LAPORAN TUGAS KECIL
IF2211 STRATEGI ALGORITMA

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

Disusun oleh:

Wesly Giovano 13520071

TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2022

DAFTAR ISI

1 Algoritma Branch and Bound untuk Persoalan 15-Puzzle

Langkah-langkah algoritma *branch and bound* dalam implementasi penyelesaian persoalan 15-puzzle adalah sebagai berikut.

1. Bangkitkan semua kemungkinan pergerakan yang satu langkah dari puzzle yang akan diselesaikan.
2. Hitung dan simpan cost untuk masing-masing puzzle yang telah dibangkitkan.
3. Masukkan semua puzzle yang telah dibangkitkan ke dalam *priority queue* berdasarkan cost, terurut dari cost terkecil.
4. Keluarkan sebuah puzzle yang berada di posisi terdepan pada *priority queue*; jika puzzle telah sesuai dengan *goal* maka algoritma selesai, jika belum maka ulangi dari nomor 1 untuk puzzle yang dikeluarkan sebelumnya.

2 Source Code Program

Program ini diimplementasikan dalam bahasa Java dengan empat buah kelas, yaitu Matrix, PuzzlePrioQueue, FifteenPuzzle, dan Main. Seluruh source code program disimpan dalam repository git dengan alamat https://github.com/weslygio/Tucil3_13520071.git.

2.1 Matrix

Kelas Matrix adalah sebuah kelas yang tidak memiliki atribut, melainkan hanya static method. Fungsi dari kelas Matrix adalah untuk melakukan operasi terhadap sebuah *array of array of integer* (*matrix of integer*), yaitu membaca dari file, melakukan output, dan membentuk matriks sembarang. Berikut adalah implementasi dari kelas Matrix.

```
import java.io.*;
import java.util.*;
import java.util.concurrent.*;

public class Matrix {
    /* A class without attributes: contains static int[][] operations
     */

    public static int[][] read_matrix_from_file(String filepath, int rows,
int cols) {
        int[][] matrix = new int[rows][cols];
        try {
            File matFile = new File(filepath);
            Scanner fileRead = new Scanner(matFile);

            for (int i = 0; i < rows; i++) {
```

```java
                String data = fileRead.nextLine();
                Scanner lineRead = new Scanner(data);
                for (int j = 0; j < cols; j++) {
                    matrix[i][j] = lineRead.nextInt();
                }
                lineRead.close();
            }
            fileRead.close();
        } catch (FileNotFoundException e) {
            System.out.println("An error occurred.");
            e.printStackTrace();
        }
        return matrix;
    }

    public static void write_matrix(int[][] matrix) {
        for (int i = 0; i < matrix.length; i++) {
            for (int j = 0; j < matrix[i].length; j++) {
                System.out.printf("%2d", matrix[i][j]);
                if (j != matrix[i].length-1)
                    System.out.print(" ");
            }
            System.out.println();
        }
    }

    // Randomize a matrix from 0 to (rows * cols - 1)
    public static int[][] random(int rows, int cols) {
        int[][] matrix = new int[rows][cols];
        List<Integer> availableElements = new ArrayList<>();

        for (int i = 0; i < rows*cols; i++) {
            availableElements.add(i);
        }

        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                int idx = ThreadLocalRandom.current().nextInt(0,
availableElements.size());
                matrix[i][j] = availableElements.get(idx);
                availableElements.remove(idx);
            }
        }

        return matrix;
    }

}
```

2.2 PuzzlePrioQueue

Kelas PuzzlePrioQueue adalah sebuah kelas dengan atribut utama yaitu *priority queue of FifteenPuzzle*, sedemikian sehingga queue selalu terurut dari FifteenPuzzle yang memiliki cost terkecil hingga terbesar. Berikut adalah implementasi dari kelas PuzzlePrioQueue.

```
import java.util.ArrayList;

class PuzzlePrioQueue {
    /* PuzzlePrioQueue is a priority queue of FifteenPuzzle-type elements.
     * The queue is sorted in ascending value of cost of the elements.
     */

    private ArrayList<FifteenPuzzle> queue;
    private int length;

    public PuzzlePrioQueue() {
        queue = new ArrayList<>();
        length = 0;
    }

    public int getLength() {
        return length;
    }

    // Insert puzzle to queue just before the next puzzle that has
    // higher cost
    public void push(FifteenPuzzle puzzle) {
        int i = 0;
        boolean found = false;
        while (i < length && !found)
            if (queue.get(i).getCost() > puzzle.getCost())
                found = true;
            else
                i += 1;
        queue.add(i, puzzle);
        length += 1;
    }

    // Remove and return puzzle from front-most element of queue
    public FifteenPuzzle pop() {
        length -= 1;
        return queue.remove(0);
    }

    // Pruning assumption: we only want a single solution
    public void prune() {
        queue.clear();
        length = 0;
    }
}
```

