LAPORAN TUGAS KECIL 3 IF2211 STRATEGI ALGORITMA
SEMESTER II 2021/2022

PENYELESAIAN PERSOALAN 15-PUZZLE DENGAN ALGORITMA BRANCH AND BOUND

Dibuat oleh:

Alifia Rahmah

13520122

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

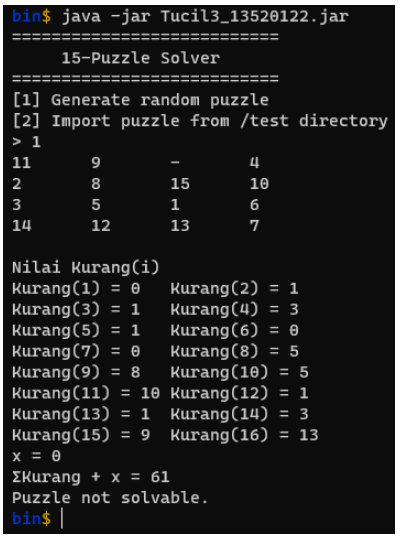
1. CARA KERJA PROGRAM

Program 15-Puzzle Solver menggunakan algoritma Branch and Bound untuk menyelesaikan persoalan 15-Puzzle yang awalnya dibangkitkan secara acak atau mengambil input dari file, sehingga susunan kotak dalam puzzle terurut dari 1 sampai 15. Pendekatan Branch and Bound dilakukan dengan memilih *cost* terkecil pada tiap langkah hingga mencapai tujuan akhir puzzle.

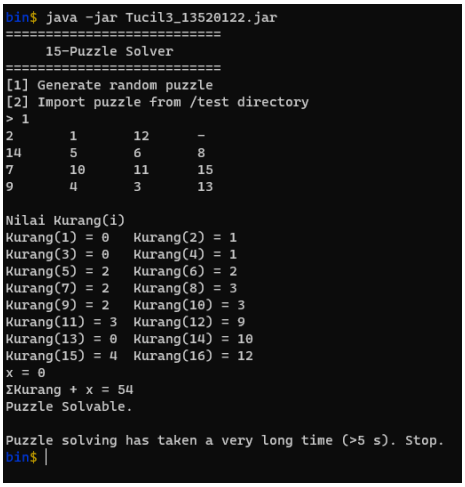
Pertama-tama, program akan menerima input terlebih dahulu, lalu mengecek apakah puzzle dapat diselesaikan dengan menghitung fungsi Kurang(i) dan posisi ubin kosong (x). Jika jumlah dari fungsi Kurang(i) dan x merupakan bilangan genap, maka puzzle dapat diselesaikan dan langsung mencari langkah-langkah menuju susunan akhir puzzle. Jika tidak, maka program akan memberi pesan bahwa puzzle tidak dapat diselesaikan lalu berhenti.

Dalam mencari langkah-langkah menuju susunan akhir puzzle, program menggunakan pendekatan Tree dengan struktur data Priority Queue untuk menyimpan kumpulan puzzle yang dimasukkan dalam Node. Awalnya, dilakukan pembangkitan langkah, yaitu membuat Node baru dari susunan puzzle awal jika kotak kosong dipindahkan ke atas, bawah, kiri, dan kanan sesuai posisi kotak kosong dalam puzzle sambil menghitung *cost* dari tiap hasil pembangkitan langkah. *Cost* dihitung dengan menjumlahkan *f(i)*, banyak langkah yang diperlukan untuk mencapai posisi puzzle tersebut dari posisi puzzle awal, dan *g(i)*, jumlah kotak kosong pada posisi puzzle tersebut yang belum sesuai tempatnya. Semua hasil dimasukkan ke dalam Priority Queue, dengan aturan puzzle dengan *cost* terkecil merupakan prioritas, berada di posisi depan dari Priority Queue). Setelah itu, Node dengan posisi paling pertama dalam PriorityQueue, yaitu Node dengan *cost* terkecil, dipilih untuk dilakukan pembangkitan langkah kembali seperti tadi hingga mencapai posisi akhir.

