# TUGAS KECIL 3

## IF2211 Strategi Algoritma

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

Dipersiapkan oleh:

Taufan Fajarama Putrawansyah Ruslanali (13520031)

**PROGRAM STUDI TEKNIK INFORMATIKA**
**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**
**INSTITUT TEKNOLOGI BANDUNG**
**2022**

# CARA KERJA PROGRAM

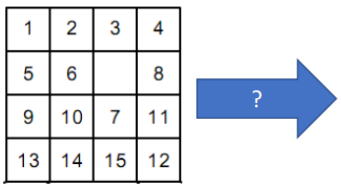
## *Pemrosesan Input*

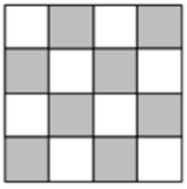
Input puzzle dapat didapat dari dua cara, yaitu dibangkitkan secara acak oleh program atau menggunakan input file. Pembangkitan puzzle secara acak memastikan semua elemen unik. Input file akan diproses menjadi tipe data PuzzleBox, yaitu sebuah matriks yang menyimpan nilai elemen puzzle.

## *Perhitungan Teorema Reachable Goal*

Teorema Reachable Goal digunakan untuk memeriksa apakah suatu susunan puzzle dapat mencapai status akhir atau tidak. Teorema ini dapat memangkas penyelesaian puzzle karena tidak semua puzzle dapat diselesaikan. Status akhir hanya dapat dicapai dari status awal jika nilai Kurang(i) + X bernilai genap. Kurang(i) adalah fungsi untuk menghitung jumlah ubin bernomor j yang memenuhi syarat j < i dan POSISI(j) > POSISI(i) pada suatu posisi ubin bernomor i. Nilai X akan bernilai 1 jika sel kosong pada posisi awal berada di sel yang diarsir dan bernilai 0 jika tidak.

Berikut adalah simulasi perhitungan Teorema Reachable Goal:



$$\sum_{i=1}^{16} Kurang(i) + X = 15 + 1 = 16$$

| 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 |
| 14 | 1 |
| 15 | 1 |
| 16 | 9 |

## *Metode Penyelesaian dengan Branch and Bound*

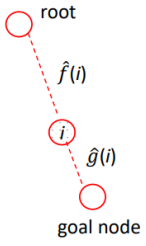
Algoritma dimulai dengan memasukkan simpul akar ke dalam sebuah antrian prioritas (PrioQueue) yang memprioritaskan antrian untuk elemen bernilai cost paling kecil. Berikut adalah nilai cost pada suatu simpul.

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

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

$\hat{f}(i)$ = ongkos mencapai simpul $i$ dari akar

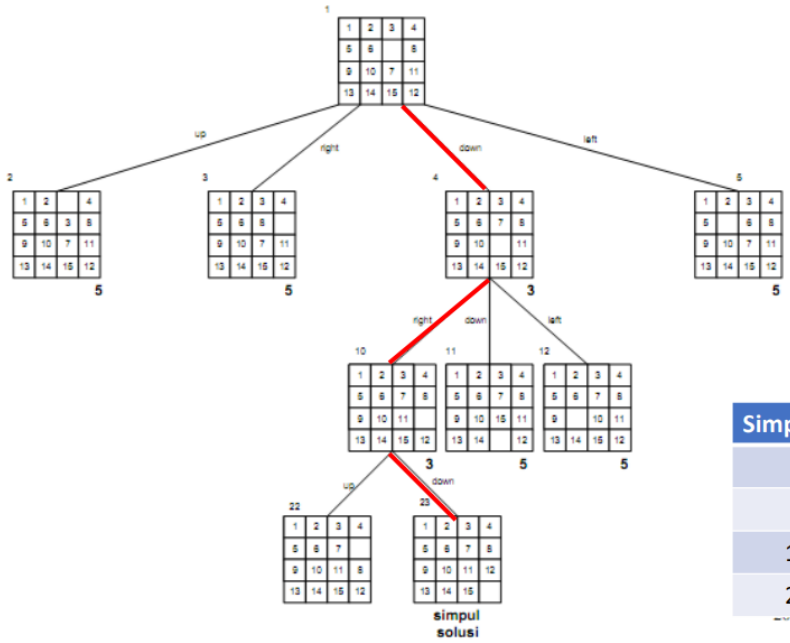
$\hat{g}(i)$ = ongkos mencapai simpul tujuan dari simpul $i$



Di mana f(i) adalah panjang lintasan yang telah dilalui dan g(i) adalah taksiran panjang lintasan dari simpul i ke simpul akhir, g(i) dalam hal ini adalah jumlah ubin tidak kosong yang tidak terdapat pada susunan akhir.