2.3 FifteenPuzzle

Kelas FifteenPuzzle adalah kelas utama dari problema yang diberikan. Kelas FifteenPuzzle utamanya menyimpan status posisi puzzle saat ini dalam tipe `int[][]`, perkiraan *cost* dari status puzzle, dan *path* yang telah dilalui dari simpul akar untuk mencapai status saat ini. Fungsi utama dari kelas ini adalah untuk menyelesaikan persoalan 15-puzzle. Berikut adalah implementasi dari kelas FifteenPuzzle.

```
import java.util.*;

public class FifteenPuzzle {
    /* FifteenPuzzle is a class of fifteen puzzle game, or generally a
     * sliding game with custom size of rows x cols and custom goal
     * position.
     *
     * Main function of this class is to solve fifteen puzzle problem with
     * branch and bound algorithm.
     */

    private static final int rows = 4;
    private static final int cols = 4;
    private static final int[][] goalMatrix =
        {{  1 ,  2 ,  3 ,  4 },
         {  5 ,  6 ,  7 ,  8 },
         {  9 , 10 , 11 , 12 },
         { 13 , 14 , 15 ,  0 }};

    private static final PuzzlePrioQueue queue = new PuzzlePrioQueue();
    private static int nodesCount = 0;

    private final int[][] matrix;
    private int blankRow;
    private int blankCol;
    private final int level;
    private final int cost;
    private ArrayList<String> path;      // Path (moves) traversed from
                                         // initial state to reach
                                         // current state.
    private final String illegalMove;    // Move that is the reverse of last
                                         // move to reach current state
                                         // to prevent moving back and forth.

    // Constructor for standard FifteenPuzzle
    public FifteenPuzzle(int[][] matrix) {
        this(matrix, 0, 0, 0, null);

        // Calculate blankRow and blankCol
        boolean found = false;
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                if (matrix[i][j] == 0) {
                    blankRow = i; blankCol = j;
                    found = true;
```

```
                        break;
                    }
                }
                if (found)
                    break;
            }
        }

    // Constructor for custom FifteenPuzzle
    private FifteenPuzzle(int[][] matrix, int blankRow, int blankCol, int
level, String illegalMove) {
        // Refresh generatedPuzzleCount
        if (level == 0)
            nodesCount = 1;

        this.matrix = matrix;
        this.blankRow = blankRow;
        this.blankCol = blankCol;
        this.level = level;
        this.cost = costEstimator();
        this.path = new ArrayList<>();
        this.illegalMove = illegalMove;
    }

    public int getCost() {
        return cost;
    }

    public ArrayList<String> getPath() {
        return path;
    }

    public int[][] getMatrix() {
        return matrix;
    }

    public static int getNodesCount() {
        return nodesCount;
    }

    // Returns number of lower numbered tiles
    // or equivalently in lecture slide "KURANG(i)"
    public int lowerNumberedTiles(int x) {
        int count = 0;
        boolean found = false;
        int val;

        if (x == 0)
            x = rows * cols;

        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                // Ignore blank tiles
                if (matrix[i][j] == 0)
                    val = rows*cols;
                else
                    val = matrix[i][j];
                if (val == x)
                    found = true;
                else if (found && val < x)
                    count += 1;
            }
        }

        return count;
    }

    // Returns the value of sum of lower numbered tiles added by 0/1
