

# **LAPORAN TUGAS KECIL 3**

## **IF2211 STRATEGI ALGORITMA**

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



Disusun oleh:

13520051 – Flavia Beatrix Leoni A. S.

**PROGRAM STUDI TEKNIK INFORMATIKA**

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**2022**

## Daftar Isi

Daftar Isi .....	i
Daftar Gambar .....	ii
BAB 1 <i>Algoritma Branch and Bound</i> .....	1
BAB 2 <i>Source Code Program</i> .....	3
BAB 3 <i>Pengujian Program</i> .....	9
3.1 <i>fail1.txt</i> .....	9
3.2 <i>fail2.txt</i> .....	9
3.3 <i>success1.txt</i> .....	10
3.4 <i>success2.txt</i> .....	12
3.5 <i>success3.txt</i> .....	13
Lampiran .....	17

## Daftar Gambar

Gambar 3.1 Instansiasi persoalan 15-puzzle fail1.txt .....	9
Gambar 3.2 Hasil Pengujian fail1.txt.....	9
Gambar 3.3 Instansiasi persoalan 15-puzzle fail2.txt .....	9
Gambar 3.4 Hasil Pengujian fail2.txt.....	10
Gambar 3.5 Instansiasi persoalan 15-puzzle success1.txt .....	10
Gambar 3.6 Hasil Pengujian success1.txt - 1.....	11
Gambar 3.7 Hasil Pengujian success1.txt – 2 .....	11
Gambar 3.8 Instansiasi persoalan 15-puzzle success2.txt .....	12
Gambar 3.9 Hasil Pengujian success2.txt - 1 .....	12
Gambar 3.10 Hasil Pengujian success2.txt - 2.....	13
Gambar 3.11 Hasil Pengujian success2.txt - 3.....	13
Gambar 3.12 Instansiasi persoalan 15-puzzle success3.txt .....	13
Gambar 3.13 Hasil Pengujian success3.txt - 1.....	14
Gambar 3.14 Hasil Pengujian success3.txt - 2.....	14
Gambar 3.15 Hasil Pengujian success3.txt - 3.....	15
Gambar 3.16 Hasil Pengujian success3.txt - 4.....	15
Gambar 3.17 Hasil Pengujian success3.txt - 5.....	16
Gambar 3.18 Hasil Pengujian success3.txt - 6.....	16

## BAB 1

### Algoritma *Branch and Bound*

Algoritma *branch and bound* merupakan algoritma yang digunakan untuk meminimalkan atau memaksimalkan suatu fungsi objektif, yang tidak melanggar batasan (*constraints*) persoalan. Setiap simpul yang terbentuk diberi sebuah nilai *cost* sehingga simpul berikutnya yang akan di-expand tidak lagi berdasarkan urutan pembangkitannya, tetapi simpul yang memiliki *cost* yang paling kecil pada kasus minimasi atau *cost* yang paling besar pada kasus maksimasi. Algoritma ini juga menerapkan “pemangkasan” pada jalur yang dianggap tidak lagi mengarah pada solusi. Secara umum, kriteria dari pemangkasan ini adalah nilai simpul tidak lebih baik dari nilai terbaik sejauh ini (*the best solution so far*), simpul tidak merepresentasikan solusi yang ‘feasible’ karena ada batasan yang dilanggar, dan jika solusi pada simpul tersebut hanya terdiri atas satu titik, ambil simpul terbaik dengan membandingkan nilai fungsi obyektif dengan solusi terbaik saat ini.

Persoalan 15-puzzle merupakan salah satu persoalan yang dapat diselesaikan dengan algoritma *branch and bound*. Penulis membuat sebuah program untuk menyelesaikan puzzle ini dalam bahasa Python. Program ini dapat menerima dua macam input matriks yang merepresentasikan posisi awal suatu instansiasi persoalan 15-puzzle, yaitu diberi oleh pengguna dari suatu file masukan, atau dibangkitkan secara acak oleh program.

Kemudian, akan dihitung nilai  $\sum_{i=1}^{16} KURANG(i) + X$  dari matriks tersebut untuk menentukan apakah *goal state* dapat dicapai dari *start state* yaitu matriks masukan. Fungsi *KURANG(i)* akan mengembalikan banyaknya ubin bernomor  $j$  sedemikian sehingga  $j < i$  dan  $POSISI(j) > POSISI(i)$  dimana sel kosong direpresentasikan dengan nilai  $i = 16$ . Sedangkan  $X$  akan bernilai 1 jika sel kosong berada pada posisi dimana jumlah indeks baris dan kolomnya bernilai ganjil dan akan bernilai 0 jika genap. *Goal state* hanya dapat dicapai apabila nilai dari  $\sum_{i=1}^{16} KURANG(i) + X$  bernilai genap.