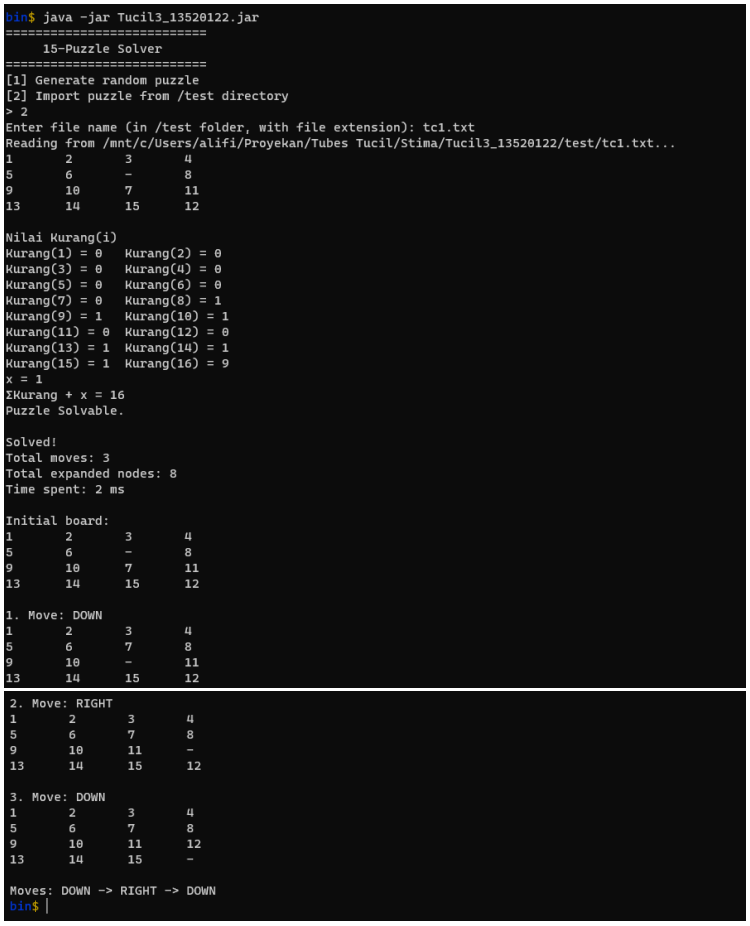
2. SCREENSHOT INPUT-OUTPUT PROGRAM



Gambar 2.1 Puzzle dibangkitkan secara acak dan tidak dapat diselesaikan



Gambar 2.2 Puzzle dibangkitkan secara acak dan waktu pengerjaan solusi melampaui batas

