

**LAPORAN TUGAS KECIL IF2211 STRATEGI ALGORITMA  
SEMESTER II TAHUN 2021/2022**

**PENYELESAIAN PERSOALAN 15-PUZZLE DENGAN  
ALGORITMA *BRANCH AND BOUND***



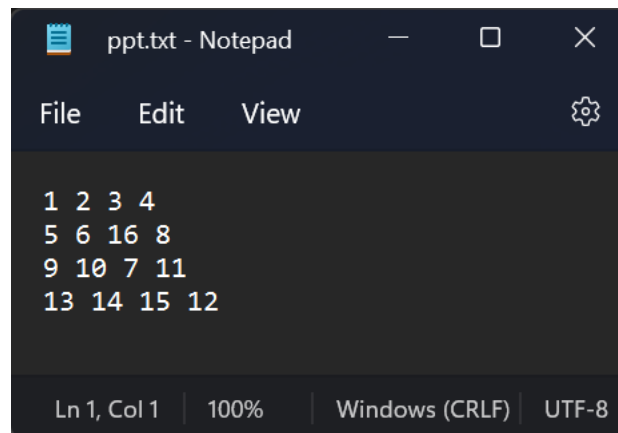
**Disusun oleh:**

**13520156 Dimas Faidh Muzaki**

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA  
INSTITUT TEKNOLOGI BANDUNG**

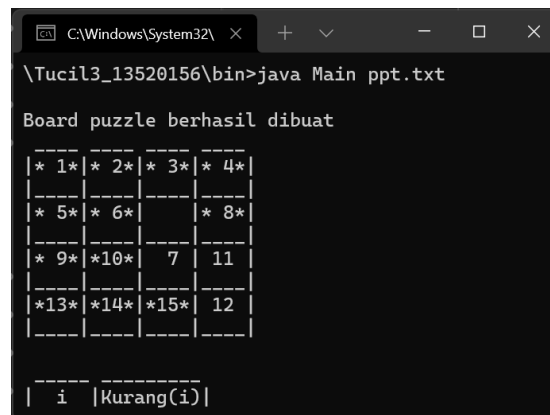
## Cara Kerja

Program dibuat menggunakan bahasa pemrograman java disusun atas empat kelas yaitu Main, Board, Pair, Puzzle Solver. Main merupakan program utama untuk menjalankan aplikasi, Board merupakan kelas yang menyimpan state permainan, Pair merupakan kelas struktur data pembantu, dan Puzzle Solver merupakan *core* dari program yang memiliki implementasi Breach and Bound di dalamnya. Dalam menyelesaikan persoalan, konfigurasi 15 Puzzle akan dimuat, dapat secara random ataupun memuat file konfigurasi. File konfigurasi adalah plain text file dengan format isi matriks 4x4 dengan elemen angka 1-16 yang masing masing hanya muncul satu kali dengan angka 16 merepresentasikan ubin kosong, sebagai contoh:



```
ppt.txt - Notepad
File Edit View
1 2 3 4
5 6 16 8
9 10 7 11
13 14 15 12
Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8
```

Setelah board dimuat, langkah selanjutnya adalah menampilkannya ke layar:



```
C:\Windows\System32\ x + - □ ×
\\Tucil3_13520156\bin>java Main ppt.txt
Board puzzle berhasil dibuat
|* 1*|* 2*|* 3*|* 4*|
|* 5*|* 6*|   |* 8*|
|* 9*|*10*| 7 | 11 |
|*13*|*14*|*15*| 12 |
|----|-----|
| i |Kurang(i)|
```

Ubin dengan bintang di kanan dan kiri angka menandakan bahwa ubin tersebut sudah pada posisinya. Selanjutnya, board akan diperiksa apakah susunan ubin tersebut dapat mencapai susuan tujuan 15 Puzzle. Caranya adalah dengan mentraversal board yang disimpan dalam struktur data matriks untuk menghitung  $\sum \text{Kurang}(i) + X$ , dengan X adalah satu jika posisi ubin kosong berada pada i genap dan nol jika posisi ubin kosong berada pada i ganjil. Board yang tidak dapat mencapai status tujuan, nilai  $\sum \text{Kurang}(i) + X$  ganjil, tidak akan dilakukan pencarian penyelesaian dengan Algoritma Breach and Bound pada dirinya. Sebaliknya, board yang dapat mencapai status tujuan akan dilakukan pencarian dengan Algoritma Breach and Bound, dengan Langkah sebagai berikut:

