

# **Tugas Kecil 3 IF2211 Strategi Algoritma**

**Semester II tahun 2021/2022**

## **Penyelesaian Persoalan 15-Puzzle dengan Algoritma Branch and Bound**

disusun oleh :

Mahesa Lizardy      13520116



**PROGRAM STUDI TEKNIK INFORMATIKA**

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**2022**

## DAFTAR ISI

<b>Algoritma</b>	<b>3</b>
<b>Program</b>	<b>4</b>
Main.java	4
State.java	5
fifteenPuzzle.java	5
<b>Testing</b>	<b>18</b>
CHECKLIST	19
LINK GITHUB	19

## 1. Algoritma

fifteen puzzle merupakan teka teki geser yang memiliki 15 ubin persegi dengan tinggi 4 ubin dan lebar 4 ubin, yang menyisakan satu ubin kosong untuk digeser. Ubin tersebut dapat digeser secara horizontal atau vertikal selama tidak keluar dari bingkai. Tujuan dari permainan ini adalah pemain dapat menggeser-geser ubin tersebut hingga ubin tersebut mencapai susunan akhir (*goal state*). Susunan akhir dari permainan fifteen puzzle adalah sebagai berikut:

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

Gambar 1.1 Susunan akhir fifteen puzzle

Pada Tugas kecil 3 kali ini akan dibuat program untuk menyelesaikan persoalan fifteen-puzzle yang telah dijelaskan sebelumnya. Pada Program Penyelesaian Persoalan fifteen-Puzzle dengan Algoritma Branch and Bound ini terdapat 3 file utama yaitu:

1. fifteenPuzzle.java
2. State.java
3. Main.java

Pada fifteenPuzzle.java berisi kelas dadri 15-puzzle, terdapat constructor dari fifteen puzzle dan fungsi/prosedur lainnya. Pembuatan 15-puzzle pada program ini dapat melalui dua cara yaitu dengan random atau dengan menggunakan file test bertipe teks(.txt) yang telah dibuat pada folder test. pada prosedur "*solution()*" merupakan Algoritma untuk menemukan solusi dari permasalahan 15-puzzle, mencari rute(*path*) dari posisi awal hingga posisi akhir untuk mencapai susunan akhir (*goal state*). Sebelum dicari rute dari puzzle tersebut akan dilakukan pengecekan oleh "*isSolvable()*". Fungsi tersebut akan mencari apakah state awal puzzle dapat dicari solusinya atau "*Reachable Goal*". Puzzle dikatakan "*Reachable Goal*" jika penjumlahan dari

$$\sum_{i=1}^{i=16} KURANG(i) + X \text{ bernilai genap}$$

fungsi kurang(i) pada program terdapat pada fungsi "*kurang()*" sedangkan fungsi X terdapat pada "*positionBlankPos()*" kedua fungsi tersebut nantinya akan dijumlahkan pada fungsi "*isSolvable()*" apakah bernilai genap atau tidak. Jika Puzzle "*Reachable Goal*" maka puzzle tersebut dapat dicari solusinya. Pada fungsi untuk mencari solusi yang kami buat akan akan digunakan Priority Queue yang akan menjadi antrian untuk menyimpan rute yang sudah ditemukan beserta dengan cost nya serta untuk menentukan rute selanjutnya yang akan dibangkitkan. Setiap membangkitkan solusi akan dihitung cost pada rute tersebut. Cost dari rute tersebut merupakan jumlah dari panjang lintasan untuk mencapai rute tersebut dari akar(posisi awal) dengan jumlah ubin tidak kosong yang tidak terdapat pada susunan akhir. Perhitungan cost tersebut akan dilakukan oleh fungsi "*cost()*"

$$\hat{c}(i) = f(i) + g(i)$$

$\hat{c}(i)$  = cost untuk simpul  $i$

$f(i)$  = panjang lintasan akar ke rute  $i$

$g(i)$  = jumlah ubin tidak kosong yang tidak terdapat pada susunan akhir

langkah-langkah Algoritma Branch and Bound yang terdapat pada solution adalah sebagai berikut

1. jika puzzle sudah merupakan susunan akhir maka stop
2. jika antrian kosong maka bangkitkan rute yang memungkinkan berdasarkan move yang diperbolehkan, urutanya adalah (“up”, “down”, “right”, “left”)
3. jika antrian tidak kosong, pilih dari antrian yang memiliki cost  $\hat{c}(i)$  paling kecil keluarkan, kemudian jika belum merupakan susunan akhir maka seperti step ke-2, bangkitkan rute yang ada.
4. jika rute  $i$  merupakan solusi maka solusi sudah ditemukan, stop
5. Jika simpul  $i$  bukan simpul solusi, maka bangkitkan semua anak-anaknya ulangi step 3.
6. untuk setiap rute akan dihitung cost dari rute tersebut untuk menentukan rute mana yang akan dibangkitkan selanjutnya

jika rute untuk menuju susunan akhir sudah ditemukan maka program akan menampilkan GUI yang memperlihatkan pergeseran move dari susunan awal ke susunan akhir.

## 2. Program

### 1. Main.java

```
public static void main(String[] args) {
    fifteenPuzzle game = new fifteenPuzzle(550,30);
    JFrame frame =new JFrame();
    SwingUtilities.invokeLater(()->{
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setTitle("Pinky-FifteenPuzzle");
        frame.setResizable(false);
        frame.add(game, BorderLayout.CENTER);
        frame.pack();
        frame.setLocationRelativeTo(null);
        frame.setVisible(true);
    });
    Scanner sc = new Scanner(System.in);
    System.out.print(">>> ");
    String str;
    str = sc.nextLine();
    while(!str.equals("QUIT")){
        if(str.equals("SOLUTION")){
            game.solution();
        }
        else if (str.equals("INPUTTEST")){
            System.out.print("masukkan nama file: ");
            String file = sc.nextLine();
            game.readFile(file);
        }
        else if(str.equals("RANDOM")){
            game.newGame();
        }
        else if(str.equals("HELP")){
            System.out.println("COMMAND LIST");

            System.out.println("=====");
            System.out.println("RANDOM          : Melakukan pengacakan pada
puzzle");
            System.out.println("INPUTTEST      : Melakukan input puzzle dari
file di dalam folder test");
            System.out.println("SOLUTION       : Menjalankan Solusi dari
puzzle tersebut");
            System.out.println("QUIT           : Keluar dari program");
            System.out.println("PS: Program yang dibuat merupakan separuh
GUI dan CLI, dimana user akan melakukan input melalui GUI " +
"dan Hasil dari input tersebut akan ditampilkan pada
GUI fifteen Puzzle");

        }
        else{
            System.out.println("INVALID COMMAND");
        }
        System.out.print(">>> ");
        str = sc.nextLine();
    }
    System.exit(0);
}
```