    // based on blank position
    public int isSolvableValue() {
        int val = 0;
        for (int i = 1; i <= rows*cols; i++)
            val += lowerNumberedTiles(i);
        val += (blankRow + blankCol) % 2;
        return val;
    }

    // Returns true if the puzzle is solvable
    public boolean isSolvable() {
        return isSolvableValue() % 2 == 0;
    }

    // Returns true if the puzzle's position is equivalent of goal's
    private boolean equalGoal() {
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                if (matrix[i][j] != goalMatrix[i][j])
                    return false;
            }
        }
        return true;
    }

    // Returns the estimated cost to the goal
    private int costEstimator() {
        int misplacedCount = 0;
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                if (matrix[i][j] != 0)
                    if (matrix[i][j] != goalMatrix[i][j])
                        misplacedCount += 1;
            }
        }
        return level + misplacedCount;
    }

    // Return a new FifteenPuzzle that has been moved from current position
    public FifteenPuzzle move(String moveName) {
        int[][] newMatrix = new int[rows][cols];
        int newBlankRow = blankRow;
        int newBlankCol = blankCol;
        String newIllegalMove = null;

        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                newMatrix[i][j] = matrix[i][j];
            }
        }

        switch (moveName) {
            case "up" -> {
                newBlankRow = blankRow - 1;
                if (newBlankRow < 0 || newBlankRow >= rows)
                    return null;
                newIllegalMove = "down";
            }
            case "down" -> {
                newBlankRow = blankRow + 1;
                if (newBlankRow < 0 || newBlankRow >= rows)
                    return null;
                newIllegalMove = "up";
            }
            case "left" -> {
                newBlankCol = blankCol - 1;
                if (newBlankCol < 0 || newBlankCol >= cols)
                    return null;
                newIllegalMove = "right";
            }
            case "right" -> {
                newBlankCol = blankCol + 1;
                if (newBlankCol < 0 || newBlankCol >= cols)
                    return null;
                newIllegalMove = "left";
            }
        }

        newMatrix[blankRow][blankCol] = matrix[newBlankRow][newBlankCol];
        newMatrix[newBlankRow][newBlankCol] = 0;

        FifteenPuzzle newPuzzle = new FifteenPuzzle(newMatrix, newBlankRow,
newBlankCol, level+1, newIllegalMove);
        newPuzzle.path = new ArrayList<>(this.path);
        newPuzzle.path.add(moveName);
        nodesCount += 1;
        return newPuzzle;
    }

    // Return new FifteenPuzzle that is solved
    // Moves taken to reach the new FifteenPuzzle can be accessed with
    // getPath()
    public FifteenPuzzle solve() {
        queue.push(this);

        FifteenPuzzle puzzle = null, newPuzzle;
        while (queue.getLength() > 0) {
            puzzle = queue.pop();
            if (puzzle.equalGoal())
                queue.prune();
            else {
                ArrayList<String> possibleMoves = new
ArrayList<>(Arrays.asList("up", "down", "left", "right"));
```

```
                if (puzzle.illegalMove != null)
                    possibleMoves.remove(puzzle.illegalMove);

                for (String moveName : possibleMoves) {
                    newPuzzle = puzzle.move(moveName);
                    if (newPuzzle != null) {
                        queue.push(newPuzzle);
                    }
                }
            }
        }

        return puzzle;
    }
}
```

