IF2211 – Strategi Algoritma

Tugas Kecil 3

Penyelesaian Persoalan 15-Puzzle dengan Algoritma Branch and Bound

Oleh:

13520118 Mohamad Daffa Argakoesoemah

PROGRAM STUDI SARJANA TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

2022

1. Cara Kerja Program

Pada program ini, terdapat beberapa tipe data yang penting, yaitu:

1. *Priority queue* Q. Elemen Q merepresentasikan sebuah simpul, yaitu tupel dengan 5 elemen. Misalnya elemen pada Q adalah e. Masing-masing elemennya merepresentasikan:
    a. Elemen pertama: Nilai fungsi ĉ(e) untuk matriks e
    b. Elemen kedua: Matriks e
    c. Elemen ketiga: Kedalaman e
    d. Elemen keempat: Simpul orang tua dari e yang merupakan sebuah tupel 4 elemen juga
    e. Elemen kelima: String yang berisi pergerakan yang dilakukan pada simpul e
2. Larik B yang menyimpan matriks yang sudah pernah dibangkitkan beserta pergerakannya. Oleh karena itu, elemen pada B adalah tupel 2 elemen yang elemen pertamanya adalah sebuah matriks dan elemennya keduanya adalah string keterangan pergerakan yang dilakukan.

Pertama-pertama, program membaca matriks 4 x 4 yang di-*generate* secara acak oleh program atau dari file teks berdasarkan masukan pengguna. Matriks ini berisi bilangan bulat 1 sampai dengan 16 yang merepresentasikan posisi awal puzzle. Ubin kosong pada puzzle direpresentasikan dengan angka 16. Setelah itu, program menampilkan pada terminal nilai fungsi KURANG(i) untuk masing-masing tile ke-i. Selanjutnya, program menghitung nilai $\sum_{i=1}^{16} KURANG(i) + X$. Jika nilai tersebut ganjil, puzzle tidak dapat diselesaikan dan program berhenti. Jika nilainya genap, puzzle dapat diselesaikan dengan algoritma pencarian langkah-langkahnya sebagai berikut:

1. Instantiasi Q dan lakukan *enqeue* dengan matriks posisi awal puzzle
2. Inisialisasi larik B dengan larik kosong.
3. Jika Q kosong, pencarian dihentikan:
4. Jika Q tidak kosong, lakukan *dequeue* pada Q untuk mendapatkan matriks i yang mempunyai ongkos terkecil (nilai fungsi ĉ(i) terkecil).
5. Jika nilai $\hat{g}(i)$, yaitu jumlah ubin pada matriks i tidak kosong yang tidak terdapat pada susunan akhir puzzle nilainya nol, solusi telah ditemukan dan pencarian dihentikan.
6. Jika nilainya bukan nol, inisialisasi larik penyimpanan sementara misalnya tempB. Inisalisasi ini dilakukan untuk setiap iterasi. Masukkan ke larik tempB setiap matriks anak dari i beserta pergerakan yang dilakukannya.
7. Untuk setiap elemen tempB, jika matriksnya (misalnya j) tidak ada pada larik B dan tidak sama dengan matriks posisi awal puzzle, lakukan hal berikut:
    a. Masukkan j pada larik B.
    b. Hitung nilai ĉ(j).
    c. Hitung level j, yaitu level i + 1.

d. Bentuk tupel dengan 5 elemen yang setiap elemennya adalah sebagai berikut:
    - Elemen pertama: Nilai pada langkah 6(b)
    - Elemen kedua: Matriks j
    - Elemen ketiga: Nilai pada langkah 6(c)
    - Elemen keempat: Matriks i
    - Elemen kelima: Pergerakan yang dilakukan untuk mendapatkan matriks j (didapatkan dari elemen kedua pada larik tempB)
e. Lakukan *enqueue* pada Q dengan tupel pada langkah 6(d).