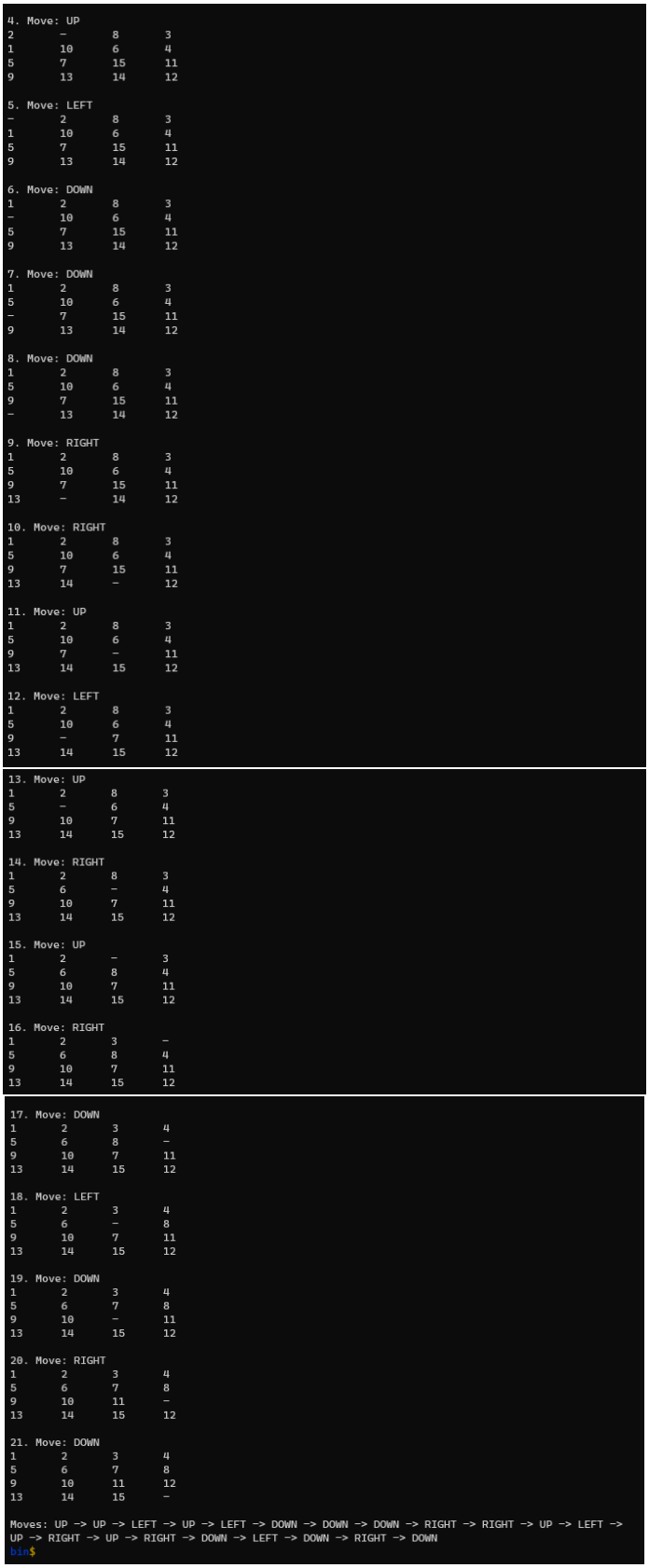


Gambar 2.3 Puzzle dibangkitkan dari file tc1.txt dan menghasilkan solusi

```
bin$ java -jar Tucil3_13520122.jar
===========================
     15-Puzzle Solver
===========================
[1] Generate random puzzle
[2] Import puzzle from /test directory
> 2
Enter file name (in /test folder, with file extension): tc2.txt
Reading from /mnt/c/Users/alifi/Proyekan/Tubes Tucil/Stima/Tucil3_13520122/test/tc2.txt...
1       3       4       15
2       -       5       12
7       6       11      14
8       9       10      13

Nilai Kurang(i)
Kurang(1) = 0   Kurang(2) = 0
Kurang(3) = 1   Kurang(4) = 1
Kurang(5) = 0   Kurang(6) = 0
Kurang(7) = 1   Kurang(8) = 0
Kurang(9) = 0   Kurang(10) = 0
Kurang(11) = 3  Kurang(12) = 6
Kurang(13) = 0  Kurang(14) = 4
Kurang(15) = 11 Kurang(16) = 10
x = 0
ΣKurang + x = 37
Puzzle not solvable.
bin$
```

Gambar 2.4 Puzzle dibangkitkan dari file tc2.txt dan tidak menghasilkan solusi

```
bin$ java -jar Tucil3_13520122.jar
===========================
     15-Puzzle Solver
===========================
[1] Generate random puzzle
[2] Import puzzle from /test directory
> 2
Enter file name (in /test folder, with file extension): tc3.txt
Reading from /mnt/c/Users/alifi/Proyekan/Tubes Tucil/Stima/Tucil3_13520122/test/tc3.txt...
2       10      8       3
1       6       15      4
5       7       14      11
9       13      -       12

Nilai Kurang(i)
Kurang(1) = 0   Kurang(2) = 1
Kurang(3) = 1   Kurang(4) = 0
Kurang(5) = 0   Kurang(6) = 2
Kurang(7) = 0   Kurang(8) = 6
Kurang(9) = 0   Kurang(10) = 8
Kurang(11) = 1  Kurang(12) = 0
Kurang(13) = 1  Kurang(14) = 4
Kurang(15) = 8  Kurang(16) = 1
x = 1
ΣKurang + x = 34
Puzzle Solvable.

Solved!
Total moves: 21
Total expanded nodes: 17714
Time spent: 34 ms

Initial board:
2       10      8       3
1       6       15      4
5       7       14      11
9       13      -       12

1. Move: UP
2       10      8       3
1       6       15      4
5       7       -       11
9       13      14      12

2. Move: UP
2       10      8       3
1       6       -       4
5       7       15      11
9       13      14      12

3. Move: LEFT
2       10      8       3
1       -       6       4
5       7       15      11
9       13      14      12
```