Program akan berjalan selama PrioQueue tidak kosong dan belum mencapai simpul akhir. Pada setiap simpul, akan dibangkitkan beberapa simpul i yang memenuhi syarat, yaitu berada pada batas index matriks. Setiap simpul yang dibangkitkan akan dimasukkan ke dalam PrioQueue dan pada loop selanjutnya akan dimulai dari simpul dengan cost terkecil (atau head dari PrioQueue).

Program akan berjalan hingga menemukan simpul solusi (pasti mencapai simpul solusi karena sudah menerapkan teorema Reachable Goal. Berikut contoh simulasinya.




| Simpul-E | Simpul Hidup |
|---|---|
| 1 | 4,2,3,5 |
| 4 | 10,2,3,5,11,12 |
| 10 | 23,2,3,5,11,12,22 |
| 23 | Solusi ketemu |

24

# SOURCE CODE

## *Class PuzzleBox*

Class PuzzleBox digunakan untuk menyimpan puzzle agar lebih mudah mengaksesnya dengan beberapa method. Berikut source codenya dalam bahasa Java:

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

You, 6 minutes ago | 1 author (You)
public class PuzzleBox {
    int[][] M;
    int rows, cols;

    // konstruktor default
    public PuzzleBox() {
        this.rows = 4;
        this.cols = 4;
        this.M = new int[rows][cols];
    }

    // fungsi helper untuk convert array of int ke ArrayList
    ArrayList<Integer> arrayToList(int[] initList) {
        ArrayList<Integer> resultList = new ArrayList<Integer>(initList.length);
        for (int i : initList) {
            resultList.add(i);
        }
        return resultList;
    }
```

```java
    // membangkitkan urutan puzzle secara acak
    ArrayList<Integer> randomizeNum() {
        int[] initNum = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 0};
        ArrayList<Integer> listNum = arrayToList(initNum);
        ArrayList<Integer> result = new ArrayList<Integer>();
        Random rand = new Random();
        // selama list awal belum kosong, pindahkan value pada index
        // yang dibangkitkan secara acak dari list awal ke list result
        while (listNum.size() > 0) {
            int randIdx = rand.nextInt(listNum.size());
            result.add(listNum.get(randIdx));
            listNum.remove(randIdx);
        }
        return result;
    }

    // konstruktor puzzle yang dibangkitkan secara acak
    public PuzzleBox(int rows, int cols) {
        this.rows = rows;
        this.cols = cols;
        this.M = new int[rows][cols];
        ArrayList<Integer> elmtList = randomizeNum();
        int iterator = 0;
        // memasukkan value ke puzzle sesuai urutan yang telah dibangkitkan
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                this.M[i][j] = elmtList.get(iterator);
                iterator++;
            }
        }
    }
```

```java
// konstruktor puzzle dari input file
public PuzzleBox(String filename) {
    String text = "";
    try {
        FileReader reader = new FileReader(filename);
        int data = reader.read();
        while (data != -1) {
            text += (char) data;
            data = reader.read();
        }
        reader.close();
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
    // dipisah per baris dari file input
    String[] def = text.split("\\r?\\n");
    this.rows = def.length;
    this.cols = def[0].split(" ").length;
    this.M = new int[rows][cols];
    // memasukkan elemen puzzle
    for (int i = 0; i < rows; i++) {
        String[] temp = def[i].split(" ");
        for (int j = 0; j < cols; j++) {
            M[i][j] = Integer.parseInt(temp[j]);
        }
    }
}
```

```java
// mencari posisi baris empty tile
public int getRowEmpty(PuzzleBox pBox) {
    for (int i = 0; i < pBox.rows; i++) {
        for (int j = 0; j < pBox.cols; j++) {
            if (pBox.M[i][j] == 0)  return i;
        }
    }
    return -1;
}