## 2. State.java

### a. constructor dan atribut State

```
public State(int Cost, Vector<String> command) {  
    this.cost = Cost;  
    this.command = new Vector<>(command);  
}  
public int cost;  
public Vector<String> command;
```

### b. State Comparator

```
class StateComparator implements Comparator<State> {  
    @Override  
    public int compare(State a, State b) {  
        if (a.cost < b.cost)  
            return -1;  
        if (a.cost > b.cost)  
            return 1;  
        if (a.cost == b.cost) {  
            if (a.command.size() < b.command.size()) {  
                return 1;  
            }  
            else {  
                return -1;  
            }  
        }  
        return 0;  
    }  
}
```

## 3. fifteenPuzzle.java

### a. Atribut

```
// Size  
private static final int size = 4;  
  
// tiles  
private int nTiles;  
  
// UI  
private int dimension;  
private int margin;  
private int gridSize;  
private int tileSize;  
  
// color  
private static final Color FOREGROUND_COLOR = new Color(255,182,213);  
private static final Color BACKGROUND_COLOR = new Color(255,217,225);  
private static final Random randomNumber = new Random();  
  
// menampung hasil isSolveable
```

```

private boolean can_Solve;
// list untuk menampung puzzle
private int[] tiles;
// posisi blankPos (kotak kosong)
private int blankPos;
// jika game sudah diselesaikan
private boolean gameOver;

// for solution
private PriorityQueue<State> pq = new PriorityQueue<State>(new
StateComparator());
private Vector<String> solutionCommand = new Vector<>();
private int[] tempTiles;

```

#### b. procedure fifteenpuzzle

```

// constructor fifteen puzzle
public fifteenPuzzle(int dimension, int margin){
    this.dimension = dimension;
    this.margin = margin;
    nTiles = size * size;
    tiles = new int[size*size];
    tempTiles = new int[size*size];
    gridSize = (dimension - 2 * margin);
    tileSize = gridSize/size;

    setPreferredSize(new Dimension(this.dimension, this.dimension +
margin));
    setBackground(BACKGROUND_COLOR);
    setForeground(FOREGROUND_COLOR);
    setFont(new Font("SansSerif", Font.BOLD, 60));
    this.gameOver = true;
    setDefault();
}

```

#### c. procedure newGame

```

// newGame random puzzle
public void newGame(){
    do{
        setDefault();
        shuffle();
    } while(!isSolvable());
}

```

```
repaint();  
}
```

d. procedure setDefault

```
// set default puzzle  
public void setDefault(){  
    for(int i=0;i<16;i++){  
        {  
            tiles[i] = i+1;  
            tempTiles[i] = i+1;  
        }  
        blankPos = 15;  
    }  
}
```

e. procedure shuffle

```
// shuffle puzzle  
public void shuffle(){  
    int i = 0;  
    while(i < 25){  
        int index = randomNumber.nextInt(0,100) % 4;  
        if(index == 0 && boolUp()){  
            commandTemp("up");  
        }  
        else if(index == 1 && boolDown()){  
            commandTemp("down");  
        }  
        else if(index == 2 && boolRight()){  
            commandTemp("right");  
        }  
        else if(index == 3 && boolLeft()){  
            commandTemp("left");  
        }  
        else{  
            i--;  
        }  
        i++;  
    }  
    for (int j = 0; j < 16; j++){  
        tiles[j] = tempTiles[j];  
    }  
    blankPos = findBlankPos(tiles);  
}
```



```
}
```

#### f. function kurang

```
// function KURANG(X)
public int kurang(){
    int count = 0;
    for (int i=0;i<16;i++){
        for (int j = i+1;j<16;j++){
            if(tiles[j]<tiles[i]){
                count+=1;
            }
        }
    }
    return count;
}
```

#### g. fungsi positionBlankPos

```
// X begining position of blankPos
public int positionBlankPos(){
    if ((blankPos <=3) || (blankPos<=11 &&blankPos>7)){
        return blankPos %2;
    }
    else{
        return (blankPos + 1)%2;
    }
}
```

#### h. function solve

```
// tiles solve
public boolean isSolve(int[] array){
    for (int i=0;i<16;i++){
        if(array[i] != i+1){
            return false;
        }
    }
    return true;
}
```

i. function isSolvable

```
// tiles can solve
public boolean isSolvable(){
    int sum = kurang() + positionBlankPos();
    System.out.println("Nilai Kurang(i) :" + kurang());
    System.out.println("Kurang(i) + X   :" + sum);
    return (sum) %2 ==0;
}
```

j. procedure drawGrid

```
// draw the grid
public void drawGrid(Graphics2D grid){
    for (int i =0; i< tiles.length;i++){
        int row = i/size;
        int col = i% size;

        int x = margin + col *tileSize;
        int y = margin + row * tileSize;

        if (tiles[i] == 16){
            if (gameOver){
                grid.setColor(FOREGROUND_COLOR);
            }

            continue;
        }
        grid.setColor((getForeground()));
        grid.fillRoundRect(x,y,tileSize-10,tileSize-10,0,0);
        grid.setColor(Color.gray);
        grid.drawRoundRect(x,y,tileSize-10,tileSize-10,0,0);
        grid.setColor(Color.WHITE);

        drawCenteredString(grid, String.valueOf(tiles[i]), x, y);
    }
}
```

k. procedure drawStart

```
// draw number puzzle
private void drawCenteredString(Graphics2D g, String s, int x, int y) {
    FontMetrics fm = g.getFontMetrics();
```

```

int asc = fm.getAscent();
int desc = fm.getDescent();
g.drawString(s, x + (tileSize-10 - fm.stringWidth(s)) / 2,
             y + (asc + (tileSize-10 - (asc + desc)) / 2));
}