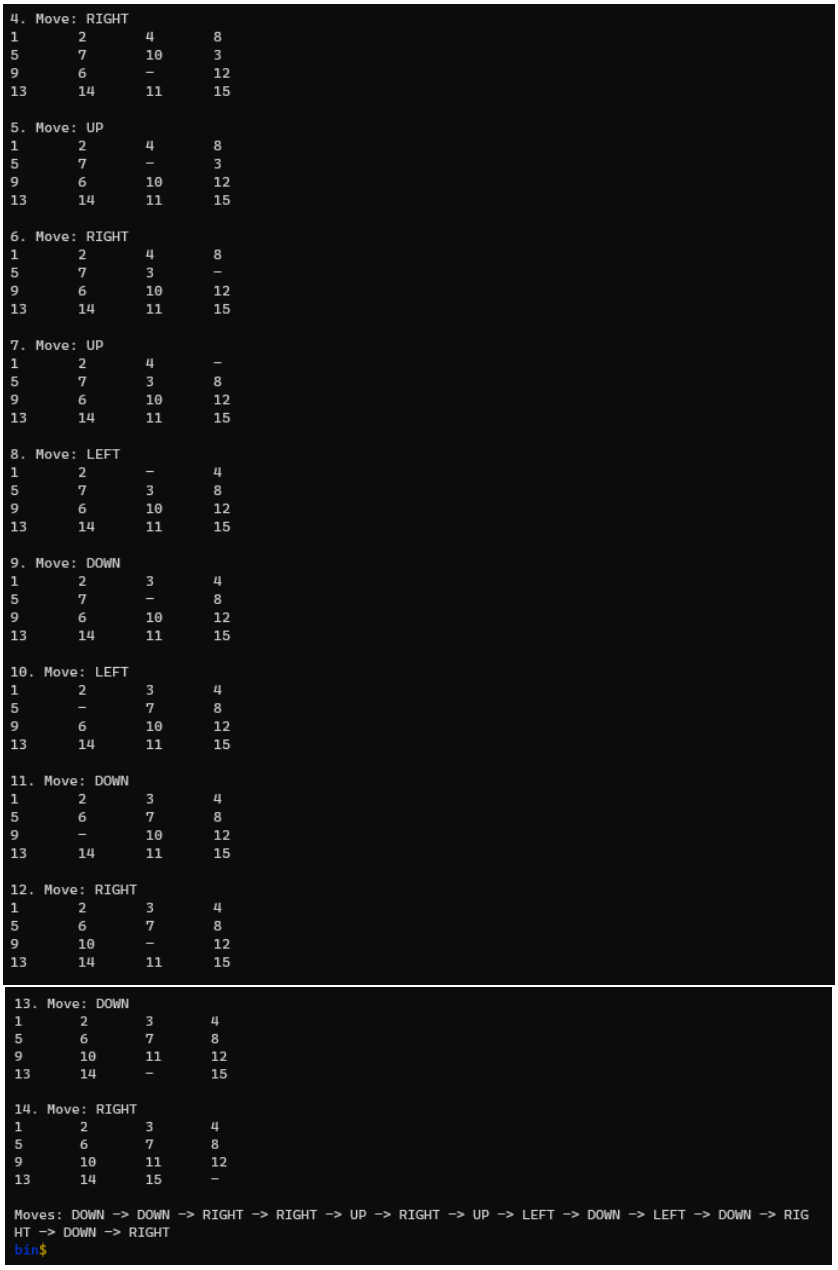


Gambar 2.5 Puzzle dibangkitkan dari file tc3.txt dan menghasilkan solusi

```
bin$ java -jar Tucil3_13520122.jar
============================
     15-Puzzle Solver
============================
[1] Generate random puzzle
[2] Import puzzle from /test directory
> 2
Enter file name (in /test folder, with file extension): tc4.txt
Reading from /mnt/c/Users/alifi/Proyekan/Tubes Tucil/Stima/Tucil3_13520122/test/tc4.txt...
15      2       1       12
8       5       6       11
4       9       10      7
3       14      13      -

Nilai Kurang(i)
Kurang(1) = 0   Kurang(2) = 1
Kurang(3) = 0   Kurang(4) = 1
Kurang(5) = 2   Kurang(6) = 2
Kurang(7) = 1   Kurang(8) = 5
Kurang(9) = 2   Kurang(10) = 2
Kurang(11) = 5  Kurang(12) = 9
Kurang(13) = 0  Kurang(14) = 1
Kurang(15) = 14 Kurang(16) = 0
x = 0
ΣKurang + x = 45
Puzzle not solvable.
bin$
```

Gambar 2.6 Puzzle dibangkitkan dari file tc4.txt dan tidak menghasilkan solusi

```
bin$ java -jar Tucil3_13520122.jar
============================
     15-Puzzle Solver
============================
[1] Generate random puzzle
[2] Import puzzle from /test directory
> 2
Enter file name (in /test folder, with file extension): tc5.txt
Reading from /mnt/c/Users/alifi/Proyekan/Tubes Tucil/Stima/Tucil3_13520122/test/tc5.txt...
-       2       4       8
1       7       10      3
5       9       6       12
13      14      11      15

Nilai Kurang(i)
Kurang(1) = 0   Kurang(2) = 1
Kurang(3) = 0   Kurang(4) = 2
Kurang(5) = 0   Kurang(6) = 0
Kurang(7) = 3   Kurang(8) = 5
Kurang(9) = 1   Kurang(10) = 4
Kurang(11) = 0  Kurang(12) = 1
Kurang(13) = 1  Kurang(14) = 1
Kurang(15) = 0  Kurang(16) = 15
x = 0
ΣKurang + x = 34
Puzzle Solvable.

Solved!
Total moves: 14
Total expanded nodes: 304
Time spent: 4 ms

Initial board:
-       2       4       8
1       7       10      3
5       9       6       12
13      14      11      15

1. Move: DOWN
1       2       4       8
-       7       10      3
5       9       6       12
13      14      11      15

2. Move: DOWN