2.4 Main

Kelas Main adalah kelas yang berfungsi sebagai *driver* dan antarmuka dengan pengguna. Pengguna dapat memasukkan input puzzle melalui file maupun puzzle yang didapat secara acak, kemudian mendapatkan output langkah-langkah menyelesaikan puzzle yang diberikan. Berikut adalah implementasi dari kelas Main.

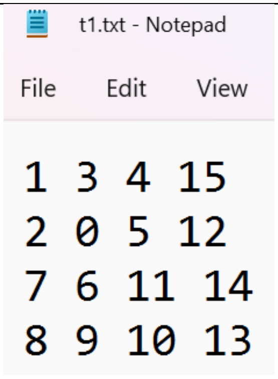
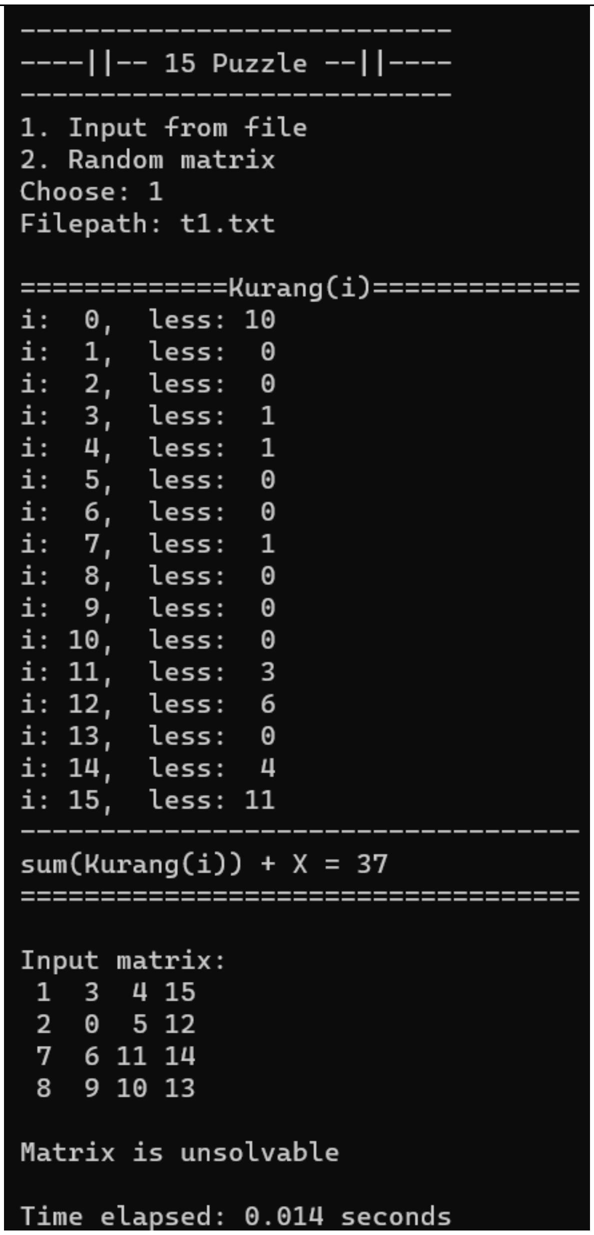
```
import java.time.*;
import java.util.*;

public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int[][] mat;

        System.out.println("----------------------------");
        System.out.println("----||-- 15 Puzzle --||----");
        System.out.println("----------------------------");

        System.out.println("1. Input from file");
        System.out.println("2. Random matrix");

        System.out.print("Choose: ");
        String c1 = in.nextLine();
        if (c1.equals("1")) {
            System.out.print("Filepath: ");
            String fp = in.nextLine();
            mat = Matrix.read_matrix_from_file("test/" + fp, 4, 4);
        }
        else
            mat = Matrix.random(4, 4);

        in.close();
```

```java
        Instant start = Instant.now();

        FifteenPuzzle p = new FifteenPuzzle(mat);

        System.out.println();
        System.out.println("=============Kurang(i)=============");
        for (int i = 0; i < 16; i++)
            System.out.printf("i: %2d,  less: %2d\n", i,
                                p.lowerNumberedTiles(i)  );
        System.out.println("----------------------------------");
        System.out.printf("sum(Kurang(i)) + X = %d\n", p.isSolvableValue());
        System.out.println("==================================");
        System.out.println();
        System.out.println("Input matrix:");
        Matrix.write_matrix(p.getMatrix());

        if (p.isSolvable()) {
            Instant startSolve = Instant.now();
            ArrayList<String> path = p.solve().getPath();
            Instant endSolve = Instant.now();
            int count = FifteenPuzzle.getNodesCount();
            for (String move : path) {
                System.out.println();
                System.out.printf("Move: %s\n", move);
                p = p.move(move);
                Matrix.write_matrix(p.getMatrix());
            }
            System.out.println();
            System.out.printf("Number of moves: %d\n", path.size());
            System.out.print("Moves list: "); System.out.println(path);
            System.out.printf("Generated nodes count: %d\n", count);
            System.out.println();
            System.out.printf("Solving time: %.3f seconds\n", (double)
                Duration.between(startSolve, endSolve).toMillis()/1000);
        }
        else {
            System.out.println();
            System.out.println("Matrix is unsolvable\n");
        }

        Instant end = Instant.now();

        System.out.printf("Time elapsed: %.3f seconds\n", (double)
                           Duration.between(start, end).toMillis()/1000);
    }
}
```

