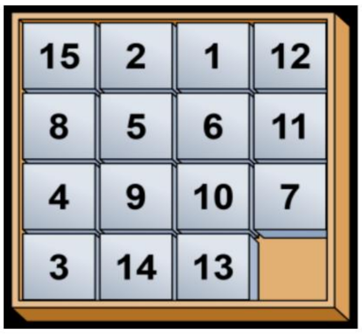
# Laporan Tugas Kecil 3 Strategi Algoritma IF2211 Semester II Tahun 2021/2022 Penyelesaian Persoalan *15-Puzzle* dengan Algoritma *Branch and Bound*

Nayotama Pradipta - 13520089
*Program Studi Teknik Informatika*
*Sekolah Teknik Elektro dan Informatika*
*Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia*

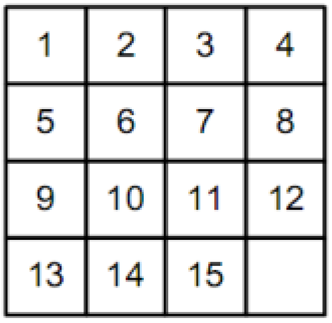
# I. Pendahuluan

## A. 15-Puzzle

15-Puzzle merupakan permainan yang terdiri atas matriks 4 x 4, yang masing-masing sel matriksnya terisi dengan sebuah angka yang teracak. Matriks ini memiliki satu sel yang tidak memiliki angka apapun, atau dengan kata lain kosong. Sel yang bersebelahan dengan sel yang kosong dapat digeser sehingga posisinya berubah. Tujuan dari permainan ini adalah mentransformasikan urutan angka pada matriks menjadi terurut membesar. Untuk lebih jelasnya dapat dilihat pada gambar berikut:



Gambar 1.1 Contoh 15-Puzzle yang belum terselesaikan



Gambar 1.2 Posisi 15-Puzzle yang telah terselesaikan

## B. Branch & Bound

Branch & Bound merupakan sebuah algoritma yang biasanya digunakan untuk menyelesaikan permasalahan optimasi kombinatorial. Optimasi berarti meminimalkan ataupun memaksimalkan suatu fungsi objektif yang tidak melanggar batasan persoalan. Branch & Bound sendiri merupakan gabungan dari Breadth-First Search dengan *least cost search*. Setiap simpul pada Branch & Bound memiliki sebuah cost *c(i)*, yaitu taksiran lintasan termurah ke simpul status tujuan melalui simpul status *i*. Simpul berikutnya yang akan ditinjau/di-*expand* adalah simpul yang memiliki cost yang paling kecil (untuk kasus minimasi).

*15-Puzzle* dapat diselesaikan menggunakan algoritma Branch & Bound, dengan cost simpul hidup adalah jumlah ubin tidak kosong yang tidak terdapat pada susunan akhir. Algoritma BnB akan mencari simpul yang memiliki cost terkecil. Sebelum BnB dilakukan, terdapat teorema untuk menentukan apakah *starting state* dari sebuah 15-Puzzle dapat diselesaikan. Teoremanya berupa:

Status tujuan hanya dapat dicapai dari status awal jika nilai

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

bernilai genap

*KURANG(i)* merupakan banyaknya ubin/*tile* bernomor j sedemikian sehingga j < i dan Posisi(j) > Posisi(i). Posisi(i) merupakan posisi ubin bernomor i pada susunan yang diperiksa. X bernilai 1 jika sel kosong terdapat pada sel {2, 4, 5, 7, 10, 12, 13, 15}, 0 jika tidak.

## II. Cara Kerja Branch & Bound pada Program

Pada program 15-Puzzle yang telah dibuat, terdapat class BranchAndBound yang memiliki beberapa atribut sebagai berikut:

1. reachable: boolean (True jika status tujuan dapat dicapai, false jika tidak)
2. X : integer (Nilai X pada teorema)
3. condition: integer (Nilai kurang(i))
4. solution: SST (State Space Tree hanya untuk solution)
5. posmove: array of String (Berisi 4 arah: “up”, “down”, “right”, “left”)
6. node: int (Jumlah simpul yang dibangkitkan)
7. path : List of SST (State Space Tree yang mengarah ke solution)

Konstruktor Class Branch & Bound akan menerima parameter berupa puzzle. Sebelum inti dari algoritmanya dijalankan, program akan mengecek apakah status tujuan dapat dicapai. Setelah objek Branch & Bound dibuat, maka objek tersebut dapat diselesaikan menggunakan fungsi solve. Fungsi solve berisi intisari dari algoritma, yaitu sebagai berikut:

1. Buat State Space Tree root yang merupakan kondisi awal puzzle
2. Jika kondisi terpenuhi maka lanjut, jika tidak maka selesai dan menampilkan pesan “Unreachable”
3. Buat Priority Queue dengan kondisi prioritas khusus, yaitu mendahulukan SST yang memiliki Cost dan depth terkecil
4. Add root ke Priority Queue, inisialisasi node menjadi 1
5. Ambil elemen pertama dari Priority Queue
6. Apabila berada pada status tujuan maka selesai
7. Apabila tidak, maka buat child State Space Tree untuk semua gerakan yang tidak repetitive dan tidak sama dengan parentnya
8. Tambahkan semua child yang berhasil dibuat ke priority queue
9. Ulangi langkah 3-7 hingga selesai (Status tujuan)
10. Define path sebagai path dari solution menggunakan fungsi getPath()

## III. Source Program

// Tile Class

```java
public abstract class Tile {
    int x;
    int y;

    public Tile() {
    }

    public abstract int getNum();

    public boolean isEqual(Object t) {
        if (t instanceof Tile) {
            Tile temp = (Tile) t;
            if (temp.getNum() == this.getNum()) {
                return true;
            } else {
                return false;
            }

        } else {
            return false;
        }
    }
}
```