1       2       4       8
5       7       10      3
-       9       6       12
13      14      11      15

3. Move: RIGHT
1       2       4       8
5       7       10      3
9       -       6       12
13      14      11      15
```



Gambar 2.7 Puzzle dibangkitkan dari file tc5.txt dan tidak menghasilkan solusi

3. CHECKLIST

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		✓

4. KODE PROGRAM

PuzzleBoard.java

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.*;

public class PuzzleBoard {
    // Deklarasi atribut
    private String[][] board;
    private int emptyRowLoc, emptyColLoc;

    // Konstruktor kosong (assign nomor secara random)
    public PuzzleBoard() {
        // Inisialisasi atribut
        board = new String[4][4];
        emptyRowLoc = emptyColLoc = 3;

        // Kocok daftar kotak
        String[] numbers = {"1", "2", "3", "4", "5", "6", "7", "8",
"9", "10", "11", "12", "13", "14", "15", "-"};
        List<String> list = Arrays.asList(numbers);
        Collections.shuffle(list);
        list.toArray(numbers);

        // Assign nomor ke board
        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 4; j++) {
                board[i][j] = String.valueOf(numbers[i * 4 + j]);
            }
        }
    }
    // Konstructor dari File
    public PuzzleBoard(String filename) {
        this.board = new String[4][4];
        String path = new File("").getAbsolutePath();
        if(path.endsWith("bin")) {
            path = path.substring(0, path.length() - 4);
        }
        String filepath = path + "/test/" + filename;
        System.out.print("Reading from " + filepath + "...\n");
        // Membaca file
        try {
            File file = new File(filepath);
            Scanner input = new Scanner(file);
            int i = 0;
            while (input.hasNextLine()) {
                String line = input.nextLine();
                String[] tokens = line.split(" ");
                for (int j = 0; j < 4; j++) {
                    this.board[i][j] = tokens[j];
```

