PENYELESAIAN PERSOALAN 15-PUZZLE DENGAN ALGORITMA *BRANCH AND BOUND*

LAPORAN TUGAS KECIL 3

Diajukan untuk memenuhi Tugas Kecil 3

IF2211 Strategi Algoritma

Oleh

Damianus Clairvoyance Diva Putra 13520035

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2022

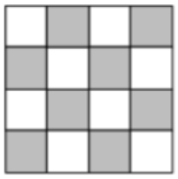
BAB I
ALGORITMA

15-Puzzle merupakan permainan papan berukuran 4x4 yang terdiri atas 15 ubin angka (berisi angka 1-15) dan 1 ubin kosong. Posisi ubin akan diacak dan pemain harus menggeser ubin sedemikian sehingga pemain berhasil mengembalikan seluruh ubin ke posisi terpecahkan (*solved*), yakni sebagai berikut.

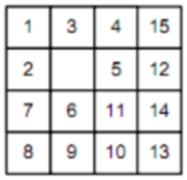
```
 1  2  3  4
 5  6  7  8
9 10 11 12
13 14 15 X
```

Algoritma Branch and Bound adalah algoritma yang menjabarkan kemungkinan *state* dari persoalan, apabila diketahui *state* tujuan dari persoalan tersebut, dengan memilih suatu fungsi pembatas. Fungsi pembatas yang lebih baik akan menyelesaikan persoalan dengan lebih efisien. Algoritma akan terus berlanjut dengan memilih kemungkinan *state* berongkos (*cost*) terkecil, dengan harapan pilihan tersebut akan mendekatkan persoalan pada *state* tujuan. Dalam kasus 15-Puzzle, state tujuannya ialah posisi terpecahkan.

Berdasarkan teorema, puzzle dapat diselesaikan (status tujuan dapat dicapai dari status awal) apabila $\sum_{i=1}^{16} Kurang(i) + X$ bernilai genap. $Kurang(i)$ didefinisikan sebagai banyaknya ubin bernomor j sedemikian sehingga $j < i$ dan $Posisi(j) > Posisi(i)$ $Posisi(i)$ didefinisikan sebagai posisi ubin bernomor i pada susunan yang diperiksa. X didefinisikan bernilai 1 apabila sel kosong pada status awal berada pada posisi sel yang diarsir berikut, sedangkan bernilai 0 apabila terjadi sebaliknya.



Gambar I.1. Ketentuan Nilai X



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

Gambar I.2. Contoh Kasus

Sebagai contoh, perhatikan kasus pada gambar I.2. $Kurang(4) = 1$, karena terdapat ubin 2 ($2 < 4$) dengan $Posisi(2) > Posisi(4)$. $X = 0$, karena ubin kosong terdapat pada sel yang tidak diarsir. Setelah diperiksa, diperoleh tabel berikut.

$\boldsymbol{i}$	$\boldsymbol{Kurang(i)}$	$\boldsymbol{i}$	$\boldsymbol{Kurang(i)}$	$\boldsymbol{i}$	$\boldsymbol{Kurang(i)}$	$\boldsymbol{i}$	$\boldsymbol{Kurang(i)}$
1	0	5	0	9	0	13	0
2	0	6	0	10	0	14	4
3	1	7	1	11	3	15	11
4	1	8	0	12	6	16	10

Tabel I.1. Nilai $\boldsymbol{i}$ dan $\boldsymbol{Kurang(i)}$ pada Contoh Kasus

Dari tabel tersebut, diperoleh nilai $\sum_{i=1}^{16} Kurang(i) + X = 37 + 0 = 37$, yang bernilai ganjil. Oleh karena itu, puzzle tidak dapat diselesaikan (status tujuan tidak dapat dicapai dari status awal).

Secara umum, algoritma Branch and Bound dalam program ini mengimplementasikan algoritma dalam Slide Bahan Kuliah IF2211 Strategi Algoritma – Algoritma Branch and Bound (Bagian 1). Berikut merupakan langkah penyelesaian 15-Puzzle tersebut.

1. Tentukan apakah puzzle dapat diselesaikan atau tidak dengan menghitung $\sum_{i=1}^{16} Kurang(i) + X$. Tampilkan hasil. Apabila bernilai genap, lanjutkan program. Apabila bernilai ganjil, hentikan program karena puzzle tidak dapat diselesaikan.
2. Apabila state saat ini merupakan status tujuan, program sukses, hentikan program. Apabila tidak, tentukan *state* apa saja yang dapat dicapai dari *state* saat ini, yakni ubin apa saja di sekitar ubin kosong yang dapat digeser.
3. Hitung ongkos taksiran, yang didefinisikan sebagai berikut.
   $$\hat{c}(i) = \hat{f}(i) + \hat{g}(i)$$
   atau, pada simpul saat ini, misalnya simpul P,
   $$\hat{c}(P) = \hat{f}(P) + \hat{g}(P),$$
   dengan $\hat{f}(P)$ adalah panjang lintasan dari simpul akar ke simpul P dan $\hat{g}(P)$ adalah taksiran panjang lintasan terpendek dari simpul P ke simpul solusi pada upapohon yang akarnya P. $\hat{g}(P)$ dinyatakan sebagai jumlah ubin tidak kosong yang tidak terdapat pada susunan akhir.
4. Letakkan setiap *state* dalam *priority queue* ”Simpul Hidup”, terurut membesar berdasarkan ongkos.
5. Ambil *state* pada priority queue ”Simpul Hidup” dengan ongkos terkecil, lalu ulangi langkah 2 sampai 4.

BAB II
SOURCE CODE PROGRAM

Tugas kecil ini diselesaikan dengan Bahasa Python. Kakas (*library*) yang dimanfaatkan ialah `time`, `random`, `itertools` dan `os` (bawaan Python), serta `tkinter`. Penggunaan kakas tersebut dilakukan setelah mencantumkan kode import pada bagian atas program.

Perancangan program ini dilakukan dalam *environment* conda dengan Python 3.9.7. Apabila pengguna hendak menjalankan program, kakas tambahan di atas, yakni `tkinter`, harus terinstalasi terlebih dahulu dalam *environment*, dengan menjalankan kode berikut.

```
conda install -c conda-forge tk
```

Program dirancang dengan paradigma pemrograman berorientasi objek, dengan 5 kelas, yaitu `InputFile`, `LiveNodeQueue`, `Node`, `Puzzle`, dan `Game`. Atribut tiap kelas dijabarkan secara implisit dalam konstruktor kelas, sedangkan tiap method dijelaskan dalam bentuk komentar pada kode program.

A. Kelas `InputFile` (dalam fail inputfile.py)

```python
# File: inputfile.py
# Description: InputFile class, containing handler for text file
# Credits: Damianus Clairvoyance DP (13520035)

class InputFile:
    # default constructor
    def __init__(self):
        pass

    # read puzzle from text file
    def readTextFile(self, fileName):
        textFile = open(fileName, "r")
        arrayLine = []
        arrayNumber = []
        for line in textFile:
            numbersInLine = line.strip("\n").split(" ")
            arrayLine.append(numbersInLine)
        textFile.close()
        for i in range (4):
            for j in range (4):
                if (arrayLine[i][j] == "X"):
                    arrayNumber.append(16)
                else:
                    arrayNumber.append(int(arrayLine[i][j]))
        return arrayNumber

    # destructor
    # not implemented, used garbage collection
```

B. Kelas `LiveNodeQueue` (dalam fail livenodequeue.py)

```
# File: livenodequeue.py
# Description: LiveNodeQueue class, containing priority queue,
#                used to store live nodes in order
# Credits: Damianus Clairvoyance DP (13520035)

class LiveNodeQueue:
    # default constructor
    def __init__(self):
        self.queue = []

    # check if priority queue is empty
    def isEmpty(self):
        return (len(self.queue) == 0)

    # insert node in increasing cost order
    def enqueue(self, node):
        index = 0
        if (not self.isEmpty()):
            while (index < len(self.queue) and self.queue[index].cost < node.cost):
                index += 1
        self.queue.insert(index, node)

    # delete node with least cost (at first position)
    def dequeue(self):
        if (not self.isEmpty()):
            node = self.queue[0]
            self.queue.pop(0)
            return node
        return None
```

C. Kelas `Node` (dalam fail node.py)

```
# File: node.py
# Description: Node class, containing id, puzzle, path from root, cost, and last move
# Credits: Damianus Clairvoyance DP (13520035)

import itertools

class Node:
    # initialize iterator for id
    id_iterator = itertools.count()

    # user-defined constructor
    def __init__(self, puzzle, arrayPath, lastMove):
        self.id = next(self.id_iterator)
        self.puzzle = puzzle
        self.path = []
        for index in range (len(arrayPath)):
            self.path.append(arrayPath[index])
        self.cost = self.countUpperCost() + self.countLowerCost()
        self.lastMove = lastMove

    # copy constructor
    def insertNode(self, inputNode):
        self.id = inputNode.id
        self.puzzle.insertPuzzle(inputNode.puzzle.matrix)
        self.path = inputNode.path
        self.cost = inputNode.cost
        self.lastMove = inputNode.lastMove

    # get path of node
    def getPath(self):
        return self.path
    # count number of moves from root to node
    # (f(P) in lecture notes)
    def countUpperCost(self):
        return len(self.path)

    # count number of numbers placed not in their respective cell
    # (g(P) in lecture notes)
    def countLowerCost(self):
        return (15-self.puzzle.countCorrectPosition())

    # returns true if next move is opposite of last move,
    # preventing infinite loop
    def isOpposite(self, command):
        return (
            (command == "up" and self.lastMove == "down") or
            (command == "right" and self.lastMove == "left") or
            (command == "down" and self.lastMove == "up") or
            (command == "left" and self.lastMove == "right")
        )

    # destructor
    # not implemented, used garbage collection
```

D. Kelas `Puzzle` (dalam fail puzzle.py)

```
# File: puzzle.py
# Description: Puzzle class, containing puzzle matrix and index of empty cell
# Credits: Damianus Clairvoyance DP (13520035)

import random

class Puzzle:
    # default constructor
    def __init__(self):
        numbers = list(range(1, 17))
        random.shuffle(numbers)
        self.matrix = numbers
        self.emptyCell = self.findEmptyCell()

    # find a solveable puzzle
    def findSolvable(self):
        numbers = list(range(1, 17))
        while (not self.isSolveable()):
            random.shuffle(numbers)
            self.matrix = numbers
            self.emptyCell = self.findEmptyCell()

    # user-defined constructor
    def insertPuzzle(self, inputMatrix):
        for i in range (4):
            for j in range (4):
                self.matrix[i*4+j] = inputMatrix[i*4+j]
        self.emptyCell = self.findEmptyCell()

    # find index of empty cell
    def findEmptyCell(self):
        index = 0
        while (self.matrix[index] != 16):
            index += 1
        return index

    # convert index of empty cell to (row, col)
    def convertEmptyCell(self):
        return (self.emptyCell // 4, self.emptyCell % 4)

    # check if puzzle is valid (unique combinations of number 1-16)
    def isValid(self):
        found = set()
        for number in self.matrix:
            if (number in found or number > 16 or number < 1):
                return False
            else:
                found.add(number)
        return True
```

```python
# count solveable factor
# (S(Kurang(i)) + X in lecture notes)
def solveableFactor(self):
    counter = 0
    for index in range (1, 17):
        counter += self.countOff(index)
    return counter + self.shadedX()

# check if puzzle is solvable (S(countOff(i)) + shadedX is even)
def isSolveable(self):
    return (self.solveableFactor() % 2 == 0)

# count number of numbers placed in their respective cell
def countCorrectPosition(self):
    counter = 0
    for index in range (16):
        if (self.matrix[index] == index + 1 and self.matrix[index] != 16):
            counter += 1
    return counter

# check if puzzle is solved
def isSolved(self):
    return (self.countCorrectPosition() == 15)

# find index of a number
def findIndex(self, i):
    index = 0
    while (index < 16 and self.matrix[index] != i):
        index += 1
    return index + 1

# count number of off placed cells
# (Kurang(i) function in lecture notes)
def countOff(self, i):
    indexI = self.findIndex(i)
    counter = 0
```

```
        for index in range (indexI, 16):
            if (self.matrix[index] < i):
                counter += 1
        return counter

    # print countOff (Kurang(i) in lecture notes) for all index
    def printOff(self):
        for index in range (16):
            print("Kurang({0:2d}) = ".format(index + 1), end = "")
            print(self.countOff(index+1))
        print()

    # find shaded X parameter
    # (X function in lecture notes)
    def shadedX(self):
        i, j = self.convertEmptyCell()
        return ((i + j) % 2 == 1)

    # check if move is valid (tile movable)
    def isMoveValid(self, command):
        i, j = self.convertEmptyCell()
        return (
            (command == "up" and i != 0) or
            (command == "right" and j != 3) or
            (command == "down" and i != 3) or
            (command == "left" and j != 0)
        )

    # switch number of a cell and empty cell
    def switchCell(self, cell, emptyCell):
        self.matrix[emptyCell] = self.matrix[cell]
        self.matrix[cell] = 16
        self.emptyCell = cell

    # move cell based on command
    def move(self, command):
        newPuzzle = Puzzle()
        newPuzzle.insertPuzzle(self.matrix)
        x = self.emptyCell
        if (command == "up"):
            newPuzzle.switchCell(x-4, x)
        elif (command == "right"):
            newPuzzle.switchCell(x+1, x)
        elif (command == "down"):
            newPuzzle.switchCell(x+4, x)
        elif (command == "left"):
            newPuzzle.switchCell(x-1, x)
        return newPuzzle

    # print puzzle
    def print(self):
        for i in range (4):
            print("---------------------")
            for j in range (4):
                if (self.matrix[i*4+j] == 16):
                    print("|    ", end = "")
                else:
                    print("|", end = "")
                    print(" {0:2d} ".format(self.matrix[i*4+j]), end = "")
            print("|")
        print("---------------------")
        print()

    # destructor
    # not implemented, used garbage collection
```