3 Eksperimen

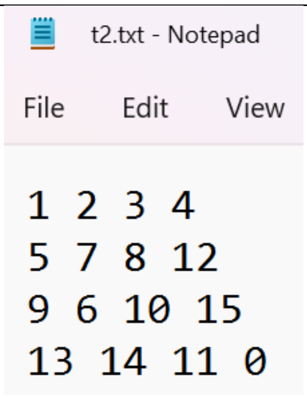
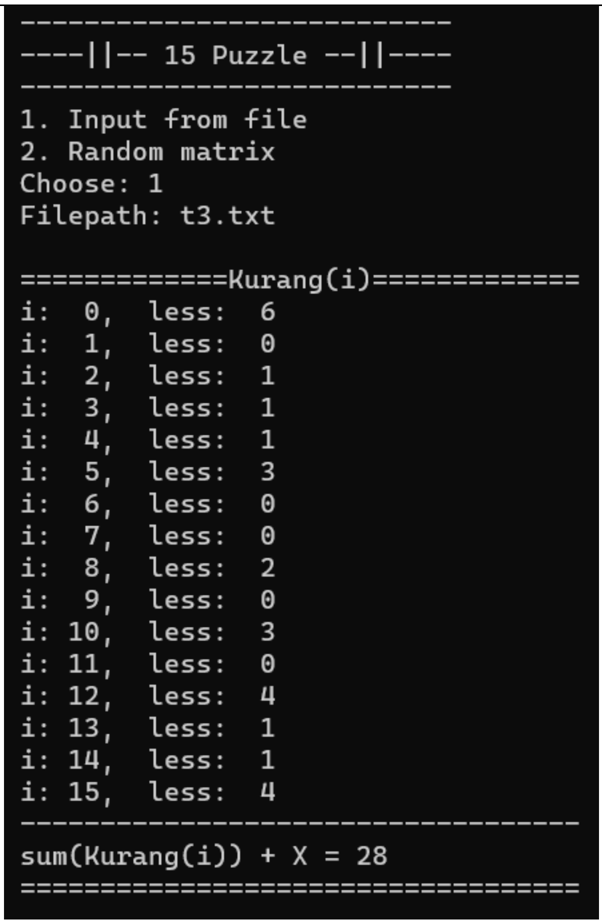
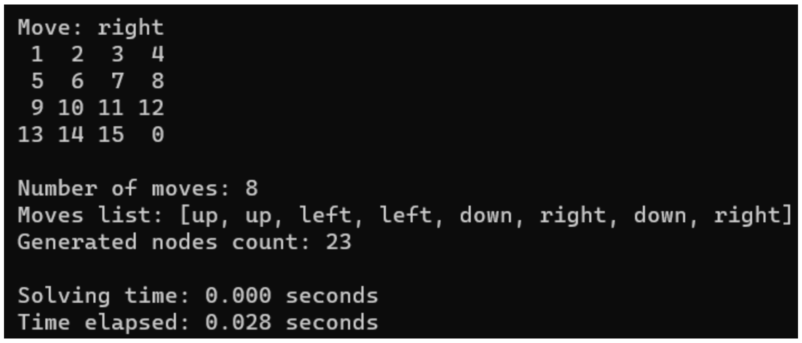
Eksperimen dilakukan terhadap lima buah persoalan 15-puzzle yang terdapat dalam folder `test` secara berurutan dari `t1.txt` hingga `t5.txt`. Berikut adalah hasil eksperimen dalam bentuk *screenshot* input dan output.