```
                    if (tokens[j].equals("-")) { // Inisialisasi posisi
kotak kosong
                        this.emptyRowLoc = i;
                        this.emptyColLoc = j;
                    }
                }
                i++;
            }
            input.close();
        } catch (FileNotFoundException e) {
            System.out.println("File not found.");
        }
    }
    public PuzzleBoard(File file) {
        try {

            this.board = new String[4][4];
            Scanner input = new Scanner(file);
            int i = 0;
            while (input.hasNextLine()) {
                String line = input.nextLine();
                String[] tokens = line.split(" ");
                for (int j = 0; j < 4; j++) {
                    this.board[i][j] = tokens[j];
                    if (tokens[j].equals("-")) { // Inisialisasi posisi
kotak kosong
                        this.emptyRowLoc = i;
                        this.emptyColLoc = j;
                    }
                }
                i++;
            }
            input.close();
        } catch (FileNotFoundException e) {
            System.out.println("File not found.");
        }
    }
    // Konstruktor dari Puzzleboard lain
    public PuzzleBoard(PuzzleBoard other) {
        this.board = new String[4][4];
        for (int i = 0; i < 4; i++) {
            System.arraycopy(other.board[i], 0, this.board[i], 0, 4);
        }
        this.emptyRowLoc = other.emptyRowLoc;
        this.emptyColLoc = other.emptyColLoc;
    }

    // Getter
    public String getValueString(int row, int col) {
        return this.board[row][col];
    }
    public int getValue(int row, int col) {
```

```
        if (this.board[row][col].equals("-")) {
            return 16;
        }
        return Integer.parseInt(this.board[row][col]);
    }
    public int getEmptyRowIdx() {
        return this.emptyRowLoc;
    }
    public int getEmptyColIdx() {
        return this.emptyColLoc;
    }

    // Setter
    public void setValue(int row, int col, String value) {
        this.board[row][col] = value;
    }
    public void setEmptyRowLoc(int row) {
        // ! jangan dipakai tanpa move
        this.emptyRowLoc = row;
    }
    public void setEmptyColLoc(int col) {
        // ! jangan dipakai tanpa move
        this.emptyColLoc = col;
    }

    // Mengembalikan board dengan posisi kotak kosong ("-") pada posisi
baru
    public PuzzleBoard movedUp() {
        PuzzleBoard newBoard = new PuzzleBoard(this);
        if (newBoard.getEmptyRowIdx() > 0) {
            newBoard.setValue(newBoard.getEmptyRowIdx(),
newBoard.getEmptyColIdx(),
newBoard.getValueString(newBoard.getEmptyRowIdx() - 1,
newBoard.getEmptyColIdx()));
            newBoard.setValue(newBoard.getEmptyRowIdx() - 1,
newBoard.getEmptyColIdx(), "-");
            newBoard.setEmptyRowLoc(newBoard.getEmptyRowIdx() - 1);
        }
        return newBoard;
    }
    public PuzzleBoard movedDown() {
        PuzzleBoard newBoard = new PuzzleBoard(this);
        if (newBoard.getEmptyRowIdx() < 3) {
            newBoard.setValue(newBoard.getEmptyRowIdx(),
newBoard.getEmptyColIdx(),
newBoard.getValueString(newBoard.getEmptyRowIdx() + 1,
newBoard.getEmptyColIdx()));
            newBoard.setValue(newBoard.getEmptyRowIdx() + 1,
newBoard.getEmptyColIdx(), "-");
            newBoard.setEmptyRowLoc(newBoard.getEmptyRowIdx() + 1);
        }
        return newBoard;
```

```
    }
    public PuzzleBoard movedLeft() {
        PuzzleBoard newBoard = new PuzzleBoard(this);
        if (newBoard.getEmptyColIdx() > 0) {
            newBoard.setValue(newBoard.getEmptyRowIdx(),
newBoard.getEmptyColIdx(),
newBoard.getValueString(newBoard.getEmptyRowIdx(),
newBoard.getEmptyColIdx() - 1));
            newBoard.setValue(newBoard.getEmptyRowIdx(),
newBoard.getEmptyColIdx() - 1, "-");
            newBoard.setEmptyColLoc(newBoard.getEmptyColIdx() - 1);
        }
        return newBoard;
    }
    public PuzzleBoard movedRight() {
        PuzzleBoard newBoard = new PuzzleBoard(this);
        if (newBoard.getEmptyColIdx() < 3) {
            newBoard.setValue(newBoard.getEmptyRowIdx(),
newBoard.getEmptyColIdx(),
newBoard.getValueString(newBoard.getEmptyRowIdx(),
newBoard.getEmptyColIdx() + 1));
            newBoard.setValue(newBoard.getEmptyRowIdx(),
newBoard.getEmptyColIdx() + 1, "-");
            newBoard.setEmptyColLoc(newBoard.getEmptyColIdx() + 1);
        }
        return newBoard;
    }