Apabila *goal state* dapat dicapai, akan dibangkitkan simpul-simpul berupa matriks yang terbentuk dengan memindahkan sel kosong pada matriks awal ke atas, bawah, kanan, dan/atau kiri. Pada tiap simpul ini, akan dihitung nilai taksiran *cost* nya dengan mencari jumlah ubin tidak kosong yang tidak berada pada susunan akhir ditambah dengan panjang lintasan dari simpul akar ke simpul tersebut. Penulis membuat sebuah *class of node* untuk mempermudah penyimpanan simpul. Kelas ini memiliki atribut berupa matriks puzzle, *level*, *parent*, *move*

yang dilakukan, dan taksiran *cost* dari simpul tersebut. Simpul-simpul yang telah dibangkitkan ini kemudian dimasukkan ke dalam sebuah *priority queue* yang diurutkan membesar berdasarkan nilai taksiran *cost* yang telah dihitung. Sementara simpul akar yang telah diperiksa akan dimasukkan ke dalam suatu *array checked* yang menyimpan seluruh simpul yang telah diperiksa agar simpul tersebut tidak diperiksa kembali nantinya.

Simpul yang akan diperiksa selanjutnya adalah simpul yang memiliki taksiran *cost* paling rendah pada *priority queue*. Jika simpul tersebut bukan merupakan *goal*, maka akan dibangkitkan simpul-simpul berisi matriks yang dapat terbentuk dengan memindahkan sel kosong. Simpul ini akan dimasukkan ke dalam *priority queue* jika belum pernah diperiksa sebelumnya. Kemudian, simpul tersebut dimasukkan ke dalam *array checked* untuk menandai bahwa simpul tersebut telah diperiksa. Langkah ini diulang hingga simpul *goal* ditemukan.

Setelah simpul *goal* ditemukan, akan ditampilkan urutan matriks dari posisi awal ke posisi akhir dan waktu eksekusi program serta jumlah simpul yang dibangkitkan untuk mencapai *goal node*.

## BAB 2

### Source Code Program

puzzle.py

```
import numpy as np
from queue import PriorityQueue
import datetime

goal = np.arange(1,17).reshape(4,4)

class node:
    # merepresentasikan node pada tree yang terbentuk
    def __init__(self, matrix, level, parent, move):
        self.matrix = matrix
        self.level = level
        self.parent = parent
        self.move = move
        self.cost = self.calculateCost()

    def __lt__(self, other):
        # digunakan untuk priority queue
        return self.cost < other.cost

    def calculateCost(self):
        # menghitung taksiran ongkos dari simpul
        count = self.level
        for i in range(4):
            for j in range(4):
                if (self.matrix[i][j] != goal[i][j] and self.matrix[i][j] != 16):
                    count += 1
        return count

    def createNode(self, move):
        # membuat child node berdasarkan move masukan
        mat = np.copy(self.matrix)
        idx_kosong = np.where(mat == 16)
        row = idx_kosong[0][0]
        col = idx_kosong[1][0]
        if (move == "up"):
            mat[row][col], mat[row-1][col] = mat[row-1][col], mat[row][col]
        elif (move == "down"):
            mat[row][col], mat[row+1][col] = mat[row+1][col], mat[row][col]
        elif (move == "left"):
            mat[row][col], mat[row][col-1] = mat[row][col-1], mat[row][col]
        elif (move == "right"):
            mat[row][col], mat[row][col+1] = mat[row][col+1], mat[row][col]
```

```

    level = self.level + 1
    newNode = node(mat, level, self, move)
    return newNode

def possibleMove(self):
    # mengembalikan move yang dapat dilakukan
    moves = ["up", "down", "left", "right"]
    # agar tidak kembali ke posisi sebelumnya
    if (self.move == "up") : moves.remove("down")
    elif (self.move == "down") : moves.remove("up")
    elif (self.move == "left") : moves.remove("right")
    elif (self.move == "right") : moves.remove("left")

    idx_kosong = np.where(self.matrix == 16)
    # hapus move yang tidak valid
    if (idx_kosong[0][0] == 0):
        if "up" in moves:
            moves.remove("up")
    if (idx_kosong[0][0] == 3):
        if "down" in moves:
            moves.remove("down")
    if (idx_kosong[1][0] == 0):
        if "left" in moves:
            moves.remove("left")
    if (idx_kosong[1][0] == 3):
        if "right" in moves:
            moves.remove("right")

    return moves

def isMatrixSame(self, mat):
    # mengecek apakah sama dengan matrix mat
    return (self.matrix == mat).all()

def checkPuzzle(self):
    # mengecek apakah node merupakan goal node
    return self.isMatrixSame(goal)

def displayPath(self):
    # menampilkan urutan matriks dari posisi awal ke posisi akhir
    if (self.parent):
        # print("Langkah:", self.move)
        self.parent.displayPath()
        print("\n=====")
        string = "MOVE " + self.move.upper()
        print(f"{string:^14}")
        print("-----")
        displayMatrix(self.matrix)

