinat;
    private int[][] gridcolor;
```

```
    public ArrayList<PuzzleState> possiblePath;
    public Deque<Integer> solutionpath;
    public int[] kurangItable;

    // Khusus untuk visualisasi di GUI
    public double time;
    public int visualidx;
    public int kurangI;
    public int X;

    // Konstruktor kelas puzzle15
    public puzzle15(boolean uF, String file){
        this.puzzle = new int[4][4];
        this.usingFile = uF;
        this.possiblePath = new ArrayList<>();
        this.solutionpath = new ArrayDeque<>();
        this.Xabsis = -1;
        this.Xordinat = -1;
        this.moveDirection = new char[]{'U','R','D','L'};
        this.kurangItable = new int[16];
        this.time = 0.0000;
        this.visualidx = 0;
        this.gridcolor = new int[][]
        {{0,1,0,1},
        {1,0,1,0},
        {0,1,0,1},
        {1,0,1,0}};

        if (this.usingFile){ // Input persoalan melalui file
            this.puzzle = FileProcess.matrixProcessing(file);
        }
        else{ // Persoalan dibuat secara acak menggunakan random()
            int[] alreadyGenerated = new int[16];
            for (int i =  0; i < this.puzzle.length; i++){
                for (int j =  0; j < this.puzzle[0].length; j++){
                    int generatedInt = (int)(Math.random()*16);
                    while (alreadyGenerated[generatedInt] == 1){
                        generatedInt = (int)(Math.random()*16);
                    }
                    this.puzzle[i][j] = generatedInt;
                    alreadyGenerated[generatedInt] = 1;
                }
            }
        }
    }
```

```
    public int[][] getPuzzle(){
        return this.puzzle;
    }

    // Method untuk tampilkan matriks puzzle
    public void displayPuzzle(int[][] puzzle){
        for (int i =  0; i < puzzle.length; i++){
            for (int j =  0; j < puzzle[0].length; j++){
                System.out.print(puzzle[i][j]);
                if (j != puzzle[0].length - 1){
                    System.out.print("\t");
                }
            }
            System.out.println("");
        }
    }

    // Method implementasi KURANG(i)
    private int kurang(int[][] onepuzzle){
        int kurangI = 0;
        int[] puzzleseq = matToArr(onepuzzle);
        for (int i = 0; i < puzzleseq.length; i++){
            int kItablevis = 0;
            for (int j = i + 1; j < puzzleseq.length; j++){
                if (puzzleseq[i] > puzzleseq[j] && puzzleseq[j] != 0 &&
puzzleseq[i] != 0){
                    kurangI++;
                    kItablevis++;
                }
                if (puzzleseq[i] == 0){
                    kurangI++;
                    kItablevis++;
                }
            }
            if (puzzleseq[i] == 0) {
                this.kurangItable[15] = kItablevis;
            }
            else {
                this.kurangItable[puzzleseq[i] - 1] = kItablevis;
            }
        }
        return kurangI;
    }
```

```java
    // Method ubah matriks menjadi array
    // Khusus digunakan untuk perhitungan KURANG(i)
    private int[] matToArr(int[][] onepuzzle){
        int size = onepuzzle.length * onepuzzle.length;
        int k = 0;
        int[] sequence = new int[size];
        for (int i =  0; i < onepuzzle.length; i++){
            for (int j =  0; j < onepuzzle[0].length; j++){
                if (onepuzzle[i][j] == 0){
                    this.Xabsis = i;
                    this.Xordinat = j;
                }
                sequence[k] = onepuzzle[i][j];
                k++;
            }
        }
        return sequence;
    }

    // Method untuk menentukan nilai X
    private int valueX(int i, int j){
        return (this.gridcolor[i][j]);
    }

    // Method untuk menentukan bisa tidaknya puzzle diselesaikan
    // Mengembalikan true apabila bisa
    // Mengembalikan false apabila tidak bisa
    public boolean isReachable(int[][] onepuzzle){
        this.kurangI = this.kurang(onepuzzle);
        this.X = this.valueX(this.Xabsis, this.Xordinat);

        int theoremresult = this.kurangI;
        theoremresult += this.X;
        if (theoremresult % 2 == 0){
            return true;
        }
        return false;

    }

    // Method untuk memulai penyelesaian puzzle
    // dengan cara mencatat koordinat 0
    // dan membangkitkan antrian child pertama
    private void solveInitial(){
        for (int i =  0; i < this.puzzle.length; i++){
```

```
            for (int j =  0; j < this.puzzle[0].length; j++){
                if (this.puzzle[i][j] == 0){
                    this.Xabsis = i;
                    this.Xordinat = j;
                }
            }
        }
        PuzzleState firstPuzzle = new PuzzleState(this.puzzle, -999, this.Xabsis,
this.Xordinat, 0, 'X', -1, false);
        firstPuzzle.alreadyChoosed = true;
        this.possiblePath.add(firstPuzzle);
        this.move(this.puzzle, this.Xabsis, this.Xordinat, 1, 'X', 0, 0);
    }

    // Method untuk melakukan pencarian setelah antrian
    // child yang pertama dibangkitkan
    private PuzzleState solveTheRest(){
        boolean finished = false;
        PuzzleState lastPS = new PuzzleState();
        while (!finished){
            PuzzleState chosenp = new PuzzleState();
            int maxDepth = 0;
            int pPathIdx = 0;
            for (int i = 0; i < this.possiblePath.size(); i++){
                if (maxDepth < this.possiblePath.get(i).depth &&
(!this.possiblePath.get(i).alreadyChoosed)){
                    chosenp.setPuzzleState(this.possiblePath.get(i));
                    pPathIdx = i;
                }
            }
            for (int k = 0; k < this.possiblePath.size(); k++){
                int cost = this.possiblePath.get(k).cost;
                if ((cost != -999) &&
(!this.possiblePath.get(k).alreadyChoosed)){
                    if (chosenp.cost > cost){
                        this.possiblePath.get(k).alreadyChoosed = true;
                        chosenp.setPuzzleState(this.possiblePath.get(k));
                        pPathIdx = k;
                    }
                    else if (chosenp.cost == cost){
                        if (chosenp.depth < this.possiblePath.get(k).depth){
                            this.possiblePath.get(k).alreadyChoosed = true;
                            chosenp.setPuzzleState(this.possiblePath.get(k));
                            pPathIdx = k;
                        }
```

```
                    }
                }
            } // Puzzle dengan cost paling kecil sudah dipilih

            // Memastikan apakah puzzle yang dipilih sudah memenuhi solusi
            if (pToSolution(chosenp) == 0){
                finished = true;
                lastPS.setPuzzleState(chosenp);
                this.solutionpath.push(pPathIdx);
            }
            else{
//              System.out.println("Chosen : ");
//                displayPuzzle(chosenp.instancepuzzle);
                this.move(chosenp.instancepuzzle, chosenp.x0, chosenp.y0, 1,
chosenp.prevMove, chosenp.depth, pPathIdx);
            }
        }
        return lastPS;
    }
    public puzzle15 solve(){
        System.out.println("Solving...");
        // Mulai waktu pencarian
        long start = System.nanoTime();

        // Lakukan pencarian
        solveInitial();
        PuzzleState finalState = solveTheRest();

        // Hentikan dan catat waktu pencarian
        long elapsedTime = System.nanoTime() - start;
        this.time = elapsedTime / 1_000_000_000.0;