// OccupiedTile Class

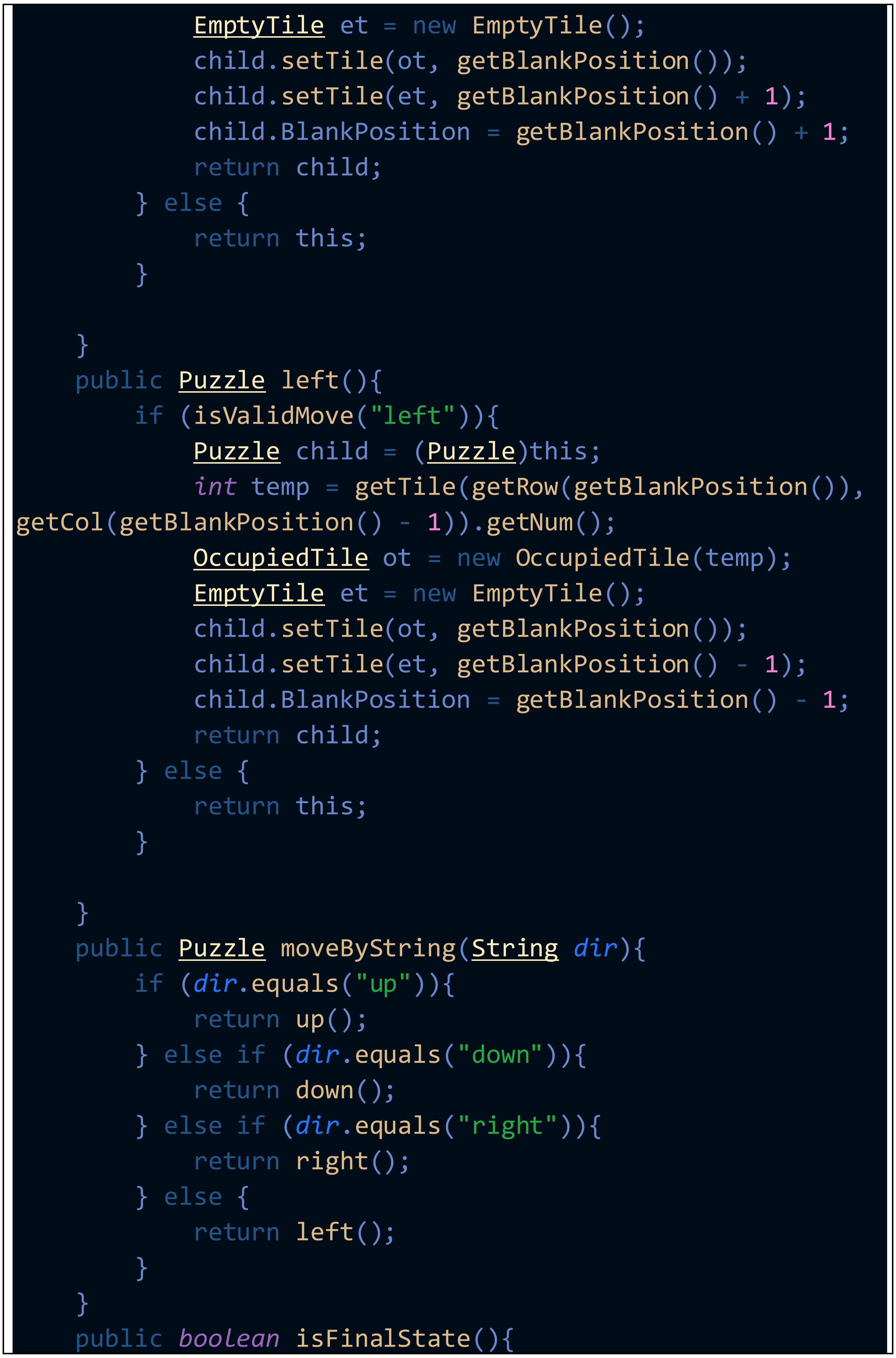
```java
public class OccupiedTile extends Tile {
    private int num;
    public OccupiedTile(int num){
        super();
        this.num = num;
    }
    public int getNum(){
        return this.num;
    }
}
```

// EmptyTile Class

```
public class EmptyTile extends Tile {
    private int num;

    public EmptyTile() {
        super();
        this.num = 16;
    }

    public int getNum() {
        return this.num;
    }
}
```

// Puzzle Class

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
import java.util.ArrayList;
import java.util.Collections;
public class Puzzle {
    private Tile[][] tile;
    private int BlankPosition;
    public Puzzle(){
        this.tile = new Tile[4][4];
        ArrayList<Integer> x = new ArrayList<>();
        for (int i = 0; i < 16; i++){
            x.add(i+1);
        }
        Collections.shuffle(x);
        int k = 0;
        for (int i = 0; i < 4; i++){
            for (int j = 0; j < 4; j++){
                if (x.get(k) != 16){
                    tile[i][j] = new
OccupiedTile(x.get(k));
                } else {
                    tile[i][j] = new EmptyTile();
                }
                k++;
            }
        }
        for (int i = 0; i < 4; i++){
            for (int j = 0; j < 4; j++){
                if (tile[i][j].getNum() == 16){
                    BlankPosition = getTilePosition(i, j);
                    break;
                }
            }
        }
    }
    public Puzzle(String filename) {
        readFileToPuzzle(filename);
        int i;
        int j;
        for (i = 0; i < 4; i++){
            for (j = 0; j < 4; j++){
                if (tile[i][j].getNum() == 16){
                    BlankPosition = getTilePosition(i, j);
                    break;
                }
            }
        }
    }

    public Puzzle(Puzzle copy){
        this.tile = new Tile[4][4];
        for (int i = 0; i < 4; i++){
            for (int j = 0; j < 4; j++){
                this.tile[i][j] = copy.tile[i][j];
            }
        }
        this.BlankPosition = copy.BlankPosition;
    }
    public Tile getTile(int i , int j){
        return this.tile[i][j];
    }
    public void readFileToPuzzle(String filename){
        try {
            File file = new File("../test/" + filename);
            Scanner input = new Scanner(file);
            tile = new Tile[4][4];
            while (input.hasNextLine()){

                for (int i = 0; i < 4;
i++){
                    for (int j = 0; j < 4; j++){
                        if (input.hasNextInt()){
                            if (input.nextInt() != 16){
                                tile[i][j] = new
OccupiedTile(input.nextInt());
                            } else {
                                tile[i][j] = new
EmptyTile();
                            }
                        }
                    }
                }
            }

            input.close();
        } catch (FileNotFoundException e){
            System.out.println("File Not Found!");
        }
    }
    public int getTilePosition(int i, int j){
        return ((4*i) + j + 1);
    }
    public int getRow(int position){
        if (position < 5){
            return 0;
        } else if (position < 9){
            return 1;
        } else if (position < 13){
            return 2;
        } else {
            return 3;
        }
    }
    public int getCol(int position){
        if (position % 4 == 0){
            return 3;
        } else {
            return (position % 4) - 1;
        }

    }
    public int getBlankPosition(){
        return this.BlankPosition;
    }
    public void setTile(Object t, int position){
        if (t instanceof Tile){
            this.tile[getRow(position)][getCol(position)] =
(Tile) t;
        }
    }

    public boolean isValidMove(String dir){
        if (dir.equals("up")){
            return (getBlankPosition() > 4);
        } else if (dir.equals("down")){
            return (getBlankPosition() < 13);
        } else if (dir.equals("right")){
            return (getBlankPosition() % 4 != 0);
        } else {
            return (getBlankPosition() % 4 != 1);
        }
    }

    public Puzzle up(){
        // Temp menyimpan nilai tile diatasnya
        if (isValidMove("up")){
            Puzzle child = (Puzzle)this;
            int temp = getTile(getRow(getBlankPosition()) -
1, getCol(getBlankPosition())).getNum();
            OccupiedTile ot = new OccupiedTile(temp);
            EmptyTile et = new EmptyTile();
            child.setTile(ot, getBlankPosition());
            child.setTile(et, getBlankPosition() - 4);
            child.BlankPosition = getBlankPosition() - 4;
            return child;
        } else {
            return this;
        }