3.1 Eksperimen I

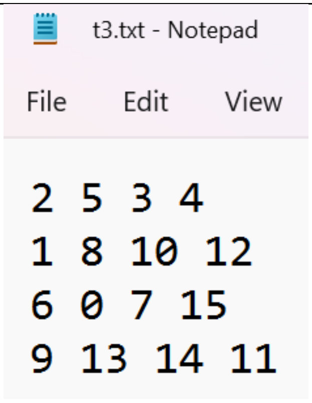
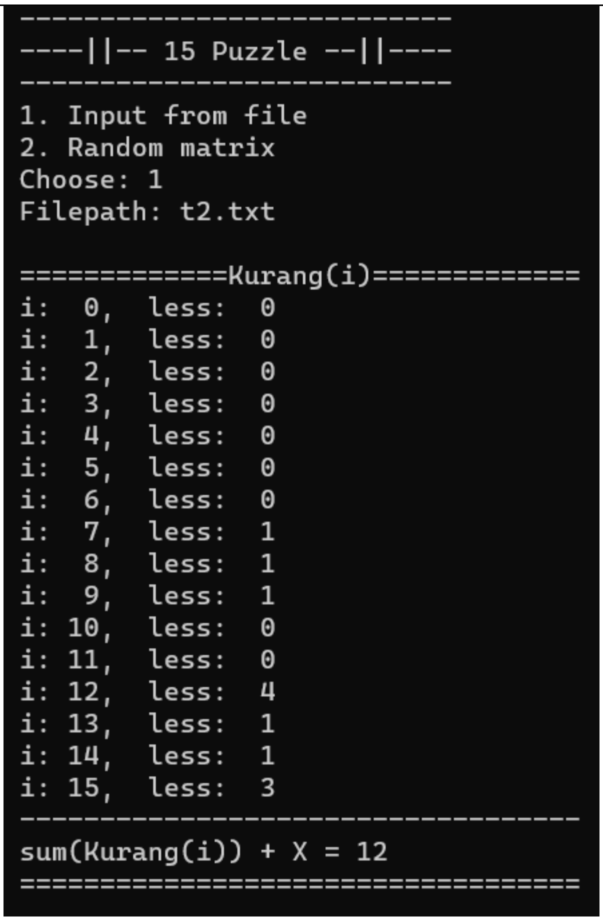
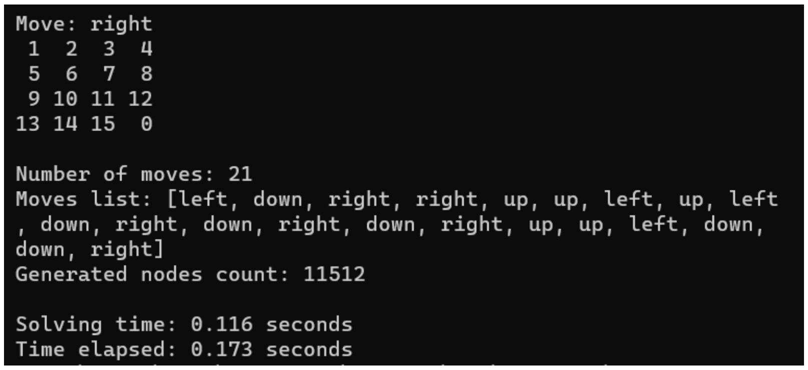
Isi Input File	Antarmuka Program
 t1.txt - Notepad File Edit View 1 3 4 15 2 0 5 12 7 6 11 14 8 9 10 13	 (terminal output below)

```
---------------------------
----||-- 15 Puzzle --||----
---------------------------
1. Input from file
2. Random matrix
Choose: 1
Filepath: t1.txt

============Kurang(i)============
i:  0,  less: 10
i:  1,  less:  0
i:  2,  less:  0
i:  3,  less:  1
i:  4,  less:  1
i:  5,  less:  0
i:  6,  less:  0
i:  7,  less:  1
i:  8,  less:  0
i:  9,  less:  0
i: 10,  less:  0
i: 11,  less:  3
i: 12,  less:  6
i: 13,  less:  0
i: 14,  less:  4
i: 15,  less: 11
-----------------------------------
sum(Kurang(i)) + X = 37
===================================

Input matrix:
 1  3  4 15
 2  0  5 12
 7  6 11 14
 8  9 10 13

Matrix is unsolvable

Time elapsed: 0.014 seconds
```

3.2 Eksperimen II

Isi Input File	Antarmuka Program
 t2.txt - Notepad File Edit View 1 2 3 4 5 7 8 12 9 6 10 15 13 14 11 0	 `----||-- 15 Puzzle --||----` 1. Input from file 2. Random matrix Choose: 1 Filepath: t3.txt `=============Kurang(i)=============` i: 0, less: 6 i: 1, less: 0 i: 2, less: 1 i: 3, less: 1 i: 4, less: 1 i: 5, less: 3 i: 6, less: 0 i: 7, less: 0 i: 8, less: 2 i: 9, less: 0 i: 10, less: 3 i: 11, less: 0 i: 12, less: 4 i: 13, less: 1 i: 14, less: 1 i: 15, less: 4 sum(Kurang(i)) + X = 28 …  Move: right 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 0 Number of moves: 8 Moves list: [up, up, left, left, down, right, down, right] Generated nodes count: 23 Solving time: 0.000 seconds Time elapsed: 0.028 seconds

3.3 Eksperimen III

Isi Input File	Antarmuka Program
	 ... 