    // Method untuk mencetak board sebagai matriks, dipisah oleh tab
("\t")
    public void printBoard() {
        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 4; j++) {
                System.out.print(this.board[i][j] + "\t");
            }
            System.out.println();
        }
    }

    // Method untuk mengecek apakah board dapat diselesaikan
    public boolean isSolvable() {
        // Menentukan x
        // x = 1 jika sel kosong berada pada posisi row genap kolom
ganjil / row ganjil kolom genap

        int x = x();
        int kurangCount = SigmaKurang();

        // Bisa diselesaikan kalau hasilnya genap
        return (kurangCount + x) % 2 == 0;
    }
    public int x() {
```

```
        if (((this.getEmptyRowIdx() % 2 == 0) &&
(this.getEmptyColIdx() % 2 == 1))
                || ((this.getEmptyRowIdx() % 2 == 1) &&
(this.getEmptyColIdx() % 2 == 0))
        ) {
            return 1;
        } else {
            return 0;
        }
    }
    public int SigmaKurang() {
        int kurangCount = 0;
        // Iterasi tiap kotak, hitung banyak kotak-kotak
        // setelahnya yang nilainya lebih kecil dari kotak tersebut
        for(int it = 0; it < 4; it++){
            for (int jt = 0; jt < 4; jt++) {
                kurangCount += Kurang(it, jt);
            }
        }
        return kurangCount;
    }
    public int Kurang(int row, int col){
        // Fungsi antara
        // KURANG(i) = banyaknya ubin bernomor j sedemikian sehingga j
< i dan POSISI(j) > POSISI(i).
        // POSISI(i) = posisi ubin bernomor i pada susunan yang
diperiksa.
        int counter = 0;
        // Cek sebelah kanan board[row][col]
        int i, j = col + 1;
        while (j < 4){
            if (this.getValue(row, j) < this.getValue(row, col)){
                counter++;
            }
            j++;
        }
        // Baris selanjutnya
        for(i = row + 1; i < 4; i++){
            for(j = 0; j < 4; j++){
                if (this.getValue(i, j) < this.getValue(row, col)){
                    counter++;
                }
            }
        }

        return counter;
    }
    public int Kurang(int value){
        for(int i = 0; i < 4; i++){
            for(int j = 0; j < 4; j++){
                if (this.getValue(i, j) == value){
                    return Kurang(i, j);
```

```
                }
            }
        }
        return 0;
    }

    // Method untuk mencari jumlah kotak yang tidak sesuai tempatnya
    public int countNotInPlace() {
        int counter = 0;
        for(int i = 0; i < 4; i++){
            for(int j = 0; j < 4; j++){
                if (this.getValue(i, j) != 4 * i + j + 1){
                    counter++;
                }
            }
        }
        return counter;
    }

    // Method untuk mengecek apakah seluruh kotak dalam board sudah
sesuai
    public boolean isSolved() {
        for(int i = 0; i < 4; i++){
            for(int j = 0; j < 4; j++){
                if (this.getValue(i, j) != 4 * i + j + 1){
                    return false;
                }
            }
        }
        return true;
    }

    // Memanggil solve() dari class Solver
    public void solve() {
        Solver.solve(this);
    }
}
```

Node.java

```
enum Move { LEFT, RIGHT, UP, DOWN }

class Node {
    private final PuzzleBoard board;
    private final int cost;
    private final int depth;
    private final Move moveFromParent;
    private final Node parent;

    // Konstruktor
    public Node(PuzzleBoard board) {
        // konstruktor root node
        this.board = board;
```