E. Program utama (dalam fail app.py)

```
# File: app.py
# Description: main program
# Credits: Damianus Clairvoyance DP (13520035)

# import libraries and classes
import time
import os.path
```

```python
from puzzle import *
from node import *
from livenodequeue import *
from inputfile import *
from gui import *

def welcomeMessage():
    # show welcome message
    print()
    print("=========================")
    print("       Welcome to        ")
    print("    15 Puzzle Solver     ")
    print("=========================")
    print("        Credits:         ")
    print(" Damianus Clairvoyance DP ")
    print("        13520035         ")
    print("=========================")
    print()

def loadSuccessful():
    # show load successful message
    print()
    print("=========================")
    print("Puzzle Loaded Successfully")
    print("=========================")
    print()

def main():
    # main program
    welcomeMessage()

    # initialize puzzle
    puzzle = Puzzle()

    # handle input puzzle source
    print("Input method:")
    print("[1] Computer Generated")
print("[2] Text File")
print("[3] Manual")
inputMethod = int(input("Pick method (1/2/3): "))
while (inputMethod > 3 or inputMethod < 0):
    print("Invalid!")
    inputMethod = int(input("Pick method (1/2/3): "))

# computer generated, ensures valid and solvable puzzle
if (inputMethod == 1):
    puzzle.findSolvable()
# text file
elif (inputMethod == 2):
    inputFile = InputFile()

    # input and validate file name
    fileName = input("Type file name: ")
    while (not os.path.isfile(fileName)):
        print("Invalid!")
        fileName = input("Type file name: ")

    # read text file line by line and put into puzzle
    arrayNumber = inputFile.readTextFile(fileName)
    puzzle.insertPuzzle(arrayNumber)

    # abort program if puzzle is invalid
    if (not puzzle.isValid()):
        print("Puzzle is invalid. Exiting.")
        exit()
# manual
else: # inputMethod == 3
    print("Insert your puzzle:")
    arrayNumber = []
    # read text file line by line and put into puzzle
    for i in range (4):
        arrayLine = input()
        numbersInLine = arrayLine.strip("\n").split(" ")
        for j in range (4):
            if (numbersInLine[j] == "X"):
                arrayNumber.append(16)
            else:
                arrayNumber.append(int(numbersInLine[j]))
    puzzle.insertPuzzle(arrayNumber)

    # abort program if puzzle is invalid
    if (not puzzle.isValid()):
        print("Puzzle is invalid. Exiting.")
        exit()

# show puzzle successfully loaded
loadSuccessful()
puzzle.print()

# print Kurang(i) and Sigma(Kurang(i)) + X
print("Kurang(i) Function")
puzzle.printOff()
print("Sigma(Kurang(i)) + X Function = ", end = "")
print(puzzle.solveableFactor())
print()

# handles puzzle based on puzzle solveability
if (not puzzle.isSolveable()):
    print("Puzzle is not solveable. Exiting.")
    exit()
else:
    print("Puzzle is solveable.")
    print()

    # initialize everything
    commands = ["up", "left", "down", "right"] # order matters, but not significantly
    arrayPath = []
    countNode = 0
    queueLiveNode = LiveNodeQueue()

    # initialize puzzle for printing purpose
```

```
printPuzzle = Puzzle()
printPuzzle.insertPuzzle(puzzle.matrix)

# start of branch and bound algorithm
# algorithm is explained further in report (see docs directory)
startTimer = time.process_time_ns()
expandNode = Node(puzzle, arrayPath, "")
newNode = Node(puzzle, arrayPath, "")
countNode += 1
queueLiveNode.enqueue(newNode)
expandNode.insertNode(queueLiveNode.dequeue())

while (not expandNode.puzzle.isSolved()):
    for command in commands:
        if (expandNode.puzzle.isMoveValid(command) and not expandNode.isOpposite(command)):
            newPuzzle = expandNode.puzzle.move(command)
            arrayPath.append(command)
            newNode = Node(newPuzzle, arrayPath, command)
            countNode += 1
            queueLiveNode.enqueue(newNode)
            arrayPath.pop()
    expandNode.insertNode(queueLiveNode.dequeue())
    arrayPath = expandNode.getPath()

# end of branch and bound algorithm
stopTimer = time.process_time_ns()

# print puzzle from initial to solved position
print("initial:")
printPuzzle.print()
for i in range (len(arrayPath)):
    print(str(arrayPath[i]) + ":")
    printPuzzle = printPuzzle.move(arrayPath[i])
    printPuzzle.print()

# print additional informations
print(countNode - 1, "nodes generated")
```

```
        print((stopTimer-startTimer)/1000000, "ms taken")

    return arrayNumber, arrayPath

if __name__ == "__main__":
    # run CLI program
    arrayNumber, arrayPath = main()

    # run GUI program
    Game(arrayNumber, arrayPath)
```

F. (Bonus) Kelas `Game` (dalam fail gui.py)

```
# File: gui.py
# Description: Game class, containing all functions and scriptings for GUI
# Credits: Damianus Clairvoyance DP (13520035)

import tkinter as tk
import configurations
from puzzle import *

class Game(tk.Frame):
    # default constructor (and main program for GUI)
    def __init__(self, arrayPuzzle, arrayPath):
        # initial scriptings
        tk.Frame.__init__(self)
        self.grid()
        self.master.title("15 Puzzle Solver")
        self.master.resizable(False, False)

        # initial functions
        self.puzzle = Puzzle()
        self.puzzle.insertPuzzle(arrayPuzzle)
        arrayPath.insert(0, "none")
        arrayPath.append("none")
        self.path = arrayPath
        self.index = 1
        self.ongoing = "stop"
```

```python
    # scripting: initialize puzzle board
    self.mainGrid = tk.Frame(
        self,
        bg = configurations.GRID_COLOR,
        bd = 3,
        width = 600,
        height = 600
    )
    # scripting: add spaces above and below puzzle board
    self.mainGrid.grid(pady = (100, 90))

    # put GUI window at the middle of the screen
    windowWidth = 445
    windowHeight = 640
    screenWidth = self.master.winfo_screenwidth()
    screenHeight = self.master.winfo_screenheight()
    x = int((screenWidth/2) - (windowWidth/2))
    y = int((screenHeight/2) - (windowHeight/2))
    self.master.geometry(f"{windowWidth}x{windowHeight}+{x}+{y}")

    self.createGUI()
    self.startGame()

    # add functionalities with keyboard arrow
    self.master.bind("<Left>", self.left)
    self.master.bind("<Right>", self.right)
    self.master.bind("<Up>", self.up)
    self.master.bind("<Down>", self.down)

    self.mainloop()

# initialize all GUI components
def createGUI(self):
    # scripting: initialize cells in puzzle board
    self.cells = []
    for i in range (4):
```

```
    for j in range (4):
        cell_frame = tk.Frame(
            self.mainGrid,
            bg = configurations.EMPTY_CELL_COLOR,
            width = 100,
            height = 100
        )
        cell_frame.grid(row = i, column = j, padx = 5, pady = 5)

        cell_number = tk.Label(
            self.mainGrid,
            bg = configurations.EMPTY_CELL_COLOR
        )
        cell_number.grid(row = i, column = j)

        cell_data = {"frame": cell_frame, "number": cell_number}
        self.cells.append(cell_data)

# scripting: initialize title frame (above puzzle board)
titleFrame = tk.Frame(self)
titleFrame.place(relx = 0.5, y = 45, anchor = "center")

self.gameTitle = tk.Label(
    titleFrame,
    text = "15 Puzzle Solver",
    font = configurations.TITLE_FONT
)
self.gameTitle.grid(row = 0)

self.creditTitle = tk.Label(
    titleFrame,
    text = "Damianus Clairvoyance DP (13520035)",
    font = configurations.CREDIT_FONT,
)
self.creditTitle.grid(row = 1)

# scripting: initialize move frame (below puzzle board)
```

```
moveFrame = tk.Frame(self)
moveFrame.place(relx = 0.5, y = 590, anchor = "center")

self.lastMoveTitle = tk.Label(
    moveFrame,
    text = "Last Move",
    font = configurations.MOVE_TITLE_FONT
)
self.lastMoveTitle.grid(row = 0, column = 0, padx = 20)

self.nextMoveTitle = tk.Label(
    moveFrame,
    text = "Next Move",
    font = configurations.MOVE_TITLE_FONT
)
self.nextMoveTitle.grid(row = 0, column = 1, padx = 20)

self.lastMove = tk.Label(
    moveFrame,
    text = self.path[self.index-1],
    font = configurations.MOVE_FONT
)
self.lastMove.grid(row = 1, column = 0, padx = 20)

self.nextMove= tk.Label(
    moveFrame,
    text = self.path[self.index],
    font = configurations.MOVE_FONT
)
self.nextMove.grid(row = 1, column = 1, padx = 20)

self.autoTitle = tk.Label(
    moveFrame,
    text = "Auto Move",
    font = configurations.MOVE_TITLE_FONT
)
self.autoTitle.grid(row = 0, column = 2, padx = 20)
```

```
        self.autoDirection = tk.Label(
            moveFrame,
            text = "stop",
            font = configurations.MOVE_FONT
        )
        self.autoDirection.grid(row = 1, column = 2, padx = 20)

    # scripting: intialize cells of puzzle board
    def startGame(self):
        for index in range (16):
            if (self.puzzle.matrix[index] != 16):
                self.cells[index]["frame"].configure(bg = configurations.CELL_COLOR)
                self.cells[index]["number"].configure(
                    bg = configurations.CELL_COLOR,
                    fg = configurations.CELL_NUMBER_COLOR,
                    font = configurations.CELL_NUMBER_FONT,
                    text = str(self.puzzle.matrix[index])
                )

    # switch two adjacent cells of puzzle board
    def switchCell(self, cell, emptyCell):
        self.puzzle.matrix[emptyCell] = self.puzzle.matrix[cell]
        self.puzzle.matrix[cell] = 16
        self.emptyCell = cell
        self.cells[emptyCell]["frame"].configure(bg = configurations.CELL_COLOR)
        self.cells[emptyCell]["number"].configure(
            bg = configurations.CELL_COLOR,
            fg = configurations.CELL_NUMBER_COLOR,
            font = configurations.CELL_NUMBER_FONT,
            text = str(self.puzzle.matrix[emptyCell])
        )
        self.cells[cell]["frame"].configure(bg = configurations.EMPTY_CELL_COLOR)
        self.cells[cell]["number"].configure(
            bg = configurations.EMPTY_CELL_COLOR,
            text = ""
        )
```

```
# update GUI to next move
def updateGUIRight(self, command):
    x = 0
    for index in range (16):
        if (self.puzzle.matrix[index] == 16):
            x = index
    if (command == "up"):
        self.switchCell(x-4, x)
    elif (command == "right"):
        self.switchCell(x+1, x)
    elif (command == "down"):
        self.switchCell(x+4, x)
    elif (command == "left"):
        self.switchCell(x-1, x)
    self.lastMove.configure(
        text = self.path[self.index]
    )
    self.nextMove.configure(
        text = self.path[self.index+1]
    )
    self.update_idletasks()

# update GUI to last move
def updateGUILeft(self, command):
    x = 0
    for index in range (16):
        if (self.puzzle.matrix[index] == 16):
            x = index
    if (command == "up"):
        self.switchCell(x+4, x)
    elif (command == "right"):
        self.switchCell(x-1, x)
    elif (command == "down"):
        self.switchCell(x-4, x)
    elif (command == "left"):
        self.switchCell(x+1, x)
    self.lastMove.configure(
        text = self.path[self.index-1]
    )
    self.nextMove.configure(
        text = self.path[self.index]
    )
    self.update_idletasks()

# update GUI when puzzle hit initial or solved position
def updateGUI(self):
    self.autoDirection.configure(
        text = self.ongoing
    )

# update GUI when left arrow is pushed
def left(self, event):
    if (self.index-1 >= 1 and self.index-1 < (len(self.path) - 1)):
        self.index -= 1
        self.updateGUILeft(self.path[self.index])
        self.updateGUI()

# update GUI when right arrow is pushed
def right(self, event):
    if (self.index >= 1 and self.index < (len(self.path) - 1)):
        self.updateGUIRight(self.path[self.index])
        self.updateGUI()
        self.index += 1

# switch auto move in the direction of last move
def switchOngoingLeft(self):
    if (self.ongoing == "right"):
        self.ongoing = "stop"
    elif (self.ongoing == "stop"):
        self.ongoing = "left"

# switch auto move in the direction of next move
def switchOngoingRight(self):
        if (self.ongoing == "left"):
            self.ongoing = "stop"
        elif (self.ongoing == "stop"):
            self.ongoing = "right"

    # functions for auto move left
    def repeatLeft(self):
        if (self.index == 1):
            self.ongoing = "stop"
            self.updateGUI()
        elif (self.index-1 >= 1 and self.index-1 < (len(self.path) - 1) and self.ongoing == "left"):
            self.index -= 1
            self.updateGUILeft(self.path[self.index])
            self.after(1000, self.repeatLeft)

    # functions for auto move right
    def repeatRight(self):
        if (self.index == (len(self.path) - 1)):
            self.ongoing = "stop"
            self.updateGUI()
        elif (self.index >= 1 and self.index < (len(self.path) - 1) and self.ongoing == "right"):
            self.updateGUIRight(self.path[self.index])
            self.index += 1
            self.after(1000, self.repeatRight)

    # start auto move right when up arrow is pushed
    def up(self, event):
        if (self.ongoing != "right"):
            self.switchOngoingRight()
            self.repeatRight()
            self.updateGUI()

    # start auto move left when down arrow is pushed
    def down(self, event):
        if (self.ongoing != "left"):
            self.switchOngoingLeft()
            self.repeatLeft()
        self.updateGUI()
```