8. Kembali ke langkah ke-3

Setelah pencarian berakhir, *graphical user interface* GUI akan tampil dan pengguna bisa melihat visualisasi perpindahan tile puzzle dari keadaan awal sampai akhir

2. Source Code Program

2.1. File fifteenPuzzle.py

```python
from queue import PriorityQueue
from copy import deepcopy
from sys import stdout
import time

def searchEmptyTile(matriks):
    '''
        Menerima masukan matriks of integer yang elemennya unik 1-16
        Mengembalikan baris dan kolom letak elemen 16 yang merepresentasikan
        tile kosong
    '''
    for row in range(4):
        for col in range(4):
            if (matriks[row][col] == 16):
                return row, col

def computeX(matriks):
    '''
        Menerima masukan matriks of integer yang elemennya unik 1-16
        Mengembalikan perhitungan nilai X dalam perhitungan reachable goal
    '''
    row, col = searchEmptyTile(matriks)
    if ((row % 2 == 0 and col % 2 != 0) or (row % 2 != 0 and col % 2 == 0)):
        return 1
    else:
        return 0
```

```python
def computeKurang(matriks):
    '''
        Menerima masukan matriks of integer yang elemennya unik 1-16
        Mengembalikan perhitungan sigma KURANG(i) dalam perhitungan reachable goal
        dan array of integer yang berisi hasil fungsi KURANG(i) untuk tiap tile
    '''
    arrayKurang = []
    sigma = 0
    for i in range(1, 17):
        sigmaOneTile = 0
        row = 0
        col = 0
        found = False
        while (row < 4 and not(found)):
            while (col < 4 and not(found)):
                if matriks[row][col] == i:
                    found = True
                else:
                    col += 1
            if (not found):
                col = 0
                row += 1
        for j in range(row, 4):
            for k in range(col, 4):
                if (matriks[j][k] < i):
                    sigma += 1
                    sigmaOneTile += 1
            col = 0
        arrayKurang.append(sigmaOneTile)
    return sigma, arrayKurang
```

```python
def computeKurangPlusX(kurang, matriks):
    '''
        Menerima hasil sigma KURANG(i) dan matriks puzzle yang ingin dihitung
        Mengembalikan hasil sigma KURANG(i) + X
    '''
    return kurang + computeX(matriks)

def isReachable(result):
    '''
        Menerima hasil sigma KURANG(i) + X
        Mengembalikan true jika goal reachable, false jika tidak
    '''
    if (result % 2 == 0):
        return True
    else:
        return False

def upMatriks(matriks):
    '''
        Menerima matriks puzzle
        Mengembalikan matriks hasil pergerakan UP
    '''
    upMatriks = deepcopy(matriks)
    row, col = searchEmptyTile(matriks)
    if (row - 1 >= 0):
        upMatriks[row][col] = upMatriks[row-1][col]
        upMatriks[row-1][col] = 16
    return upMatriks
```

```python
def downMatriks(matriks):
    '''
        Menerima matriks puzzle
        Mengembalikan matriks hasil pergerakan DOWN
    '''
    downMatriks = deepcopy(matriks)
    row, col = searchEmptyTile(matriks)
    if (row + 1 < 4):
        downMatriks[row][col] = downMatriks[row+1][col]
        downMatriks[row+1][col] = 16
    return downMatriks

def leftMatriks(matriks):
    '''
        Menerima matriks puzzle
        Mengembalikan matriks hasil pergerakan LEFT
    '''
    leftMatriks = deepcopy(matriks)
    row, col = searchEmptyTile(matriks)
    if (col - 1 >= 0):
        leftMatriks[row][col] = leftMatriks[row][col-1]
        leftMatriks[row][col-1] = 16
    return leftMatriks

def rightMatriks(matriks):
    '''
        Menerima matriks puzzle
        Mengembalikan matriks hasil pergerakan RIGHT
    '''
    rightMatriks = deepcopy(matriks)
    row, col = searchEmptyTile(matriks)
    if (col + 1 < 4):
        rightMatriks[row][col] = rightMatriks[row][col+1]
        rightMatriks[row][col+1] = 16
    return rightMatriks
```

```python
def computeTaksiran(matriks):
    '''
        Menerima matriks puzzle
        Mengembalikan nilai taksiran atau fungsi g(i), yaitu ongkos untuk
        mencapai simpul tujuan dari simpul i
    '''
    count = 0
    counter = 1
    for i in range(4):
        for j in range(4):
            if (matriks[i][j] != counter):
                count += 1
            counter += 1
    return count

def printPuzzle(matriks):
    '''
        Melakukan print matriks puzzle ke layar
    '''
    for i in range(4):
        for j in range(4):
            if (matriks[i][j] == 16):
                print('    ', end = '')
            else:
                if (matriks[i][j] < 10):
                    print(matriks[i][j], '  ', end = '')
                else:
                    print(matriks[i][j], ' ', end = '')
        print('\n')
```

```python
def solvePuzzle(matriks):
    '''
        Menerima matriks puzzle yang ingin di-solve
        Mengembalikan simpul goal dan jumlah simpul yang sudah dibangkitkan
    '''
    queue = PriorityQueue() # instantiasi priority queue
    queue.put((0, matriks, 0, None, None)) # tupel: (cost, matriks, level, parentNode, move)
    nodeBangkit = [] # inisialisasi array penampung simpul yg sudah dibangkitkan

    print("\nSearching...\nPlease wait")

    while (not(queue.empty())):
        currentNode = queue.get() # dequeue (node yg memiliki cost minimum)
        currentMatriks = currentNode[1] # dapatkan matriksnya (array 2D)
        levelNode = currentNode[2] # dapatkan levelnya

        if (computeTaksiran(currentMatriks) == 0): # jika sudah sama dengan goal, maka pencarian solusi berakhir
            break

        # tambahkan kemungkinan semua pergerakan ke dalam array bangkitTemp
        bangkitTemp = []
        bangkitTemp.append((upMatriks(currentMatriks), "UP"))
        bangkitTemp.append((downMatriks(currentMatriks), "DOWN"))
        bangkitTemp.append((leftMatriks(currentMatriks), "LEFT"))
        bangkitTemp.append((rightMatriks(currentMatriks), "RIGHT"))

        for mat in bangkitTemp:
            if (mat[0] not in nodeBangkit and mat[0] != matriks): # cek apakah node sudah pernah dibangkitkan
                nodeBangkit.append(mat[0])
                taksiran = computeTaksiran(mat[0])
                cost = taksiran + levelNode + 1
                queue.put((cost, mat[0], levelNode + 1, currentNode, mat[1])) # masukkan node ke antrian

        stdout.write("\rJumlah simpul yang telah dibangkitkan sejauh ini: {0}".format(len(nodeBangkit)))
        stdout.flush()

    return currentNode, len(nodeBangkit)
```

```python
def printSolution(finalNode, matriks):
    '''
        Melakukan print ke layar langkah-langkah yang dilakukan untuk memecahkan puzzle
    '''

    path = getPath(finalNode, matriks)

    for i in range (len(path)):
        print("\nLangkah ke-" + str(i + 1), ': ' + path[i][4] + '\n')
        printPuzzle(path[i][1])
        print("================================")

def getPath(finalNode, matriks):
    path = []
    while True:
        path.insert(0, finalNode)
        if (finalNode[3][1] == matriks):
            break
        finalNode = finalNode[3]

    return path
```

```python
def startSolve(matriks):
    '''
        Prosedur utama untuk memecahkan puzzle yang diinginkan
        Mengecek apakah reachable goal, jika iya maka puzzle akan dipecahkan
        Jika tidak maka program berhenti
    '''
    print("\nPosisi awal: \n")
    printPuzzle(matriks)

    print("================================\n")

    totalKurang, arrayKurang = computeKurang(matriks)
    for i in range(len(arrayKurang)):
        print("Nilai KURANG(" + str(i+1) + ") = ", arrayKurang[i])
    kurangPlusX = computeKurangPlusX(totalKurang, matriks)
    print("\nNilai sigma KURANG(i) + X = ", kurangPlusX)

    print("\n================================")

    if (isReachable(kurangPlusX)):
        start = time.time()
        finalNode, nodeBangkit = solvePuzzle(matriks)
        print('\n')
        printSolution(finalNode, matriks)
        print("\nWaktu eksekusi = ", time.time() - start, "s")
        print("Jumlah simpul yang dibangkitkan = ", nodeBangkit, '\n')
        return finalNode
    else:
        print("\nBerdasarkan hasil sigma KURANG(i) + x, persoalan tidak dapat diselesaikan (hasil ganjil)\n")
        return None
```