```
        this.cost = 0;
        this.depth = 0;
        this.moveFromParent = null;
        this.parent = null;
    }
    public Node(PuzzleBoard board, Move moveFromParent, Node parent) {
        this.board = board;
        // c(i) = f(i) + g(i)
        // c(i) -> cost
        // f(i) -> cost from start to this node (cost parent) -->
asumsi cost = depth parent
        // g(i) -> countNotInPlace
        this.cost = parent.getDepth() + board.countNotInPlace();
        this.depth = parent.getDepth() + 1;
        this.moveFromParent = moveFromParent;
        this.parent = parent;
    }

    // Getter
    public PuzzleBoard getBoard() {
        return board;
    }
    public int getCost() {
        return cost;
    }
    public int getDepth() {
        return depth;
    }
    public Move getMoveFromParent() {
        return moveFromParent;
    }
    public Node getParent() {
        return parent;
    }
}
```

Solver.java

```
enum Move { LEFT, RIGHT, UP, DOWN }

class Node {
    private final PuzzleBoard board;
    private final int cost;
    private final int depth;
    private final Move moveFromParent;
    private final Node parent;

    // Konstruktor
    public Node(PuzzleBoard board) {
        // konstruktor root node
        this.board = board;
        this.cost = 0;
        this.depth = 0;
```

```
        this.moveFromParent = null;
        this.parent = null;
    }
    public Node(PuzzleBoard board, Move moveFromParent, Node parent) {
        this.board = board;
        // c(i) = f(i) + g(i)
        // c(i) -> cost
        // f(i) -> cost from start to this node (cost parent) -->
asumsi cost = depth parent
        // g(i) -> countNotInPlace
        this.cost = parent.getDepth() + board.countNotInPlace();
        this.depth = parent.getDepth() + 1;
        this.moveFromParent = moveFromParent;
        this.parent = parent;
    }

    // Getter
    public PuzzleBoard getBoard() {
        return board;
    }
    public int getCost() {
        return cost;
    }
    public int getDepth() {
        return depth;
    }
    public Move getMoveFromParent() {
        return moveFromParent;
    }
    public Node getParent() {
        return parent;
    }
}
```

Main.java

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        // Main menu
        System.out.println("============================");
        System.out.println("     15-Puzzle Solver      ");
        System.out.println("============================");
        System.out.println("[1] Generate random puzzle");
        System.out.println("[2] Import puzzle from /test directory");
        System.out.print("> ");

        Scanner scanner = new Scanner(System.in);
        String input = scanner.nextLine();
        PuzzleBoard board;
        while(!input.equals("1") && !input.equals("2")) {
            System.out.println("Invalid input.");
```

```
            System.out.print("> ");
            input = scanner.nextLine();
        }
        if (input.equals("1")) {
            // Generate random puzzle
            board = new PuzzleBoard();
        } else {
            // Import puzzle
            System.out.print("Enter file name (in /test folder, with
file extension): ");
            input = scanner.nextLine();
            board = new PuzzleBoard(input);
        }

        if(board.getValueString(0,0) != null){
            // Cetak board
            board.printBoard();

            // Cetak Kurang(i), x, dan isSolvable
            System.out.println("\nNilai Kurang(i)");
            for(int i = 1; i < 16; i+=2) {
                System.out.print("Kurang(" + i + ") = " +
board.Kurang(i));
                System.out.println("\tKurang(" + (i+1) + ") = " +
board.Kurang(i+1));
            }
            System.out.println("x = " + board.x());
            System.out.printf("ΣKurang + x = %d\n", board.SigmaKurang()
+ board.x());

            // Solve puzzle
            if (board.isSolvable()) {
                System.out.println("Puzzle Solvable.\n");
                board.solve();
            } else {
                System.out.println("Puzzle not solvable.");
            }
        }
    }
}
```

5. BERKAS TEKS INSTANSIASI PERSOALAN

tc1.txt

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

tc2.txt

```
1 3 4 15
2 - 5 12
```

```
7 6 11 14
8 9 10 13
```

tc3.txt

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

tc4.txt

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

tc5.txt

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

6. ALAMAT KODE PROGRAM

Repository GitHub:

https://github.com/alifiarahmah/Tucil3_13520122