// mencari posisi kolom empty tile
public int getColEmpty(PuzzleBox pBox) {
    for (int i = 0; i < pBox.rows; i++) {
        for (int j = 0; j < pBox.cols; j++) {
            if (pBox.M[i][j] == 0)  return j;
        }
    }
    return -1;
}

// fungsi untuk menyalin PuzzleBox
public PuzzleBox copyPBox(PuzzleBox pBox){
    PuzzleBox pbCopy = new PuzzleBox();
    for (int i = 0; i < pBox.rows; i++){
        for(int j = 0; j < pBox.cols; j++){
            pbCopy.M[i][j] = pBox.M[i][j];
        }
    }
    return pbCopy;
}
```

```java
// mencari posisi baris empty tile
public int getRowEmpty(PuzzleBox pBox) {
    for (int i = 0; i < pBox.rows; i++) {
        for (int j = 0; j < pBox.cols; j++) {
            if (pBox.M[i][j] == 0)  return i;
        }
    }
    return -1;
}

// mencari posisi kolom empty tile
public int getColEmpty(PuzzleBox pBox) {
    for (int i = 0; i < pBox.rows; i++) {
        for (int j = 0; j < pBox.cols; j++) {
            if (pBox.M[i][j] == 0)  return j;
        }
    }
    return -1;
}

// fungsi untuk menyalin PuzzleBox
public PuzzleBox copyPBox(PuzzleBox pBox){
    PuzzleBox pbCopy = new PuzzleBox();
    for (int i = 0; i < pBox.rows; i++){
        for(int j = 0; j < pBox.cols; j++){
            pbCopy.M[i][j] = pBox.M[i][j];
        }
    }
    return pbCopy;
}

// convert puzzle ke list untuk mempermudah pemeriksaan
ArrayList<Integer> puzzleToList(PuzzleBox pBox) {
    ArrayList<Integer> pList = new ArrayList<Integer>();
    for(int i = 0; i < pBox.rows; i++) {
        for(int j = 0; j < pBox.cols; j++) {
            pList.add(pBox.M[i][j]);
        }
    }
    return pList;
}
```

```java
// teorema untuk menghitung reachable goal
public int[] teoremaKurang(PuzzleBox pBox) {
    int[] posisiX = {1, 3, 4, 6, 9, 11, 12, 14};    // posisi nilai X=1
    ArrayList<Integer> listX = arrayToList(posisiX);
    ArrayList<Integer> pList = puzzleToList(pBox);  // urutan puzzle
    int[] kurang = new int[17];     // elemen terakhir adalah nilai X
    int nilaiX = 0;
    for (int i = 0; i < pList.size(); i++) {
        // jika sel adalah kotak kosong
        if (pList.get(i) == 0) {
            if (listX.contains(i)) {
                nilaiX = 1;
            } else {
                nilaiX = 0;
            }
            kurang[15] = pList.size() - (i+1);
            kurang[16] = nilaiX;
        } else {    // sel bukan kotak kosong
            for (int j = i+1; j < pList.size(); j++) {
                if (pList.get(j) < pList.get(i) && pList.get(j) != 0) {
                    kurang[(pList.get(i) - 1)]++;
                }
            }
        }
    }
    // menampilkan nilai kurang(i) tiap ubin
    for (int i = 0; i < 16; i++) {
        System.out.println("Nilai Kurang(" + Integer.toString(i+1) +
        ") = " + Integer.toString(kurang[i]));
    }
    return kurang;
}
```

```java
// memeriksa apakah goal dapat dicapai
public boolean isGoalReachable(int[] kurang) {
    int total = kurang[16];     // inisialisasi dengan nilai X
    for (int i = 0; i < 16; i++) {
        total += kurang[i];     // menghitung total kurang(i)
    }
    System.out.println("Nilai dari teorema Reachable Goal: " + Integer.toString(total));
    return (total % 2 == 0);
}

// menampilkan puzzle ke terminal
public void printPuzzleBox() {
    for (int i = 0; i < this.rows; i++) {
        for (int j = 0; j < this.cols; j++) {
            System.out.print(Integer.toString(this.M[i][j]) + " ");
        }
        System.out.println();
    }
}