1. Membuat sebuah array of board bernama boardList untuk menyimpan state board dari simpul-simpul hidup
2. Board pertama/akar akan dimasukan ke sebuah Pair bernama CurrPair dengan key adalah board dan value adalah string arah dari perpindahan terakhir dari ubin kosong
3. Dari CurrPair akan dibangkitkan node-node yaitu board-board proyeksi state jika ubin kosong dipindahkan ke arah tertentu.
4. Board-board tersebut akan dihitung nilai estimasi  $c = f + g$  nya, nilai disimpan pada objek board itu sendiri.
5. Lalu board-board tersebut akan dimasukkan ke boardList
6. State/board pada boardList dengan nilai c terkecil akan dijadikan curr-pair
7. Apabila curr-pair bukan merupakan solusi, proses akan diulang dari step 3 untuk mencari state solusi.
8. Apabila curr-pair merupakan solusi, maka state tersebut akan disimpan sebagai solutionSoFar dan boardList akan ditraversal untuk dieliminasi board/state yang memiliki nilai c lebih tinggi dari c solusi
9. Jika boardList kosong, maka pencarian selesai. Jika boardList tidak kosong, akan dipilih board/state dari boardList dengan nilai c terendah untuk dijadikan currPair dan pencarian diulang dari step 3.
10. Selama pencarian, jumlah board/state yang dibangkitkan akan dicatat untuk dikembalikan ke pemanggilan fungsi solve bersama solutionSoFar
11. Tiap board juga mencatat Langkah-langkah menuju dirinya dari board root, maka dari solutionSoFar akan menampilkan Langkah-langkah yang dimilikinya yang merupakan Langkah-langkah menuju solusi.

## Screen-shot

Menggunakan contoh board di spesifikasi:

```

C:\Windows\System32\cmd.exe X + v
D:\OneDrive - Institut Teknologi Bandung\Arsip Kuliah\WOW INFORMATIKA\Semester 4 waw\Stima\Tucil3_13520156\src>make fromfile FileName
="ppt.txt"
javac Board.java PuzzleSolver.java Main.java Pair.java
java Main ppt.txt

Board puzzle berhasil dibuat

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

| i | Kurang(i) |
|---|
| 1 | 0 |
| 2 | 0 |
| 3 | 0 |
| 4 | 0 |
| 5 | 0 |
| 6 | 0 |
| 7 | 0 |
| 8 | 1 |
| 9 | 1 |
|10 | 1 |
|11 | 0 |
|12 | 0 |
|13 | 1 |

```

```

C:\Windows\System32\cmd.exe
12 | 0 |
13 | 1 |
14 | 1 |
15 | 1 |
16 | 9 |
tot | 15 |
X | 1 |
-----+
16 |
-----

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

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

* 1* * 2* * 3* * 4* |
-----

```

```

C:\Windows\System32\cmd.exe
* 1* * 2* * 3* * 4* |
-----
* 5* * 6* * 7* * 8* |
-----
* 9* *10* | 11 |
-----
*13* *14* *15* | 12 |
-----

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

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

Waktu eksekusi: 0.0024231 seconds
Jumlah node dibangkitkan: 10

D:\OneDrive - Institut Teknologi Bandung\Arsip Kuliah\WOW INFORMATIKA\Semester 4 waw\Stima\Tucil3_13520156\src>

```

Input dimasukkan sebagai argument saat pemanggilan Main menggunakan java.