```

## l. procedure readFile

```

// read from file in folder Test
public void readFile(String file){
    String filePath = "../test/"+ file;
    try{
        Scanner sc = new Scanner(new BufferedReader(new
FileReader(filePath)));
        int i =0;
        while(sc.hasNextLine()){
            while (i<tiles.length) {
                String[] line = sc.nextLine().trim().split(" ");
                for (int j=0; j<line.length; j++) {
                    tiles[i] = Integer.parseInt(line[j]);
                    i++;
                }
            }
        }
    } catch (IOException e) {
        System.out.println("tidak ada file dengan nama " + file);
    }
    blankPos=findBlankPos(tiles);
    if (!isSolvable()){
        repaint();
        System.out.println("Puzzle tidak dapat diselesaikan");
        try {
            TimeUnit.SECONDS.sleep(3);
        }
        catch (InterruptedException ex) {
        }
        setDefault();
    }
    repaint();
}

```

## m. function findBlankPos

```
// find position blankPos
public int findBlankPos(int[] array){
    for(int i=0;i<16;i++){
        if (array[i]==16){
            return i;
        }
    }
    return -1;
}
```

#### n. function cost

```
// cost command
public int cost(int[] temp){
    int cost = 0;
    for(int i= 0;i<temp.length;i++){
        if(temp[i]!=i+1 && temp[i] != 16){
            cost++;
        }
    }
    return cost;
}
```

#### o. procedure commandTemp

```
// move tempTiles
private void commandTemp(String command) {
    int temp;
    if (command.equals("up")) {
        temp = tempTiles[blankPos - 4];
        tempTiles[blankPos - 4] = tempTiles[blankPos];
        tempTiles[blankPos] = temp;
        blankPos = blankPos - 4;
    } else if (command.equals("down")) {
        temp = tempTiles[blankPos + 4];
        tempTiles[blankPos + 4] = tempTiles[blankPos];
        tempTiles[blankPos] = temp;
        blankPos = blankPos + 4;
    } else if (command.equals("left")) {
        temp = tempTiles[blankPos - 1];
        tempTiles[blankPos - 1] = tempTiles[blankPos];
        tempTiles[blankPos] = temp;
        blankPos = blankPos - 1;
    } else if (command.equals("right")) {
        temp = tempTiles[blankPos + 1];
    }
}
```

```

        tempTiles[blankPos + 1] = tempTiles[blankPos];
        tempTiles[blankPos] = temp;
        blankPos = blankPos + 1;
    } else {
        System.out.println("invalid Command");
    }
}

// overloading
private void commandTemp(Vector<String> vectorCommand) {
    int temp;
    String command;
    for(int i = 0; i < vectorCommand.size(); i++) {
        command = vectorCommand.get(i);
        if (command.equals("up")) {
            temp = tempTiles[blankPos - 4];
            tempTiles[blankPos - 4] = tempTiles[blankPos];
            tempTiles[blankPos] = temp;
            blankPos = blankPos - 4;
        } else if (command.equals("down")) {
            temp = tempTiles[blankPos + 4];
            tempTiles[blankPos + 4] = tempTiles[blankPos];
            tempTiles[blankPos] = temp;
            blankPos = blankPos + 4;
        } else if (command.equals("left")) {
            temp = tempTiles[blankPos - 1];
            tempTiles[blankPos - 1] = tempTiles[blankPos];
            tempTiles[blankPos] = temp;
            blankPos = blankPos - 1;
        } else if (command.equals("right")) {
            temp = tempTiles[blankPos + 1];
            tempTiles[blankPos + 1] = tempTiles[blankPos];
            tempTiles[blankPos] = temp;
            blankPos = blankPos + 1;
        } else {
            System.out.println("invalid Command");
        }
    }
}
}

```

#### p. procedure command

```

// move puzzle with delay
public void command(String command) {
    int temp;

```

```

// wait 1 second
try {
    TimeUnit.SECONDS.sleep(1);
}
catch (InterruptedException ex) {
}
if (command.equals("up")) {
    temp = tiles[blankPos-4];
    tiles[blankPos-4] = tiles[blankPos];
    tiles[blankPos] = temp;
    blankPos = blankPos-4;
}
else if (command.equals("down")) {
    temp = tiles[blankPos+4];
    tiles[blankPos+4] = tiles[blankPos];
    tiles[blankPos] = temp;
    blankPos = blankPos+4;
}
else if (command.equals("left")) {
    temp = tiles[blankPos-1];
    tiles[blankPos-1] = tiles[blankPos];
    tiles[blankPos] = temp;
    blankPos = blankPos-1;
}
else if (command.equals("right")) {
    temp = tiles[blankPos+1];
    tiles[blankPos+1] = tiles[blankPos];
    tiles[blankPos] = temp;
    blankPos = blankPos+1;
}
else {
    System.out.println("invalid Command");
}
this.repaint();
}

```

#### q. boolean move

```

// boolean move left
public boolean boolLeft() {
    return blankPos % 4 != 0;
}

// boolean move right
public boolean boolRight() {
    return blankPos % 4 != 3;
}

//// boolean move up
public boolean boolUp() {
    return blankPos-4 >= 0;
}

// boolean move down
public boolean boolDown() {

