Tugas Kecil 2 IF2211 Strategi Algoritma

Semester II tahun 2021/2022

Penyelesaian Persoalan 15-Puzzle dengan Algorigtma *Branch and Bound*

Disusun oleh:

Brianaldo Phandiarta 13520113

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

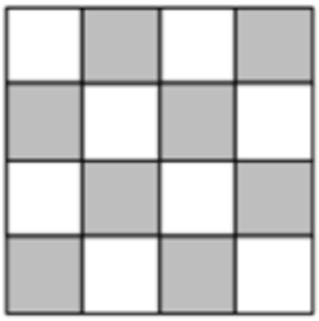
INSTITUT TEKNOLOGI BANDUNG

2022

I. Algoritma *Branch and Bound* dalam Penyelesaian Persoalan 15-Puzzle

Berikut merupakan langkah-langkah algoritma *branch and bound* dalam penyelesaian persoalan 15-puzzle:

1. Periksa apakah state awal dapat mencapai state tujuan menggunakan teorema status tujuan hanya dapat dicapai dari status awal jika $\sum_{i=1}^{16} KURANG(i) + X$ bernilai genap:
   - Menghitung $\sum_{i=1}^{16} KURANG(i)$:
     $KURANG(i)$ merupakan banyaknya sel bernomor i sedemikian sehingga $j < i$ dan $POSISI(j) > POSISI(i)$ dengan $POSISI(i)$ merupakan posisi sel bernomor i pada susunan yang diperiksa.
   - Menghitung X:
     Jika pada state awal posisi sel kosong puzzle berada pada sel yang diarsir, X bernilai 1. Jika sebaliknya, X bernilai 0.



Gambar 1.1. Puzzle Arsir untuk Menghitung Nilai X

2. Jika state tujuan tidak dapat dicapai, algoritma selesai. Jika state tujuan dapat dicapai, lanjut ke langkah ketiga.
3. Lakukan iterasi langkah keempat sampai langkah ketujuh hingga susunan puzzle sama dengan state tujuan.
4. Bangkitkan simpul anak yang mungkin, yaitu sel kosong bergerak ke atas, ke bawah, ke kiri, dan ke kanan. Periksa apakah pergerakan sel valid.
5. Pada setiap simpul anak yang dibangkitkan, hitung cost dari simpul menggunakan rumus berikut.
6. Masukkan simpul anak yang telah dibangkitkan ke dalam antrian berprioritas berdasarkan cost terkecil.
7. Pilih simpul dengan cost terkecil untuk dilakukan evaluasi.

II. Kode Program

1. Pranala Kode Program

Google Drive:

https://drive.google.com/drive/folders/1hVUu6vsqai2heM7Zv7Qed7w1QWycQv0L?usp=sharing

Github:

https://github.com/Brianaldo/Tucil3_13520113

2. *Source Code*

Constant.py

Modul ini berisi konstanta yang akan digunakan dalam program, seperti state tujuan, konstanta pergerakan sel, dan hash table untuk menyimpan simpul yang telah diperiksa.

```
import numpy as np

target = np.array([[ 1,  2,  3,  4],
                   [ 5,  6,  7,  8],
                   [ 9, 10, 11, 12],
                   [13, 14, 15,  0]])

move = np.array([[-1,  0], # up
                 [ 1,  0], # down
                 [ 0, -1], # left
                 [ 0,  1]] # right
                )

visited = {}
```

Node.py

Modul ini berisi kelas Node yang digunakan sebagai simpul.

```
class Node:
    # Class Attribute:
    # parent      : Node Reference
    # puzzle      : np.array([[x, x, x],
    #                         [x, x, x],
    #                         [x, x, x]])
    # position    : np.array([x, y])
    # level       : integer
    # cost        : integer
    def __init__(self, parent, puzzle, position, level, cost):
        self.parent = parent
        self.puzzle = puzzle
        self.position = position
        self.level = level
        self.cost = cost + level

    # Less than definition (>)
    def __gt__(self, next):
        return self.cost > next.cost
```