## Checklist

| Poin  | Ya | Tidak |
|---|----|-------|
| 1. Program berhasil dikompilasi                       | ✓  |       |
| 2. Program berhasil <i>running</i>                    | ✓  |       |
| 3. Program dapat menerima input dan menuliskan output | ✓  |       |
| 4. Luaran sudah benar untuk semua data                | ✓  |       |

## Kode Program

```
/**
 * Main.java
 */
public class Main {
    public static void main(String[] args) {

        Board b = null;
        try {
            b = Board.fromFile(args[0]);
        } catch (Exception e) {
            if (e instanceof NoSuchFieldException) {
                System.out.println("File tidak ditemukan, membuat board
random...");
            }
            b = Board.random();
        }

        System.out.println();
        System.out.println("Board puzzle berhasil dibuat");
        b.show();

        int kurang = 0;
        System.out.println();
        System.out.println(" _____ ");
        System.out.println("|   i   |Kurang(i)|");
        System.out.println("|_____|_____|");
        for (int i = 1; i <= 16; i++) {
            int kurangi = PuzzleSolver.kurang(b, i);
            kurang += kurangi;
            if (i < 10) {
                System.out.print("|   " + i + "   |");
            } else {
                System.out.print("|  " + i + "  |");
            }
            if (kurangi < 10) {
                System.out.println("      " + kurangi + "      |");
            } else {
                System.out.println("       " + kurangi + "       |");
            }
        }
    }
}
```

```

    }
    System.out.println("| tot |      "+kurang+"      |");
    System.out.println("|_____|_____|");
    System.out.println("|  X  |      " + PuzzleSolver.xBesar(b) + "      |");
    System.out.println("|_____|_____|+");
    System.out.println("|      "+(kurang +
PuzzleSolver.xBesar(b))+      |");
    System.out.println("|_____|");

    if (!b.isSolvable()) {
        System.out.println("Status tujuan tidak dapat dicapai.");
    } else{
        long startTime = System.nanoTime();
        // ArrayList<Board> solution =
        Pair<Board,Integer> result = PuzzleSolver.solve(b);
        long stopTime = System.nanoTime();
        // 1 second = 1_000_000_000 nano seconds
        double elapsedTimeInSeconds = (double) (stopTime - startTime) /
1_000_000_000;

        b.showSolution(result.getKey().getTail());

        System.out.println("Waktu eksekusi: " + elapsedTimeInSeconds + "
seconds");
        System.out.println("Jumlah node dibangkitkan: " + result.getValue());
    }
}
}

```

```

/**
 * Board.java
 */
import java.io.BufferedReader;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.List;
import java.util.Random;
import java.util.stream.Collectors;

```

```

public class Board {
    private ArrayList<String> tail;
    private int f;
    private int c;
    private int g;
    private int N;
    private int empty_x;
    private int empty_y;
    private int[][] board;

    public Board() {
        super();
        this.tail = new ArrayList<String>();
        this.f = 0;
        this.N = 4;
        this.empty_x = 0;
        this.empty_y = 0;
        this.c = 0;
        this.g = 0;
        this.board = new int[4][4];
    }

    public Board(Board inputBoard) {
        super();
        this.tail = new ArrayList<String>();
        this.N = inputBoard.N;
        this.f = inputBoard.f;
        this.c = inputBoard.c;
        this.g = inputBoard.g;
        this.empty_x = inputBoard.empty_x;
        this.empty_y = inputBoard.empty_y;
        this.board = new int[this.N][this.N];
        for (int i = 0; i < this.N; i++) {
            for (int j = 0; j < this.N; j++) {
                this.board[i][j] = inputBoard.board[i][j];
            }
        }
    }

    public void addTail(String input) {
        this.tail.add(input);
    }

    public ArrayList<String> getTail() {
        return tail;
    }
}

```

```
}  
public int getC() {  
    return c;  
}  
public int getG() {  
    return g;  
}  
  
public int getElmt(int i, int j){  
    return this.board[i][j];  
}  
  
public int getN() {  
    return N;  
}  
  
public int getF() {  
    return f;  
}  
  
public int getEmpty_x() {  
    return empty_x;  
}  
  
public int getEmpty_y() {  
    return empty_y;  
}  
  
public void setF(int f) {  
    this.f = f;  
}  
  
public void setN(int n) {  
    N = n;  
}  
  
public void setEmpty_x(int empty_x) {  
    this.empty_x = empty_x;  
}  
  
public void setEmpty_y(int empty_y) {  
    this.empty_y = empty_y;  
}  
  
public void setElmt(int i, int j, int X){
```



```

        this.board[i][j] = X;
    }

    private void swap(int i, int j, int k, int l) {
        int temp = this.board[i][j];
        this.board[i][j] = this.board[k][l];
        this.board[k][l] = temp;
    }

    public static Board fromFile(String fileName) {
        Board A = new Board();

        String currDir = System.getProperty("user.dir");
        fileName = currDir + "\\config\\" + fileName;

        List<String> list = new ArrayList<>();

        try (BufferedReader br = Files.newBufferedReader(Paths.get(fileName))) {

            //br returns as stream and convert it into a List
            list = br.lines().collect(Collectors.toList());
            int i = 0;
            for (String string : list) {
                int j = 0;

                String[] splited = string.split("\\s+");
                for (String string2 : splited) {
                    A.board[i][j] = Integer.parseInt(string2);
                    if (Integer.parseInt(string2) == 16) {
                        A.empty_x = i;
                        A.empty_y = j;
                    }
                    j++;
                }
                i++;
            }
        } catch (IOException e) {
            e.printStackTrace();
        }

        return A;
    }

    public static Board random() {
        Board A = new Board();
    }