```

```
        return blankPos+4<16;
    }
}
```

#### r. procedure solution

```
// find solution for puzzle
public void solution(){
    // menampung command sementara
    long nano_startTime = System.nanoTime();
    Vector<String> tempCommand = new Vector<>();
    solutionCommand.clear();
    pq.clear();
    int compare =0;
    //menampung cost
    int tempCost;
    while(!isSolve(tiles)){
        tempTiles = this.tiles.clone();
        blankPos = findBlankPos(tempTiles);
        // jika pq empty
        if(pq.isEmpty()){
            if (boolUp()){
                tempCommand.add("up");
                commandTemp("up");
                tempCost = cost(tempTiles)+ tempCommand.size();
                pq.add(new State(tempCost,tempCommand));

                if (isSolve(tempTiles)){
                    break;
                }
                commandTemp("down");
                tempCommand.clear();
                compare++;
            }
            if(boolDown()){
                tempCommand.add("down");
                commandTemp("down");
                tempCost = cost(tempTiles)+ tempCommand.size();
                pq.add(new State(tempCost,tempCommand));

                if (isSolve(tempTiles)){
                    break;
                }
                commandTemp("up");
            }
        }
    }
}
```

```

        tempCommand.clear();
        compare++;
    }
    if(boolRight()){
        tempCommand.add("right");
        commandTemp("right");
        tempCost = cost(tempTiles)+ tempCommand.size();
        pq.add(new State(tempCost,tempCommand));

        if (isSolve(tempTiles)){
            break;
        }
        commandTemp("left");
        tempCommand.clear();
        compare++;
    }
    if(boolLeft()){
        tempCommand.add("left");
        commandTemp("left");
        tempCost = cost(tempTiles)+ tempCommand.size();
        pq.add(new State(tempCost,tempCommand));

        if (isSolve(tempTiles)){
            break;
        }
        commandTemp("right");
        tempCommand.clear();
        compare++;
    }
}
// jika tidak empty
else{
    State temp = pq.poll();
    commandTemp(temp.command);
    tempCommand = new Vector<>(temp.command);
    if (boolUp() &&
!tempCommand.get(tempCommand.size()-1).equals("down")){
        tempCommand.add("up");
        commandTemp("up");
        tempCost = cost(tempTiles)+ tempCommand.size();

        pq.add(new State(tempCost,tempCommand));

        if (isSolve(tempTiles)){

```



```

        break;
    }
    commandTemp("down");
    tempCommand.remove(tempCommand.size()-1);
    compare++;
}

if(boolDown() && !tempCommand.get(tempCommand.size()-1).equals("up")) {
    tempCommand.add("down");
    commandTemp("down");
    tempCost = cost(tempTiles)+ tempCommand.size();
    pq.add(new State(tempCost,tempCommand));

    if (isSolve(tempTiles)) {
        break;
    }
    commandTemp("up");
    tempCommand.remove(tempCommand.size()-1);
    compare++;
}

if(boolRight() && !tempCommand.get(tempCommand.size()-1).equals("left")) {
    tempCommand.add("right");
    commandTemp("right");
    tempCost = cost(tempTiles)+ tempCommand.size();
    pq.add(new State(tempCost,tempCommand));

    if (isSolve(tempTiles)) {
        break;
    }
    commandTemp("left");
    tempCommand.removeElementAt(tempCommand.size()-1);
    compare++;
}

if(boolLeft() && !tempCommand.get(tempCommand.size()-1).equals("right")) {
    tempCommand.add("left");
    commandTemp("left");
    tempCost = cost(tempTiles)+ tempCommand.size();
    pq.add(new State(tempCost,tempCommand));

    if (isSolve(tempTiles)) {
        break;
    }
}

```

```

        commandTemp("right");
        tempCommand.remove(tempCommand.size()-1);
        compare++;
    }

}

}

```

#### s. procedure paintComponent

```

@Override
public void paintComponent(Graphics g) {
    super.paintComponent(g);
    Graphics2D g2D = (Graphics2D) g;
    g2D.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
    drawGrid(g2D);
}

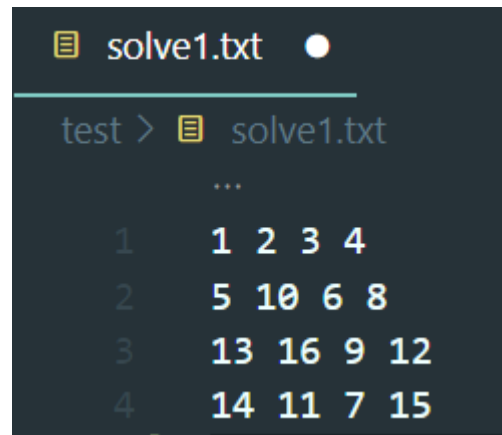
```

### 3. Testing

Program yang telah dibuat akan dilakukan pengujian terhadap testing yang terdapat pada folder test. file test terdiri dari 3 file yang berisi puzzle yang dapat diselesaikan, dan 2 file puzzle yang tidak dapat diselesaikan

- a. file solve1.txt

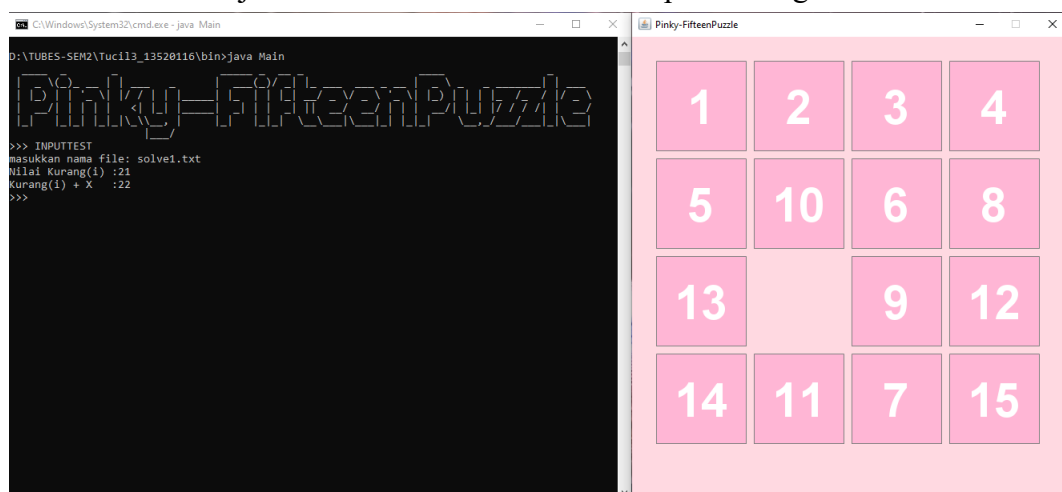
berikut isi dari file solve1.txt



```
test > solve1.txt
...
1    1  2  3  4
2    5 10  6  8
3   13 16  9 12
4   14 11  7 15
```