    }

    public Puzzle down(){
        if (isValidMove("down")){
            Puzzle child = (Puzzle)this;
            int temp = getTile(getRow(getBlankPosition()) +
1, getCol(getBlankPosition())).getNum();
            OccupiedTile ot = new OccupiedTile(temp);
            EmptyTile et = new EmptyTile();
            child.setTile(ot, getBlankPosition());
            child.setTile(et, getBlankPosition() + 4);
            child.BlankPosition = getBlankPosition() + 4;
            return child;
        } else {
            return this;
        }

    }
    public Puzzle right(){
        if (isValidMove("right")){
            Puzzle child = (Puzzle)this;
            int temp = getTile(getRow(getBlankPosition()),
getCol(getBlankPosition() + 1)).getNum();
            OccupiedTile ot = new OccupiedTile(temp);
```



```
            EmptyTile et = new EmptyTile();
            child.setTile(ot, getBlankPosition());
            child.setTile(et, getBlankPosition() + 1);
            child.BlankPosition = getBlankPosition() + 1;
            return child;
        } else {
            return this;
        }

    }
    public Puzzle left(){
        if (isValidMove("left")){
            Puzzle child = (Puzzle)this;
            int temp = getTile(getRow(getBlankPosition()),
getCol(getBlankPosition() - 1)).getNum();
            OccupiedTile ot = new OccupiedTile(temp);
            EmptyTile et = new EmptyTile();
            child.setTile(ot, getBlankPosition());
            child.setTile(et, getBlankPosition() - 1);
            child.BlankPosition = getBlankPosition() - 1;
            return child;
        } else {
            return this;
        }

    }
    public Puzzle moveByString(String dir){
        if (dir.equals("up")){
            return up();
        } else if (dir.equals("down")){
            return down();
        } else if (dir.equals("right")){
            return right();
        } else {
            return left();
        }
    }
    public boolean isFinalState(){
        for (int i = 0; i < 4; i++){
            for (int j = 0; j < 4; j++){
                if (this.tile[i][j].getNum() !=
getTilePosition(i, j)){
                    return false;
                }
            }
        }
        return true;
    }
    public void printTiles(){
        for (int i = 0; i < 4; i++){
            for (int j = 0; j < 4; j++){
                if (this.tile[i][j].getNum() < 10){
                    System.out.print(this.tile[i][j].getNum
() + "  ");
                } else {
                    if (this.tile[i][j].getNum() != 16){
                        System.out.print(this.tile[i][j].ge
tNum() + " ");
                    } else {
                        System.out.print("X  ");
                    }
                }
            }
            System.out.println();
        }
        System.out.println();
    }

}
```

// SST class

```
import java.util.List;
import java.util.ArrayList;
public class SST {
    public Puzzle root;
    public SST parent;
```

```java
    public String move;
    public int depth;
    public SST(Puzzle root){
        this.root = root;
        this.move = "";
        this.depth = 0;
    }
    public SST(Puzzle root, SST parent, String move, int
depth){
        this.root = root;
        this.parent = parent;
        this.move = move;
        this.depth = depth;
    }
    public SST(SST x){
        this.root = x.root;
        this.parent = x.parent;
        this.move = x.move;
        this.depth = x.depth;

    }
    public boolean isRepetitive(String dir){
        if (this.move.equals("up")){
            return (dir.equals("down"));
        } else if (this.move.equals("down")){
            return (dir.equals("up"));
        } else if (this.move.equals("right")){
            return (dir.equals("left"));
        } else if (this.move.equals("left")){
            return (dir.equals("right"));
        } else {
            return false;
        }
    }
    public List<SST> getPath(){
        List<SST> solution = new ArrayList<SST>();
        SST parent = this.parent;
        SST child = this;
        while (parent != null){
            solution.add(0, child);
            child = parent;
            parent = parent.parent;
        }
        return solution;
    }
    public int getCost(){
        return (BranchAndBound.misplaced(this.root) +
this.depth);
    }
}
```

// BranchAndBound Class

```java
import java.util.ArrayList;
import java.util.List;
import java.util.PriorityQueue;
public class BranchAndBound {
    private boolean reachable;
    private int X;
    private int condition;
    private SST solution;
    private String[] posmove;
    private int node;
    private List<SST> path;
    public BranchAndBound(Puzzle puzzle){
        posmove = new String[]{"up", "down", "right",
"left"};
        // Pertama cek apakah goal dapat dicapai
menggunakan Teorema pada pdf
        // Ambil nilai X
        if ((puzzle.getRow(puzzle.getBlankPosition()) +
puzzle.getCol(puzzle.getBlankPosition())) % 2 == 1){
            X = 1;
        } else {
            X = 0;
        }
        // Hitung total Kurang(i)
        int k = 1;
        this.condition = 0;
        int ctr[] = new int[16];
        while (k <= 16){
            int x = puzzle.getRow(k);
            int y = puzzle.getCol(k);
            int temp = 0;
            for (int i = 0; i < 4; i++){
                for (int j = 0; j < 4; j++){
                    if ((puzzle.getTile(i, j).getNum() <
puzzle.getTile(x, y).getNum()) &&
(puzzle.getTilePosition(i, j) > puzzle.getTilePosition(x,
y))){
                        this.condition += 1;
                        temp +=1;
                    }

                }
            }
            ctr[puzzle.getTile(puzzle.getRow(k),
puzzle.getCol(k)).getNum() - 1] = temp;

            k += 1;
        }
        for (int i = 0; i < 16; i++){
            System.out.println("Kurang[" + (i+1) + "] : " +
ctr[i]);
        }
        System.out.println("Total Kurang[i] : " +
this.condition);
        this.condition += this.X;
        if (this.condition % 2 == 0){
            reachable = true;
        } else {
            reachable = false;
        }
    }
    public boolean isReachable(){
        return this.reachable;
    }
    public int getX(){
        return this.X;
    }
    public int getCondition(){
        return this.condition;
    }


    public static int misplaced(Puzzle puzzle){
        int ctr = 0;
        for (int i = 0; i < 4; i++){
            for (int j = 0; j < 4; j++){
                if (puzzle.getTile(i, j).getNum() !=
puzzle.getTilePosition(i, j) && puzzle.getTile(i,
j).getNum() != 16){
                    ctr += 1;
                }
            }
        }
        return ctr;
    }
    // Buat Priority Queue

    public void solve(Puzzle puzzle){
        long startTime = System.nanoTime();
        if (reachable){
            // Algoritma Branch & Bound
            SST root = new SST(puzzle);
            PriorityQueue<SST> pqbnb = new
PriorityQueue<SST>((m,n) -> m.getCost() - n.getCost());

            pqbnb.add(root);
            this.node = 1;
            while (!pqbnb.isEmpty()){
                SST curr = pqbnb.poll();
                if (curr.root.isFinalState()){
                    this.solution = curr;
                    break;
                } else {
                    for (int i = 0; i < 4; i++){
                        if
(!curr.isRepetitive(posmove[i])){
                            Puzzle temp = new
Puzzle(curr.root);
                            SST child = new
SST(temp.moveByString(this.posmove[i]), curr,
this.posmove[i], curr.depth + 1);
                            if
(child.root.getBlankPosition() !=
curr.root.getBlankPosition() && child.root != null){
                                this.node += 1;
                                pqbnb.add(child);
                            }