```

```

        ArrayList<Integer> l = new ArrayList<Integer>();
        Random rand = new Random();
        for (int i = 1; i<=16;i++){
            l.add(i);
        }
        for (int i = 0; i < 4; i++)
            for (int j = 0; j < 4; j++){
                int randomIndex = rand.nextInt(l.size());
                int randomElement = l.remove(randomIndex);
                if (randomElement == 16){
                    A.empty_x = i;
                    A.empty_y = j;
                }
                A.board[i][j] = randomElement;
            }

        return A;
    }

```

```

public void show() {
    System.out.println(" ____ ____ ____ ____");
    for (int i = 0; i < N; i++) {
        System.out.print("|");
        for (int j = 0; j < N; j++) {

            int x = this.board[i][j];
            if (x == 16) {
                System.out.print("   |");
            } else {
                if (x == (i*4)+j+1){
                    if (x < 10) {
                        System.out.print("*");
                        System.out.print(" " + x);
                        System.out.print("*|");
                    } else{
                        System.out.print("*");
                        System.out.print(x);
                        System.out.print("*|");
                    }
                }
            } else {
                if (x < 10) {
                    System.out.print(" ");
                    System.out.print(" " + x);
                    System.out.print(" |");
                } else{

```

```

        System.out.print(" ");
        System.out.print(x);
        System.out.print(" |");
    }
}

}

}
System.out.println();
System.out.println("|____|____|____|____|");
}

}

public void updateG() {
    this.g = PuzzleSolver.gTopi(this);
}

public void updateC() {
    this.updateG();
    this.c = this.f + this.g;
}

public void showSolution(ArrayList<String> list) {
    for (String string : list) {
        if (string.equals("up")) {
            this.up();
            this.show();
        } else if (string.equals("down")) {
            this.down();
            this.show();
        } else if (string.equals("left")) {
            this.left();
            this.show();
        } else if (string.equals("right")) {
            this.right();
            this.show();
        }
    }
}

}

public void up() {

```

```

        swap(empty_x, empty_y, empty_x-1, empty_y);
        empty_x--;
        this.updateC();
    }

    public void down() {
        swap(empty_x, empty_y, empty_x+1, empty_y);
        empty_x++;
        this.updateC();
    }

    public void left() {
        swap(empty_x, empty_y, empty_x, empty_y-1);
        empty_y--;
        this.updateC();
    }

    public void right() {
        swap(empty_x, empty_y, empty_x, empty_y+1);
        empty_y++;
        this.updateC();
    }

    public boolean isSolvable() {
        int kurang = 0;
        for (int i = 1; i <= 16; i++) {
            int kurangi = PuzzleSolver.kurang(this, i);
            kurang += kurangi;
        }
        kurang += PuzzleSolver.xBesar(this);
        return (kurang % 2 == 0);
    }

    public boolean equals(Board b){
        for (int i = 0; i < N; i++) {
            for (int j = 0; j < N; j++) {
                if (this.board[i][j] != b.board[i][j]){
                    return false;
                }
            }
        }
        return true;
    }
}

```

```

}

/**
 * Pair.java
 */
public class Pair<T,B> {
    T key;
    B value;

    public Pair() {
        super();
        this.key = null;
        this.value = null;
    }

    public Pair(T key, B value) {
        super();
        this.key = key;
        this.value = value;
    }

    public T getKey() {
        return key;
    }

    public B getValue() {
        return value;
    }
}