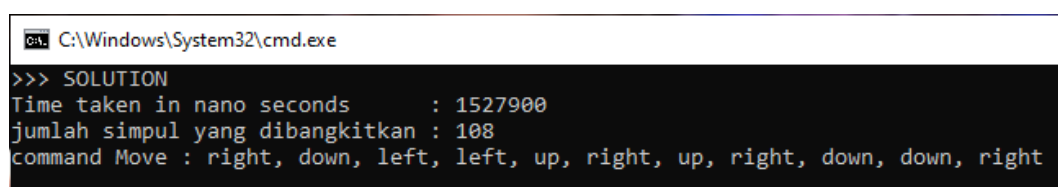
*Gambar 3.1 isi file test solve1.txt*

pada test solve1.txt jika di run akan memberikan tampilan sebagai berikut



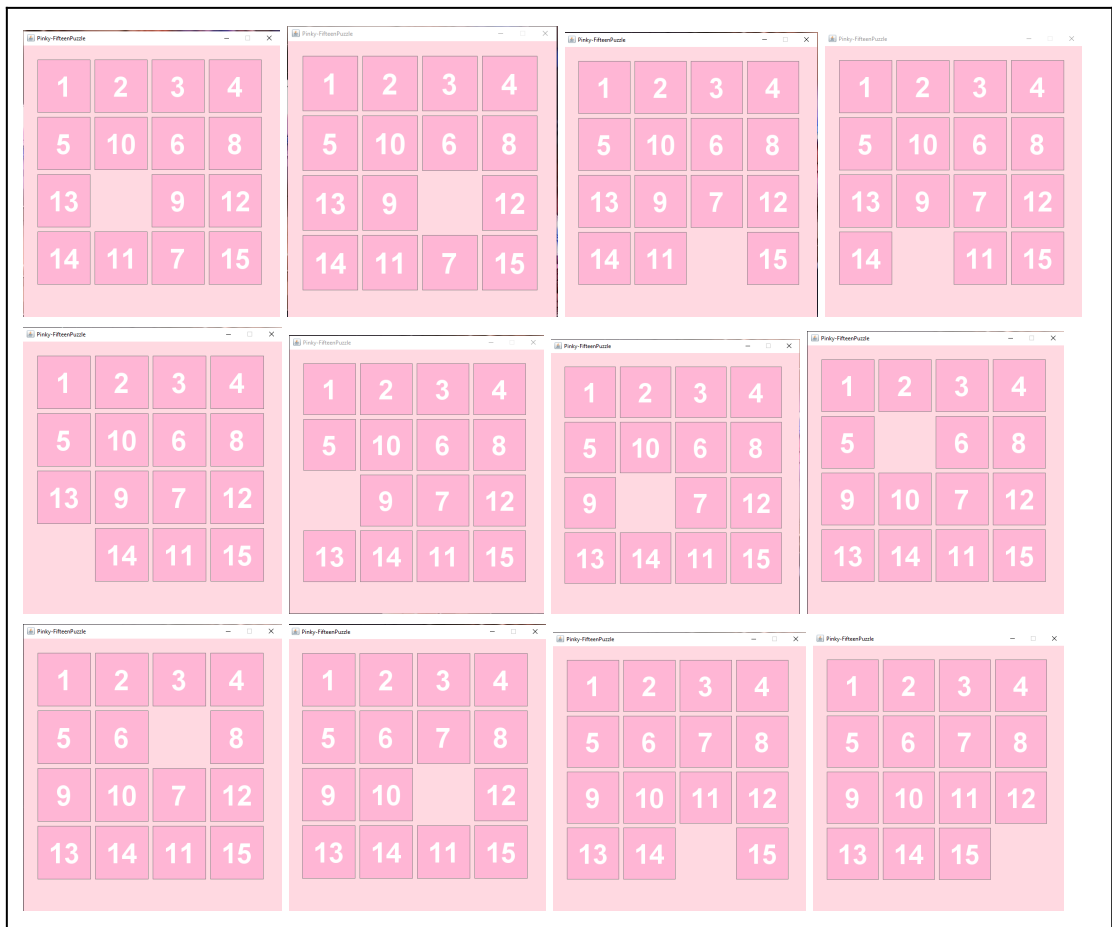
*Gambar 3.2 hasil run solve1.txt*

karena puzzle tersebut dapat diselesaikan atau *Reachable Goal* dapat dilihat bahwa nilai kurang(i) ditambah X merupakan genap. Maka puzzle tersebut dapat dicari solusinya. Berikut merupakan tampilan yang keluar jika kita mencari solusinya, melakukan command “SOLUTION”



```
C:\Windows\System32\cmd.exe
>>> SOLUTION
Time taken in nano seconds      : 1527900
jumlah simpul yang dibangkitkan : 108
command Move : right, down, left, left, up, right, up, right, down, down, right
```

*Gambar 3.3 pencarian solusi dari puzzle solve1.txt*



Gambar 3.4 tampilan GUI dari solusi puzzle solve1.txt dari kiri ke kanan lalu bawah