2.2. File main.py

```python
import os
import os.path
import random
import tkinter as tk
import numpy as np
from fifteenPuzzle import startSolve, getPath

print("Metode pembangkitan posisi awal puzzle:\n1. Random\n2. File teks")
choice = input("Pilih metode pembangkitan posisi awal puzzle (1 atau 2): ")
matriks = []
if (choice == '1'): # random
    sequence = []
    for i in range(1, 17): # array berisi bilangan 1-16
        sequence.append(i)

    random.shuffle(sequence) # shuffle array untuk mendapatkan urutan yg random
    arr = np.reshape(sequence, (4,4)) # ubah array ke dalam matriks
    matriks = arr.tolist()
elif (choice == '2'): # file teks
    fileName = input("Masukkan nama file yang ada pada folder test: ")
    try:
        if (os.path.basename(os.path.normpath(os.getcwd())) == 'src'):
            os.chdir("..")
        currentDir = os.getcwd() # direktori tempat menjalankan program

        # mendapatkan semua path yang terdapat pada direktori repository (parent directory dari src)
        arrayOfPath = [os.path.join(currentDir, arrayOfPath) for arrayOfPath in os.listdir(currentDir)
            if os.path.isdir(os.path.join(currentDir, arrayOfPath))]

        # pengecekan apakah terdapat file tsb pada folder test
        for item in arrayOfPath:
            if os.path.exists(item + '\\' + fileName) and os.path.basename(os.path.normpath(item)) == 'test':
                file = item + '\\' + fileName
        f = open(file, 'r')
    except: # file tidak ditemukan
        print("File tidak ditemukan pada folder test!")
        quit()
```

```
    matriks = [[int(elmt) for elmt in line.split()] for line in f] # ubah file ke dalam matriks
else:
    print("Silakan pilih 1 atau 2!")
    quit()

finalNode = startSolve(matriks) # cari solusi puzzle

# jika puzzle bisa dipecahkan, tampilkan UI
if (finalNode != None):
    puzzleUI = tk.Tk(className = "15-Puzzle")
    puzzleUI.resizable(False, False)

    tiles = [] # array penampung label yang merepresentasikan tile pada UI

    # inisialisasi puzzle dengan keadaan awal
    for i in range(4):
        for j in range(4):
            if (matriks[i][j] == 16): # jika tile kosong
                relief = "flat"
                text = ''
                bgColor = "#ffffff"
                fgColor = "#ffffff"
            else:
                relief = "raised"
                bgColor = "#247881"
                fgColor = "#17252a"
                text = str(matriks[i][j])
            tile = tk.Label(puzzleUI, text = text, width = 4, height = 2, font=("Segoe UI Semibold", 30), bg= bgColor,
                fg = fgColor, relief = relief)
            tiles.append(tile)

    # posisikan tile dengan grid
    idx = 0
    for i in range(4):
        for j in range(4):
            tiles[idx].grid(row = i, column = j, padx = 0.5, pady = 0.5)
            idx += 1
```