```

```

def displayMatrix(mat):
    # menampilkan matrix
    for i in range(4):
        for j in range(4):
            if (mat[i][j] == 16):
                print("   ", end= "")
            else:
                print("{:3}".format(mat[i][j]), end= "")
        print()

def Kurang(i, mat):
    # menghitung banyaknya ubin bernomor j sedemikian sehingga j < 1 dan
    POSISI(j) > POSISI(i)
    arr = mat.reshape(-1)
    idx = np.where(arr == i)
    count = 0
    for a in range(idx[0][0]+1, 16):
        if (arr[a] < i):
            count += 1
    return count

def valueX(mat):
    # menentukan nilai X berdasarkan posisi awal sel kosong
    idx_kosong = np.where(mat == 16)
    if ((idx_kosong[0][0] + idx_kosong[1][0]) % 2 == 1):
        return 1
    return 0

def sumKurangPlusX(mat):
    # menghitung nilai dari ΣKurang(i) + X
    sum = valueX(mat)
    for i in range(1,17):
        sum += Kurang(i, mat)
    return sum

def isReachable(mat):
    # menentukan apakah puzzle dapat diselesaikan berdasarkan nilai ΣKurang(i)
    + X
    return (sumKurangPlusX(mat) % 2 == 0)

def hasChecked(currNode, checked):
    # mengecek apakah node telah diperiksa sebelumnya
    for mat in checked:
        if (currNode.isMatrixSame(mat)):
            return True
    return False

```



```

def solvePuzzle(puzzle):
    # menampilkan matriks posisi awal 15-puzzle
    print("\nMatriks posisi awal:")
    displayMatrix(puzzle)

    # nilai dari fungsi Kurang(i) untuk setiap ubin tidak kosong pada posisi awal
    print("\nNilai dari fungsi Kurang(i) untuk setiap ubin:")
    for i in range(1,17):
        print("Kurang({}) = {}".format(i, Kurang(i, puzzle)))

    # nilai dari  $\sum Kurang(i) + X$ 
    print("\nNilai dari  $\sum Kurang(i) + X$  adalah", sumKurangPlusX(puzzle))

    startTime = datetime.datetime.now()
    if (isReachable(puzzle)):
        root = node(puzzle, 0, None, "-")
        liveNodes = PriorityQueue()
        liveNodes.put(root)
        checked = []
        moves = root.possibleMove()
        countNode = 1

        while (True):
            currNode = liveNodes.get()
            checked.append(currNode.matrix)

            if (not(currNode.checkPuzzle())):
                moves = currNode.possibleMove()
                for move in moves:
                    newNode = currNode.createNode(move)
                    countNode += 1
                    if (not(hasChecked(newNode, checked))):
                        liveNodes.put(newNode)
            else: # goal node ditemukan
                break

            if (liveNodes.empty()):
                break

        endTime = datetime.datetime.now()
        if (currNode.checkPuzzle()): # goal node ditemukan
            # menampilkan urutan matriks dari posisi awal hingga akhir
            print("\nUrutan matriks dari posisi awal hingga akhir:\n")
            currNode.displayPath()
            print("\nPuzzle berhasil diselesaikan!")
            # waktu eksekusi program
            executionTime = (endTime - startTime).total_seconds() * 1000

```

```

        print("Waktu eksekusi program adalah", executionTime, "milliseconds")
        # jumlah simpul yang dibangkitkan
        print("Jumlah simpul yang dibangkitkan sebanyak", countNode)

    else:
        print("Puzzle tidak dapat diselesaikan")