PriorityQueue.py

Modul ini berisi kelas PriorityQueue yang digunakan sebagai antrian berprioritas yang menyimpan simpul-simpul yang telah dibangkitkan.

```
class PriorityQueue:
    # Class Attribute:
    # pq : array of Node
    #      Node terurut menurun
    def __init__(self):
        self.pq = []

    # isEmpty
    def isEmpty(self):
        return len(self.pq) == 0

    # enqueue
    def enqueue(self, node):
        if (self.isEmpty()):
            self.pq.append(node)
        else:
            for i in range (0, len(self.pq)):
                if (node > self.pq[i]):
                    self.pq.insert(i, node)
                    return
            self.pq.append(node)

    # dequeue
    def dequeue(self):
        return self.pq.pop()
```

Solver.py

Modul ini berisi fungsi-fungis yang digunakan untuk menyelesaikan persoalan 15-Puzzle.

```
from Constant import *
from Node import *
from PriorityQueue import *

def getEmptySpacePosition(puzzle):
    for i in range (4):
        for j in range (4):
            if (puzzle[i, j] == 0):
                return i, j

def isMoveSafe(x, y):
    return x >= 0 and x < 4 and y >= 0 and y < 4

def isSolvable(puzzle):
    sum = 0
    x = 0
    kurang = [0 for i in range (16)]
    for row in range (4):
        for col in range (4):
            count = 0
            if puzzle[row, col] == 0:
                if (row + col) % 2 == 1:
                    sum += 1
                    x = 1
                count += 16 - (4 * row + col + 1)
            else:
                for k in range (4 * row + col + 1, 16):
                    subRow = k // 4
                    subCol = k % 4
                    if puzzle[row, col] > puzzle[subRow, subCol] and
puzzle[subRow, subCol] != 0:
                        count += 1
            sum += count
            kurang[puzzle[row, col]] = count
    print()
    print("KURANG(i): ")
    for i in range (1, 16):
        print("Kurang(" + str(i) +  ") = " + str(kurang[i]))
    print("Kurang(16) = " + str(kurang[0]))
    print()
    print("X = " + str(x))
    print()
    print("Sum(KURANG(i)) + X = " + str(sum))
    print()
    if (sum % 2 == 0):
        print("Puzzle dapat diselesaikan!")
        print()
        return True
    else:
        print("Puzzle tidak dapat diselesaikan!")
        print()
        return False

def calcCost(puzzle):
    sum = 0
    for row in range (4):
        for col in range (4):
            if (puzzle[row, col] and
                puzzle[row, col] != target[row, col]):
                sum += 1
    return sum

def printPuzzle(puzzle):
    print()
    for i in range (4):
        for j in range (4):
            if (puzzle[i, j] != 0):
                if (puzzle[i , j] < 10):
                    print(" " + str(puzzle[i, j]), end=" ")
                else:
                    print(puzzle[i, j], end=" ")
            else:
                print("  ", end=" ")
        print()

def printPath(root):
    if root == None:
        return
    printPath(root.parent)
    printPuzzle(root.puzzle)
```

```
def solve(puzzle):
    total_node = 0
    pq = PriorityQueue()

    root = Node(None, puzzle, getEmptySpacePosition(puzzle), 0,
calcCost(puzzle))
    pq.enqueue(root)
    total_node += 1

    while not pq.isEmpty():
        currNode = pq.dequeue()

        if np.array_equal(currNode.puzzle, target):
            print("Total simpul yang dibangkitkan = " + str(total_node))
            return currNode
        else:
            row, col = currNode.position

            for x, y in move:
                newRow, newCol = row + x, col + y
                if (isMoveSafe(newRow, newCol)):
                    tempPuzzle = np.copy(currNode.puzzle)
                    tempPuzzle[row, col], tempPuzzle[newRow, newCol] =
tempPuzzle[newRow, newCol], tempPuzzle[row, col]
                    if not np.array_str(tempPuzzle) in visited.keys():
                        child = Node(currNode, tempPuzzle, (newRow,
newCol), currNode.level + 1, calcCost(tempPuzzle))
                        pq.enqueue(child)
                        visited[np.array_str(tempPuzzle)] = child
                        total_node += 1

            row, col = currNode.position

            for x, y in move:
                newRow, newCol = row + x, col + y
                if (isMoveSafe(newRow, newCol)):
                    tempPuzzle = np.copy(currNode.puzzle)
                    tempPuzzle[row, col], tempPuzzle[newRow, newCol] =
tempPuzzle[newRow, newCol], tempPuzzle[row, col]
                    if not np.array_str(tempPuzzle) in visited.keys():
                        child = Node(currNode, tempPuzzle, (newRow,
newCol), currNode.level + 1, calcCost(tempPuzzle))
                        pq.enqueue(child)
                        visited[np.array_str(tempPuzzle)] = child
                        total_node += 1
```