        // Menuliskan indeks dari jalan start menuju solusi puzzle
        this.writePathIdx(finalState);
        System.out.println("Puzzle Solved!");
        return this;
    }

    // Method untuk mencatat indeks solusi
    // Dilakukan rekursi dari daun ke akar pohon
    private void writePathIdx(PuzzleState ps){
        // Melakukan pencatatan indeks solusi
        // dengan syarat indeks != -1
        if (ps.prevStateIdx != -1){
            this.solutionpath.push(ps.prevStateIdx);
```

```
            this.writePathIdx(this.possiblePath.get(ps.prevStateIdx));
        }

    }

    // Method untuk memeriksa apakah puzzle yang dibangkitkan
    // sudah pernah ada di possible path
    public boolean isPuzzleInPP(int[][] a) {
        int p = 0;
        while (p < this.possiblePath.size()) {
            int[][] b = this.possiblePath.get(p).instancepuzzle;
            for (int i = 0; i < 4; i++) {
                for (int j = 0; j < 4; j++) {
                    if(a[i][j] != b[i][j]) {
                        return false;
                    }
                }
            }
        }
        return true;
    }

    // Method untuk menghitung cost
    private int cost(PuzzleState puzzle){
        return rootToP(puzzle) + pToSolution(puzzle);
    }

    // Method untuk menghitung f(P) dari permasalahan puzzle
    // f(P) adalah jarak dari akar ke node P
    private int rootToP(PuzzleState puzzle){
        return puzzle.depth;
    }

    // Method untuk menghitung g(P) dari permasalahan puzzle
    // g(P) adalah estimasi jarak dari P ke solusi yang diharapkan
    private int pToSolution(PuzzleState puzzle){
        int k = 1; // Value acuan posisi puzzle
        int gridstillwrong = 0; // Jumlah posisi yang tidak tepat
        for (int i =  0; i < puzzle.instancepuzzle.length; i++){
            for (int j =  0; j < puzzle.instancepuzzle[0].length; j++){
                if (puzzle.instancepuzzle[i][j] == 0){ // Penanganan khusus untuk
value nol
                    // Gagalkan pengecekan ketidakcocokan
                }
```

```
                else if (puzzle.instancepuzzle[i][j] != k){ // Value di [i][j]
tidak tepat
                    gridstillwrong++;
                }
                k++;
            }
        }
        return gridstillwrong;
    }

    // Method untuk membangkitkan semua kemungkinan langkah
    private void move(int[][] puzzle, int zeroX, int zeroY, int zeroMov, char
lastMovement, int lastLevel, int thisidx){
        int[][] arisedstate = new int[4][4];
        for (int i =  0; i < arisedstate.length; i++){
            for (int j =  0; j < arisedstate[0].length; j++){
                arisedstate[i][j] = puzzle[i][j];
            }
        }
        int x = zeroX;
        int y = zeroY;
        int temp = puzzle[x][y];
        int currentZM = lastLevel + zeroMov;

        for (int i = 0; i < this.moveDirection.length; i++){
            switch(this.moveDirection[i]) {
                case 'U':
                    if (x != 0 && lastMovement != 'D'){
                        // Lakukan pergerakan 0
                        arisedstate[x][y] = arisedstate[x-1][y];
                        arisedstate[x-1][y] = temp;

                        if (!isPuzzleInPP(arisedstate)) {
                            // Bangkitkan puzzle dengan posisi baru
                            // Beserta informasi relevan seperti cost, posisi x
dan y nya 0
                            PuzzleState possiblestate;
                            possiblestate = new PuzzleState(arisedstate, 0, x-1,
y, currentZM, 'U', thisidx, false);
                            possiblestate.cost = cost(possiblestate);
                            this.possiblePath.add(possiblestate);
                        }
                        // Kembalikan ke kondisi semula
                        temp = arisedstate[x-1][y];
                        arisedstate[x-1][y] = arisedstate[x][y];
```

```
                        arisedstate[x][y] = temp;
                    }
                    break;
                case 'R':
                    if (y != arisedstate[0].length - 1  && lastMovement != 'L'){
                        arisedstate[x][y] = arisedstate[x][y+1];
                        arisedstate[x][y+1] = temp;

                        if (!isPuzzleInPP(arisedstate)) {
                            // Bangkitkan puzzle dengan posisi baru
                            // Beserta informasi relevan seperti cost, posisi x
dan y nya 0
                            PuzzleState possiblestate;
                            possiblestate = new PuzzleState(arisedstate, 0, x,
y+1, currentZM, 'R', thisidx, false);
                            possiblestate.cost = cost(possiblestate);
                            this.possiblePath.add(possiblestate);
                        }

                        temp = arisedstate[x][y+1];
                        arisedstate[x][y+1] = arisedstate[x][y];
                        arisedstate[x][y] = temp;
                    }
                    break;
                case 'D':
                    if (x != arisedstate.length - 1 && lastMovement != 'U'){
                        arisedstate[x][y] = arisedstate[x+1][y];
                        arisedstate[x+1][y] = temp;

                        if (!isPuzzleInPP(arisedstate)) {
                            // Bangkitkan puzzle dengan posisi baru
                            // Beserta informasi relevan seperti cost, posisi x
dan y nya 0
                            PuzzleState possiblestate;
                            possiblestate = new PuzzleState(arisedstate, 0, x+1,
y, currentZM, 'D', thisidx, false);
                            possiblestate.cost = cost(possiblestate);
                            this.possiblePath.add(possiblestate);
                        }

                        temp = arisedstate[x+1][y];
                        arisedstate[x+1][y] = arisedstate[x][y];
                        arisedstate[x][y] = temp;
                    }
                    break;
```

```
                case 'L':
                    if (y != 0 && lastMovement != 'R'){
                        arisedstate[x][y] = arisedstate[x][y-1];
                        arisedstate[x][y-1] = temp;

                        if (!isPuzzleInPP(arisedstate)) {
                            // Bangkitkan puzzle dengan posisi baru
                            // Beserta informasi relevan seperti cost, posisi x
dan y nya 0
                            PuzzleState possiblestate;
                            possiblestate = new PuzzleState(arisedstate, 0, x, y-
1, currentZM, 'L', thisidx, false);
                            possiblestate.cost = cost(possiblestate);
                            this.possiblePath.add(possiblestate);
                        }

                        temp = arisedstate[x][y-1];
                        arisedstate[x][y-1] = arisedstate[x][y];
                        arisedstate[x][y] = temp;
                    }
                    break;
                default:
                    break;
            }
        }
    }

//    public static void main(String[] args) {
//        puzzle15 test = new puzzle15(true);
//        if (test.isReachable(test.puzzle)){
//            test.solve();
//        }
//        else{
//            System.out.println("Unreachable");
//        }
//    }
}

// Kelas untuk menampung data hasil pembangkitan
// Bertujuan untuk efisiensi waktu algoritma
class PuzzleState{
    public int[][] instancepuzzle;
    public int cost;
    public int x0;
    public int y0;
```