```

inputPuzzle.py

```

import numpy as np

goal = np.arange(1,17).reshape(4,4)

def isPuzzleValid(puzzle):
    # mengecek apakah puzzle masukan merupakan puzzle yang valid
    arr = (np.sort(puzzle, axis=None)).reshape(4,4)
    return (arr == goal).all()

def inputPuzzle():
    print("Pilih cara untuk memberi masukan puzzle:")
    print("1. Dari file teks masukan")
    print("2. Dibangkitkan secara acak oleh program")

    try:
        opt = int(input(" > "))
        if (opt == 1):
            # input dari file
            print("\n*NOTE: posisi ubin kosong direpresentasikan dengan 16 pada file teks")
            filename = "../test/" + input("Masukkan nama file: ")
            return readFile(filename)

        elif (opt == 2):
            # posisi awal puzzle dibangkitkan secara acak
            puzzle = np.arange(1,17)
            np.random.shuffle(puzzle)
            puzzle = puzzle.reshape(4,4)
            return puzzle

        else:
            print("Masukan tidak valid!\n")
            return inputPuzzle()

    except:
        print("Masukan tidak valid!\n")
        return inputPuzzle()

```

```
def readFile(filename):
    # membaca matrix dari file teks masukan
    try:
        with open(filename) as file:
            puzzle = np.zeros([4,4], dtype = "int")
            for i in range(4):
                line = file.readline().split()
                for j in range(4):
                    puzzle[i][j] = int(line[j])
            if (isPuzzleValid(puzzle)):
                return puzzle
            else:
                print("Puzzle tidak valid!\n")
                return inputPuzzle()
    except FileNotFoundError:
        print("File tidak ditemukan!\n")
        return inputPuzzle()
```

main.py
---------

```
from inputPuzzle import inputPuzzle
from puzzle import solvePuzzle

print("  _ _ _ _ _ _ _ _ _ _")
print(" / || _| _| _ \ _ _ _ _| | _ / _| _| | _ _ _ _ _")
print(" | || _ \ _|| _/| || || _ /| _ /| / -_) \_ \_ _ \ | \_ \_ // -_)|
'_|")
print(" | || _/ _| _ \_ _/ _| _| _/ _| _| _ \_ /| | \_ \_|||")
print("\nby Flavia Beatrix Leoni A. S. - 13520051\n")

puzzle = inputPuzzle()
solvePuzzle(puzzle)
```