```

```

/**
 * PuzzleSolver.java
 */
import java.util.ArrayList;

public class PuzzleSolver {

    public static int kurang(Board b, int x) {
        boolean start = false;
        int counter = 0;
        for (int i = 0; i < b.getN(); i++) {
            for (int j = 0; j < b.getN(); j++) {

```

```

        if (b.getElmt(i, j) == x){
            start = true;
        }
        if(start && b.getElmt(i, j) < x && b.getElmt(i, j) != 0){
            counter++;
        }
    }
}
return counter;
}

public static int xBesar(Board b) {
    int i = 0;
    int j = 0;
    int ires = 0;
    int jres = 0;
    for (i = 0; i < b.getN(); i++) {
        for (j = 0; j < b.getN(); j++) {
            if (b.getElmt(i, j) == 16){
                ires = i;
                jres = j;
            }
        }
    }
    return (((ires + jres) % 2) == 0 ? 0: 1);
}

public static int gTopi(Board b) {
    int counter = 0;
    for (int i = 0; i < b.getN(); i++) {
        for (int j = 0; j < b.getN(); j++) {
            // System.out.println(Integer.toString(b.getElmt(i, j)) + " == "
+ Integer.toString((i*4)+j+1));
            if(b.getElmt(i, j) != (i*4)+j+1 && b.getElmt(i, j) != 16)
                counter++;
        }
    }
    return counter;
}

public static int fTopi(Board b) {
    return b.getF();
}

public static int cTopi(Board b) {

```

```

        return gTopi(b) + fTopi(b);
    }

    public static boolean isSolution(Board b) {
        return gTopi(b) == 0;
    }

    public static int minFour(int one, int two, int three, int four) {
        return Math.min(one, Math.min(two, Math.min(three, four)));
    }

    public static int findMin(ArrayList<Pair<Board, String>> list) {
        int min = Integer.MAX_VALUE;
        int idx = 0;
        int i = 0;
        for (Pair<Board, String> pair : list) {
            if (pair.getKey().getC() < min) {
                min = pair.getKey().getC();
                idx = i;
            }
            i++;
        }
        return idx;
    }

    public static boolean isBoardEqual(Pair<Board, String> b1, Pair<Board,
String> b2) {
        return b1.getKey().equals(b2.getKey());
    }

    public static boolean has(ArrayList<Pair<Board, String>> list, Pair<Board,
String> b) {
        for (Pair<Board, String> pair : list) {
            if (isBoardEqual(pair, b)) {
                return true;
            }
        }
        return false;
    }

    public static Pair<Board, Integer> solve(Board b){
        ArrayList<Pair<Board,String>> boardList = new
ArrayList<Pair<Board,String>>();
        Board currBoard = new Board(b);
        boolean end = false;

```

```

Pair<Board, String> currPair = new Pair<Board, String>(currBoard, "");
int nodeCount = 1;

while (!end && currPair.getKey().isSolvable()) {
    ArrayList<Pair<Board, String>> localBoardList = new
ArrayList<Pair<Board, String>>();
    Pair<Board, String> upPair;
    Pair<Board, String> rightPair;
    Pair<Board, String> leftPair;
    Pair<Board, String> downPair;
    if (!(currPair.getKey().getEmpty_x() == 0 ||
currPair.getValue().equals("down"))) {
        upPair = new Pair<Board, String>(new Board(currPair.getKey()),
"up");

        upPair.getKey().setF(currPair.getKey().getF() + 1);
        for (String path : currPair.getKey().getTail()) {
            upPair.getKey().addTail(path);
        }
        upPair.getKey().addTail("up");
        upPair.getKey().up();
        localBoardList.add(upPair);
        nodeCount++;

        // if (!has(boardList, upPair)) {
        //     localBoardList.add(upPair);
        // }
    }
    if (!(currPair.getKey().getEmpty_x() == currPair.getKey().getN() - 1
|| currPair.getValue().equals("up"))) {
        downPair = new Pair<Board, String>(new Board(currPair.getKey()),
"down");

        downPair.getKey().setF(currPair.getKey().getF() + 1);
        for (String path : currPair.getKey().getTail()) {
            downPair.getKey().addTail(path);
        }
        downPair.getKey().addTail("down");
        downPair.getKey().down();
        localBoardList.add(downPair);
        nodeCount++;

        // if (!has(boardList, downPair)) {
        //     localBoardList.add(downPair);
        // }
    }
}