App.py

Modul ini berisi program utama beserta tampilan GUI.

```
import time
import random
import threading
import sys

from PyQt5.QtWidgets import *
from PyQt5 import QtCore
from PyQt5.QtGui import *
from PyQt5.QtCore import QPropertyAnimation, QRect

from Solver import *
```

```
class Window(QWidget):
    def __init__(self, solution):
        super().__init__()
        self.setGeometry(100, 100, 200, 200)

        self.solution = solution

        self.Puzzle01Label = QLabel( "1", self)
        self.Puzzle02Label = QLabel( "2", self)
        self.Puzzle03Label = QLabel( "3", self)
        self.Puzzle04Label = QLabel( "4", self)
        self.Puzzle05Label = QLabel( "5", self)
        self.Puzzle06Label = QLabel( "6", self)
        self.Puzzle07Label = QLabel( "7", self)
        self.Puzzle08Label = QLabel( "8", self)
        self.Puzzle09Label = QLabel( "9", self)
        self.Puzzle10Label = QLabel("10", self)
        self.Puzzle11Label = QLabel("11", self)
        self.Puzzle12Label = QLabel("12", self)
        self.Puzzle13Label = QLabel("13", self)
        self.Puzzle14Label = QLabel("14", self)
        self.Puzzle15Label = QLabel("15", self)
        self.Puzzle00Label = QLabel( " ", self)

        self.Puzzle01Label.setAlignment(QtCore.Qt.AlignCenter)
        self.Puzzle02Label.setAlignment(QtCore.Qt.AlignCenter)
        self.Puzzle03Label.setAlignment(QtCore.Qt.AlignCenter)
        self.Puzzle04Label.setAlignment(QtCore.Qt.AlignCenter)
        self.Puzzle05Label.setAlignment(QtCore.Qt.AlignCenter)
        self.Puzzle06Label.setAlignment(QtCore.Qt.AlignCenter)
        self.Puzzle07Label.setAlignment(QtCore.Qt.AlignCenter)
        self.Puzzle08Label.setAlignment(QtCore.Qt.AlignCenter)
        self.Puzzle09Label.setAlignment(QtCore.Qt.AlignCenter)
        self.Puzzle10Label.setAlignment(QtCore.Qt.AlignCenter)
        self.Puzzle11Label.setAlignment(QtCore.Qt.AlignCenter)
        self.Puzzle12Label.setAlignment(QtCore.Qt.AlignCenter)
        self.Puzzle13Label.setAlignment(QtCore.Qt.AlignCenter)
        self.Puzzle14Label.setAlignment(QtCore.Qt.AlignCenter)
        self.Puzzle15Label.setAlignment(QtCore.Qt.AlignCenter)
        self.Puzzle00Label.setAlignment(QtCore.Qt.AlignCenter)

        self.Puzzle01Label.resize(50, 50)
        self.Puzzle02Label.resize(50, 50)
        self.Puzzle03Label.resize(50, 50)
        self.Puzzle04Label.resize(50, 50)
        self.Puzzle05Label.resize(50, 50)
        self.Puzzle06Label.resize(50, 50)
        self.Puzzle07Label.resize(50, 50)
        self.Puzzle08Label.resize(50, 50)
        self.Puzzle09Label.resize(50, 50)
        self.Puzzle10Label.resize(50, 50)
        self.Puzzle11Label.resize(50, 50)
        self.Puzzle12Label.resize(50, 50)
        self.Puzzle13Label.resize(50, 50)
        self.Puzzle14Label.resize(50, 50)
        self.Puzzle15Label.resize(50, 50)
        self.Puzzle00Label.resize(50, 50)

        self.Puzzle01Label.setStyleSheet("border: 1px solid black;
background-color: black; color: white")
        self.Puzzle02Label.setStyleSheet("border: 1px solid black;
background-color: black; color: white")
        self.Puzzle04Label.setStyleSheet("border: 1px solid black;
background-color: black; color: white")
        self.Puzzle03Label.setStyleSheet("border: 1px solid black;
background-color: black; color: white")
```

```
        self.Puzzle05Label.setStyleSheet("border: 1px solid black;
background-color: black; color: white")
        self.Puzzle06Label.setStyleSheet("border: 1px solid black;
background-color: black; color: white")
        self.Puzzle07Label.setStyleSheet("border: 1px solid black;
background-color: black; color: white")
        self.Puzzle08Label.setStyleSheet("border: 1px solid black;
background-color: black; color: white")
        self.Puzzle09Label.setStyleSheet("border: 1px solid black;
background-color: black; color: white")
        self.Puzzle10Label.setStyleSheet("border: 1px solid black;