G. *Test Case* 1 (notsolveable1.txt)

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

H. *Test Case* 2 (notsolveable2.txt)

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

I. *Test Case* 3 (solveable1.txt)

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

J. *Test Case* 4 (solveable2.txt)

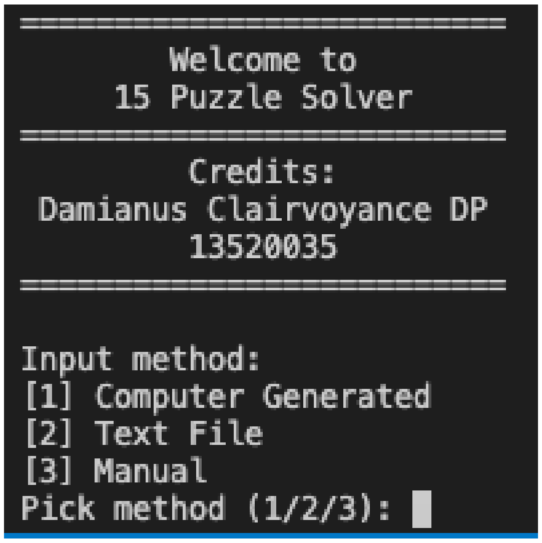
```
1 2 3 4
5 6 7 8
11 12 15 14
10 9 13 X
```

K. *Test Case* 5 (solveable3.txt)

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

BAB III
HASIL PROGRAM DAN *TEST CASE*

Program dijalankan dengan menjalankan file utama app.py. Program akan menyambut dengan salam pembukaan berikut.



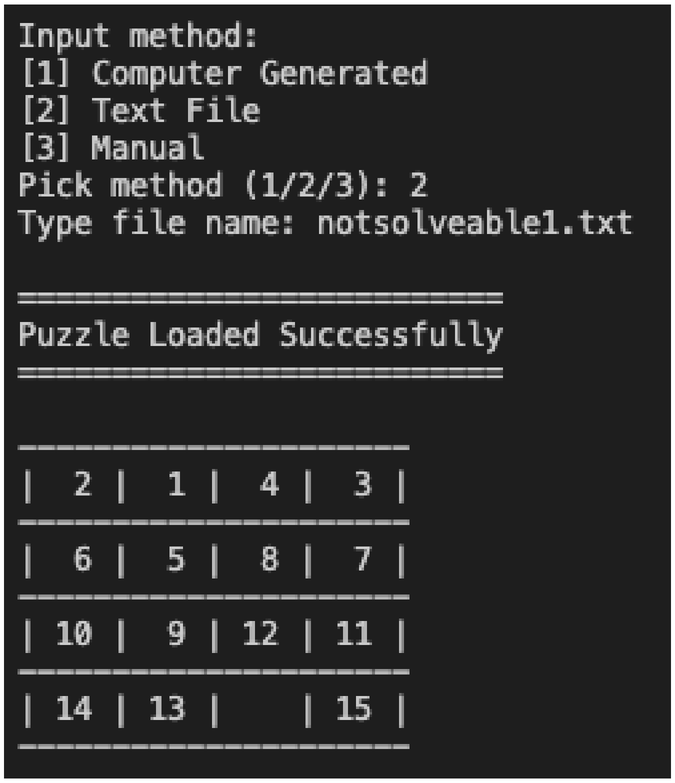
```
==========================
       Welcome to
    15 Puzzle Solver
==========================
        Credits:
 Damianus Clairvoyance DP
        13520035
==========================


Input method:
[1] Computer Generated
[2] Text File
[3] Manual
Pick method (1/2/3): 
```

Gambar III.1. Salam Pembukaan

Sesuai spesifikasi, input puzzle dapat berasal dari input pengguna, baik dalam bentuk file teks maupun tik langsung di *command line* saat *runtime*, serta dari pembangkitan acak oleh program. Input puzzle harus mematuhi contoh format berikut.

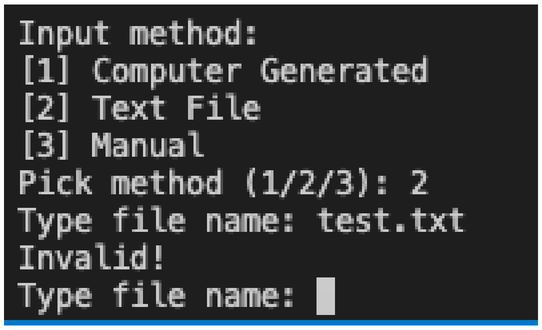
```
 1  2  3  4
 5  6  7  8
 9 10 11 12
13 14 15  X
```

Perhatikan bahwa elemen kosong puzzle diilustrasikan sebagai ‘X’.



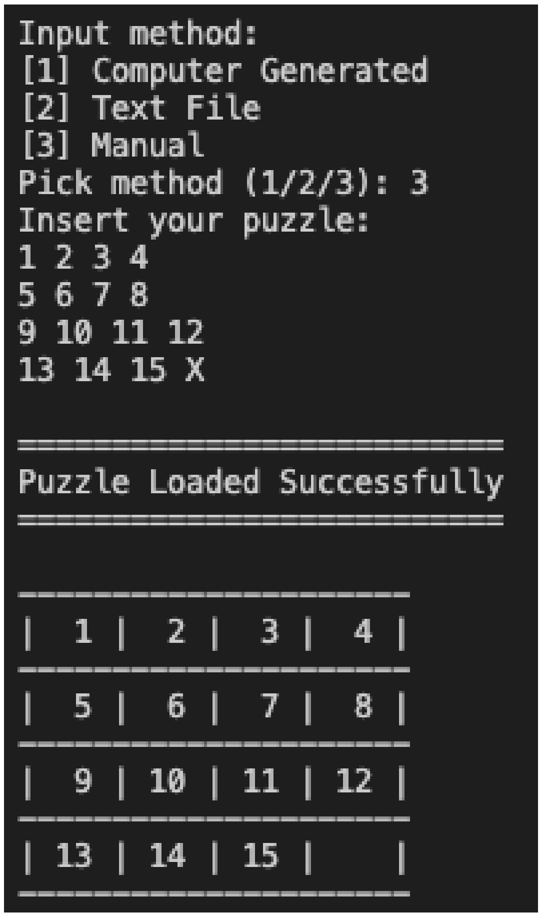
```
Input method:
[1] Computer Generated
[2] Text File
[3] Manual
Pick method (1/2/3): 2
Type file name: notsolveable1.txt

==========================
Puzzle Loaded Successfully
==========================


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

Gambar III.2. Input Fail Teks Berhasil



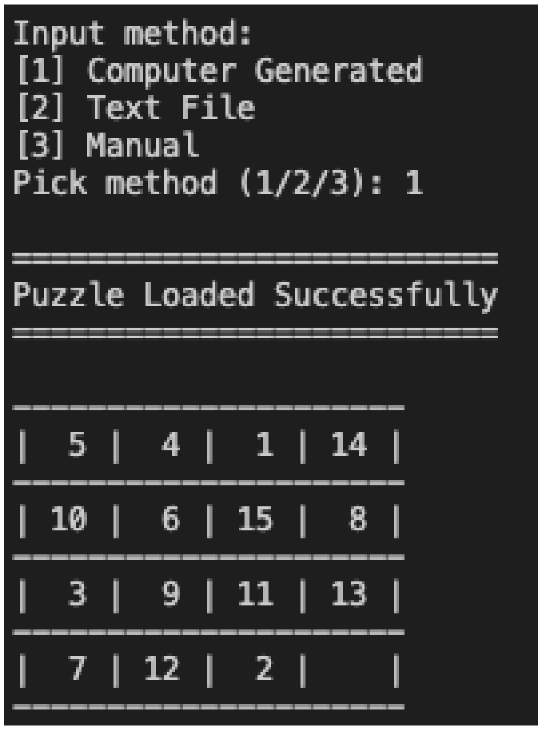
```
Input method:
[1] Computer Generated
[2] Text File
[3] Manual
Pick method (1/2/3): 2
Type file name: test.txt
Invalid!
Type file name: 
```

Gambar III.3. Input Fail Teks Gagal



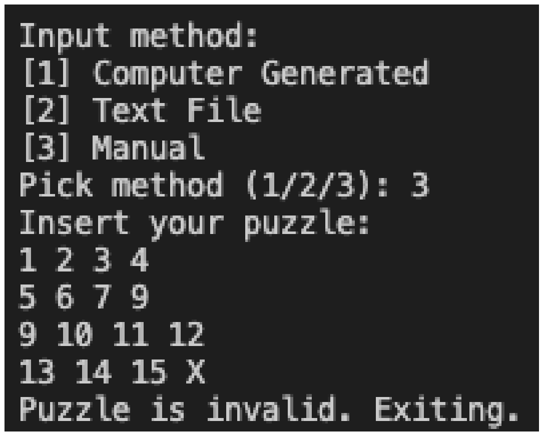
```
Input method:
[1] Computer Generated
[2] Text File
[3] Manual
Pick method (1/2/3): 3
Insert your puzzle:
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 X

==========================
Puzzle Loaded Successfully
==========================


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

Gambar III.4. Input Tik Manual Berhasil



```
Input method:
[1] Computer Generated
[2] Text File
[3] Manual
Pick method (1/2/3): 1

==========================
Puzzle Loaded Successfully
==========================


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

Gambar III.5. Input Random Berhasil