```



```

        if (!(currPair.getKey().getEmpty_y() == 0 ||
currPair.getValue().equals("right"))) {
            leftPair = new Pair<Board, String>(new Board(currPair.getKey()),
"left");

            leftPair.getKey().setF(currPair.getKey().getF() + 1);
            for (String path : currPair.getKey().getTail()) {
                leftPair.getKey().addTail(path);
            }
            leftPair.getKey().addTail("left");
            leftPair.getKey().left();
            localBoardList.add(leftPair);
            nodeCount++;

            // if (!has(boardList, leftPair)) {
            //     localBoardList.add(leftPair);
            // }
        }
        if (!(currPair.getKey().getEmpty_y() == currPair.getKey().getN() - 1
|| currPair.getValue().equals("left"))) {
            rightPair = new Pair<Board, String>(new Board(currPair.getKey()),
"right");

            rightPair.getKey().setF(currPair.getKey().getF() + 1);
            for (String path : currPair.getKey().getTail()) {
                rightPair.getKey().addTail(path);
            }
            rightPair.getKey().addTail("right");
            rightPair.getKey().right();
            localBoardList.add(rightPair);
            nodeCount++;

            // if (!has(boardList, rightPair)) {
            //     localBoardList.add(rightPair);
            // }
        }

        for (Pair<Board, String> pair : localBoardList) {
            boardList.add(pair);
        }

        int iMin = PuzzleSolver.findMin(boardList);
        currPair = boardList.get(iMin);
        boardList.remove(iMin);

```

```

        localBoardList.clear();
        if (isSolution(currPair.getKey())) {
            ArrayList<Pair<Board,String>> fooListCopy = new
ArrayList<Pair<Board,String>>();
            for (Pair<Board,String> pair : boardList) {
                if (pair.getKey().getC() < currPair.getKey().getC()) {
                    fooListCopy.add(pair);
                }
            }

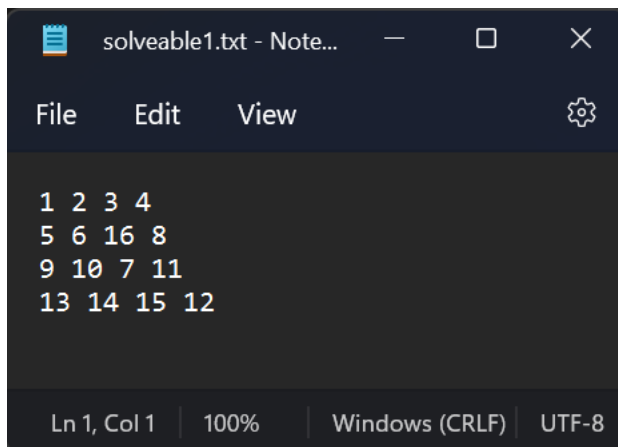
            boardList = fooListCopy;

            if(boardList.isEmpty()){
                end = true;
            }
            else{
                currPair = boardList.get(PuzzleSolver.findMin(boardList));
            }
        }
    }
    return new Pair<Board,Integer>(currPair.getKey(), nodeCount);
}
}

```

## ***Test Case***

1. Solveable1.txt



```

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

```

Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8

2. Solveable2.txt

```
solveable2.txt - Note...
File Edit View
2 3 4 11
1 5 10 8
9 6 12 15
13 14 16 7
Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8
```

3. Solveable3.txt

```
solveable3.txt - Notepad
File Edit View
1 3 7 6
5 2 16 4
13 10 12 8
14 9 11 15
Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8
```

4. Unsolveable1.txt

```
unsolvable1.txt - Notepad
File Edit View
1 3 4 15
2 16 5 12
7 6 11 14
8 9 10 13
Ln 4, Col 10 | 100% | Windows (CRLF) | UTF-8
```

5. Unsolveable2.txt

```
unsolvable2.txt - Notepad
File Edit View
5 14 1 8
15 3 2 10
12 6 9 11
16 7 4 13
Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8
```

### ***Link Kode Program***

<https://github.com/maspaitujaki/15PuzzleSolver.git>