                        }
                    }
                }
            }

            this.path = this.solution.getPath();
            System.out.println("Steps taken: " +
this.path.size());
        } else {
            System.out.println("Goal Unreachable!");
        }
        long stopTime = System.nanoTime();
        if (reachable){
            showPath();
        }
        System.out.print("Nilai fungsi Sigma Kurang[i] + X:
");
        System.out.println(getCondition());
        System.out.print("Nodes generated: ");
        System.out.println(this.node);
        System.out.println("Execution time: " + ((stopTime
- startTime) / 1000000) + " ms");


    }
    public void showPath(){
        for (int i = 0; i < this.path.size();i++){
            System.out.println("Step " + (i+1) + ": " +
this.path.get(i).move);
            this.path.get(i).root.printTiles();
        }
    }
}
```

// Main class

```
import java.io.FileNotFoundException;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) throws
FileNotFoundException {
        // Read input
        System.out.println("Choose Input Format: ");
        System.out.println("1. Randomly Generated Puzzle");
        System.out.println("2. Input from File");
        Scanner sc = new Scanner(System.in);
        Integer z;
        do {
            System.out.print("Input Number: ");
            z = sc.nextInt();
            if (z == 1){
                Puzzle x = new Puzzle();
                // Output Starting position
                System.out.println("Starting Tiles: ");
                x.printTiles();
                // Initialize branch and bound
                BranchAndBound bnb = new BranchAndBound(x);
                // Solve
                bnb.solve(x);
            } else if (z==2) {
                Scanner s = new Scanner(System.in);
                System.out.print("Enter file name
(with .txt): ");
                String filename = s.nextLine();
                s.close();
                // Create Puzzle
                Puzzle x = new Puzzle(filename);
                // Output Starting position
                System.out.println("Starting Tiles: ");
                x.printTiles();

                // Initialize branch and bound
                BranchAndBound bnb = new BranchAndBound(x);
                // Solve
                bnb.solve(x);
            } else {
                System.out.println("Invalid Input");
            }
        } while (z != 1 && z != 2);

    }
}
```

## IV. Input & Output Program

Input Files:

Reachable1.txt

```
0 1 0 2 0 3 0 4
0 5 0 6 0 7 0 8
0 9 0 12 0 15 0 14
0 13 0 11 0 16 0 10
```

Reachable2.txt

```
0 5 0 1 0 3 0 4
0 9 0 2 0 7 0 8
0 16 0 6 0 15 0 11
0 13 0 10 0 14 0 12
```

Reachable3.txt

```
0 1 0 2 0 4 0 7
0 5 0 6 0 16 0 3
0 9 0 11 0 12 0 8
0 13 0 10 0 14 0 15
```

Unreachable1.txt

```
0 3 0 9 0 1 0 15
0 14 0 11 0 4 0 6
0 13 0 16 0 10 0 12
0 2 0 7 0 8 0 5
```

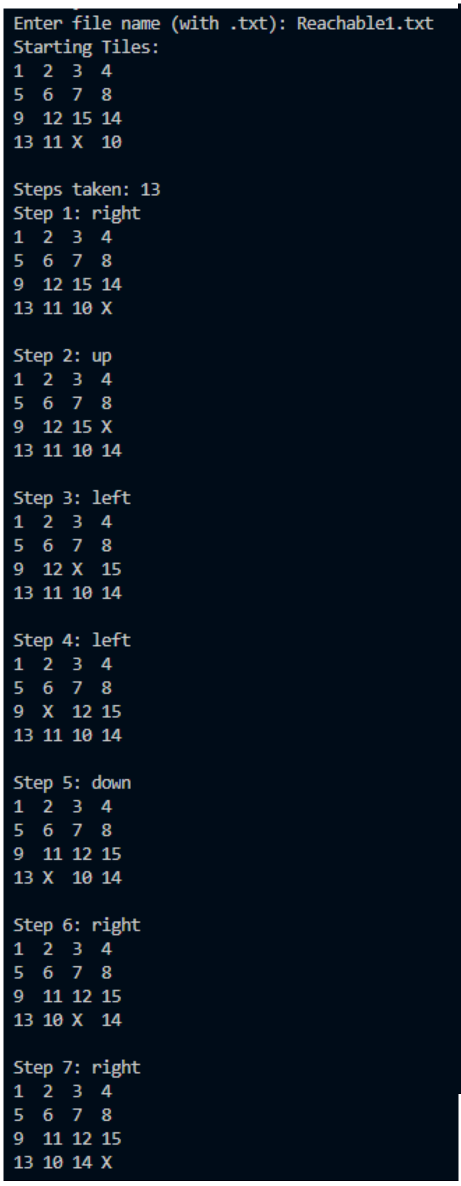
Unreachable2.txt