```
Input method:
[1] Computer Generated
[2] Text File
[3] Manual
Pick method (1/2/3): 3
Insert your puzzle:
1 2 3 4
5 6 7 9
9 10 11 12
13 14 15 X
Puzzle is invalid. Exiting.
```

Gambar III.6. Input Puzzle Tidak Valid

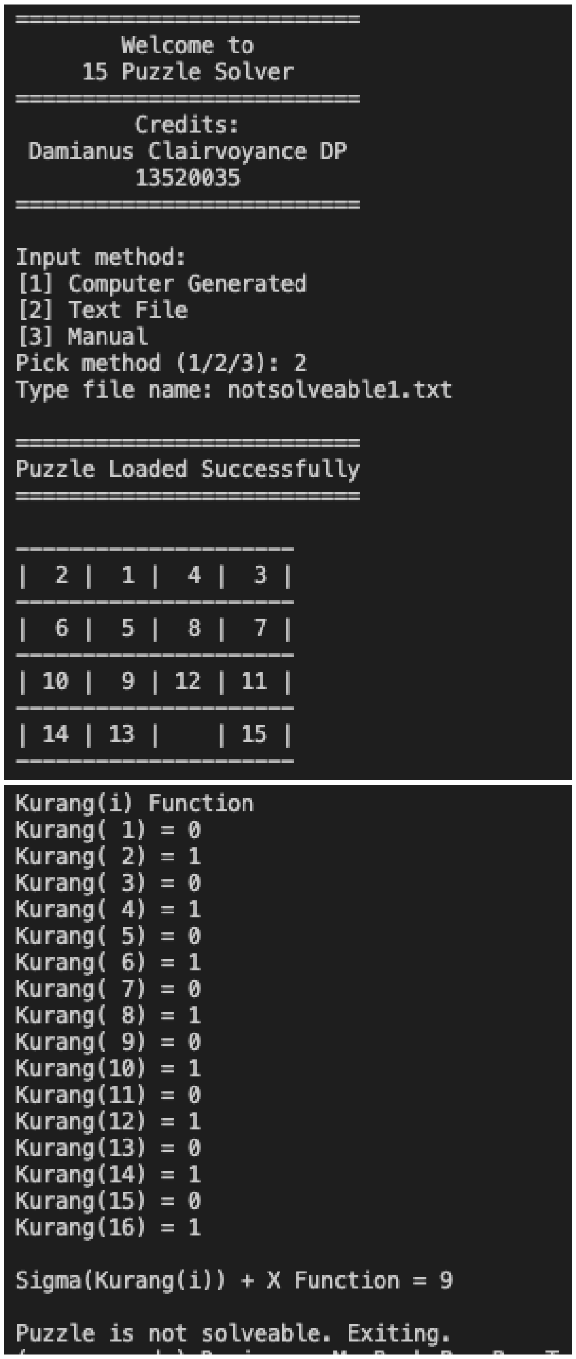
Terdapat penanganan terhadap kesalahan masukan pengguna, yakni tidak tersedianya fail teks dalam direktori dengan nama yang sesuai dan tidak valid-nya *puzzle*. *Puzzle* disebut tidak valid apabila *puzzle* memenuhi salah satu deskripsi berikut.

- Sel kosong *puzzle* tidak diilustrasikan dengan 'X' atau 16.
- Terdapat elemen angka di luar rentang 1-15.
- Terdapat elemen angka duplikat.

Setelah berhasil *import puzzle*, program akan menampilkan nilai dari fungsi $Kurang(i)$ untuk semua nilai i (sesuai ketentuan *slide* bahan kuliah), nilai $\sum_{i=1}^{16} Kurang(i) + X$, dan matriks posisi awal 15-Puzzle. Apabila puzzle tidak dapat diselesaikan, program akan menampilkan pesan dan exit. Sebaliknya, apabila puzzle dapat diselesaikan, program akan menampilkan matriks dari posisi awal ke posisi akhir 15-Puzzle. Program akan menampilkan pula waktu eksekusi program dan banyak simpul yang dibangkitkan dalam pohon ruang status pencarian.

Berikut merupakan tangkapan layar untuk 5 *test case*, dengan rincian 2 *puzzle* yang tidak dapat diselesaikan dan 3 *puzzle* yang dapat diselesaikan. Input puzzle akan menggunakan motode 2 (pembacaan fail teks).

A. *Test Case* 1: notsolveable1.txt

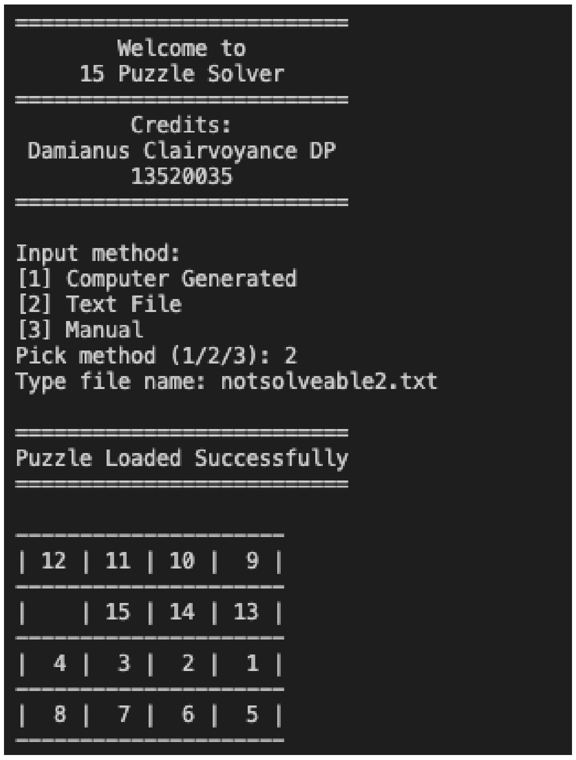


```
==========================
        Welcome to
     15 Puzzle Solver
==========================
         Credits:
 Damianus Clairvoyance DP
         13520035
==========================

Input method:
[1] Computer Generated
[2] Text File
[3] Manual
Pick method (1/2/3): 2
Type file name: notsolveable1.txt

==========================
Puzzle Loaded Successfully
==========================

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

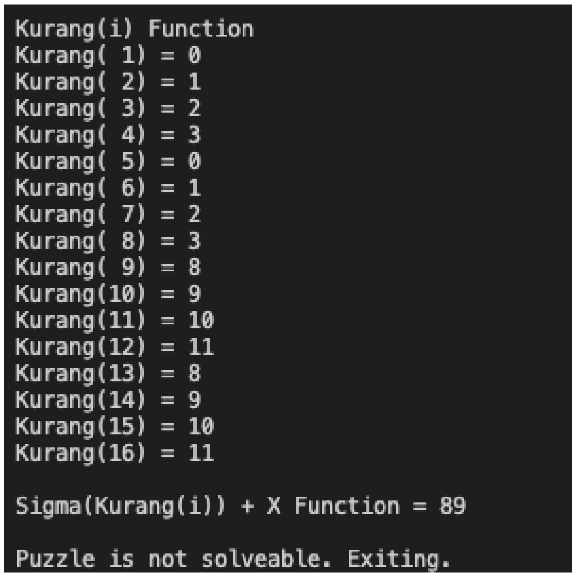
```
Kurang(i) Function
Kurang( 1) = 0
Kurang( 2) = 1
Kurang( 3) = 0
Kurang( 4) = 1
Kurang( 5) = 0
Kurang( 6) = 1
Kurang( 7) = 0
Kurang( 8) = 1
Kurang( 9) = 0
Kurang(10) = 1
Kurang(11) = 0
Kurang(12) = 1
Kurang(13) = 0
Kurang(14) = 1
Kurang(15) = 0
Kurang(16) = 1

Sigma(Kurang(i)) + X Function = 9

Puzzle is not solveable. Exiting.
```

Gambar III.7. Proses *Test Case* 1

B. *Test Case* 2: notsolveable2.txt



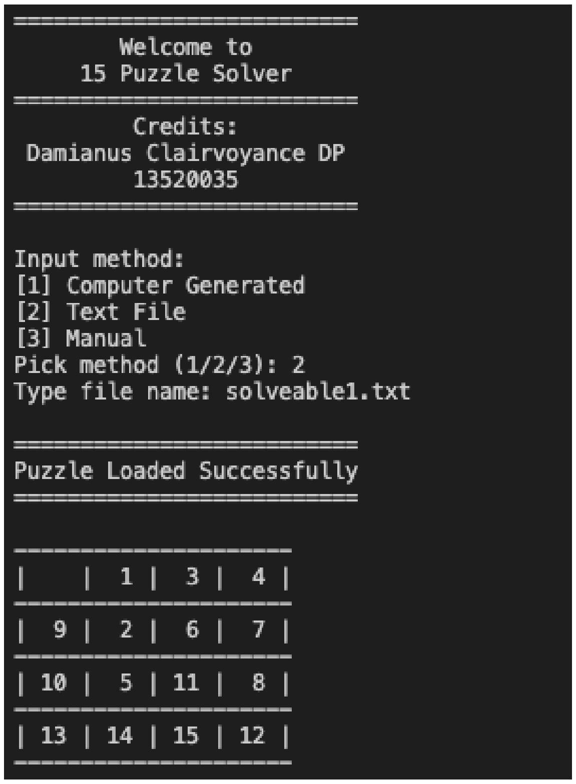
```
===========================
        Welcome to
     15 Puzzle Solver
===========================
         Credits:
 Damianus Clairvoyance DP
         13520035
===========================

Input method:
[1] Computer Generated
[2] Text File
[3] Manual
Pick method (1/2/3): 2
Type file name: notsolveable2.txt

===========================
Puzzle Loaded Successfully
===========================


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



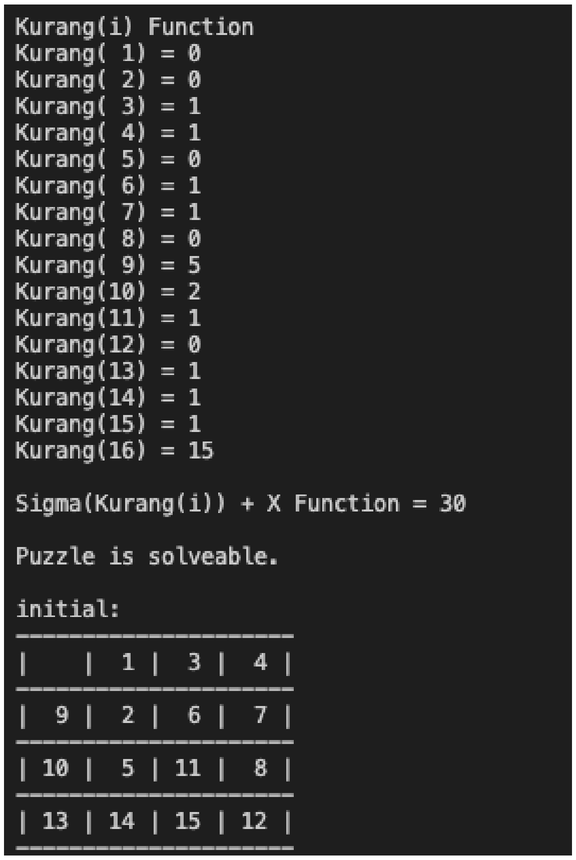
```
Kurang(i) Function
Kurang( 1) = 0
Kurang( 2) = 1
Kurang( 3) = 2
Kurang( 4) = 3
Kurang( 5) = 0
Kurang( 6) = 1
Kurang( 7) = 2
Kurang( 8) = 3
Kurang( 9) = 8
Kurang(10) = 9
Kurang(11) = 10
Kurang(12) = 11
Kurang(13) = 8
Kurang(14) = 9
Kurang(15) = 10
Kurang(16) = 11

Sigma(Kurang(i)) + X Function = 89

Puzzle is not solveable. Exiting.
```

Gambar III.8. Proses *Test Case* 2

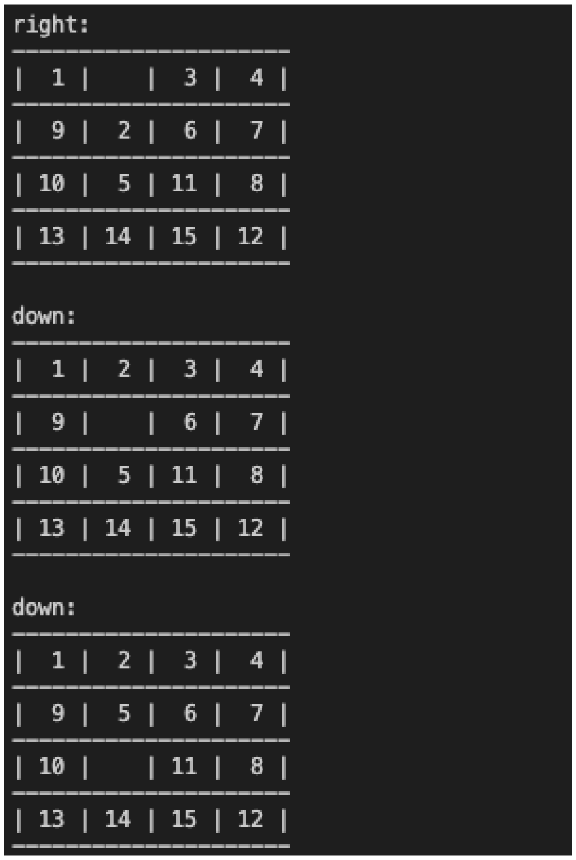
C. *Test Case* 3: solveable1.txt



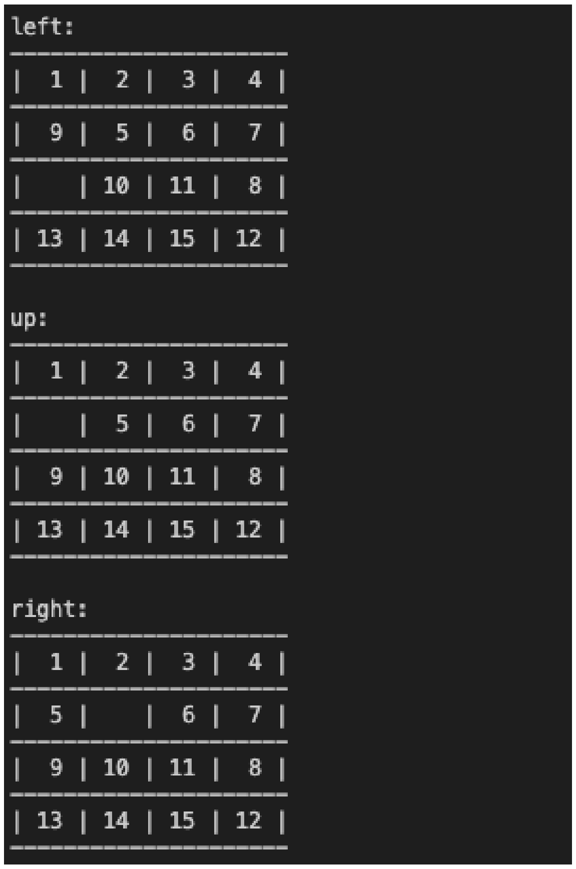
```
==========================
        Welcome to
     15 Puzzle Solver
==========================
         Credits:
 Damianus Clairvoyance DP
         13520035
==========================