```
    public int depth;
    public char prevMove;
    public int prevStateIdx;
    public boolean alreadyChoosed;

    public PuzzleState(){
        this.instancepuzzle = new int[4][4];
        this.cost = 0;
        this.x0 = -1;
        this.y0 = -1;
        this.depth = 0;
        this.prevMove = 'X';
        this.prevStateIdx = -1;
        this.alreadyChoosed = false;
    }
    public PuzzleState(int[][] p, int c, int x, int y, int d, char m, int psi,
boolean ac){
        this.instancepuzzle = new int[4][4];
        this.cost = c;
        this.x0 = x;
        this.y0 = y;
        this.depth += d;
        this.prevMove = m;
        this.prevStateIdx = psi;
        this.alreadyChoosed = ac;
        for (int i =  0; i < this.instancepuzzle.length; i++){
            for (int j =  0; j < this.instancepuzzle[0].length; j++){
                this.instancepuzzle[i][j] = p[i][j];
            }
        }
    }
    public void setPuzzleState(PuzzleState ps){
        this.instancepuzzle = new int[4][4];
        this.cost = ps.cost;
        this.x0 = ps.x0;
        this.y0 = ps.y0;
        this.depth = ps.depth;
        this.prevMove = ps.prevMove;
        this.prevStateIdx = ps.prevStateIdx;
        this.alreadyChoosed = ps.alreadyChoosed;

        for (int i =  0; i < this.instancepuzzle.length; i++){
            for (int j =  0; j < this.instancepuzzle[0].length; j++){
                this.instancepuzzle[i][j] = ps.instancepuzzle[i][j];
            }
```

```
        }
    }
}
```

Implementasi pemrosesan file

```
import java.io.FileReader;
import java.io.IOException;

public class FileProcess {
    public static int[][] matrixProcessing(String nameFile){
        String firstStep = fileInput(nameFile);
        int[][] matrixstr = strToMat(firstStep);

        return matrixstr;
    }

    public static String fileInput(String strFile)
    {
        String strConv = "";
        String namaFile = "../test/" + strFile;
        try {
            FileReader fRead = new FileReader(namaFile);

            int ch;
            while ((ch = fRead.read()) != -1) {
                strConv += (char)ch;
            }
            fRead.close();
        }
        catch (IOException e) {
            System.out.println("Pembacaan file masukan error.");
        }
        return strConv;
    }

    public static int[][] strToMat(String mat){
        String[] hasilSplit = mat.split("\n");
        int[][] matrix = new int[4][4];
        for (int i = 0; i < 4; i++){
            String[] arr2 = hasilSplit[i].split(" ");
            for (int j = 0; j < 4; j++){
                matrix[i][j] = strToInt(arr2[j]);
            }
        }
```

```
        return matrix;
    }

    public static int[][] convertToInt(String[][] matrixstr){
        int[][] matrixint = new int[4][4];
        for (int i = 0; i < 4; i++){
            for (int j = 0; j < 4; j++){
                matrixint[i][j] = Integer.valueOf(matrixstr[i][j]);
            }
        }

        return matrixint;
    }

    private static int strToInt(String numstr){
        int intresult = 0;
        numstr = numstr.replace("\n", "").replace("\r", "");
        int strlen = numstr.length();
        if(strlen == 1){
            intresult = numstr.charAt(0);
            intresult -= 48;
        }
        else{
            for (int i = 0; i < strlen; i++){
                char tempchar = numstr.charAt(i);
                int tempint = tempchar;
                tempint -= 48;
                intresult = intresult + tempint * (int)Math.pow(10, strlen - 1 -
i);

            }
        }
        return intresult;
    }
}
```

Program GUI (menggunakan Eclipse)

```
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JTextField;
import javax.swing.JButton;
import javax.swing.JCheckBox;
```

```java
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.JPanel;
import javax.swing.JLabel;
import java.awt.GridLayout;
import java.awt.Image;

import javax.swing.border.BevelBorder;
import javax.swing.text.SimpleAttributeSet;
import javax.swing.text.StyleConstants;
import javax.swing.text.StyledDocument;

import java.awt.Font;
import javax.swing.JTextPane;
import java.awt.SystemColor;
import java.awt.Toolkit;
import java.awt.Label;

public class Frame1 {

    private JFrame frame;
    private JTextField textField;
    private JLabel lblNewLabel;
    private JButton solveButton;
    private JCheckBox chckbxNewCheckBox;
    private JPanel panel_1;
    private JPanel panel_2;
    private JPanel panel_3;
    private JPanel panel_4;
    private JPanel panel_5;
    private JPanel panel_6;
    private JPanel panel_7;
    private JPanel panel_8;
    private JPanel panel_9;
    private JPanel panel_10;
    private JPanel panel_11;
    private JPanel panel_12;
    private JPanel panel_13;
    private JPanel panel_14;
    private JPanel panel_15;
    private JPanel panel_16;
    private JPanel panel_17;
    private JPanel panel_18;
    private JPanel panel_19;
    private JPanel panel_20;
```

```
    private JPanel panel_21;
    private JPanel panel_22;
    private JPanel panel_23;
    private JPanel panel_24;
    private JPanel panel_25;
    private JPanel panel_26;
    private JPanel panel_27;
    private JPanel panel_28;
    private JTextPane textBoxMat;
    private JTextPane textBoxMat_1;
    private JTextPane textBoxMat_2;
    private JTextPane textBoxMat_3;
    private JTextPane textBoxMat_4;
    private JTextPane textBoxMat_5;
    private JTextPane textBoxMat_6;
    private JTextPane textBoxMat_7;
    private JTextPane textBoxMat_8;
    private JTextPane textBoxMat_9;
    private JTextPane textBoxMat_10;
    private JTextPane textBoxMat_11;
    private JTextPane textBoxMat_12;
    private JTextPane textBoxMat_13;
    private JTextPane textBoxMat_14;
    private JTextPane textBoxMat_15;

    private puzzle15 puzzleResult;
    private Label label;
    private JTextField timeTextField;
    private JLabel lblarisedNode;
    private JTextField nodeTextField;
    private JLabel lblSec;
    private JTextField kurangXTextField;
    private boolean fileError;
    private boolean reachable;

    // Special function for visual center align
    public void setToCenter(JTextPane tbm) {
        StyledDocument th1 = tbm.getStyledDocument();
        SimpleAttributeSet centerN1 = new SimpleAttributeSet();
        StyleConstants.setAlignment(centerN1, StyleConstants.ALIGN_CENTER);
        th1.setParagraphAttributes(0, th1.getLength(), centerN1, false);
    }

    /**
     * Launch the application.
```

```
     */
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    Frame1 window = new Frame1();
                    window.frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    /**
     * Create the application.
     */
    public Frame1() {
        initialize();
    }

    /**
     * Initialize the contents of the frame.
     */
    private void initialize() {
        frame = new JFrame("15-Puzzle Solver");
        frame.setBounds(100, 100, 750, 455);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLocationRelativeTo(null);
        frame.setResizable(false);
        frame.getContentPane().setLayout(null);

        Image icon = Toolkit.getDefaultToolkit().getImage("puzzleicon.png");
        frame.setIconImage(icon);

        JPanel panel = new JPanel();
        panel.setBounds(0, 0, 315, 413);
        frame.getContentPane().add(panel);
        panel.setLayout(null);

        lblNewLabel = new JLabel("File Name");
        lblNewLabel.setBounds(41, 119, 62, 14);
        panel.add(lblNewLabel);

        textField = new JTextField();
```

```
        textField.setBounds(113, 116, 86, 20);
        panel.add(textField);
        textField.setColumns(10);