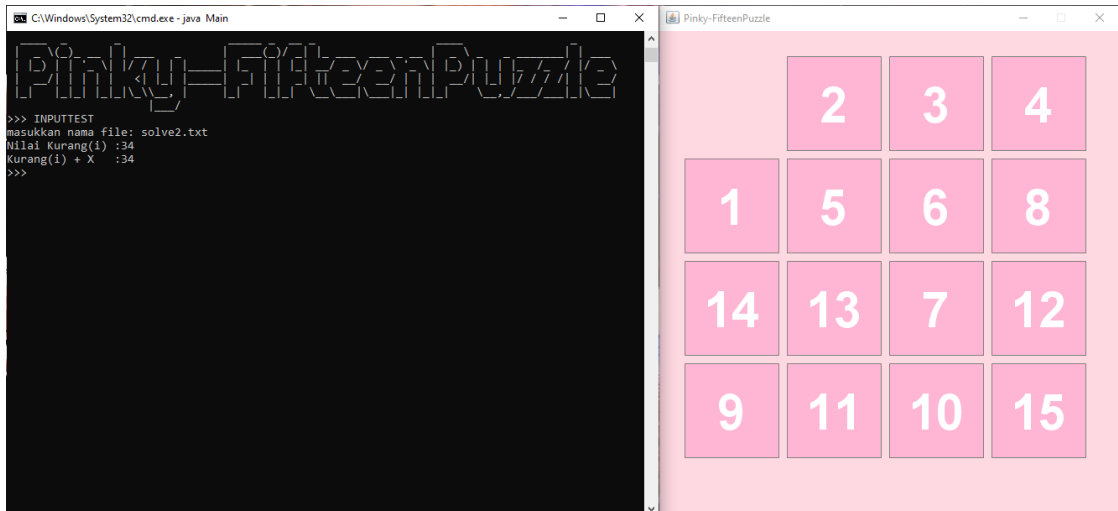
- b. file solve2.txt  
berikut isi dari file solve2.txt

```

solve2.txt
test > solve2.txt
...
1 16 2 3 4
2 1 5 6 8
3 14 13 7 12
4 9 11 10 15
  
```

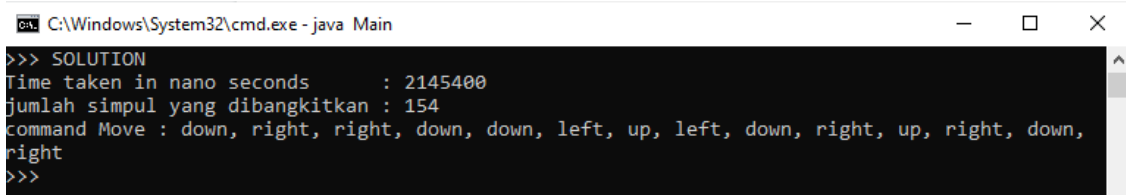
Gambar 3.5 isi file test solve2.txt

pada test solve2.txt jika di run akan memberikan tampilan sebagai berikut

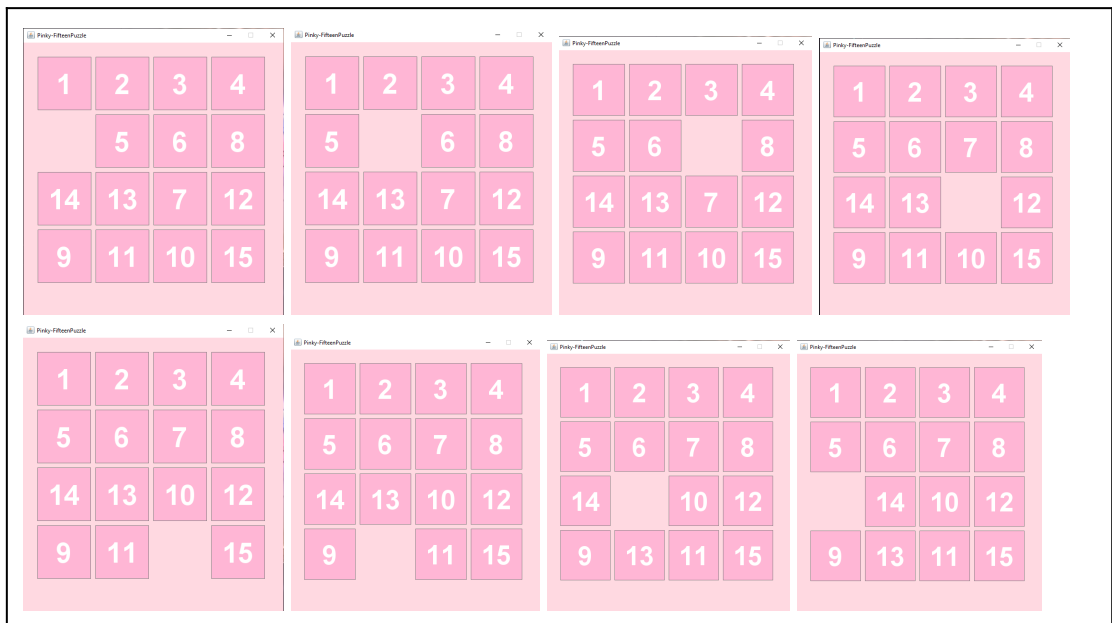


Gambar 3.6 hasil run solve2.txt

karena puzzle tersebut dapat diselesaikan atau *Reachable Goal* dapat dilihat bahwa nilai kurang(i) ditambah X merupakan genap. Maka puzzle tersebut dapat dicari solusinya. Berikut merupakan tampilan yang keluar jika kita mencari solusinya, melakukan command “SOLUTION”



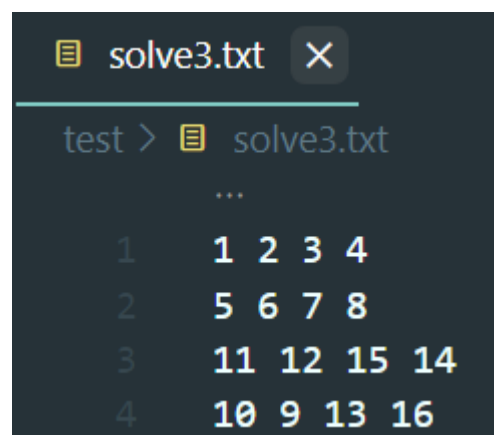
Gambar 3.7 pencarian solusi dari puzzle solve2.txt