Input method:
[1] Computer Generated
[2] Text File
[3] Manual
Pick method (1/2/3): 2
Type file name: solveable1.txt

==========================
Puzzle Loaded Successfully
==========================

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



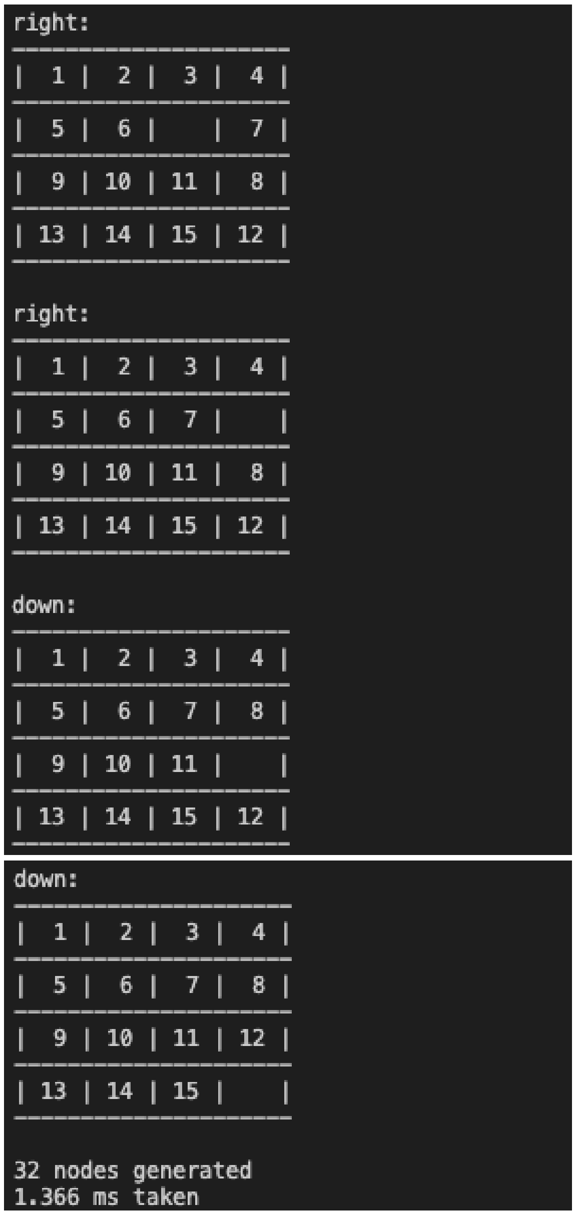
```
Kurang(i) Function
Kurang( 1) = 0
Kurang( 2) = 0
Kurang( 3) = 1
Kurang( 4) = 1
Kurang( 5) = 0
Kurang( 6) = 1
Kurang( 7) = 1
Kurang( 8) = 0
Kurang( 9) = 5
Kurang(10) = 2
Kurang(11) = 1
Kurang(12) = 0
Kurang(13) = 1
Kurang(14) = 1
Kurang(15) = 1
Kurang(16) = 15

Sigma(Kurang(i)) + X Function = 30

Puzzle is solveable.

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



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

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

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



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

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

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

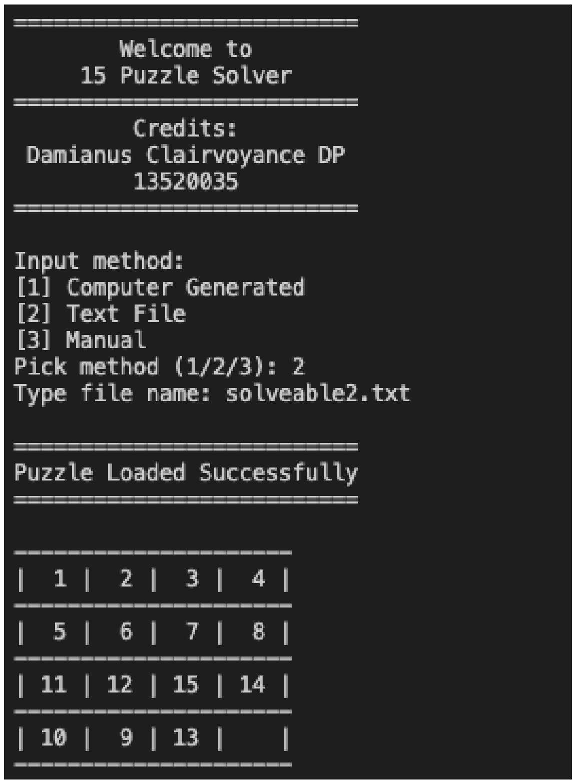


```
right:
---------------------
|  1 |  2 |  3 |  4 |
---------------------
|  5 |  6 |    |  7 |
---------------------
|  9 | 10 | 11 |  8 |
---------------------
| 13 | 14 | 15 | 12 |
---------------------

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

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

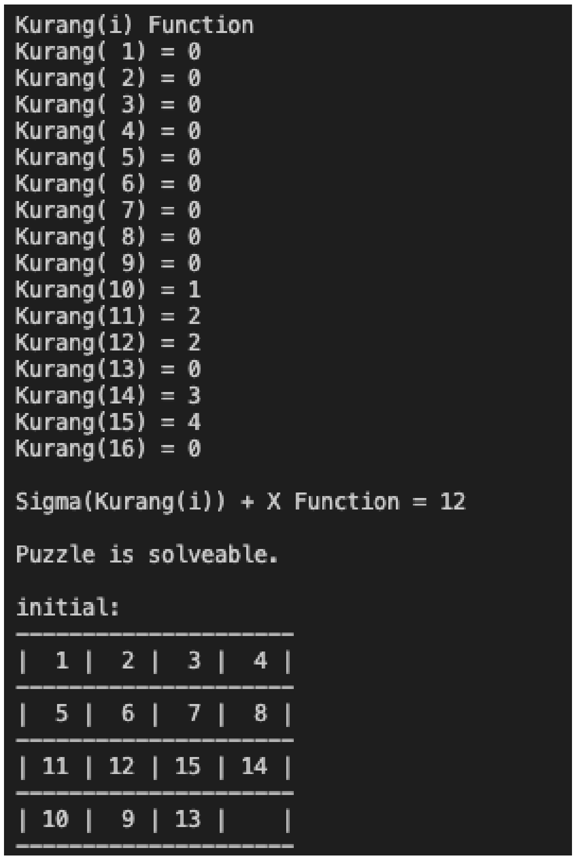
```
down:
---------------------
|  1 |  2 |  3 |  4 |
---------------------
|  5 |  6 |  7 |  8 |
---------------------
|  9 | 10 | 11 | 12 |
---------------------
| 13 | 14 | 15 |    |
---------------------

32 nodes generated
1.366 ms taken
```

Gambar III.9. Proses *Test Case* 3

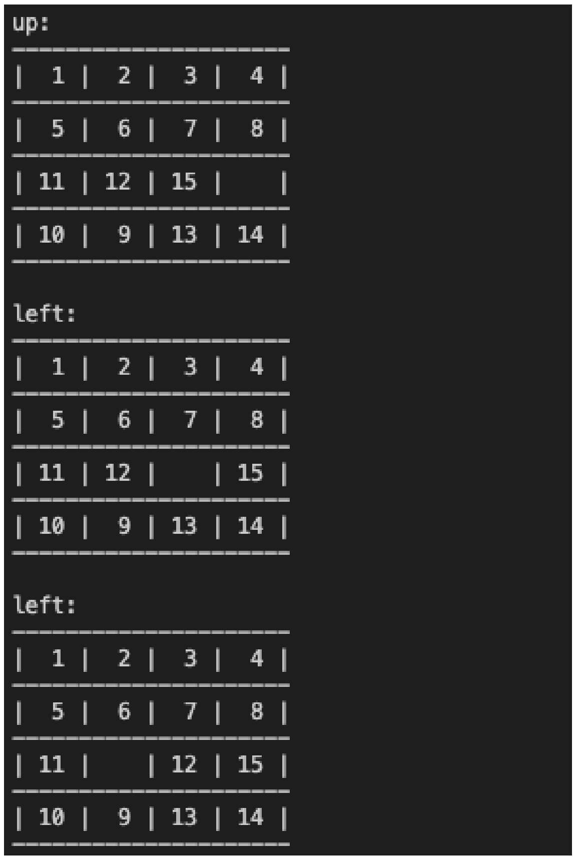
D. *Test Case* 4: solveable2.txt



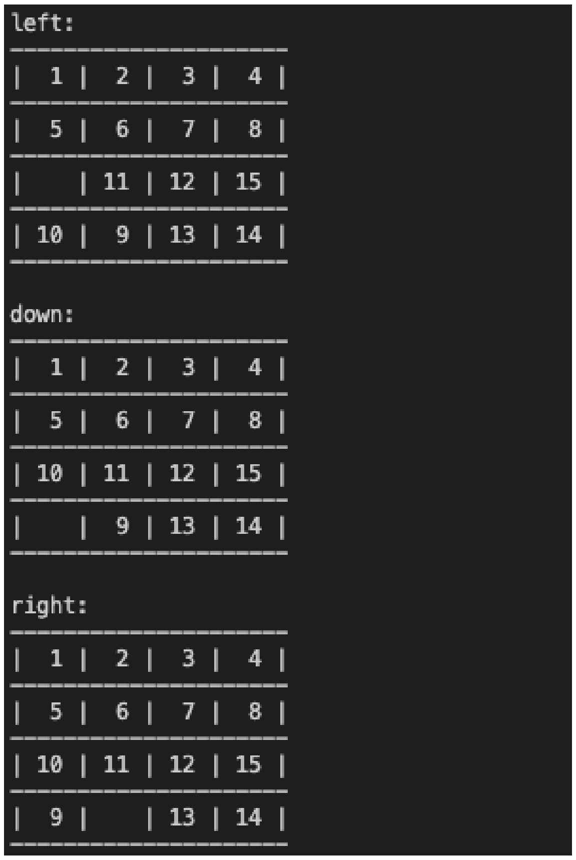
```
==========================
        Welcome to
     15 Puzzle Solver
==========================
         Credits:
 Damianus Clairvoyance DP
         13520035
==========================

Input method:
[1] Computer Generated
[2] Text File
[3] Manual
Pick method (1/2/3): 2
Type file name: solveable2.txt

==========================
Puzzle Loaded Successfully
==========================

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



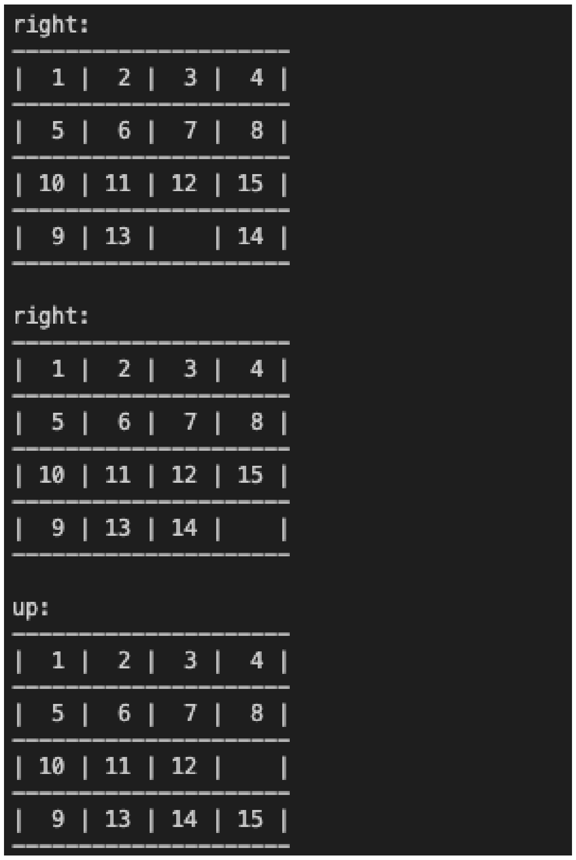
```
Kurang(i) Function
Kurang( 1) = 0
Kurang( 2) = 0
Kurang( 3) = 0
Kurang( 4) = 0
Kurang( 5) = 0
Kurang( 6) = 0
Kurang( 7) = 0
Kurang( 8) = 0
Kurang( 9) = 0
Kurang(10) = 1
Kurang(11) = 2
Kurang(12) = 2
Kurang(13) = 0
Kurang(14) = 3
Kurang(15) = 4
Kurang(16) = 0

Sigma(Kurang(i)) + X Function = 12

Puzzle is solveable.