        JLabel lblSolveTime = new JLabel("Solve Duration");
        lblSolveTime.setBounds(41, 351, 107, 14);
        panel.add(lblSolveTime);

        timeTextField = new JTextField();
        timeTextField.setEditable(false);
        timeTextField.setColumns(10);
        timeTextField.setBounds(30, 365, 107, 20);
        panel.add(timeTextField);

        lblarisedNode = new JLabel("Arised Node");
        lblarisedNode.setBounds(212, 351, 80, 14);
        panel.add(lblarisedNode);

        nodeTextField = new JTextField();
        nodeTextField.setEditable(false);
        nodeTextField.setColumns(10);
        nodeTextField.setBounds(212, 365, 68, 20);
        panel.add(nodeTextField);

        JButton nextButton = new JButton("Next");
        JButton prevButton = new JButton("Prev");
        nextButton.setEnabled(false);
        prevButton.setEnabled(false);

        nextButton.setBounds(191, 186, 89, 23);
        panel.add(nextButton);
        prevButton.setBounds(30, 186, 89, 23);
        panel.add(prevButton);

        JLabel sumKurangX = new JLabel("Sum(KURANG(i)) + X");
        sumKurangX.setBounds(51, 259, 118, 14);
        panel.add(sumKurangX);

        kurangXTextField = new JTextField();
        kurangXTextField.setEditable(false);
        kurangXTextField.setBounds(174, 256, 86, 20);
        panel.add(kurangXTextField);
        kurangXTextField.setColumns(10);

        /*--- Main Feature of the GUI ---*/
```

```
        JButton insertButton = new JButton("Insert");
        insertButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                // File process & panelMatrix initialized
                String fileName = textField.getText();
                boolean usingFile = chckbxNewCheckBox.isSelected();
                fileError = false;
                try {
                    puzzleResult = new puzzle15(usingFile, fileName);
                }
                catch(Exception e2){
                    JOptionPane.showMessageDialog(null, "File not found /
error");
                    fileError = true;
                }
                JTextPane[] panelMatrix
=  {textBoxMat,textBoxMat_1,textBoxMat_2,textBoxMat_3,
                        textBoxMat_4,textBoxMat_5,textBoxMat_6,textBoxMat_7,
                        textBoxMat_8,textBoxMat_9,textBoxMat_10,textBoxMat_11,
                        textBoxMat_12,textBoxMat_13,textBoxMat_14,textBoxMat_15};

                // Visual Matrix Cleaner
                int f = 0;
                for (int i = 0; i < 4; i++) {
                    for (int j = 0; j < 4; j++) {
                        panelMatrix[f].setText(" ");
                        f++;
                    }
                }

                int g = 0;
                for (int i = 0; i < 4; i++) {
                    for (int j = 0; j < 4; j++) {
                        panelMatrix[g].setText(Integer.toString(puzzleResult.getP
uzzle()[i][j]));
                        g++;
                    }
                }
                // isReachable check
                reachable = puzzleResult.isReachable(puzzleResult.getPuzzle());
                System.out.println("|----------Tabel KurangI----------|");
                System.out.println("\ti\t    Kurang(i)");
                System.out.println(">---------------------------------<");
                for (int i = 0; i < puzzleResult.kurangItable.length; i++) {
```

```
                    System.out.println("\t" + (i+1) + "\t\t" +
puzzleResult.kurangItable[i]);
                }
                System.out.println("|--------------------------------|");
                kurangXTextField.setText(Integer.toString(puzzleResult.X +
puzzleResult.kurangI));
                solveButton.setEnabled(true);
            }
        });
        insertButton.setBounds(209, 115, 71, 23);
        panel.add(insertButton);


        solveButton = new JButton("Solve");
        solveButton.setBounds(209, 143, 71, 23);
        solveButton.setEnabled(false);
        solveButton.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {
                // Prev Next button initialized
                nextButton.setEnabled(true);
                prevButton.setEnabled(true);

                JTextPane[] panelMatrix
=  {textBoxMat,textBoxMat_1,textBoxMat_2,textBoxMat_3,
                        textBoxMat_4,textBoxMat_5,textBoxMat_6,textBoxMat_7,
                        textBoxMat_8,textBoxMat_9,textBoxMat_10,textBoxMat_11,
                        textBoxMat_12,textBoxMat_13,textBoxMat_14,textBoxMat_15};

                int[] solution;
                int solutionlen;

                if (reachable){
                    // Disable execute button
                    solveButton.setEnabled(false);

                    // Solve the puzzle
                    puzzleResult = puzzleResult.solve();

                    // Put arised node and time to screen
                    int arisedNodeCount = puzzleResult.possiblePath.size() - 1;
                    double time = puzzleResult.time;
                    nodeTextField.setText(Integer.toString(arisedNodeCount));
                    timeTextField.setText(Double.toString(time));
```

```
                    // Transfer solutionpath value to solution
                    // Functionalizing prev & next
                    solutionlen= puzzleResult.solutionpath.size();
                    solution = new int[solutionlen];

                    for (int q = 0; q < solutionlen; q++) {
                        solution[q] = puzzleResult.solutionpath.removeFirst();

                    }

                    // Next button pressed
                    nextButton.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                            // visualidx is in the last idx
                            if (puzzleResult.visualidx != solutionlen - 1) {
                                puzzleResult.visualidx++;
                                int[][] visualizerMat =
puzzleResult.possiblePath.get(solution[puzzleResult.visualidx]).instancepuzzle;
                                int n = 0;
                                for (int i = 0; i < 4; i++) {
                                    for (int j = 0; j < 4; j++) {
                                        panelMatrix[n].setText(Integer.toString(v
isualizerMat[i][j]));
                                        n++;
                                    }
                                }
                            }
                        }
                    });

                    // Prev button pressed
                    prevButton.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                            if (puzzleResult.visualidx != 0) {
                                puzzleResult.visualidx--;
                                int[][] visualizerMat =
puzzleResult.possiblePath.get(solution[puzzleResult.visualidx]).instancepuzzle;
                                int n = 0;
                                for (int i = 0; i < 4; i++) {
                                    for (int j = 0; j < 4; j++) {
                                        panelMatrix[n].setText(Integer.toString(v
isualizerMat[i][j]));
                                        n++;
                                    }
                                }
```

```
                            }
                        }
                    });
                    JOptionPane.showMessageDialog(null, "Puzzle Solved!!\nPlease
close and reopen this app to try again.\nThank You :D");
                    insertButton.setEnabled(false);
                }
                else{
                    if (!fileError) {
                        JOptionPane.showMessageDialog(null, "Unreachable");
                    }
                }
            }
        });
        panel.add(solveButton);
        /*--- end of button fiture ---*/

        chckbxNewCheckBox = new JCheckBox("Using test file");
        chckbxNewCheckBox.setBounds(85, 143, 113, 23);
        panel.add(chckbxNewCheckBox);

        label = new Label("15 - PUZZLE SOLVER");
        label.setFont(new Font("Cambria Math", Font.BOLD, 23));
        label.setBounds(41, 33, 235, 46);
        panel.add(label);

        lblSec = new JLabel("sec");
        lblSec.setBounds(141, 368, 36, 14);
        panel.add(lblSec);

        panel_1 = new JPanel();
        panel_1.setBounds(318, 0, 95, 95);
        frame.getContentPane().add(panel_1);
        panel_1.setLayout(new GridLayout(1, 2, 3, 4));

        panel_2 = new JPanel();
        panel_2.setBorder(new BevelBorder(BevelBorder.LOWERED, null, null, null,
null));
        panel_1.add(panel_2);
        panel_2.setLayout(null);

        textBoxMat = new JTextPane();
        textBoxMat.setFont(new Font("Tahoma", Font.PLAIN, 56));
        textBoxMat.setEditable(false);
        textBoxMat.setBackground(SystemColor.menu);
```