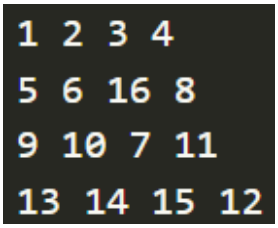
```
    path = getPath(finalNode, matriks)
    step = -1 # counter untuk menghitung step yang sudah ditampilkan pada UI

    def resetPuzzle():
        '''
            Mereset tampilan puzzle kembali ke tampilan awal
        '''
        idx = 0
        for i in range(4):
            for j in range(4):
                if (matriks[i][j] == 16): # jika tile kosong
                    relief = "flat"
                    text = ''
                    bgColor = "#ffffff"
                    fgColor = "#ffffff"
                else:
                    relief = "raised"
                    bgColor = "#247881"
                    fgColor = "#17252a"
                    text = str(matriks[i][j])
                tiles[idx].config(text = text, background = bgColor, fg = fgColor, relief = relief)
                idx += 1

    def startPuzzle():
        '''
            Prosedur untuk memulai menampilkan pergerakan puzzle pada UI
        '''
        global step
        if step > -1:
            step = -1 # reset step ke -1
        updatePuzzle()
```

```python
def updatePuzzle():
    '''
        Melakukan pergantian tampilan puzzle pada UI
        sesuai dengan urutan pergerakan yang dilakukan
    '''
    global step
    startButton["state"] = "disabled" # disable tombol Mulai
    if (step < len(path)): # jika belum mencapai step terakhir
        if step == -1:
            resetPuzzle()
        else :
            idx = 0
            for i in range(4):
                for j in range(4):
                    if (path[step][1][i][j] == 16): # jika tile kosong
                        relief = "flat"
                        text = ''
                        bgColor = "#ffffff"
                        fgColor = "#ffffff"
                    else:
                        relief = "raised"
                        bgColor = "#247881"
                        fgColor = "#17252a"
                        text = str(path[step][1][i][j])
                    tiles[idx].config(text = text, background = bgColor, fg = fgColor, relief = relief)
                    idx += 1
        step += 1
        puzzleUI.after(1000, updatePuzzle) # panggil kembali prosedur updatePuzzle setelah 1 detik
    else: # pergerakan sudah selesai
        startButton["state"] = "normal" # aktifkan kembali tombol Mulai

# tambahkan tombol Mulai
startButton = tk.Button(puzzleUI, text = "Mulai", font=("Segoe UI Semibold", 12), command = startPuzzle, width = 5, height = 2)
startButton.grid(row = 4, column = 1, columnspan = 2, padx = 1, pady = 1)

puzzleUI.mainloop()
```