t3.txt - Notepad

File Edit View

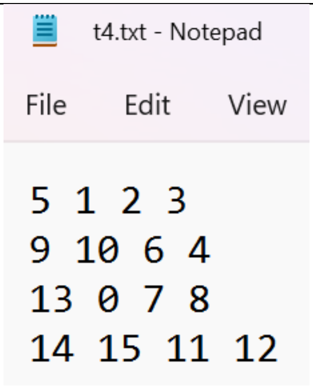
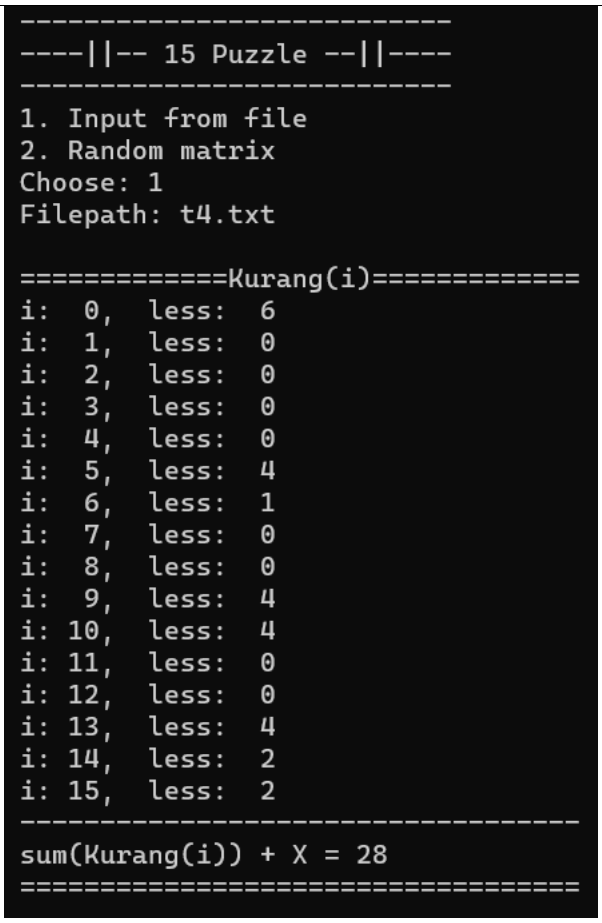
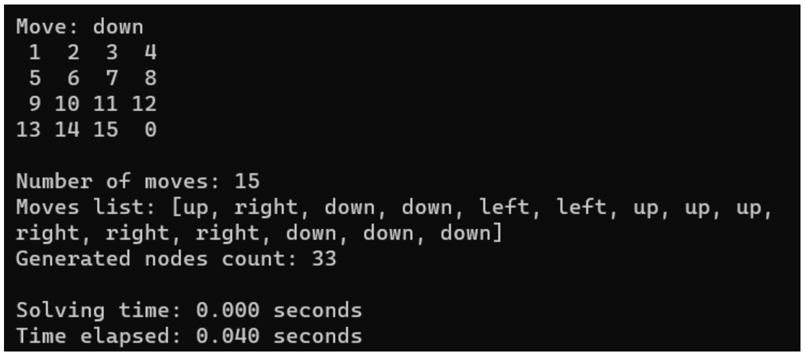
```
2 5 3 4
1 8 10 12
6 0 7 15
9 13 14 11
```

```
---------------------------
----||-- 15 Puzzle --||----
---------------------------
1. Input from file
2. Random matrix
Choose: 1
Filepath: t2.txt

=============Kurang(i)=============
i:  0,  less:  0
i:  1,  less:  0
i:  2,  less:  0
i:  3,  less:  0
i:  4,  less:  0
i:  5,  less:  0
i:  6,  less:  0
i:  7,  less:  1
i:  8,  less:  1
i:  9,  less:  1
i: 10,  less:  0
i: 11,  less:  0
i: 12,  less:  4
i: 13,  less:  1
i: 14,  less:  1
i: 15,  less:  3
-----------------------------------
sum(Kurang(i)) + X = 12
===================================
```

...

```
Move: right
 1  2  3  4
 5  6  7  8
 9 10 11 12
13 14 15  0

Number of moves: 21
Moves list: [left, down, right, right, up, up, left, up, left
, down, right, down, right, down, right, up, up, left, down,
down, right]
Generated nodes count: 11512

Solving time: 0.116 seconds
Time elapsed: 0.173 seconds
```