```
        textBoxMat.setBounds(10, 11, 75, 75);
        StyledDocument first = textBoxMat.getStyledDocument();
        SimpleAttributeSet center1 = new SimpleAttributeSet();
        StyleConstants.setAlignment(center1, StyleConstants.ALIGN_CENTER);
        first.setParagraphAttributes(0, first.getLength(), center1, false);
        panel_2.add(textBoxMat);

        panel_3 = new JPanel();
        panel_3.setBounds(423, 0, 95, 95);
        frame.getContentPane().add(panel_3);
        panel_3.setLayout(new GridLayout(1, 2, 3, 4));

        panel_4 = new JPanel();
        panel_4.setBorder(new BevelBorder(BevelBorder.LOWERED, null, null, null,
null));
        panel_3.add(panel_4);
        panel_4.setLayout(null);

        textBoxMat_1 = new JTextPane();
        textBoxMat_1.setFont(new Font("Tahoma", Font.PLAIN, 56));
        textBoxMat_1.setEditable(false);
        textBoxMat_1.setBackground(SystemColor.menu);
        textBoxMat_1.setBounds(10, 11, 75, 75);
        setToCenter(textBoxMat_1);
        panel_4.add(textBoxMat_1);

        panel_5 = new JPanel();
        panel_5.setBounds(528, 0, 95, 95);
        frame.getContentPane().add(panel_5);
        panel_5.setLayout(new GridLayout(1, 2, 3, 4));

        panel_6 = new JPanel();
        panel_6.setBorder(new BevelBorder(BevelBorder.LOWERED, null, null, null,
null));
        panel_5.add(panel_6);
        panel_6.setLayout(null);

        textBoxMat_2 = new JTextPane();
        textBoxMat_2.setFont(new Font("Tahoma", Font.PLAIN, 56));
        textBoxMat_2.setEditable(false);
        textBoxMat_2.setBackground(SystemColor.menu);
        textBoxMat_2.setBounds(10, 11, 75, 75);
        setToCenter(textBoxMat_2);
        panel_6.add(textBoxMat_2);
```

```
        panel_7 = new JPanel();
        panel_7.setBounds(634, 0, 95, 95);
        frame.getContentPane().add(panel_7);
        panel_7.setLayout(new GridLayout(1, 2, 3, 4));

        panel_8 = new JPanel();
        panel_8.setBorder(new BevelBorder(BevelBorder.LOWERED, null, null, null,
null));
        panel_7.add(panel_8);
        panel_8.setLayout(null);

        textBoxMat_3 = new JTextPane();
        textBoxMat_3.setFont(new Font("Tahoma", Font.PLAIN, 56));
        textBoxMat_3.setEditable(false);
        textBoxMat_3.setBackground(SystemColor.menu);
        textBoxMat_3.setBounds(10, 11, 75, 75);
        setToCenter(textBoxMat_3);
        panel_8.add(textBoxMat_3);

        panel_9 = new JPanel();
        panel_9.setBounds(318, 106, 95, 95);
        frame.getContentPane().add(panel_9);
        panel_9.setLayout(new GridLayout(1, 2, 3, 4));

        panel_10 = new JPanel();
        panel_10.setBorder(new BevelBorder(BevelBorder.LOWERED, null, null, null,
null));
        panel_9.add(panel_10);
        panel_10.setLayout(null);

        textBoxMat_4 = new JTextPane();
        textBoxMat_4.setFont(new Font("Tahoma", Font.PLAIN, 56));
        textBoxMat_4.setEditable(false);
        textBoxMat_4.setBackground(SystemColor.menu);
        textBoxMat_4.setBounds(10, 11, 75, 75);
        setToCenter(textBoxMat_4);
        panel_10.add(textBoxMat_4);

        panel_11 = new JPanel();
        panel_11.setBounds(423, 106, 95, 95);
        frame.getContentPane().add(panel_11);
        panel_11.setLayout(new GridLayout(1, 2, 3, 4));

        panel_12 = new JPanel();
```

```
        panel_12.setBorder(new BevelBorder(BevelBorder.LOWERED, null, null, null,
null));
        panel_11.add(panel_12);
        panel_12.setLayout(null);

        textBoxMat_5 = new JTextPane();
        textBoxMat_5.setFont(new Font("Tahoma", Font.PLAIN, 56));
        textBoxMat_5.setEditable(false);
        textBoxMat_5.setBackground(SystemColor.menu);
        textBoxMat_5.setBounds(10, 11, 75, 75);
        setToCenter(textBoxMat_5);
        panel_12.add(textBoxMat_5);

        panel_13 = new JPanel();
        panel_13.setBounds(528, 106, 95, 95);
        frame.getContentPane().add(panel_13);
        panel_13.setLayout(new GridLayout(1, 2, 3, 4));

        panel_14 = new JPanel();
        panel_14.setBorder(new BevelBorder(BevelBorder.LOWERED, null, null, null,
null));
        panel_13.add(panel_14);
        panel_14.setLayout(null);

        textBoxMat_6 = new JTextPane();
        textBoxMat_6.setFont(new Font("Tahoma", Font.PLAIN, 56));
        textBoxMat_6.setEditable(false);
        textBoxMat_6.setBackground(SystemColor.menu);
        textBoxMat_6.setBounds(10, 11, 75, 75);
        setToCenter(textBoxMat_6);
        panel_14.add(textBoxMat_6);

        panel_15 = new JPanel();
        panel_15.setBounds(634, 106, 95, 95);
        frame.getContentPane().add(panel_15);
        panel_15.setLayout(new GridLayout(1, 2, 3, 4));

        panel_16 = new JPanel();
        panel_16.setBorder(new BevelBorder(BevelBorder.LOWERED, null, null, null,
null));
        panel_15.add(panel_16);
        panel_16.setLayout(null);

        textBoxMat_7 = new JTextPane();
        textBoxMat_7.setFont(new Font("Tahoma", Font.PLAIN, 56));
```

```
        textBoxMat_7.setEditable(false);
        textBoxMat_7.setBackground(SystemColor.menu);
        textBoxMat_7.setBounds(10, 11, 75, 75);
        setToCenter(textBoxMat_7);
        panel_16.add(textBoxMat_7);

        panel_17 = new JPanel();
        panel_17.setBounds(318, 212, 95, 95);
        frame.getContentPane().add(panel_17);
        panel_17.setLayout(new GridLayout(1, 2, 3, 4));

        panel_18 = new JPanel();
        panel_18.setBorder(new BevelBorder(BevelBorder.LOWERED, null, null, null,
null));
        panel_17.add(panel_18);
        panel_18.setLayout(null);

        textBoxMat_8 = new JTextPane();
        textBoxMat_8.setFont(new Font("Tahoma", Font.PLAIN, 56));
        textBoxMat_8.setEditable(false);
        textBoxMat_8.setBackground(SystemColor.menu);
        textBoxMat_8.setBounds(10, 11, 75, 75);
        setToCenter(textBoxMat_8);
        panel_18.add(textBoxMat_8);