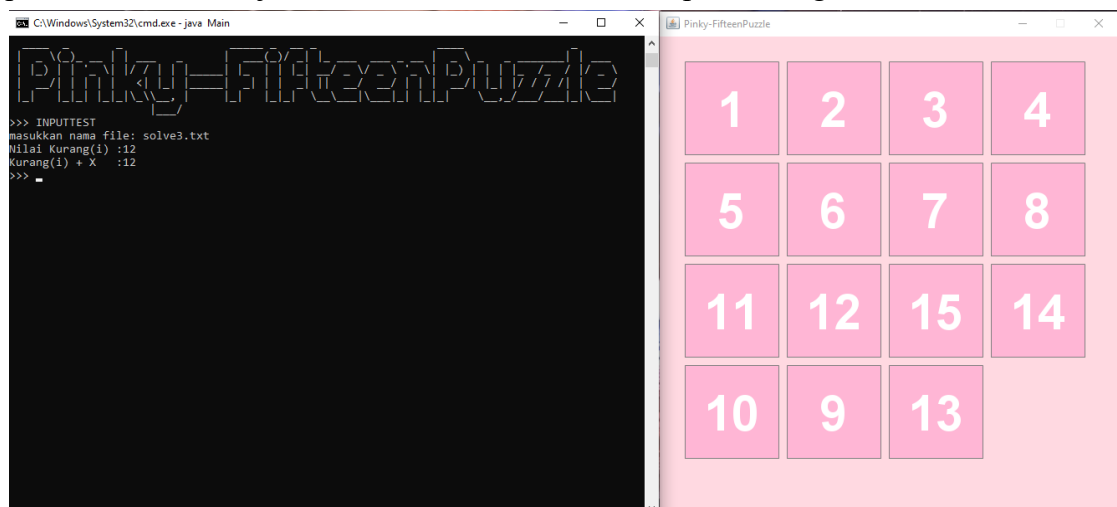
Gambar 3.8 tampilan GUI dari solusi puzzle solve2.txt dari kiri ke kanan lalu bawah

- c. file solve3.txt  
berikut isi dari file solve3.txt



Gambar 3.9 isi file test solve3.txt

pada test solve3.txt jika di run akan memberikan tampilan sebagai berikut

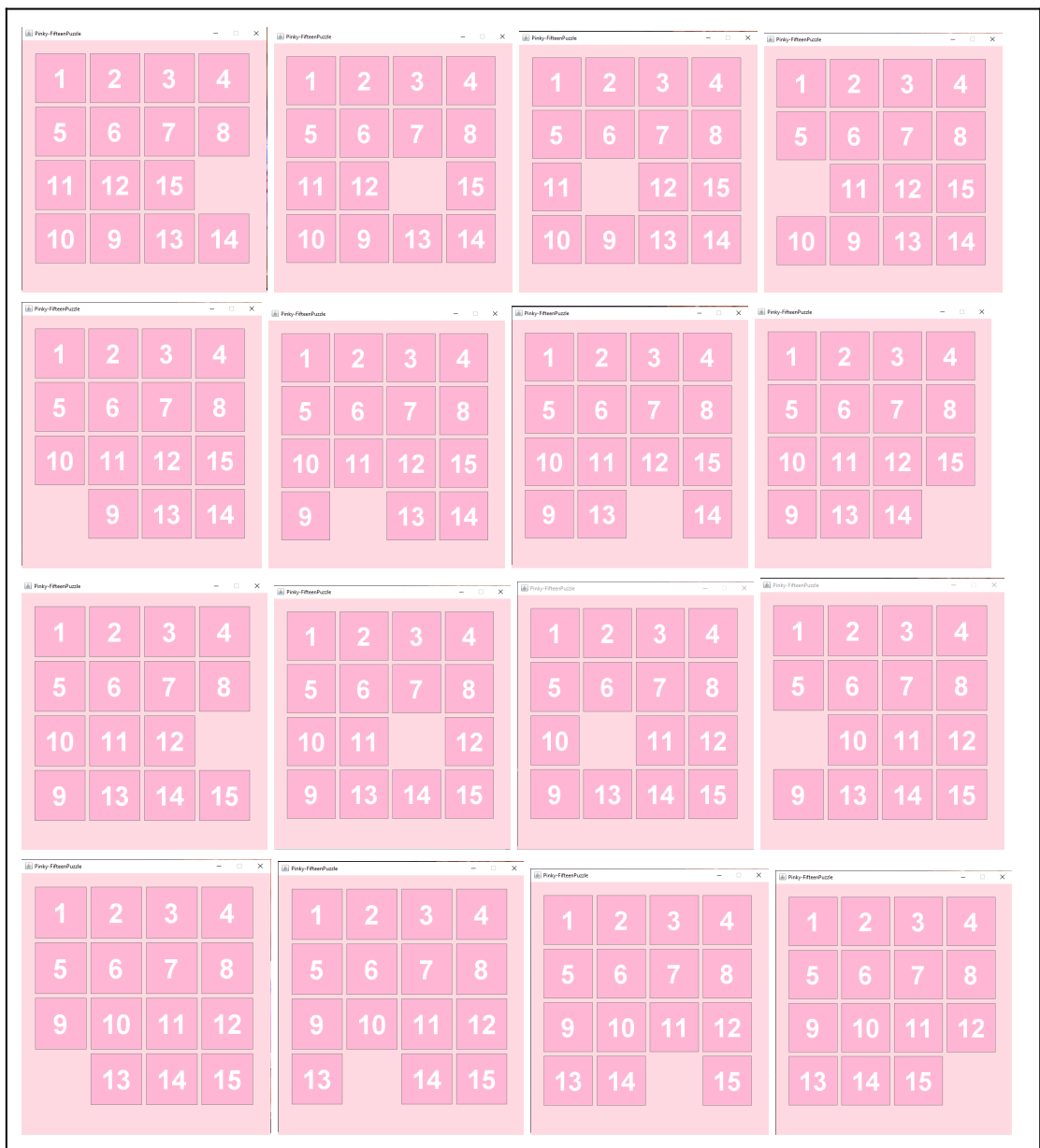


*Gambar 3.10 hasil run solve3.txt*

karena puzzle tersebut dapat diselesaikan atau *Reachable Goal* dapat dilihat bahwa nilai kurang(i) ditambah X merupakan genap. Maka puzzle tersebut dapat dicari solusinya. Berikut merupakan tampilan yang keluar jika kita mencari solusinya, melakukan command “SOLUTION”

```
C:\Windows\System32\cmd.exe - java Main
>>> SOLUTION
Time taken in nano seconds      : 4169400
jumlah simpul yang dibangkitkan : 814
command Move : up, left, left, left, down, right, right, right, up, left, left, left, down,
right, right, right
>>>
```