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



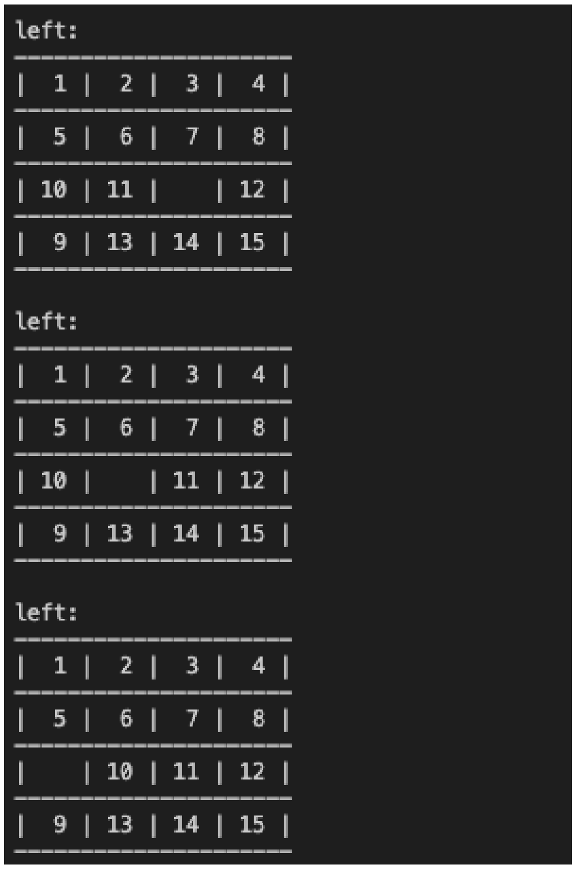
```
up:
---------------------
|  1 |  2 |  3 |  4 |
---------------------
|  5 |  6 |  7 |  8 |
---------------------
| 11 | 12 | 15 |    |
---------------------
| 10 |  9 | 13 | 14 |
---------------------

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

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



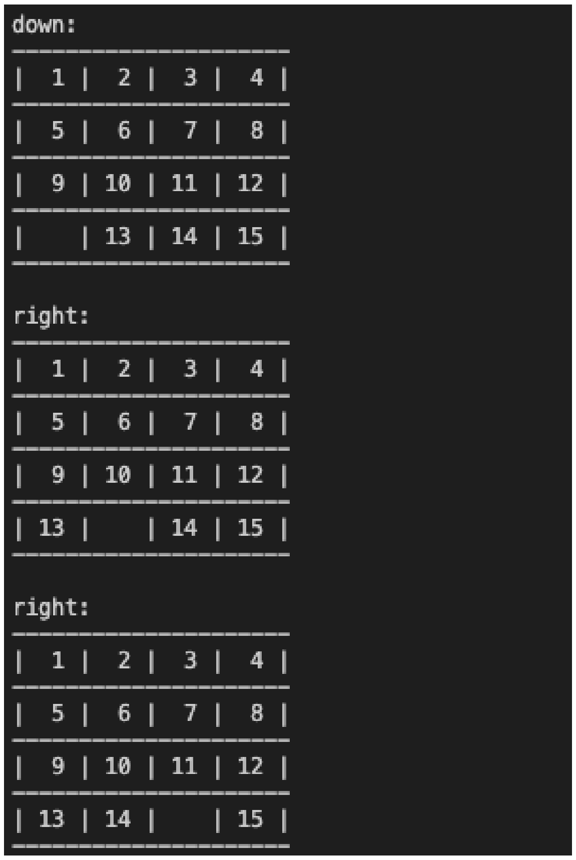
```
left:
---------------------
|  1 |  2 |  3 |  4 |
---------------------
|  5 |  6 |  7 |  8 |
---------------------
|    | 11 | 12 | 15 |
---------------------
| 10 |  9 | 13 | 14 |
---------------------

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

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



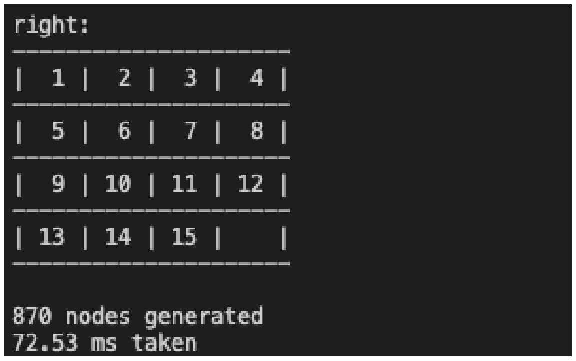
```
right:
---------------------
|  1 |  2 |  3 |  4 |
---------------------
|  5 |  6 |  7 |  8 |
---------------------
| 10 | 11 | 12 | 15 |
---------------------
|  9 | 13 |    | 14 |
---------------------

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

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



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

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

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



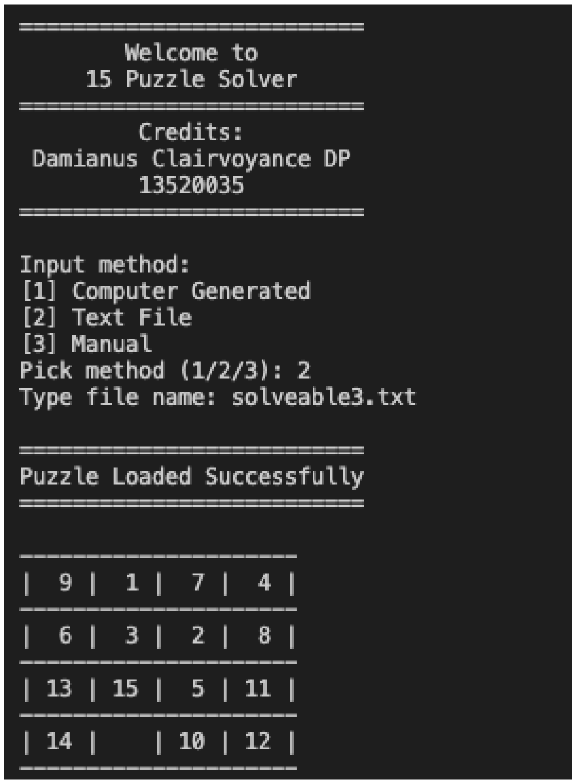
```
down:
---------------------
|  1 |  2 |  3 |  4 |
---------------------
|  5 |  6 |  7 |  8 |
---------------------
|  9 | 10 | 11 | 12 |
---------------------
|    | 13 | 14 | 15 |
---------------------

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

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



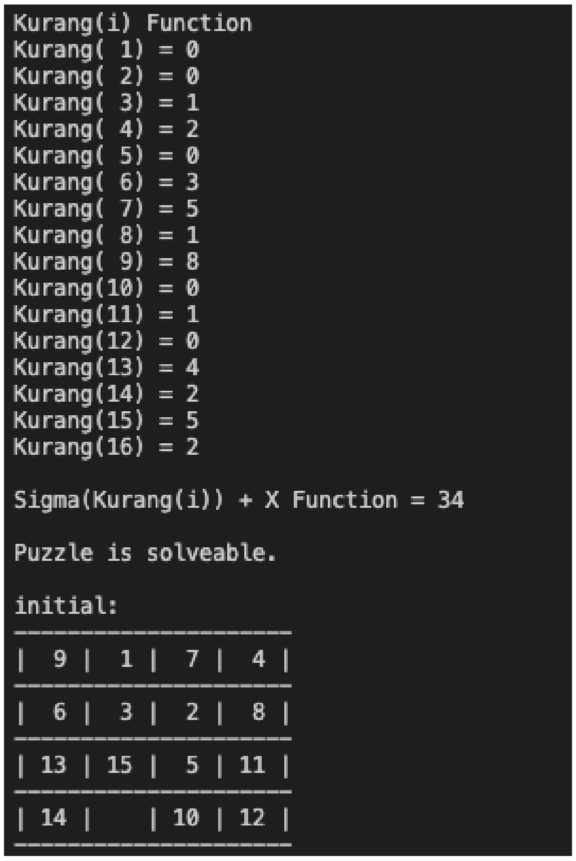
```
right:
---------------------
|  1 |  2 |  3 |  4 |
---------------------
|  5 |  6 |  7 |  8 |
---------------------
|  9 | 10 | 11 | 12 |
---------------------
| 13 | 14 | 15 |    |
---------------------

870 nodes generated
72.53 ms taken
```