        panel_19 = new JPanel();
        panel_19.setBounds(423, 212, 95, 95);
        frame.getContentPane().add(panel_19);
        panel_19.setLayout(new GridLayout(1, 2, 3, 4));

        panel_20 = new JPanel();
        panel_20.setBorder(new BevelBorder(BevelBorder.LOWERED, null, null, null,
null));
        panel_19.add(panel_20);
        panel_20.setLayout(null);

        textBoxMat_9 = new JTextPane();
        textBoxMat_9.setFont(new Font("Tahoma", Font.PLAIN, 56));
        textBoxMat_9.setEditable(false);
        textBoxMat_9.setBackground(SystemColor.menu);
        textBoxMat_9.setBounds(10, 11, 75, 75);
        setToCenter(textBoxMat_9);
        panel_20.add(textBoxMat_9);

        panel_21 = new JPanel();
```

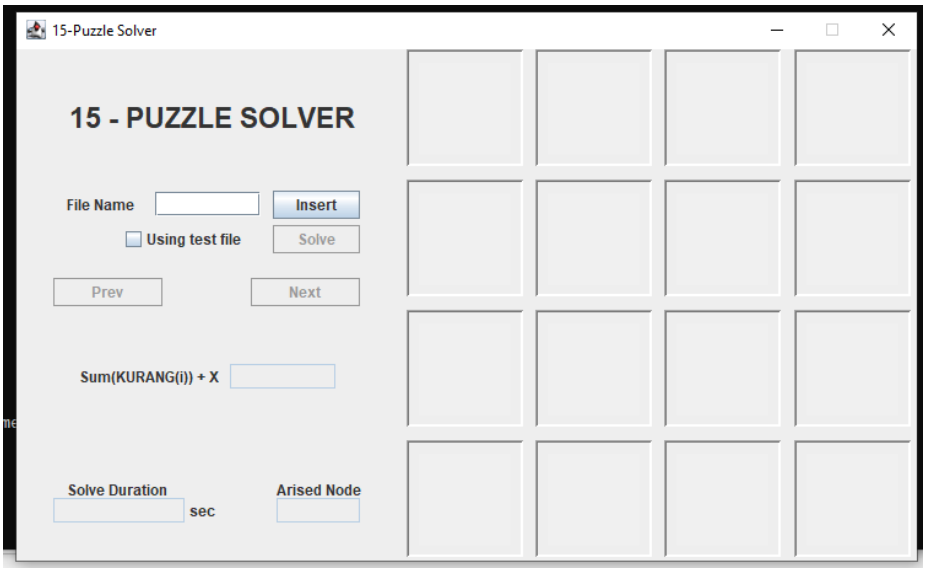
```
        panel_21.setBounds(528, 212, 95, 95);
        frame.getContentPane().add(panel_21);
        panel_21.setLayout(new GridLayout(1, 2, 3, 4));

        panel_22 = new JPanel();
        panel_22.setBorder(new BevelBorder(BevelBorder.LOWERED, null, null, null,
null));
        panel_21.add(panel_22);
        panel_22.setLayout(null);

        textBoxMat_10 = new JTextPane();
        textBoxMat_10.setFont(new Font("Tahoma", Font.PLAIN, 56));
        textBoxMat_10.setEditable(false);
        textBoxMat_10.setBackground(SystemColor.menu);
        textBoxMat_10.setBounds(10, 11, 75, 75);
        setToCenter(textBoxMat_10);
        panel_22.add(textBoxMat_10);

        panel_23 = new JPanel();
        panel_23.setBounds(634, 212, 95, 95);
        frame.getContentPane().add(panel_23);
        panel_23.setLayout(new GridLayout(1, 2, 3, 4));

        panel_24 = new JPanel();
        panel_24.setBorder(new BevelBorder(BevelBorder.LOWERED, null, null, null,
null));
        panel_23.add(panel_24);
        panel_24.setLayout(null);

        textBoxMat_11 = new JTextPane();
        textBoxMat_11.setFont(new Font("Tahoma", Font.PLAIN, 56));
        textBoxMat_11.setEditable(false);
        textBoxMat_11.setBackground(SystemColor.menu);
        textBoxMat_11.setBounds(10, 11, 75, 75);
        setToCenter(textBoxMat_11);
        panel_24.add(textBoxMat_11);

        panel_25 = new JPanel();
        panel_25.setBorder(new BevelBorder(BevelBorder.LOWERED, null, null, null,
null));
        panel_25.setBounds(318, 318, 95, 95);
        frame.getContentPane().add(panel_25);
        panel_25.setLayout(null);

        textBoxMat_12 = new JTextPane();
```

```
        textBoxMat_12.setFont(new Font("Tahoma", Font.PLAIN, 56));
        textBoxMat_12.setEditable(false);
        textBoxMat_12.setBackground(SystemColor.menu);
        textBoxMat_12.setBounds(10, 11, 75, 75);
        setToCenter(textBoxMat_12);
        panel_25.add(textBoxMat_12);

        panel_26 = new JPanel();
        panel_26.setBorder(new BevelBorder(BevelBorder.LOWERED, null, null, null,
null));
        panel_26.setBounds(423, 318, 95, 95);
        frame.getContentPane().add(panel_26);
        panel_26.setLayout(null);

        textBoxMat_13 = new JTextPane();
        textBoxMat_13.setFont(new Font("Tahoma", Font.PLAIN, 56));
        textBoxMat_13.setEditable(false);
        textBoxMat_13.setBackground(SystemColor.menu);
        textBoxMat_13.setBounds(10, 11, 75, 75);
        setToCenter(textBoxMat_13);
        panel_26.add(textBoxMat_13);

        panel_27 = new JPanel();
        panel_27.setBorder(new BevelBorder(BevelBorder.LOWERED, null, null, null,
null));
        panel_27.setBounds(528, 318, 95, 95);
        frame.getContentPane().add(panel_27);
        panel_27.setLayout(null);

        textBoxMat_14 = new JTextPane();
        textBoxMat_14.setFont(new Font("Tahoma", Font.PLAIN, 56));
        textBoxMat_14.setEditable(false);
        textBoxMat_14.setBackground(SystemColor.menu);
        textBoxMat_14.setBounds(10, 11, 75, 75);
        setToCenter(textBoxMat_14);
        panel_27.add(textBoxMat_14);

        panel_28 = new JPanel();
        panel_28.setBorder(new BevelBorder(BevelBorder.LOWERED, null, null, null,
null));
        panel_28.setBounds(634, 318, 95, 95);
        frame.getContentPane().add(panel_28);
        panel_28.setLayout(null);

        textBoxMat_15 = new JTextPane();
```

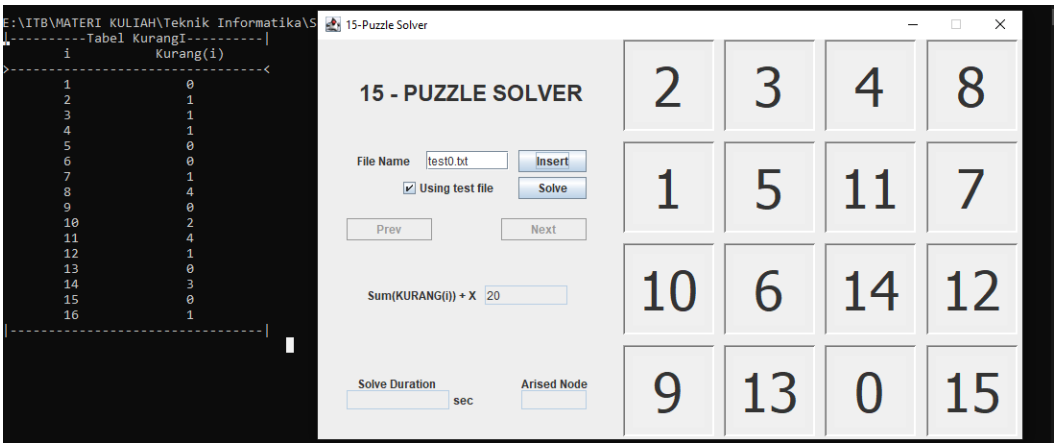
```
        textBoxMat_15.setFont(new Font("Tahoma", Font.PLAIN, 56));
        textBoxMat_15.setEditable(false);
        textBoxMat_15.setBackground(SystemColor.menu);
        textBoxMat_15.setBounds(10, 11, 75, 75);
        setToCenter(textBoxMat_15);
        panel_28.add(textBoxMat_15);