background-color: black; color: white")
        self.Puzzle11Label.setStyleSheet("border: 1px solid black;
background-color: black; color: white")
        self.Puzzle12Label.setStyleSheet("border: 1px solid black;
background-color: black; color: white")
        self.Puzzle13Label.setStyleSheet("border: 1px solid black;
background-color: black; color: white")
        self.Puzzle14Label.setStyleSheet("border: 1px solid black;
background-color: black; color: white")
        self.Puzzle15Label.setStyleSheet("border: 1px solid black;
background-color: black; color: white")
        self.Puzzle00Label.setStyleSheet("border: 1px solid black;")

        self.Puzzle01Label.move(0, 0)
        self.Puzzle02Label.move(0, 0)
        self.Puzzle03Label.move(0, 0)
        self.Puzzle04Label.move(0, 0)
        self.Puzzle05Label.move(0, 0)
        self.Puzzle06Label.move(0, 0)
        self.Puzzle07Label.move(0, 0)
        self.Puzzle08Label.move(0, 0)
        self.Puzzle09Label.move(0, 0)
        self.Puzzle10Label.move(0, 0)
        self.Puzzle11Label.move(0, 0)
        self.Puzzle12Label.move(0, 0)
        self.Puzzle13Label.move(0, 0)
        self.Puzzle14Label.move(0, 0)
        self.Puzzle15Label.move(0, 0)
        self.Puzzle00Label.move(0, 0)

        threading.Thread(target=self.callback, daemon=True).start()

    def callback(self):
        self.drawPath(self.solution)

    def drawPath(self, root):
        if root == None:
            return
        self.drawPath(root.parent)
        self.drawPuzzle(root.puzzle)

    def drawPuzzle(self, puzzle):
        post = [(0, 0) for _ in range (16)]

        for i in range (4):
            for j in range (4):
                post[puzzle[i, j]] = i * 50, j * 50

        self.Puzzle01Label.move(post[ 1][1], post[ 1][0])
        self.Puzzle02Label.move(post[ 2][1], post[ 2][0])
        self.Puzzle03Label.move(post[ 3][1], post[ 3][0])
        self.Puzzle04Label.move(post[ 4][1], post[ 4][0])
        self.Puzzle05Label.move(post[ 5][1], post[ 5][0])
        self.Puzzle06Label.move(post[ 6][1], post[ 6][0])
        self.Puzzle07Label.move(post[ 7][1], post[ 7][0])
        self.Puzzle08Label.move(post[ 8][1], post[ 8][0])
        self.Puzzle09Label.move(post[ 9][1], post[ 9][0])
        self.Puzzle10Label.move(post[10][1], post[10][0])
        self.Puzzle11Label.move(post[11][1], post[11][0])
        self.Puzzle12Label.move(post[12][1], post[12][0])
        self.Puzzle13Label.move(post[13][1], post[13][0])
        self.Puzzle14Label.move(post[14][1], post[14][0])
        self.Puzzle15Label.move(post[15][1], post[15][0])
        self.Puzzle00Label.move(post[ 0][1], post[ 0][0])

        time.sleep(1)

if __name__ == '__main__':
    print("Pilih jenis input: ")
    print("1. Masukkan File")
    print("2. Random")

    print()
    choose = input("Pilihan : ")
    print()

    if (choose == "1"):
        file = input("Masukkan nama file: ")
        with open ('../test/' + file, 'r') as f:
            puz = []
            for line in f.readlines():
                puz.append([int(x) for x in line.split(' ')])
            puzzle = np.array(puz)
    elif (choose == "2"):
        puz = [i for i in range(0,16)]
        random.shuffle(puz)
        puzzle = np.array(puz, dtype='int8').reshape(4,4)
    else:
        sys.exit("Masukkan tidak valid")

    printPuzzle(puzzle)

    start_time = time.time()
    if (isSolvable(puzzle)):
        solution = solve(puzzle)
        runtime = time.time() - start_time
        printPath(solution, 0)
        print()
        app = QApplication(sys.argv)
        app.setStyleSheet("QLabel{font-size: 25pt;}")
        w = Window(solution)
        w.show()
        print("Waktu eksekusi program = " + str(runtime) + " detik")
        sys.exit(app.exec())
    else:
        runtime = time.time() - start_time
        print("Waktu eksekusi program = " + str(runtime) + " detik")
```