```
0 1 0 3 0 4 0 15
0 2 0 16 0 5 0 12
0 7 0 6 0 11 0 14
0 8 0 9 0 10 0 13
```

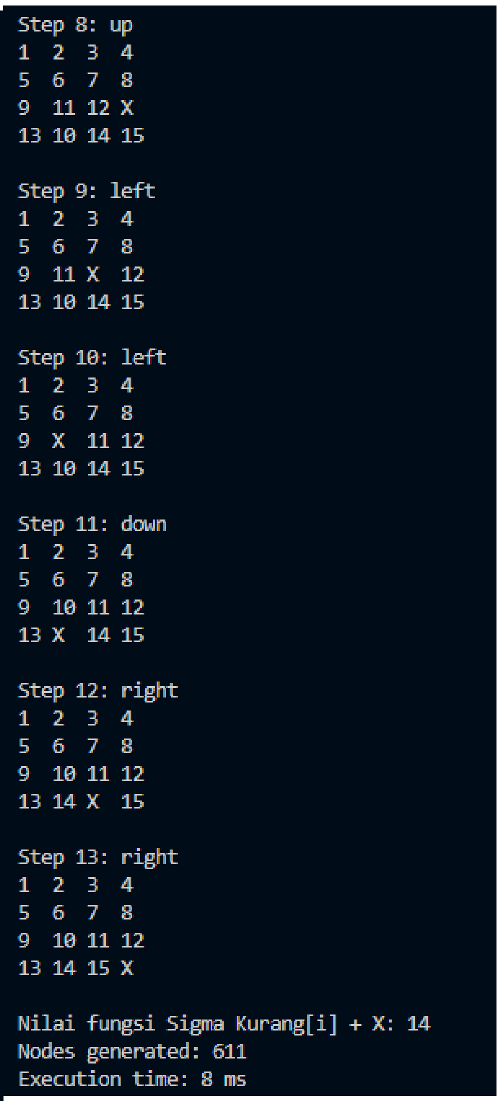
Penjelasan Input Files: Input Files diatas semuanya berisi angka nol diantara dua buah angka karena read Files pada program saya membaca satu angka kemudian skip angka berikutnya. Saya sendiri kurang tahu bagaimana cara memperbaikinya. Apabila tidak ada angka 0, maka nanti puzzle yang terbaca tidak akan sesuai. Untuk file lain sebenarnya tidak harus menyisipkan 0, bisa digantikan dengan angka lain selama tidak berkisar di range 1-16 (Nanti *confusing*).

ScreenShot Output:

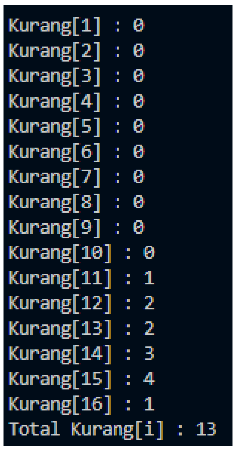
1. Output Reachable1.txt



```
Enter file name (with .txt): Reachable1.txt
Starting Tiles:
1  2  3  4
5  6  7  8
9  12 15 14
13 11 X  10

Steps taken: 13
Step 1: right
1  2  3  4
5  6  7  8
9  12 15 14
13 11 10 X

Step 2: up
1  2  3  4
5  6  7  8
9  12 15 X
13 11 10 14

Step 3: left
1  2  3  4
5  6  7  8
9  12 X  15
13 11 10 14

Step 4: left
1  2  3  4
5  6  7  8
9  X  12 15
13 11 10 14

Step 5: down
1  2  3  4
5  6  7  8
9  11 12 15
13 X  10 14

Step 6: right
1  2  3  4
5  6  7  8
9  11 12 15
13 10 X  14

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

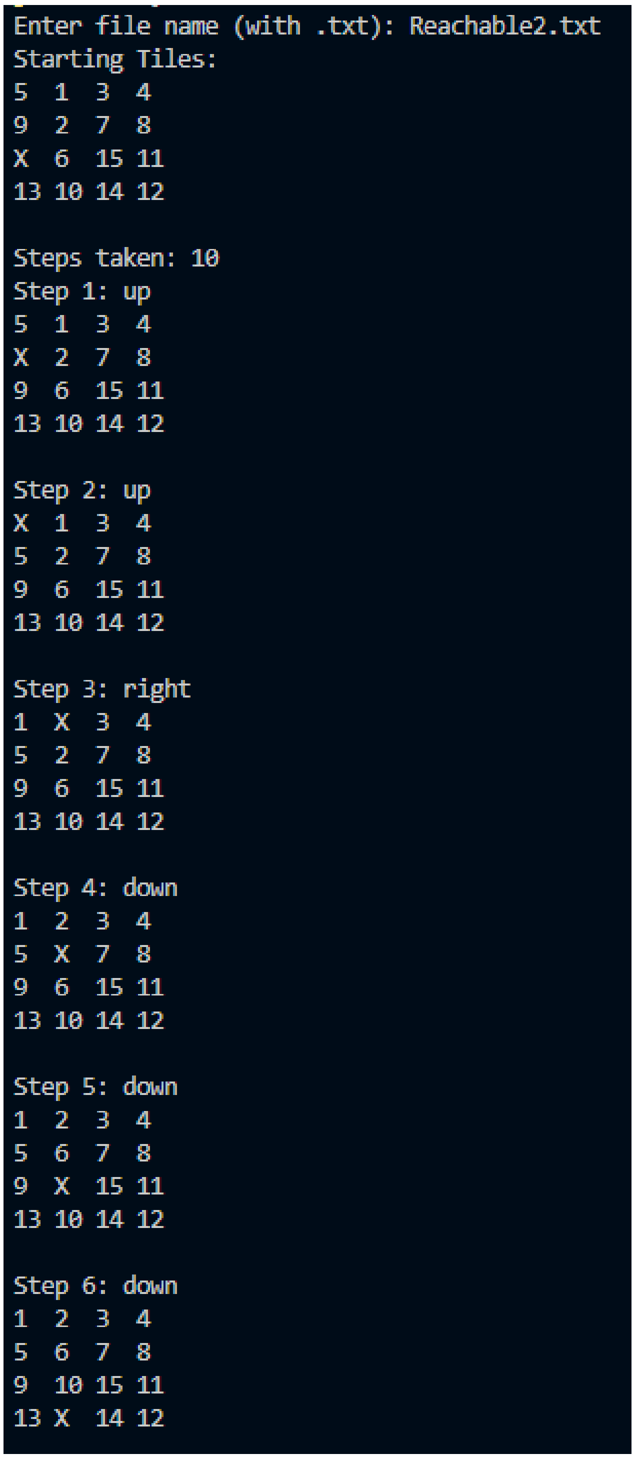


```
Step 8: up
1  2  3  4
5  6  7  8
9  11 12 X
13 10 14 15

Step 9: left
1  2  3  4
5  6  7  8
9  11 X  12
13 10 14 15

Step 10: left
1  2  3  4
5  6  7  8
9  X  11 12
13 10 14 15

Step 11: down
1  2  3  4
5  6  7  8
9  10 11 12
13 X  14 15

Step 12: right
1  2  3  4
5  6  7  8
9  10 11 12
13 14 X  15

Step 13: right
1  2  3  4
5  6  7  8
9  10 11 12
13 14 15 X

Nilai fungsi Sigma Kurang[i] + X: 14
Nodes generated: 611
Execution time: 8 ms
```