3. *Screenshot Input dan Output Program*

3.1. *Test Case Reachable* 1

Input:



Output:

```
Metode pembangkitan posisi awal puzzle:
1. Random
2. File teks
Pilih metode pembangkitan posisi awal puzzle (1 atau 2): 2
Masukkan nama file yang ada pada folder test: reach1.txt

Posisi awal:

1   2   3   4

5   6       8

9   10  7   11

13  14  15  12

================================

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 sigma KURANG(i) + X =  16
```

```
================================

Searching...
Please wait
Jumlah simpul yang telah dibangkitkan sejauh ini: 9


Langkah ke-1 : DOWN

1   2   3   4

================================

Langkah ke-2 : RIGHT

1   2   3   4

5   6   7   8

9   10  11

13  14  15  12

================================

Langkah ke-3 : DOWN

1   2   3   4

5   6   7   8

9   10  11  12

13  14  15

================================
```

```
Waktu eksekusi =  0.009597063064575195 s
Jumlah simpul yang dibangkitkan =  9
```

GUI keadaan puzzle awal:

3.2. *Test Case Reachable* 2

Input:

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

Output:

```
Metode pembangkitan posisi awal puzzle:
1. Random
2. File teks
Pilih metode pembangkitan posisi awal puzzle (1 atau 2): 2
Masukkan nama file yang ada pada folder test: reach2.txt

Posisi awal:

5   1   3   4

9   2   7   8

    6   15  11

13  10  14  12

================================

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 sigma KURANG(i) + X =  28
```

```
==============================

Searching...
Please wait
Jumlah simpul yang telah dibangkitkan sejauh ini: 24


Langkah ke-1 : UP

5   1   3   4

    2   7   8

9   6   15  11

13  10  14  12

==============================

Langkah ke-2 : UP

    1   3   4

5   2   7   8

9   6   15  11

13  10  14  12

==============================
```

```
Langkah ke-3 : RIGHT

1       3   4

5   2   7   8

9   6   15  11

13  10  14  12

==============================

Langkah ke-4 : DOWN

1   2   3   4

5       7   8

9   6   15  11

13  10  14  12

==============================

Langkah ke-5 : DOWN

1   2   3   4

5   6   7   8

9       15  11

13  10  14  12

==============================
```

```
Langkah ke-6 : DOWN

1   2   3   4

5   6   7   8

9   10  15  11

13      14  12

================================

Langkah ke-7 : RIGHT

1   2   3   4

5   6   7   8

9   10  15  11

13  14      12

================================

Langkah ke-8 : UP

1   2   3   4

5   6   7   8

9   10      11

13  14  15  12

================================
```

```
Langkah ke-9 : RIGHT

1   2   3   4

5   6   7   8

9   10  11

13  14  15  12

================================

Langkah ke-10 : DOWN

1   2   3   4

5   6   7   8

9   10  11  12

13  14  15

================================

Waktu eksekusi =  0.026250600814819336 s
Jumlah simpul yang dibangkitkan =  24
```

GUI keadaan puzzle awal:

3.3. *Test Case Reachable* 3

Input:

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

Output:

```
Metode pembangkitan posisi awal puzzle:
1. Random
2. File teks
Pilih metode pembangkitan posisi awal puzzle (1 atau 2): 2
Masukkan nama file yang ada pada folder test: reach3.txt

Posisi awal:

1   2   3   4

5   6   7   11

9   10  12  8

13  14  15

================================

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) =  1
Nilai KURANG(10) =  1
Nilai KURANG(11) =  3
Nilai KURANG(12) =  1
Nilai KURANG(13) =  0
Nilai KURANG(14) =  0
Nilai KURANG(15) =  0
Nilai KURANG(16) =  0

Nilai sigma KURANG(i) + X =  6
```

```
================================

Searching...
Please wait
Jumlah simpul yang telah dibangkitkan sejauh ini: 773


Langkah ke-1 : UP

1   2   3   4

5   6   7   11

9   10  12

13  14  15  8

================================

Langkah ke-2 : UP

1   2   3   4

5   6   7

9   10  12  11

13  14  15  8

================================
```

```
Langkah ke-3 : LEFT

1   2   3   4

5   6       7

9   10  12  11

13  14  15  8

================================

Langkah ke-4 : DOWN

1   2   3   4

5   6   12  7

9   10      11

13  14  15  8

================================

Langkah ke-5 : RIGHT

1   2   3   4

5   6   12  7

9   10  11

13  14  15  8

================================
```

```
Langkah ke-6 : DOWN

1   2   3   4

5   6   12  7

9   10  11  8

13  14  15

================================

Langkah ke-7 : LEFT

1   2   3   4

5   6   12  7

9   10  11  8

13  14      15

================================

Langkah ke-8 : UP

1   2   3   4

5   6   12  7

9   10      8

13  14  11  15

================================
```

```
Langkah ke-9 : UP

1   2   3   4

5   6       7

9   10  12  8

13  14  11  15

================================

Langkah ke-10 : RIGHT

1   2   3   4

5   6   7

9   10  12  8

13  14  11  15

================================

Langkah ke-11 : DOWN

1   2   3   4

5   6   7   8

9   10  12

13  14  11  15

================================
```

```
Langkah ke-12 : LEFT

1   2   3   4

5   6   7   8

9   10      12

13  14  11  15

================================

Langkah ke-13 : DOWN

1   2   3   4

5   6   7   8

9   10  11  12

13  14      15

================================

Langkah ke-14 : RIGHT

1   2   3   4

5   6   7   8

9   10  11  12

13  14  15

================================
Waktu eksekusi =  0.09020781517028809 s
Jumlah simpul yang dibangkitkan =  773
```

GUI keadaan puzzle awal:

3.4. *Test Case Unreachable* 1

Input:

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

Output:

```
Metode pembangkitan posisi awal puzzle:
1. Random
2. File teks
Pilih metode pembangkitan posisi awal puzzle (1 atau 2): 2
Masukkan nama file yang ada pada folder test: unreach1.txt

Posisi awal:

1   3   4   15

2       5   12

7   6   11  14

8   9   10  13

================================

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 sigma KURANG(i) + X =  37
================================

Berdasarkan hasil sigma KURANG(i) + x, persoalan tidak dapat diselesaikan (hasil ganjil)
```

3.5. *Test Case Unreachable* 2

Input:

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

Output:

```
Metode pembangkitan posisi awal puzzle:
1. Random
2. File teks
Pilih metode pembangkitan posisi awal puzzle (1 atau 2): 2
Masukkan nama file yang ada pada folder test: unreach2.txt

Posisi awal:

1   3   2   4

5   11      8

13  10  7   6

9   14  15  12

================================

Nilai KURANG(1) =  0
Nilai KURANG(2) =  0
Nilai KURANG(3) =  1
Nilai KURANG(4) =  0
Nilai KURANG(5) =  0
Nilai KURANG(6) =  0
Nilai KURANG(7) =  1
Nilai KURANG(8) =  2
Nilai KURANG(9) =  0
Nilai KURANG(10) =  3
Nilai KURANG(11) =  5
Nilai KURANG(12) =  0
Nilai KURANG(13) =  5
Nilai KURANG(14) =  1
Nilai KURANG(15) =  1
Nilai KURANG(16) =  9

Nilai sigma KURANG(i) + X =  29
================================

Berdasarkan hasil sigma KURANG(i) + x, persoalan tidak dapat diselesaikan (hasil ganjil)
```

4. Berkas Teks Contoh Instantiasi Persoalan 15-Puzzle

Berikut lima berkas teks contoh instantitasi persoalan 15-Puzzle. Angka 16 pada matriks merepresentasikan ubin kosong pada puzzle.

Berkas 1 (reachable):

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

Berkas 2 (reachable):

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

Berkas 3 (reachable):

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

Berkas 4 (unreachable):

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

Berkas 5 (unreachable):

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

5. Alamat Kode Program

https://github.com/daffarg/Tucil3-Stima

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