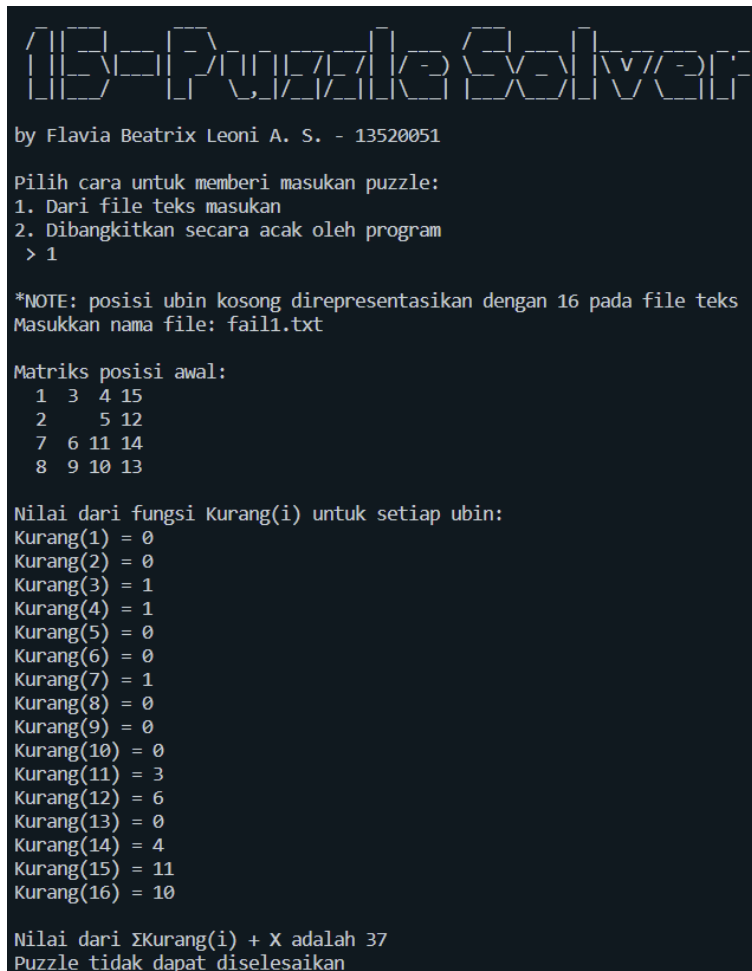
## BAB 3

### Pengujian Program

#### 3.1 fail1.txt

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

Gambar 3.1 Instansiasi persoalan 15-puzzle fail1.txt



```
15-Puzzle Solver
by Flavia Beatrix Leoni A. S. - 13520051

Pilih cara untuk memberi masukan puzzle:
1. Dari file teks masukan
2. Dibangkitkan secara acak oleh program
> 1

*NOTE: posisi ubin kosong direpresentasikan dengan 16 pada file teks
Masukkan nama file: fail1.txt

Matriks posisi awal:
  1  3  4 15
  2   5 12
  7  6 11 14
  8  9 10 13

Nilai dari fungsi Kurang(i) untuk setiap ubin:
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

Nilai dari  $\sum \text{Kurang}(i) + X$  adalah 37
Puzzle tidak dapat diselesaikan
```

Gambar 3.2 Hasil Pengujian fail1.txt

#### 3.2 fail2.txt

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

Gambar 3.3 Instansiasi persoalan 15-puzzle fail2.txt

```

15-Puzzle Solver
by Flavia Beatrix Leoni A. S. - 13520051

Pilih cara untuk memberi masukan puzzle:
1. Dari file teks masukan
2. Dibangkitkan secara acak oleh program
> 1

*NOTE: posisi ubin kosong direpresentasikan dengan 16 pada file teks
Masukkan nama file: fail2.txt

Matriks posisi awal:
 3 10 14
13  4  7  1
 5 12 15  6
 2 11  9  8

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

Nilai dari  $\sum \text{Kurang}(i) + X$  adalah 65
Puzzle tidak dapat diselesaikan

```

Gambar 3.4 Hasil Pengujian fail2.txt

### 3.3 success1.txt

```

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

```

Gambar 3.5 Instansiasi persoalan 15-puzzle success1.txt

```

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

by Flavia Beatrix Leoni A. S. - 13520051

Pilih cara untuk memberi masukan puzzle:
1. Dari file teks masukan
2. Dibangkitkan secara acak oleh program
> 1

*NOTE: posisi ubin kosong direpresentasikan dengan 16 pada file teks
Masukkan nama file: success1.txt

Matriks posisi awal:
 1  2  3  4
 5  6      8
 9 10  7 11
13 14 15 12

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

Nilai dari  $\sum \text{Kurang}(i) + X$  adalah 16

```

Gambar 3.6 Hasil Pengujian success1.txt - 1

```

Urutan matriks dari posisi awal hingga akhir:

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

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

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

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

Puzzle berhasil diselesaikan!
Waktu eksekusi program adalah 0.0 milliseconds
Jumlah simpul yang dibangkitkan sebanyak 10

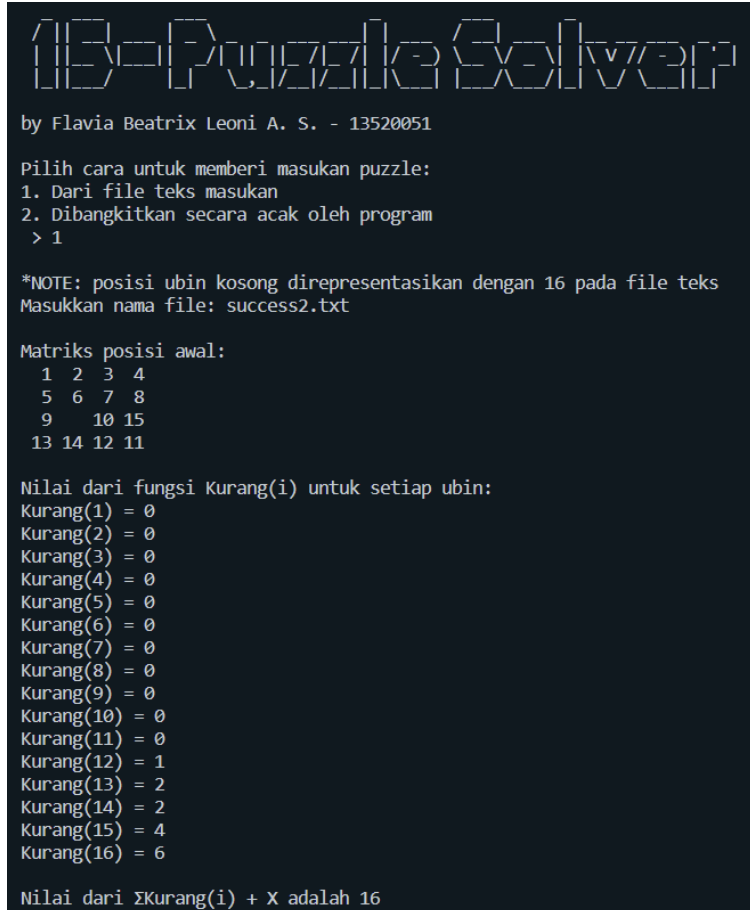
```