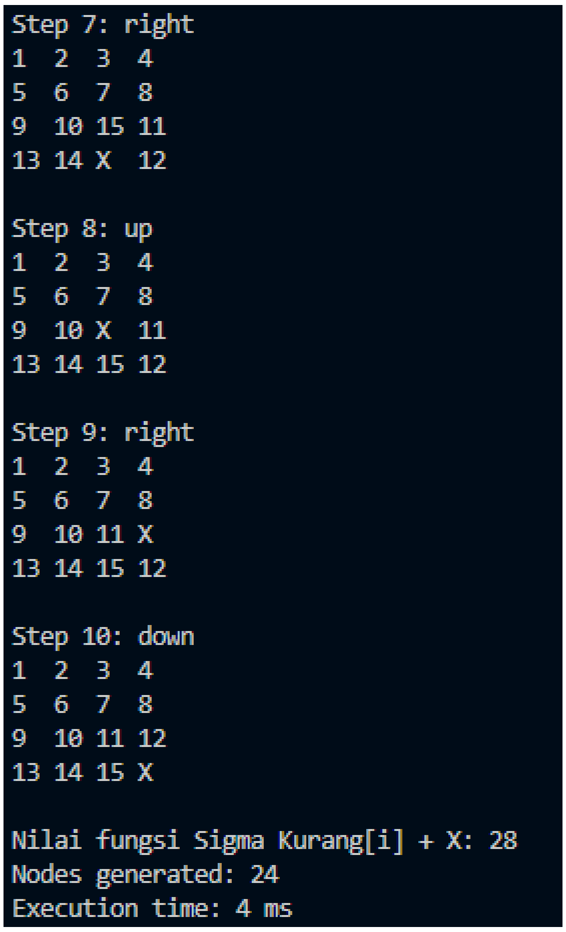


```
Kurang[1] : 0
Kurang[2] : 0
Kurang[3] : 0
Kurang[4] : 0
Kurang[5] : 0
Kurang[6] : 0
Kurang[7] : 0
Kurang[8] : 0
Kurang[9] : 0
Kurang[10] : 0
Kurang[11] : 1
Kurang[12] : 2
Kurang[13] : 2
Kurang[14] : 3
Kurang[15] : 4
Kurang[16] : 1
Total Kurang[i] : 13
```

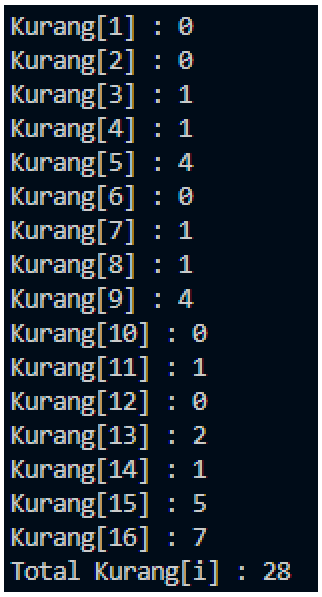
2. Output Reachable2.txt



```
Enter file name (with .txt): Reachable2.txt
Starting Tiles:
5  1  3  4
9  2  7  8
X  6  15 11
13 10 14 12

Steps taken: 10
Step 1: up
5  1  3  4
X  2  7  8
9  6  15 11
13 10 14 12

Step 2: up
X  1  3  4
5  2  7  8
9  6  15 11
13 10 14 12

Step 3: right
1  X  3  4
5  2  7  8
9  6  15 11
13 10 14 12

Step 4: down
1  2  3  4
5  X  7  8
9  6  15 11
13 10 14 12

Step 5: down
1  2  3  4
5  6  7  8
9  X  15 11
13 10 14 12

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

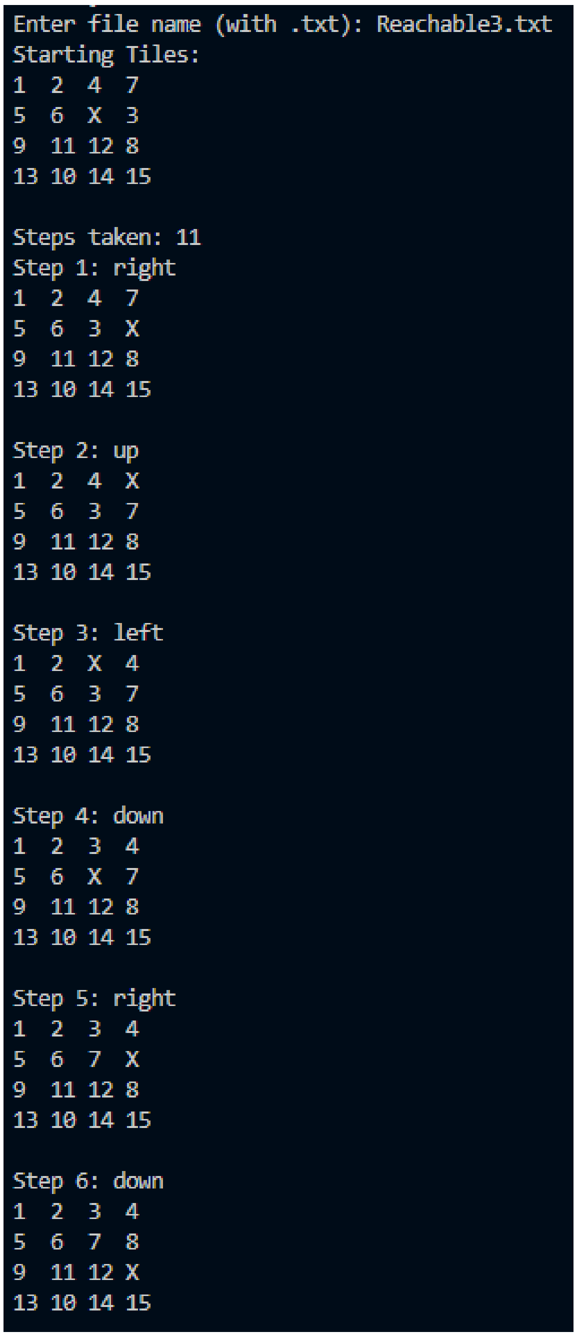


```
Step 7: right
1  2  3  4
5  6  7  8
9  10 15 11
13 14 X  12

Step 8: up
1  2  3  4
5  6  7  8
9  10 X  11
13 14 15 12

Step 9: right
1  2  3  4
5  6  7  8
9  10 11 X
13 14 15 12

Step 10: down
1  2  3  4
5  6  7  8
9  10 11 12
13 14 15 X

Nilai fungsi Sigma Kurang[i] + X: 28
Nodes generated: 24
Execution time: 4 ms
```

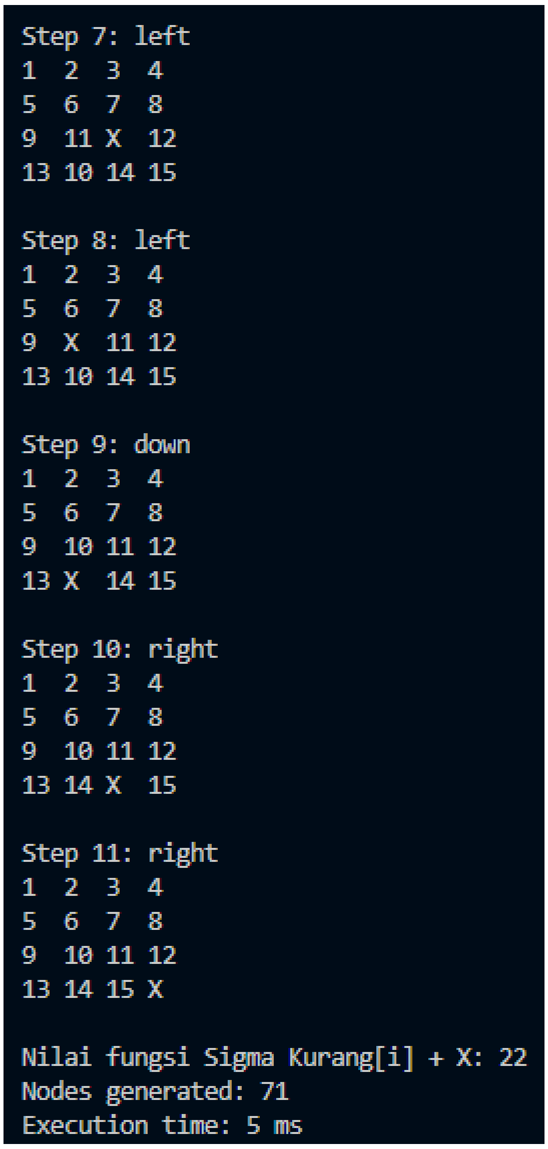