    }
}
```

3. ***Screenshot* input dan output**

LAYAR UTAMA

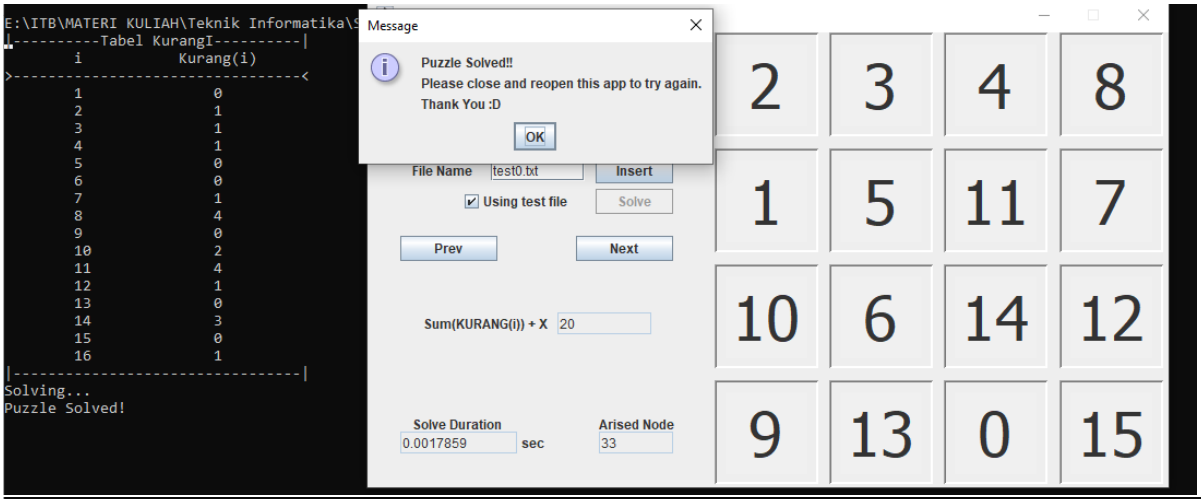


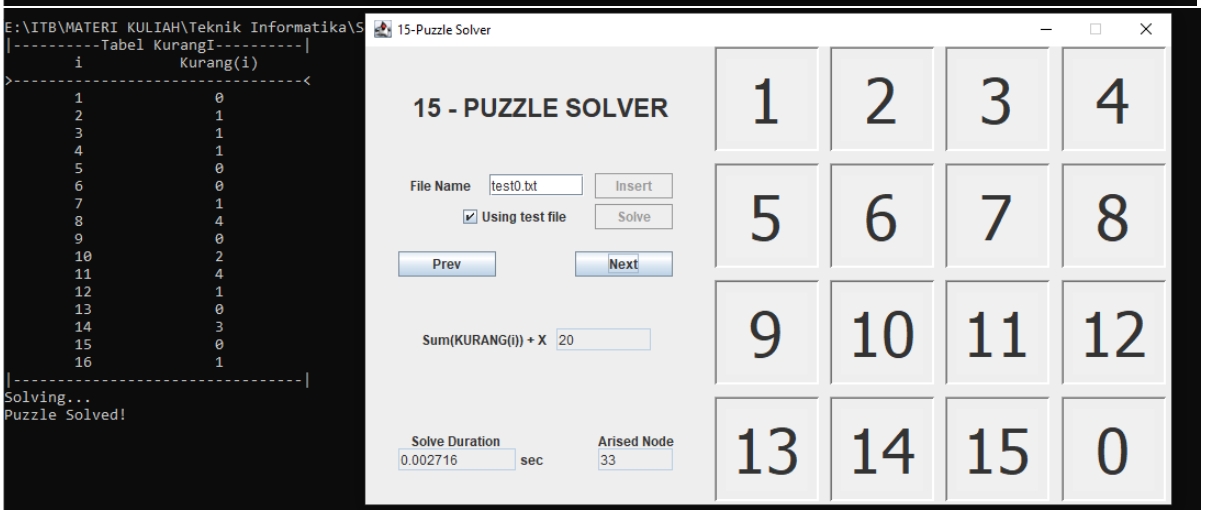
LAYAR SETELAH KLIK INSERT



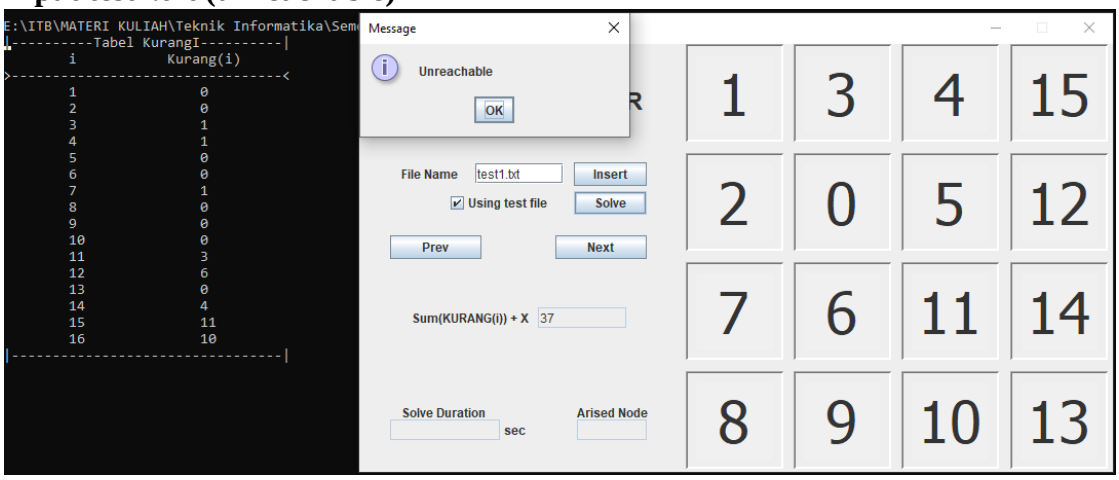
***** Tabel setiap ubin dari Kurang(i) ditampilkan melalui cmd

a. **Input test0.txt (solved)**

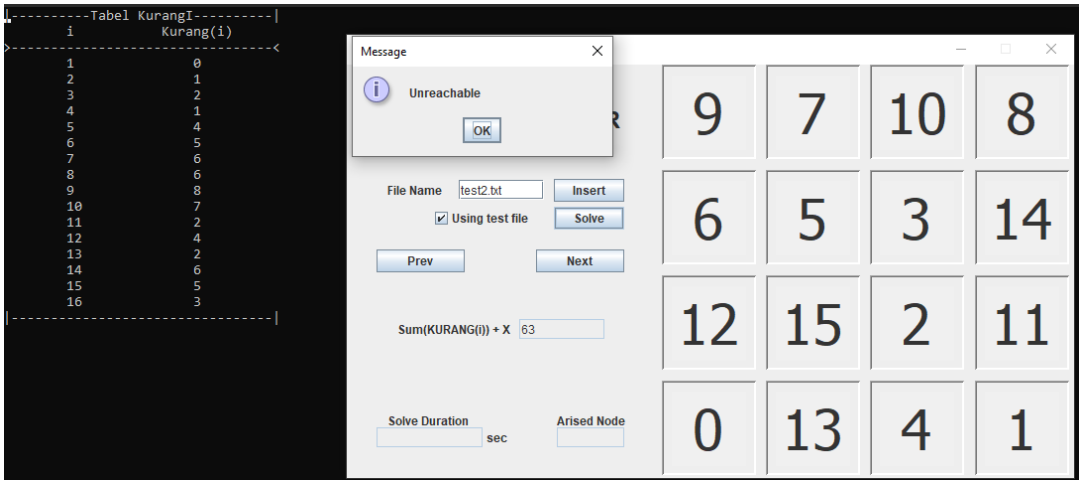




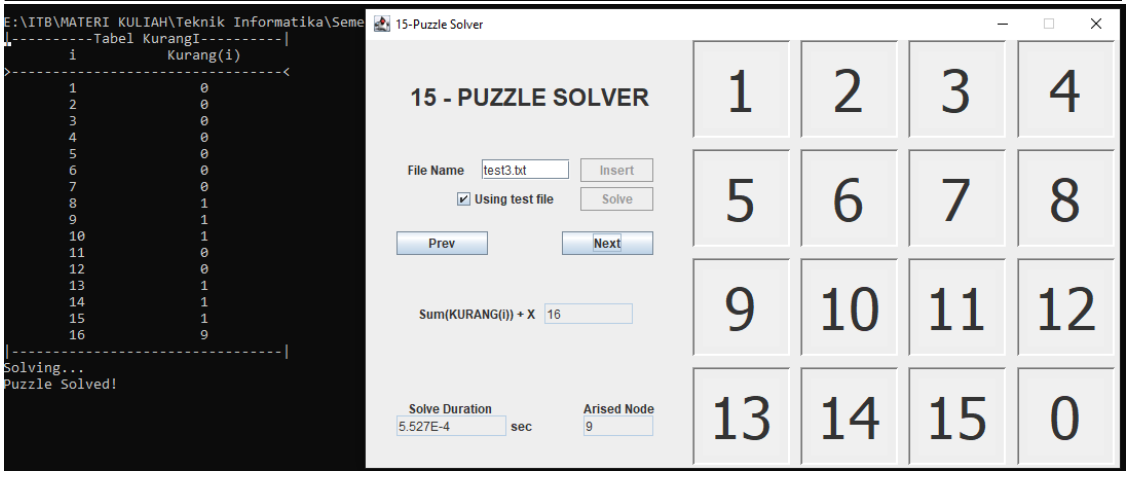
b. **Input test1.txt (unreachable)**



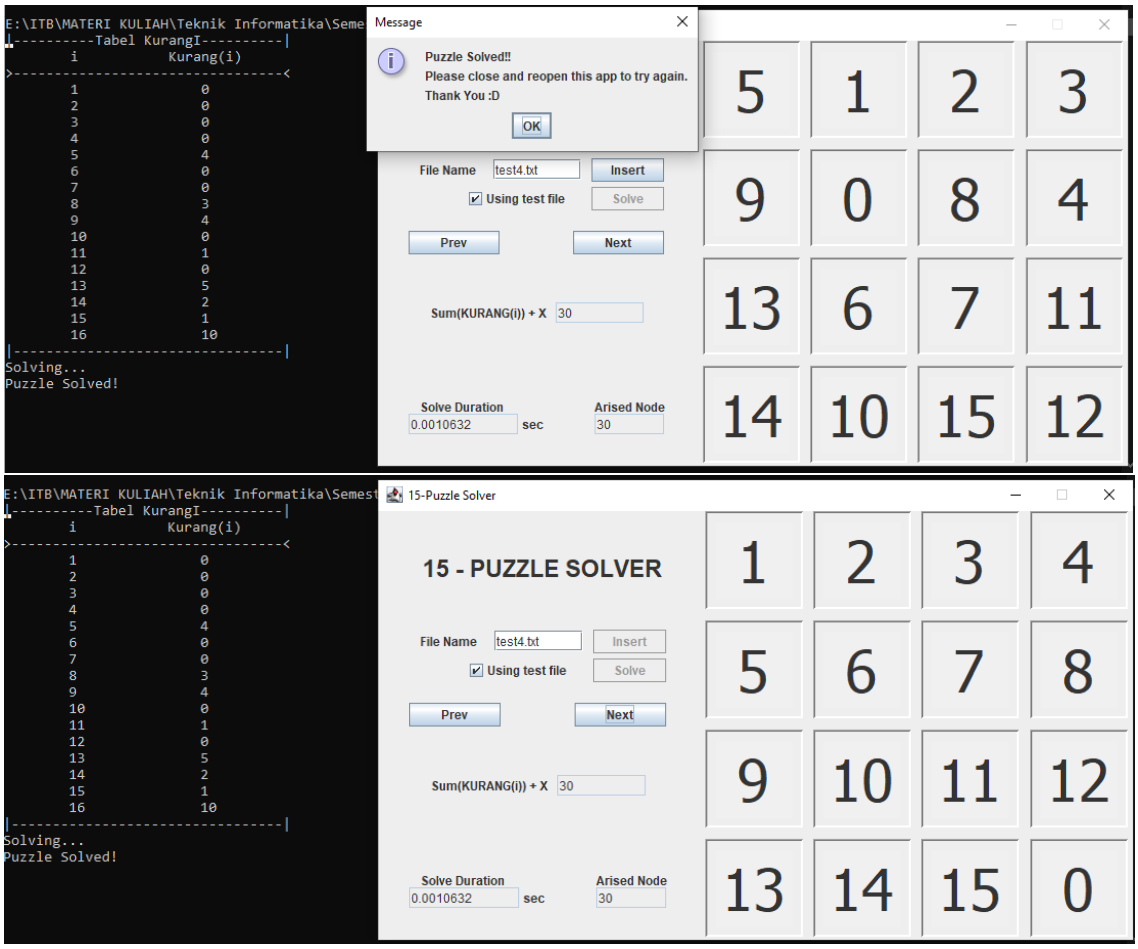
c. **Input test2.txt (unreachable)**



d. **Input test3.txt (solved)**



e. **Input test4.txt (solved)**



4. Alamat *drive* berisi kode program.

Paket program dapat diakses melalui repository github saya:

https://github.com/rkvilena/Tucil3_13520134.git

Poin	Ya	Tidak
1. Program berhasil dikompilasi	√	
2. Program berhasil running	√	
3. Program dapat menerima input dan menuliskan output	√	
4. Luaran sudah benar untuk semua data uji	√	
5. Bonus dibuat	√	