III. *Screenshot* dari *Input-Output* Program

1. TC1.txt

Input:

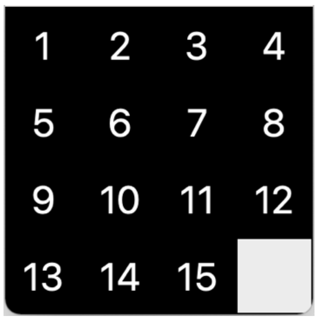
```
10 5 2 4

0 1 3 8

6 14 7 12

9 13 11 15
```

Output:

```
Pilih jenis input:
1. Masukkan File
2. Random

Pilihan : 1

Masukkan nama file: TC1.txt

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

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

X = 1

Sum(KURANG(i)) + X = 38

Puzzle dapat diselesaikan!

Total simpul yang dibangkitkan = 780

Jumlah langkah = 17

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Waktu eksekusi program = 0.18842601776123047 detik
```



Gambar 3.1. GUI TC1.txt

2. **TC2.txt**

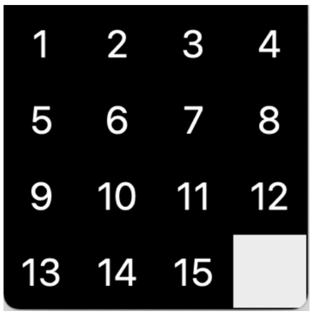
Input:

```
1 6 2 4

5 0 3 8

9 7 15 11

13 14 10 12
```

Output:

```
Pilih jenis input:
1. Masukkan File
2. Random

Pilihan : 1

Masukkan nama file: TC2.txt

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

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

X = 0

Sum(KURANG(i)) + X = 28

Puzzle dapat diselesaikan!

Total simpul yang dibangkitkan = 5921

Jumlah langkah = 18

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Waktu eksekusi program = 2.4007680416107178 detik
```



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

Gambar 3.2. GUI TC2.txt

3. TC3.txt

Input:

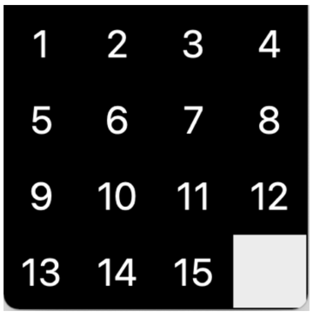
```
2 3 4 11

1 5 10 8

9 6 12 15

13 14 0 7
```

Output:

```
Pilih jenis input:
1. Masukkan File
2. Random

Pilihan : 1

Masukkan nama file: TC3.txt

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

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

X = 1

Sum(KURANG(i)) + X = 26

Puzzle dapat diselesaikan!

Total simpul yang dibangkitkan = 5997

Jumlah langkah = 21

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Waktu eksekusi program = 2.5067808628082275 detik
```



Gambar 3.3. GUI TC3.txt

4. TC4.txt

Input:

```
13 1 8 10

12 9 4 7

6 15 3 5

14 0 2 11
```

Output:

```
Pilih jenis input:
1. Masukkan File
2. Random

Pilihan : 1

Masukkan nama file: TC4.txt

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

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

X = 0

Sum(KURANG(i)) + X = 59

Puzzle tidak dapat diselesaikan!

Waktu eksekusi program = 0.0003020763397216797 detik
```

5. **TC5.txt**

Input:

```
13 6 11 2

4 15 1 3

0 12 7 8

5 9 14 10
```

Output:

```
Pilih jenis input:
1. Masukkan File
2. Random

Pilihan : 1

Masukkan nama file: TC5.txt

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

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

X = 0

Sum(KURANG(i)) + X = 53

Puzzle tidak dapat diselesaikan!

Waktu eksekusi program = 0.0002760887145996094 detik
```

IV. *Checklist*

Poin	Ya	Tidak
1. Program berhasil dikompilasi	✓	
2. Program berhasil *running*	✓	
3. Luaran sudah benar untuk semua data uji	✓	
4. Bonus dibuat	✓	