Gambar 3.7 Hasil Pengujian success1.txt – 2

### 3.4 success2.txt

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

Gambar 3.8 Instansiasi persoalan 15-puzzle success2.txt



```
15-Puzzle Solver
by Flavia Beatrix Leoni A. S. - 13520051

Pilih cara untuk memberi masukan puzzle:
1. Dari file teks masukan
2. Dibangkitkan secara acak oleh program
> 1

*NOTE: posisi ubin kosong direpresentasikan dengan 16 pada file teks
Masukkan nama file: success2.txt

Matriks posisi awal:
 1  2  3  4
 5  6  7  8
 9 10 15
13 14 12 11

Nilai dari fungsi Kurang(i) untuk setiap ubin:
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) = 0
Kurang(12) = 1
Kurang(13) = 2
Kurang(14) = 2
Kurang(15) = 4
Kurang(16) = 6

Nilai dari  $\sum \text{Kurang}(i) + X$  adalah 16
```

Gambar 3.9 Hasil Pengujian success2.txt - 1

```

Urutan matriks dari posisi awal hingga akhir:

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

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

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

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

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

```

Gambar 3.10 Hasil Pengujian success2.txt - 2

```

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

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

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

Puzzle berhasil diselesaikan!
Waktu eksekusi program adalah 3.945999999999997 milliseconds
Jumlah simpul yang dibangkitkan sebanyak 44

```

Gambar 3.11 Hasil Pengujian success2.txt - 3

### 3.5 success3.txt

```

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

```

Gambar 3.12 Instansiasi persoalan 15-puzzle success3.txt



```

16-Puzzle Solver
by Flavia Beatrix Leoni A. S. - 13520051

Pilih cara untuk memberi masukan puzzle:
1. Dari file teks masukan
2. Dibangkitkan secara acak oleh program
> 1

*NOTE: posisi ubin kosong direpresentasikan dengan 16 pada file teks
Masukkan nama file: success3.txt

Matriks posisi awal:
  1  2  3  4
  9  5 11  8
  7 10 15 14
 13 12   6

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

Nilai dari  $\sum \text{Kurang}(i) + X$  adalah 24

```

Gambar 3.13 Hasil Pengujian success3.txt - 1

```

Urutan matriks dari posisi awal hingga akhir:

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

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

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

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

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

```

Gambar 3.14 Hasil Pengujian success3.txt - 2

```

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

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

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

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

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

```

Gambar 3.15 Hasil Pengujian success3.txt - 3

```

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

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

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

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

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

```

Gambar 3.16 Hasil Pengujian success3.txt - 4

```

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

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

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

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

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

```

Gambar 3.17 Hasil Pengujian success3.txt - 5

```

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

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

Puzzle berhasil diselesaikan!
Waktu eksekusi program adalah 329714.938 milliseconds
Jumlah simpul yang dibangkitkan sebanyak 28565

```

Gambar 3.18 Hasil Pengujian success3.txt - 6

## Lampiran

Link kode program:

[https://github.com/leoniantoinette/Tucil3\\_13520051.git](https://github.com/leoniantoinette/Tucil3_13520051.git)

Poin	Ya	Tidak
1. Program berhasil dikompilasi	√	
2. Program berhasil <i>running</i>	√	
3. Program dapat menerima input dan menuliskan output	√	
4. Luaran sudah benar untuk semua data uji	√	
5. Bonus dibuat		√