```
Kurang[1] : 0
Kurang[2] : 0
Kurang[3] : 1
Kurang[4] : 1
Kurang[5] : 4
Kurang[6] : 0
Kurang[7] : 1
Kurang[8] : 1
Kurang[9] : 4
Kurang[10] : 0
Kurang[11] : 1
Kurang[12] : 0
Kurang[13] : 2
Kurang[14] : 1
Kurang[15] : 5
Kurang[16] : 7
Total Kurang[i] : 28
```

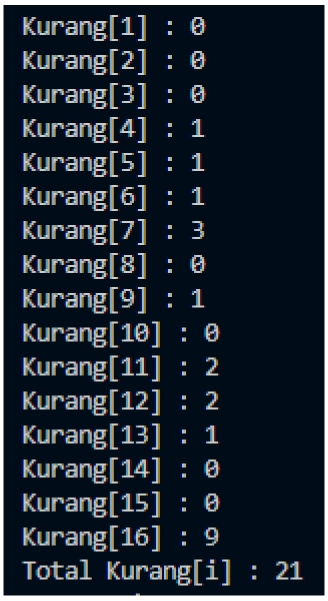
3. Output Reachable3.txt



```
Enter file name (with .txt): Reachable3.txt
Starting Tiles:
1  2  4  7
5  6  X  3
9  11 12 8
13 10 14 15

Steps taken: 11
Step 1: right
1  2  4  7
5  6  3  X
9  11 12 8
13 10 14 15

Step 2: up
1  2  4  X
5  6  3  7
9  11 12 8
13 10 14 15

Step 3: left
1  2  X  4
5  6  3  7
9  11 12 8
13 10 14 15

Step 4: down
1  2  3  4
5  6  X  7
9  11 12 8
13 10 14 15

Step 5: right
1  2  3  4
5  6  7  X
9  11 12 8
13 10 14 15

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

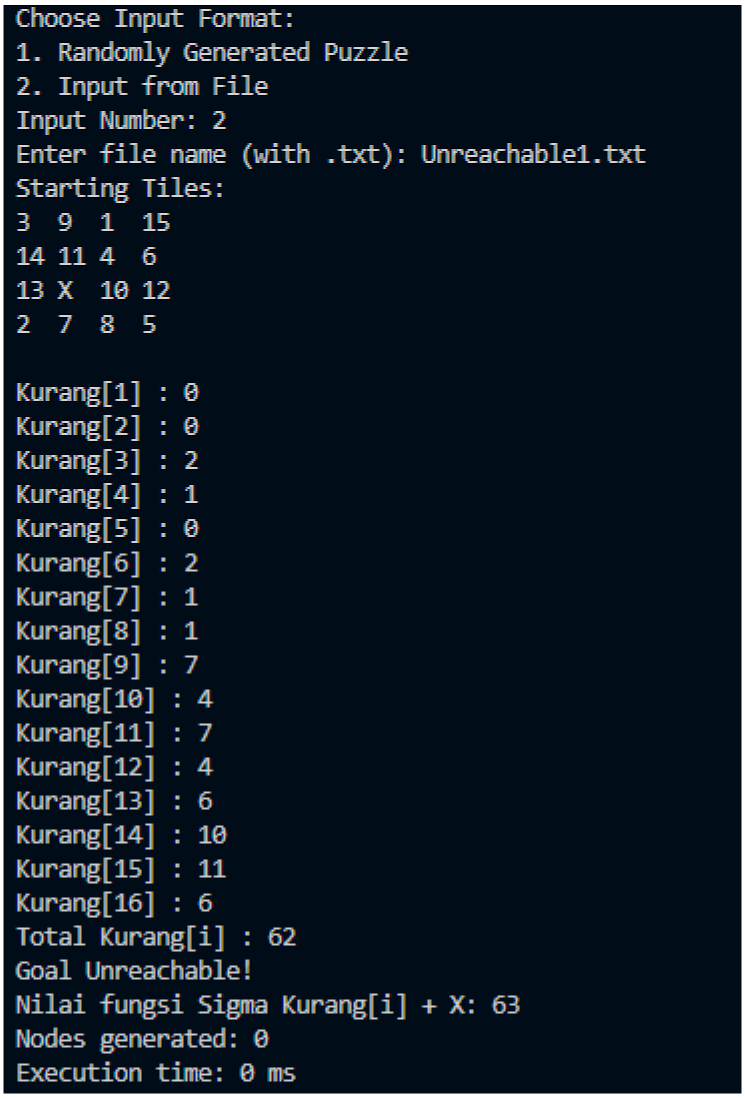


```
Step 7: left
1  2  3  4
5  6  7  8
9  11 X  12
13 10 14 15

Step 8: left
1  2  3  4
5  6  7  8
9  X  11 12
13 10 14 15

Step 9: down
1  2  3  4
5  6  7  8
9  10 11 12
13 X  14 15

Step 10: right
1  2  3  4
5  6  7  8
9  10 11 12
13 14 X  15

Step 11: right
1  2  3  4
5  6  7  8
9  10 11 12
13 14 15 X

Nilai fungsi Sigma Kurang[i] + X: 22
Nodes generated: 71
Execution time: 5 ms
```