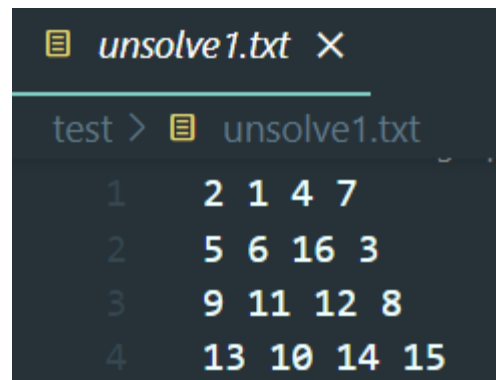
*Gambar 3.11 pencarian solusi dari puzzle solve3.txt*



*Gambar 3.12 tampilan GUI dari solusi puzzle solve3.txt dari kiri ke kanan lalu bawah*

- d. file unsolve1.txt

berikut isi dari file unsolve1.txt

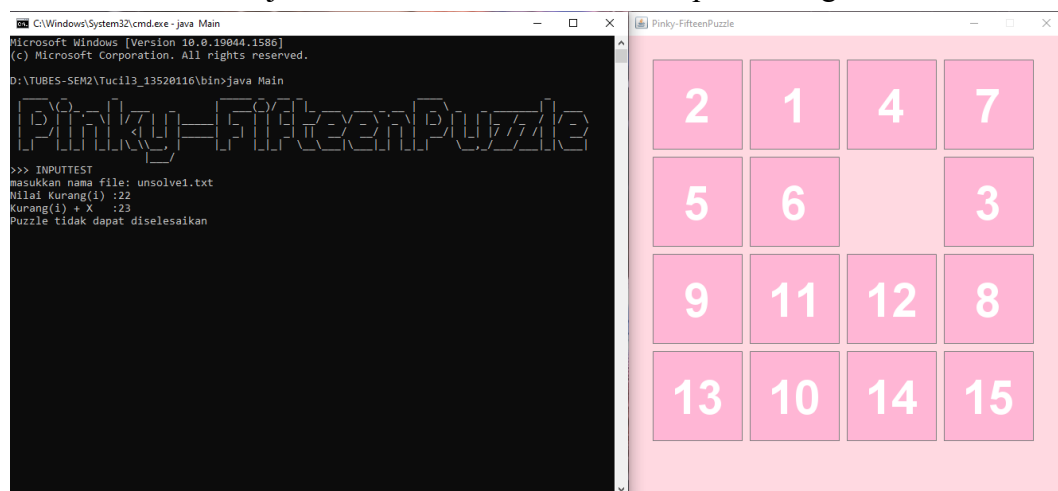


```
test > unsolve1.txt

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

*Gambar 3.13 isi file test unsolve1.txt*

pada test unsolve1.txt jika di run akan memberikan tampilan sebagai berikut

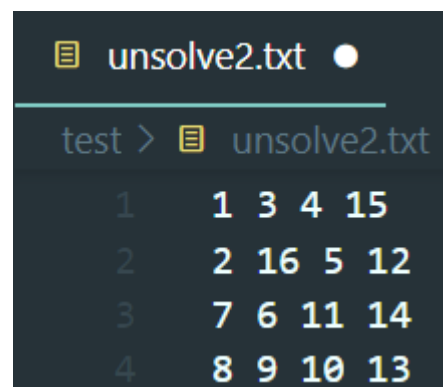


*Gambar 3.14 hasil run unsolve1.txt*

karena puzzle tersebut tidak dapat diselesaikan atau tidak *Reachable Goal* dapat dilihat bahwa nilai kurang(i) ditambah x bukanlah genap. Maka pada terminal akan ditampilkan bahwa “puzzle tidak dapat diselesaikan”.

e. file unsolve2.txt

berikut isi dari file unsolve2.txt



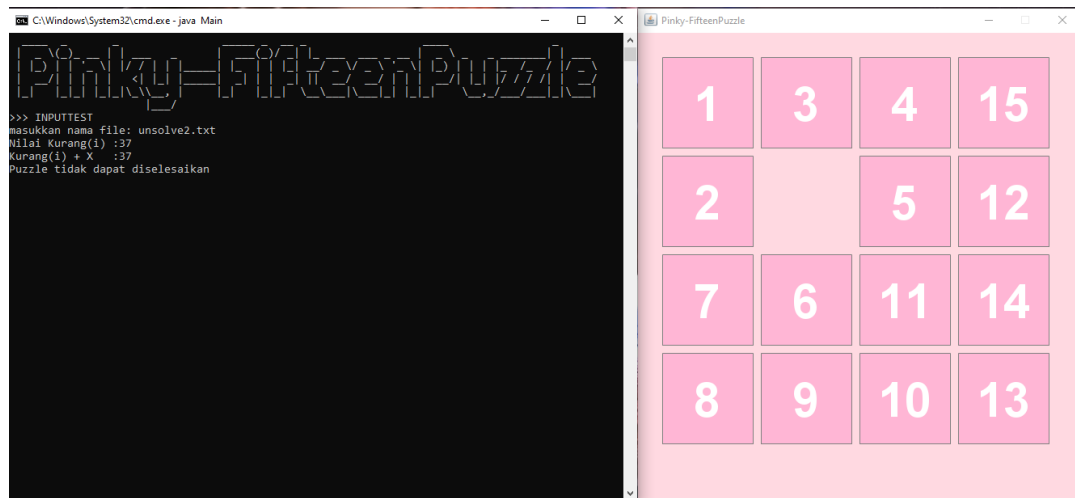
```
test > unsolve2.txt

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

*Gambar 3.15 isi file test unsolve2.txt*

pada test unsolve2.txt jika di run akan memberikan tampilan sebagai berikut





*Gambar 3.16 hasil run unsolve1.txt*

karena puzzle tersebut tidak dapat diselesaikan atau tidak *Reachable Goal* dapat dilihat bahwa nilai kurang(i) ditambah x bukanlah genap. Maka pada terminal akan ditampilkan bahwa “puzzle tidak dapat diselesaikan”.

## CHECKLIST

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

## LINK GITHUB

[https://github.com/lizardyy/Tucil3\\_13520116](https://github.com/lizardyy/Tucil3_13520116)