}
```

## *Class BranchNBound*

Class BranchNBound digunakan untuk menerapkan algoritma Branch and Bound dan menampilkan path solusi. Berikut source codenya

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

You, 3 minutes ago | 1 author (You)
public class BranchNBound {

    You, 3 minutes ago | 1 author (You)
    class Node {
        Node parent;
        PuzzleBox pBox;
        int x, y;
        int cost, moves;
    }

    // membuat node baru
    public Node newNode(PuzzleBox pBox, int x, int y, int newX, int newY, int moves, Node parent) {
        Node node = new Node();
        node.parent = parent;
        node.pBox = pBox.copyPBox(pBox);
        // swap empty tile ke suatu arah
        int temp = node.pBox.M[x][y];
        node.pBox.M[x][y] = node.pBox.M[newX][newY];
        node.pBox.M[newX][newY] = temp;
        // menghitung cost dan jumlah simpul
        node.cost = costToGoal(node.pBox);
        node.moves = moves;
        // update index empty tile
        node.x = newX;
        node.y = newY;
        return node;
    }
```

```java
// menghitung cost untuk mencapai goal node dari node saat ini
public int costToGoal(PuzzleBox pBox) {
    int[] goal = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 0};
    ArrayList<Integer> goalList = pBox.arrayToList(goal);
    ArrayList<Integer> pList = pBox.puzzleToList(pBox);
    int cost = 0;
    for (int i = 0; i < pList.size(); i++) {
        if (pList.get(i) == 0)  continue;
        else if (pList.get(i) != goalList.get(i))    cost++;
    }
    return cost;
}

// down, left, up, right
int row[] = {1, 0, -1, 0};
int col[] = {0, -1, 0, 1};

// memeriksa jika pergerakan valid
boolean isValid(int x, int y) {
    return (x >= 0 && x < 4 && y >= 0 && y < 4);
}

// print path dari root node ke goal node
void printPath(Node root) {
    if (root == null)    return;
    printPath(root.parent);
    root.pBox.printPuzzleBox();
    System.out.println();
}
```

```java
// comparator untuk mengatur urutan priority queue
class pqComparator implements Comparator<Node> {
    public int compare(Node node1, Node node2) {
        if (node1.cost > node2.cost) {
            return 1;
        } else if (node1.cost < node2.cost) {
            return -1;
        }
        return 0;
    }
}

// fungsi penyelesaian
void solve(PuzzleBox pBox, int x, int y) {
    PriorityQueue<Node> pq = new PriorityQueue<Node>(new pqComparator());
    Node root = newNode(pBox, x, y, x, y, 0, null);
    // push root ke prio queue dan melakukan pencarian
    // selama prio queue belum kosong
    pq.add(root);
    while (!pq.isEmpty()) {
        // mengambil head dari priority queue
        Node min = pq.poll();
        // jika node adalah simpul solusi (tidak memiliki cost ke goal)
        if (min.cost == 0) {
            printPath(min);
            System.out.println("\nJumlah simpul: " + Integer.toString(min.moves));
            return;
        }
```

```
            // membangkitkan simpul dari perpindahan yang valid
            for (int i = 0; i < 4; i++) {
                if (isValid(min.x + row[i], min.y + col[i])) {
                    // membuat child node
                    Node child = newNode(min.pBox, min.x, min.y,
                                         min.x + row[i], min.y + col[i],
                                         min.moves + 1, min);
                    // menambah child ke prio queue
                    pq.add(child);
                }
            }
        }
    }

}
```

## *Class Main*

Class main adalah tempat untuk menjalankan program, berikut kodenya

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

You, 2 minutes ago | 1 author (You)
public class main {
    Run | Debug
    public static void main(String[] args) throws IOException {
        PuzzleBox pBox = new PuzzleBox();
        Scanner inputInt = new Scanner(System.in);
        Scanner inputTxt = new Scanner(System.in);
        BranchNBound search = new BranchNBound();

        System.out.println("PILIH TIPE MASUKAN PUZZLE");
        System.out.println("1. Dibangkitkan secara acak oleh program");
        System.out.println("2. Menggunakan file teks");
        System.out.println("Tipe masukan yang dipilih (ketik angkanya):");
        int tipeInput = inputInt.nextInt();
        if (tipeInput == 1) {
            pBox = new PuzzleBox(4, 4);
        } else if (tipeInput == 2) {
            System.out.println("Masukkan nama file:");
            String filename = inputTxt.nextLine();
            pBox = new PuzzleBox(filename);
        }
        System.out.println("\nPosisi awal puzzle sebagai berikut:");
        pBox.printPuzzleBox();
```