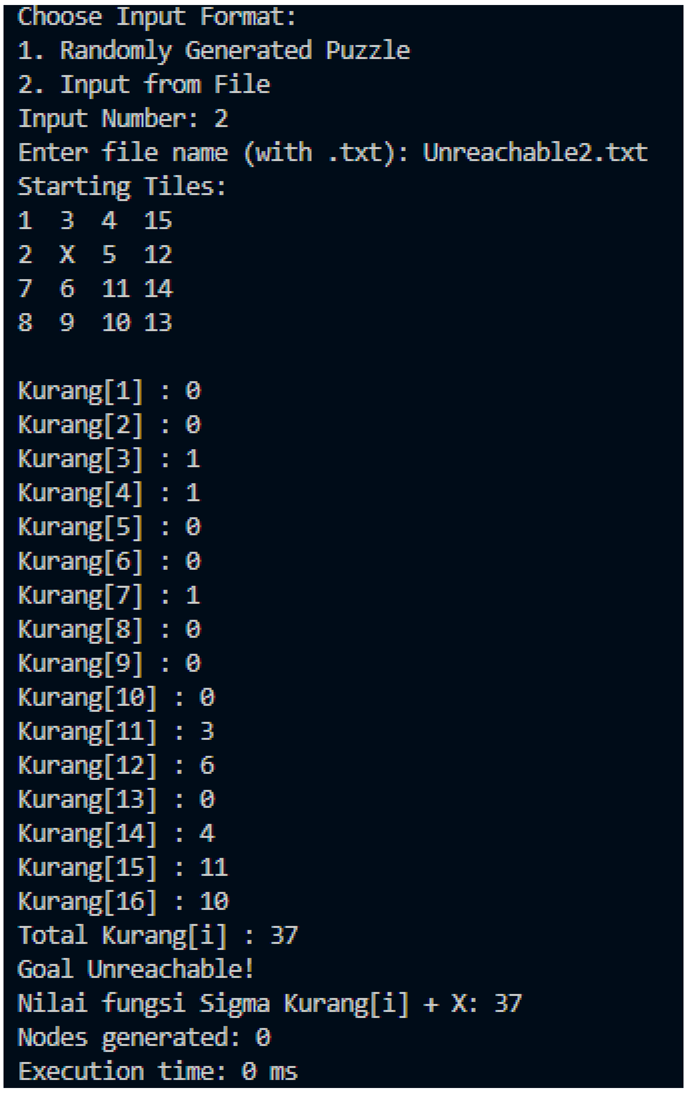
```
Kurang[1] : 0
Kurang[2] : 0
Kurang[3] : 0
Kurang[4] : 1
Kurang[5] : 1
Kurang[6] : 1
Kurang[7] : 3
Kurang[8] : 0
Kurang[9] : 1
Kurang[10] : 0
Kurang[11] : 2
Kurang[12] : 2
Kurang[13] : 1
Kurang[14] : 0
Kurang[15] : 0
Kurang[16] : 9
Total Kurang[i] : 21
```

4. Output Unreachable1.txt



```
Choose Input Format:
1. Randomly Generated Puzzle
2. Input from File
Input Number: 2
Enter file name (with .txt): Unreachable1.txt
Starting Tiles:
3  9  1  15
14 11 4  6
13 X  10 12
2  7  8  5

Kurang[1] : 0
Kurang[2] : 0
Kurang[3] : 2
Kurang[4] : 1
Kurang[5] : 0
Kurang[6] : 2
Kurang[7] : 1
Kurang[8] : 1
Kurang[9] : 7
Kurang[10] : 4
Kurang[11] : 7
Kurang[12] : 4
Kurang[13] : 6
Kurang[14] : 10
Kurang[15] : 11
Kurang[16] : 6
Total Kurang[i] : 62
Goal Unreachable!
Nilai fungsi Sigma Kurang[i] + X: 63
Nodes generated: 0
Execution time: 0 ms
```

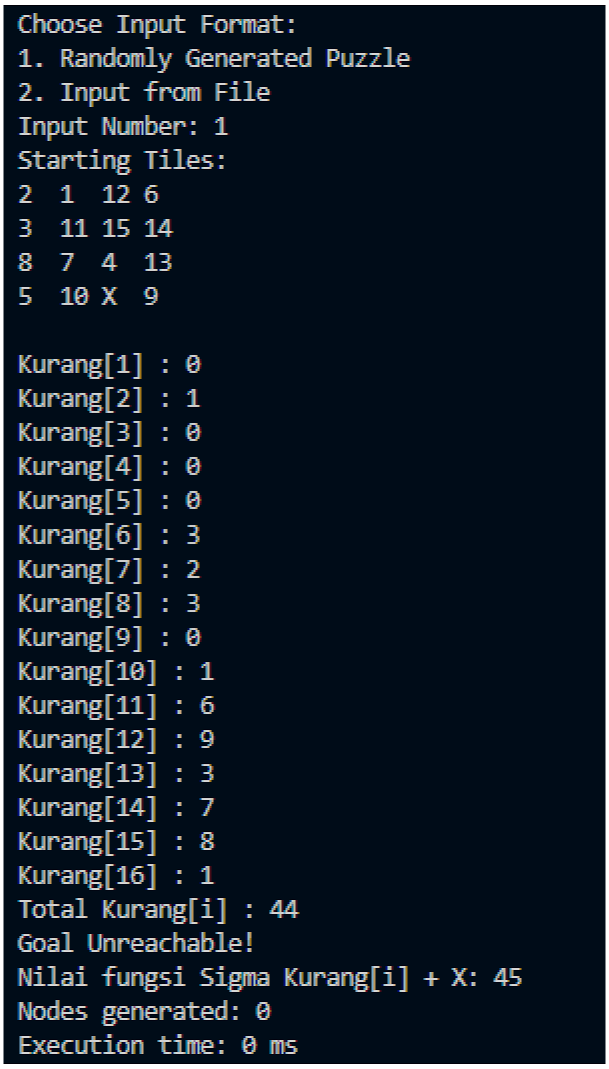
5. Output Unreachable2.txt



```
Choose Input Format:
1. Randomly Generated Puzzle
2. Input from File
Input Number: 2
Enter file name (with .txt): Unreachable2.txt
Starting Tiles:
1  3  4  15
2  X  5  12
7  6  11 14
8  9  10 13

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
Total Kurang[i] : 37
Goal Unreachable!
Nilai fungsi Sigma Kurang[i] + X: 37
Nodes generated: 0
Execution time: 0 ms
```

6. Input & Output Randomized

Untuk input randomized yang reachable terlalu lama, maka saya akan berikan input dan output yang unreachable



```
Choose Input Format:
1. Randomly Generated Puzzle
2. Input from File
Input Number: 1
Starting Tiles:
2  1  12 6
3  11 15 14
8  7  4  13
5  10 X  9

Kurang[1] : 0
Kurang[2] : 1
Kurang[3] : 0
Kurang[4] : 0
Kurang[5] : 0
Kurang[6] : 3
Kurang[7] : 2
Kurang[8] : 3
Kurang[9] : 0
Kurang[10] : 1
Kurang[11] : 6
Kurang[12] : 9
Kurang[13] : 3
Kurang[14] : 7
Kurang[15] : 8
Kurang[16] : 1
Total Kurang[i] : 44
Goal Unreachable!
Nilai fungsi Sigma Kurang[i] + X: 45
Nodes generated: 0
Execution time: 0 ms
```

## V. Checklist Fitur Program

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

## VI. Github Link

| https://github.com/NayotamaPradipta/15PuzzleBnB |
|---|