3.4 Eksperimen IV

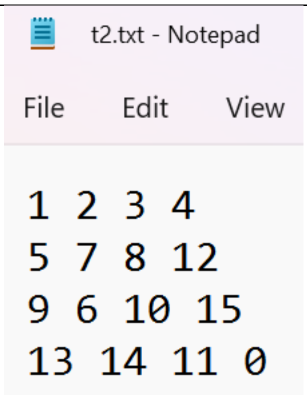
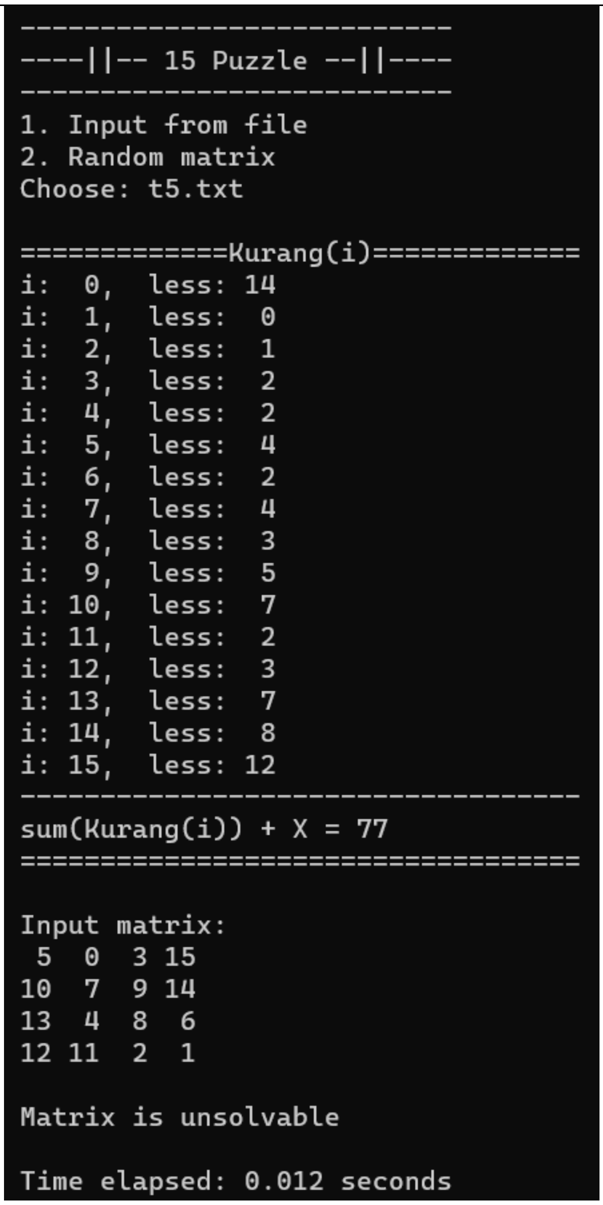
Isi Input File	Antarmuka Program
 t4.txt - Notepad File Edit View 5 1 2 3 9 10 6 4 13 0 7 8 14 15 11 12	 ... 

```
----------------------------
----||-- 15 Puzzle --||----
----------------------------
1. Input from file
2. Random matrix
Choose: 1
Filepath: t4.txt

=============Kurang(i)=============
i:  0,  less:  6
i:  1,  less:  0
i:  2,  less:  0
i:  3,  less:  0
i:  4,  less:  0
i:  5,  less:  4
i:  6,  less:  1
i:  7,  less:  0
i:  8,  less:  0
i:  9,  less:  4
i: 10,  less:  4
i: 11,  less:  0
i: 12,  less:  0
i: 13,  less:  4
i: 14,  less:  2
i: 15,  less:  2
-----------------------------------
sum(Kurang(i)) + X = 28
===================================
```

...

```
Move: down
 1  2  3  4
 5  6  7  8
 9 10 11 12
13 14 15  0

Number of moves: 15
Moves list: [up, right, down, down, left, left, up, up, up,
right, right, right, down, down, down]
Generated nodes count: 33

Solving time: 0.000 seconds
Time elapsed: 0.040 seconds
```

3.5 Eksperimen V

Isi Input File	Antarmuka Program
 t2.txt - Notepad File Edit View 1 2 3 4 5 7 8 12 9 6 10 15 13 14 11 0	

```
---------------------------
----||-- 15 Puzzle --||----
---------------------------
1. Input from file
2. Random matrix
Choose: t5.txt

=============Kurang(i)=============
i:  0,  less: 14
i:  1,  less:  0
i:  2,  less:  1
i:  3,  less:  2
i:  4,  less:  2
i:  5,  less:  4
i:  6,  less:  2
i:  7,  less:  4
i:  8,  less:  3
i:  9,  less:  5
i: 10,  less:  7
i: 11,  less:  2
i: 12,  less:  3
i: 13,  less:  7
i: 14,  less:  8
i: 15,  less: 12
-----------------------------------
sum(Kurang(i)) + X = 77
===================================

Input matrix:
 5  0  3 15
10  7  9 14
13  4  8  6
12 11  2  1

Matrix is unsolvable

Time elapsed: 0.012 seconds
```

4 Tabel Penyelesaian

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		✓