```
        long startTime = System.nanoTime();

        System.out.println("\nNilai fungsi Kurang(i) setiap ubin:");
        int[] kurang = pBox.teoremaKurang(pBox);

        if (!pBox.isGoalReachable(kurang)) {
            System.out.println("\nBerdasarkan teorema Reachable Goal, puzzle ini tidak ada solusinya.");
        } else {    // goal dapat dicapai
            System.out.println("\nBerikut urutan puzzle dari awal hingga mencapai goal:");
            search.solve(pBox, pBox.getRowEmpty(pBox), pBox.getColEmpty(pBox));
        }
        long endTime = System.nanoTime();
        long totalTime = endTime - startTime;
        System.out.println("\nWaktu eksekusi program (dalam nanosecond): " + totalTime);
        inputInt.close();
        inputTxt.close();
    }
}
```

# CONTOH INPUT/OUTPUT

## *Test Solvable*

### 1) testSolvable1.txt

```
PILIH TIPE MASUKAN PUZZLE
1. Dibangkitkan secara acak oleh program
2. Menggunakan file teks
Tipe masukan yang dipilih (ketik angkanya):
2
Masukkan nama file:
../test/testSolvable1.txt

Posisi awal puzzle sebagai berikut:
1 2 3 4
5 6 0 8
9 10 7 11
13 14 15 12
```

```
Nilai fungsi Kurang(i) setiap ubin:
Nilai Kurang(1) = 0
Nilai Kurang(2) = 0
Nilai Kurang(3) = 0
Nilai Kurang(4) = 0
Nilai Kurang(5) = 0
Nilai Kurang(6) = 0
Nilai Kurang(7) = 0
Nilai Kurang(8) = 1
Nilai Kurang(9) = 1
Nilai Kurang(10) = 1
Nilai Kurang(11) = 0
Nilai Kurang(12) = 0
Nilai Kurang(13) = 1
Nilai Kurang(14) = 1
Nilai Kurang(15) = 1
Nilai Kurang(16) = 9
Nilai dari teorema Reachable Goal: 16
```

```
Berikut urutan puzzle dari awal hingga mencapai goal:
1 2 3 4
5 6 0 8
9 10 7 11
13 14 15 12

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

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

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


Jumlah simpul: 3

Waktu eksekusi program (dalam nanosecond): 37518900
```

## 2) testSolvable2.txt

```
PILIH TIPE MASUKAN PUZZLE
1. Dibangkitkan secara acak oleh program
2. Menggunakan file teks
Tipe masukan yang dipilih (ketik angkanya):
2
Masukkan nama file:
../test/testSolvable2.txt

Posisi awal puzzle sebagai berikut:
5 1 3 4
9 2 7 8
0 6 15 11
13 10 14 12
```

```
Nilai fungsi Kurang(i) setiap ubin:
Nilai Kurang(1) = 0
Nilai Kurang(2) = 0
Nilai Kurang(3) = 1
Nilai Kurang(4) = 1
Nilai Kurang(5) = 4
Nilai Kurang(6) = 0
Nilai Kurang(7) = 1
Nilai Kurang(8) = 1
Nilai Kurang(9) = 4
Nilai Kurang(10) = 0
Nilai Kurang(11) = 1
Nilai Kurang(12) = 0
Nilai Kurang(13) = 2
Nilai Kurang(14) = 1
Nilai Kurang(15) = 5
Nilai Kurang(16) = 7
Nilai dari teorema Reachable Goal: 28
```

```
Berikut urutan puzzle dari awal hingga mencapai goal:
5 1 3 4
9 2 7 8
0 6 15 11
13 10 14 12

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

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

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

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

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

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

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

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

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

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


Jumlah simpul: 10

Waktu eksekusi program (dalam nanosecond): 61633100
```

### 3) testSolvable3.txt

```
PILIH TIPE MASUKAN PUZZLE
1. Dibangkitkan secara acak oleh program
2. Menggunakan file teks
Tipe masukan yang dipilih (ketik angkanya):
2
Masukkan nama file:
../test/testSolvable3.txt