Gambar III.10. Proses *Test Case* 4

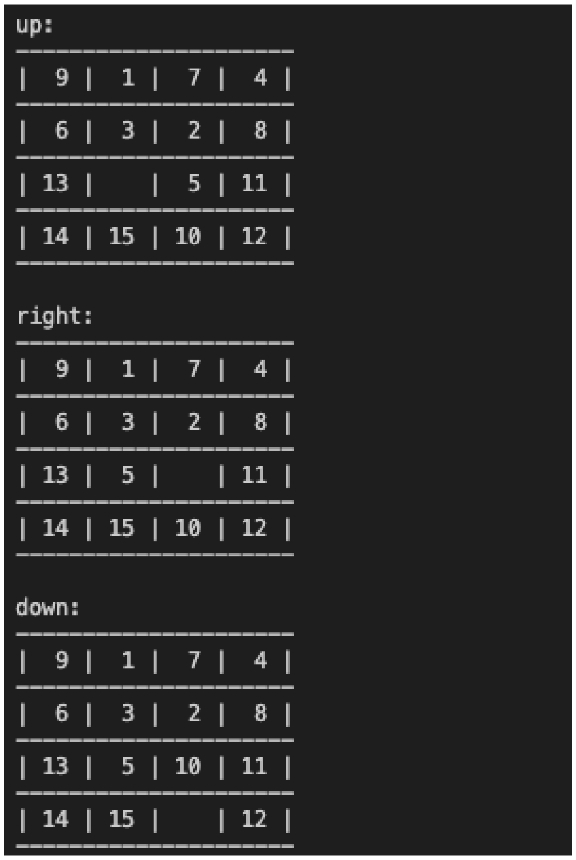
E. *Test Case* 5: solveable3.txt



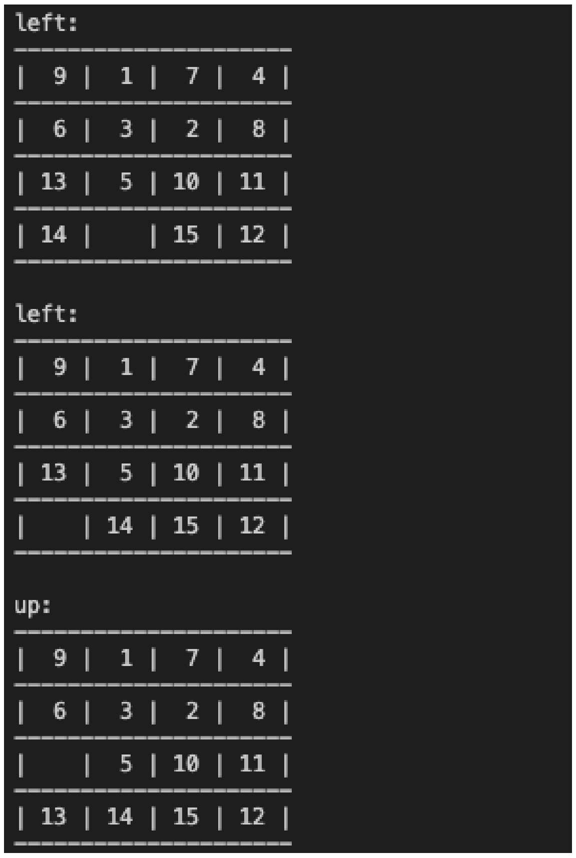
```
==========================
        Welcome to
     15 Puzzle Solver
==========================
         Credits:
 Damianus Clairvoyance DP
         13520035
==========================

Input method:
[1] Computer Generated
[2] Text File
[3] Manual
Pick method (1/2/3): 2
Type file name: solveable3.txt

==========================
Puzzle Loaded Successfully
==========================

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



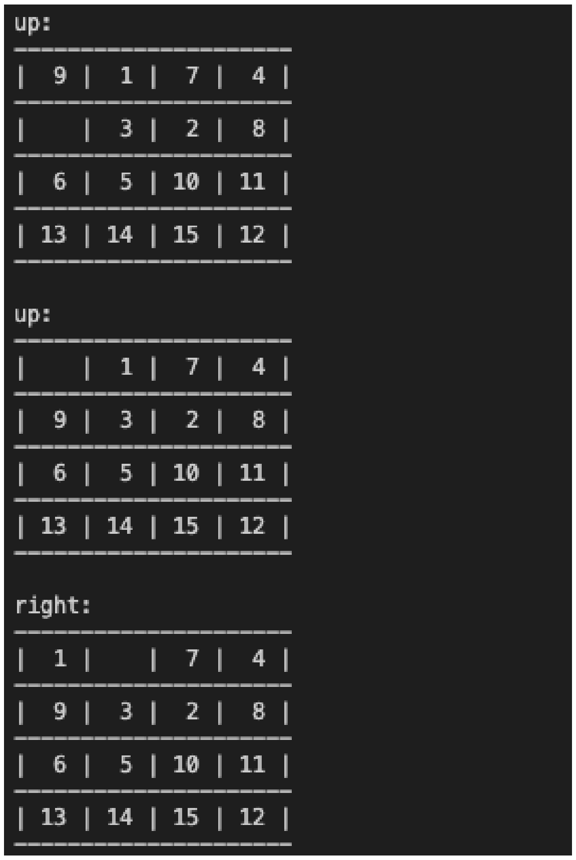
```
Kurang(i) Function
Kurang( 1) = 0
Kurang( 2) = 0
Kurang( 3) = 1
Kurang( 4) = 2
Kurang( 5) = 0
Kurang( 6) = 3
Kurang( 7) = 5
Kurang( 8) = 1
Kurang( 9) = 8
Kurang(10) = 0
Kurang(11) = 1
Kurang(12) = 0
Kurang(13) = 4
Kurang(14) = 2
Kurang(15) = 5
Kurang(16) = 2

Sigma(Kurang(i)) + X Function = 34

Puzzle is solveable.

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



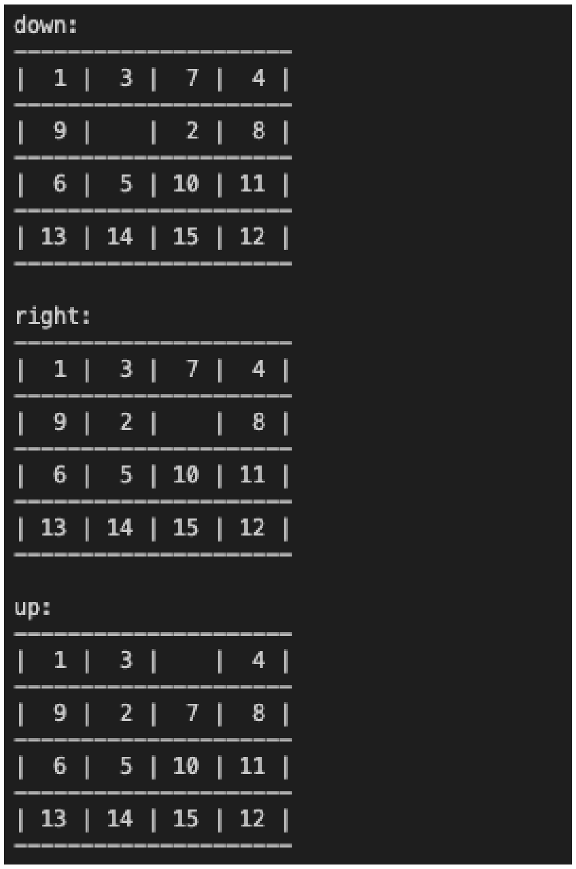
```
up:
---------------------
|  9 |  1 |  7 |  4 |
---------------------
|  6 |  3 |  2 |  8 |
---------------------
| 13 |    |  5 | 11 |
---------------------
| 14 | 15 | 10 | 12 |
---------------------

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

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



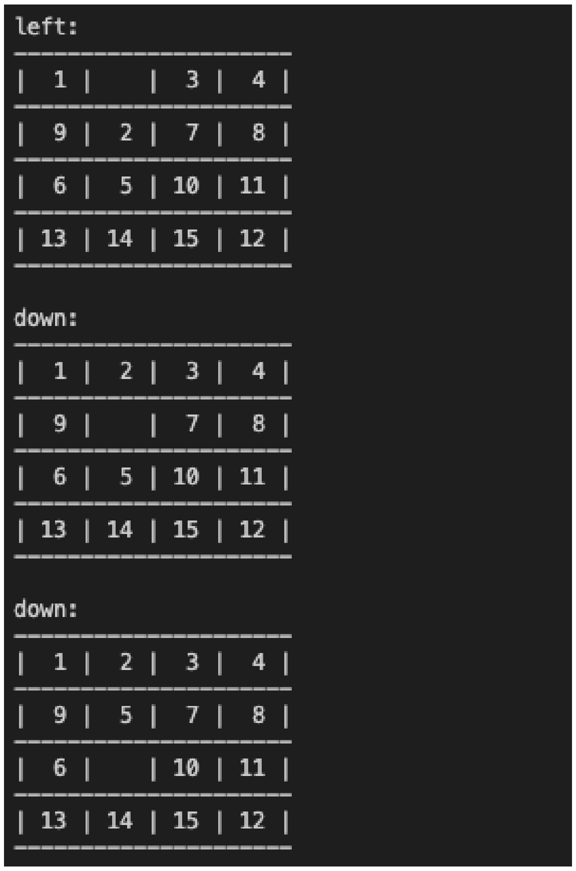
```
left:
---------------------
|  9 |  1 |  7 |  4 |
---------------------
|  6 |  3 |  2 |  8 |
---------------------
| 13 |  5 | 10 | 11 |
---------------------
| 14 |    | 15 | 12 |
---------------------

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

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



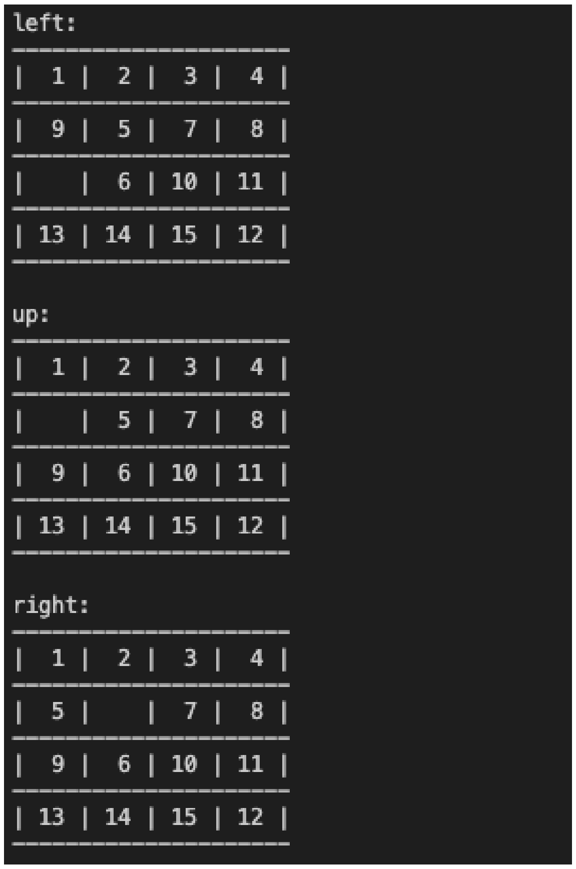
```
up:
---------------------
|  9 |  1 |  7 |  4 |
---------------------
|    |  3 |  2 |  8 |
---------------------
|  6 |  5 | 10 | 11 |
---------------------
| 13 | 14 | 15 | 12 |
---------------------

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

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



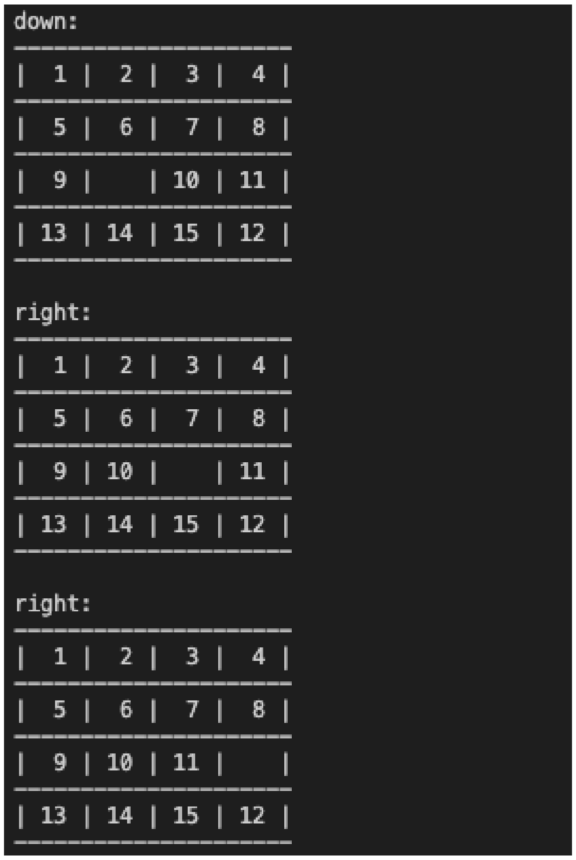
```
down:
---------------------
|  1 |  3 |  7 |  4 |
---------------------
|  9 |    |  2 |  8 |
---------------------
|  6 |  5 | 10 | 11 |
---------------------
| 13 | 14 | 15 | 12 |
---------------------

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

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



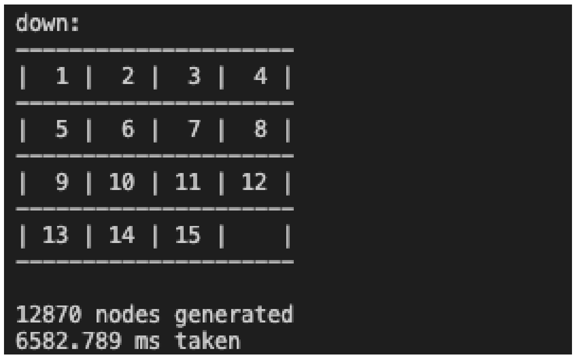
```
left:
---------------------
|  1 |    |  3 |  4 |
---------------------
|  9 |  2 |  7 |  8 |
---------------------
|  6 |  5 | 10 | 11 |
---------------------
| 13 | 14 | 15 | 12 |
---------------------

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

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



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

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

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



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

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

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



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

12870 nodes generated
6582.789 ms taken
```

Gambar III.11. Proses *Test Case* 5

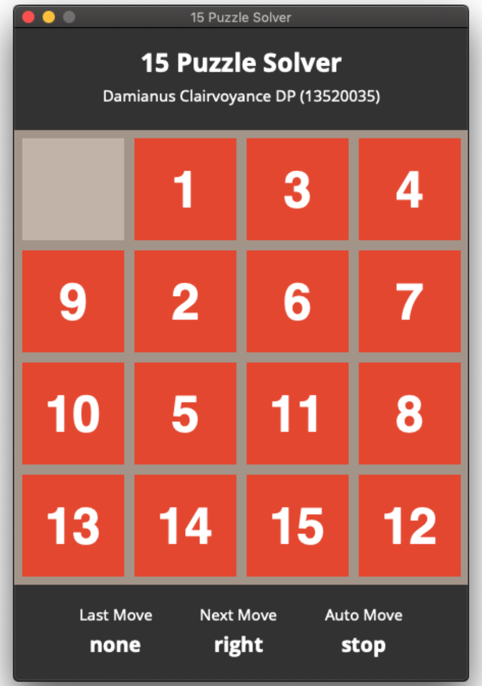
Bonus: Graphical User Interface (GUI)

Sebagai bonus, dirancang Graphical User Interface (GUI) dengan memanfaatkan kakas `tkinter`. Sesuai spesifikasi, GUI hanya berperan menampilkan papan 15-Puzzle secara menarik dan pergeseran ubin secara interaktif. GUI hanya akan dibangkitkan pada akhir program Command Line Interface (CLI), khusus untuk *puzzle* yang dapat diselesaikan.