Posisi awal puzzle sebagai berikut:
1 2 3 4
5 6 7 8
9 10 0 11
13 14 15 12
```

```
Nilai fungsi Kurang(i) setiap ubin:
Nilai Kurang(1) = 0
Nilai Kurang(2) = 0
Nilai Kurang(3) = 0
Nilai Kurang(4) = 0
Nilai Kurang(5) = 0
Nilai Kurang(6) = 0
Nilai Kurang(7) = 0
Nilai Kurang(8) = 0
Nilai Kurang(9) = 0
Nilai Kurang(10) = 0
Nilai Kurang(11) = 0
Nilai Kurang(12) = 0
Nilai Kurang(13) = 1
Nilai Kurang(14) = 1
Nilai Kurang(15) = 1
Nilai Kurang(16) = 5
Nilai dari teorema Reachable Goal: 8
```

```
Berikut urutan puzzle dari awal hingga mencapai goal:
1 2 3 4
5 6 7 8
9 10 0 11
13 14 15 12

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

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


Jumlah simpul: 2

Waktu eksekusi program (dalam nanosecond): 32748300
```

*Test Unsolvable*

## 1) Pembangkitan acak oleh program

```
PILIH TIPE MASUKAN PUZZLE
1. Dibangkitkan secara acak oleh program
2. Menggunakan file teks
Tipe masukan yang dipilih (ketik angkanya):
1

Posisi awal puzzle sebagai berikut:
12 10 6 3
2 1 0 14
11 15 13 9
5 7 4 8
```

```
Nilai fungsi Kurang(i) setiap ubin:
Nilai Kurang(1) = 0
Nilai Kurang(2) = 1
Nilai Kurang(3) = 2
Nilai Kurang(4) = 0
Nilai Kurang(5) = 1
Nilai Kurang(6) = 5
Nilai Kurang(7) = 1
Nilai Kurang(8) = 0
Nilai Kurang(9) = 4
Nilai Kurang(10) = 9
Nilai Kurang(11) = 5
Nilai Kurang(12) = 11
Nilai Kurang(13) = 5
Nilai Kurang(14) = 7
Nilai Kurang(15) = 6
Nilai Kurang(16) = 9
Nilai dari teorema Reachable Goal: 67

Berdasarkan teorema Reachable Goal, puzzle ini tidak ada solusinya.

Waktu eksekusi program (dalam nanosecond): 22642900
```

## 2) testUnsolvable1.txt

```
PILIH TIPE MASUKAN PUZZLE
1. Dibangkitkan secara acak oleh program
2. Menggunakan file teks
Tipe masukan yang dipilih (ketik angkanya):
2
Masukkan nama file:
../test/testUnsolvable1.txt

Posisi awal puzzle sebagai berikut:
1 3 4 15
2 0 5 12
7 6 11 14
8 9 10 13
```

```
Nilai fungsi Kurang(i) setiap ubin:
Nilai Kurang(1) = 0
Nilai Kurang(2) = 0
Nilai Kurang(3) = 1
Nilai Kurang(4) = 1
Nilai Kurang(5) = 0
Nilai Kurang(6) = 0
Nilai Kurang(7) = 1
Nilai Kurang(8) = 0
Nilai Kurang(9) = 0
Nilai Kurang(10) = 0
Nilai Kurang(11) = 3
Nilai Kurang(12) = 6
Nilai Kurang(13) = 0
Nilai Kurang(14) = 4
Nilai Kurang(15) = 11
Nilai Kurang(16) = 10
Nilai dari teorema Reachable Goal: 37

Berdasarkan teorema Reachable Goal, puzzle ini tidak ada solusinya.

Waktu eksekusi program (dalam nanosecond): 17223999
```

# LAMPIRAN

## *Link Github/Google Drive*

https://github.com/roastland/Tucil3_13520031.git
https://drive.google.com/drive/folders/1x6SqNbPMvvIh4ffzOzO5oAzfqW0LQ2Cc?usp=sharing

## *Check List Asisten*

| No. | 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 | | ✓ |

## *Catatan*

- Program dapat membangkitkan puzzle secara acak.
- Program belum dapat menangani puzzle dengan jumlah teorema Reachable Goal yang besar (40-an ke atas).