GUI terdiri atas 3 bagian utama, yakni informasi program, papan puzzle, dan kondisi permainan. Bagian informasi program terdiri atas nama permainan dan pemrogram. Bagian kondisi permainan terdiri atas langkah terakhir (*last move*), langkah selanjutnya (*next move*), dan kondisi otomasi (*auto move*). *Auto move* dapat berupa ”stop” (*puzzle* tidak bergerak secara otomatis), ”right” (*puzzle* bergerak maju secara otomatis menuju kondisi akhir), atau ”left” (*puzzle* bergerak mundur secara otomatis menuju kondisi awal).

Secara default, GUI akan menampilkan kondisi awal *puzzle* dengan auto move “*stop*”. GUI dapat dioperasikan oleh pengguna dengan menekan tanda panah pada papan tik (*keyboard*). Pengguna dapat menekan panah kanan untuk maju ke langkah selanjutnya atau panah kiri untuk mundur ke langkah sebelumnya. Pengguna dapat menekan panah atas untuk melangkah maju secara otomatis setiap 1 detik atau panah bawah untuk melangkah mundur secara otomatis setiap 1 detik. Pengguna dapat menonaktifkan *auto move* dengan menekan panah sebaliknya dari kondisi *auto move* sekarang.

Sebagai demonstrasi, akan digunakan *test case* 1, yaitu solveable1.txt.


15 Puzzle Solver

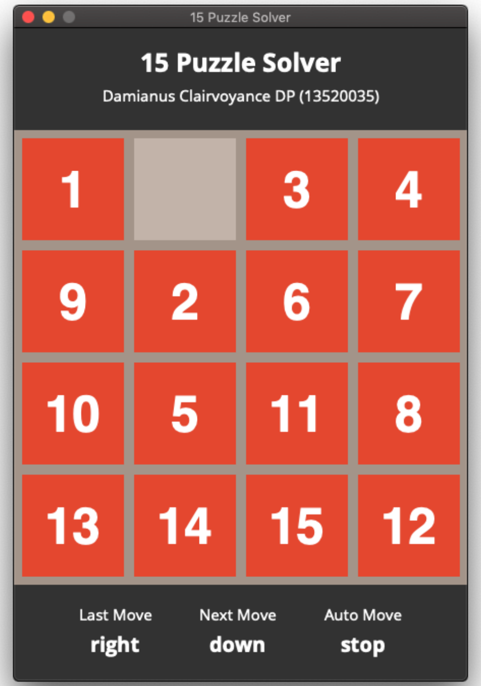
15 Puzzle Solver

Damianus Clairvoyance DP (13520035)

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

Last Move	Next Move	Auto Move
none	**right**	**stop**

Kondisi Awal


15 Puzzle Solver

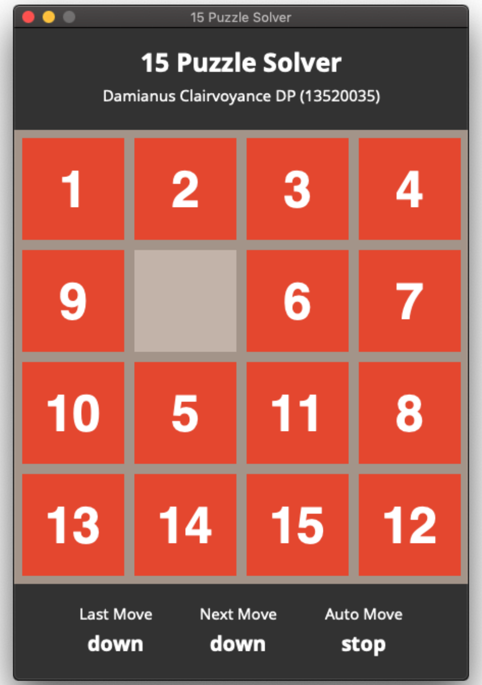
15 Puzzle Solver

Damianus Clairvoyance DP (13520035)

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

Last Move	Next Move	Auto Move
right	**down**	**stop**

Tekan Panah Kanan (Next Move)


15 Puzzle Solver

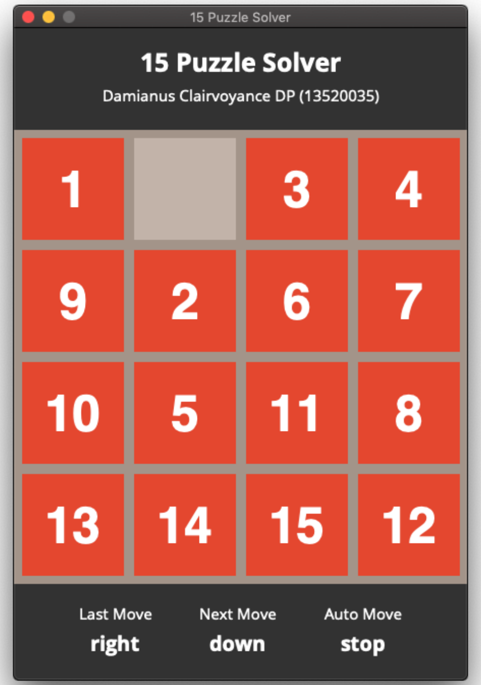
15 Puzzle Solver

Damianus Clairvoyance DP (13520035)

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

Last Move	Next Move	Auto Move
down	**down**	**stop**

Tekan Panah Kanan (Next Move)


15 Puzzle Solver

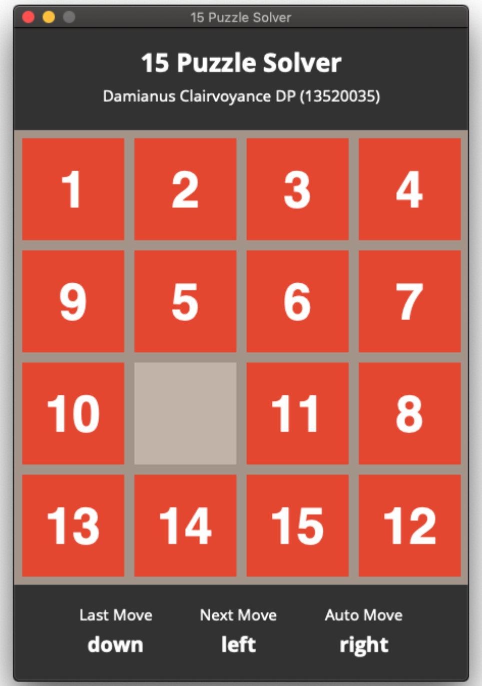
15 Puzzle Solver

Damianus Clairvoyance DP (13520035)

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

Last Move	Next Move	Auto Move
right	**down**	**stop**

Tekan Panah Kiri (Last Move)



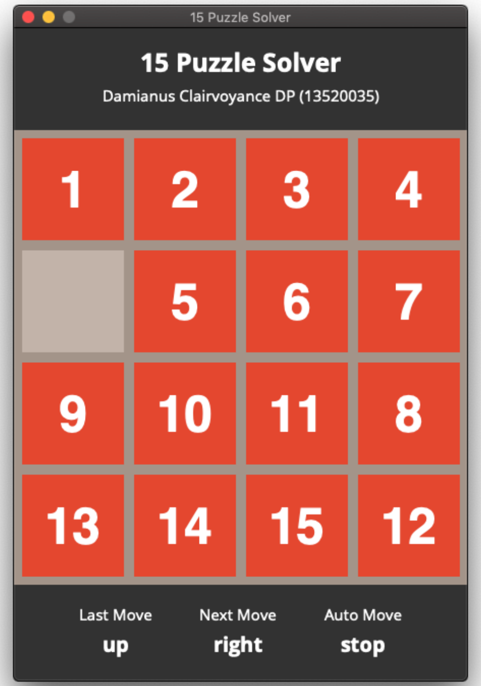
15 Puzzle Solver

Damianus Clairvoyance DP (13520035)

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

Last Move	Next Move	Auto Move
down	left	right

Tekan Panah Atas (Next Move per 1 s)



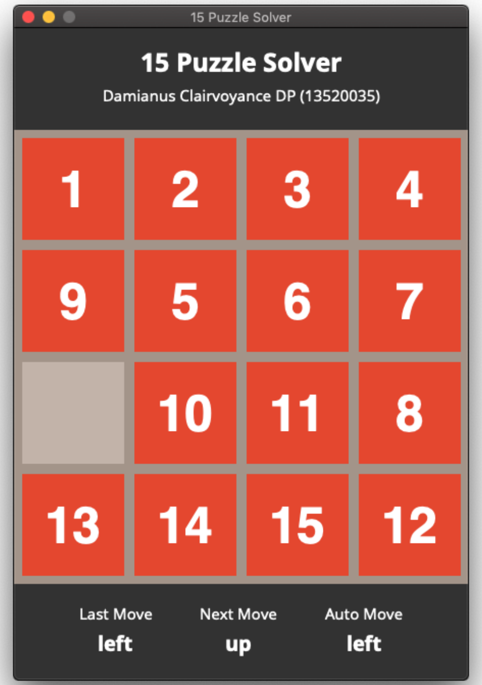
15 Puzzle Solver

Damianus Clairvoyance DP (13520035)

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

Last Move	Next Move	Auto Move
up	right	stop

Tekan Panah Bawah (Stop Auto Move)



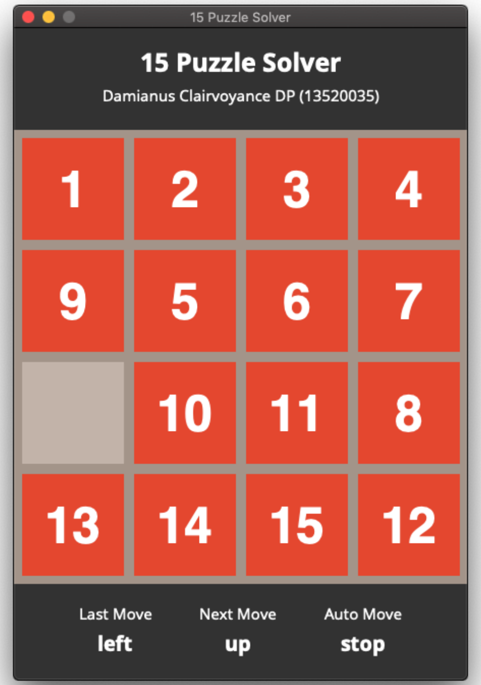
15 Puzzle Solver

Damianus Clairvoyance DP (13520035)

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

Last Move	Next Move	Auto Move
left	up	left

Tekan Panah Bawah (Last Move per 1 s)



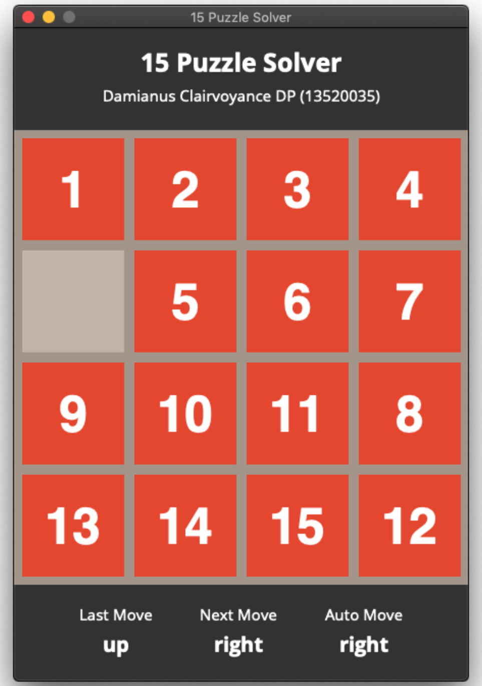
15 Puzzle Solver

Damianus Clairvoyance DP (13520035)

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

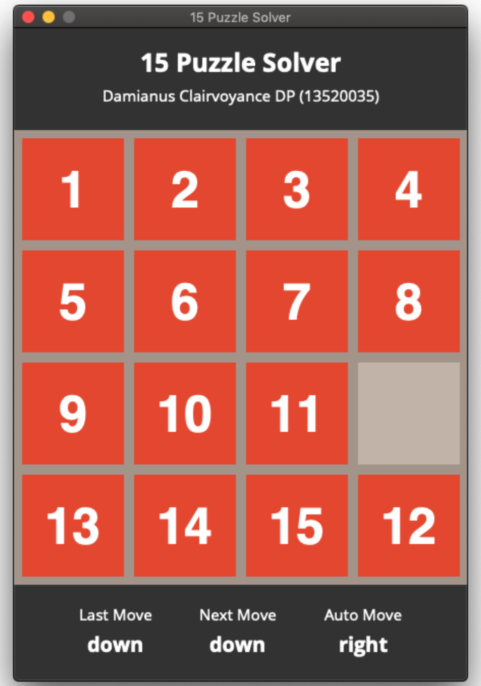
Last Move	Next Move	Auto Move
left	up	stop

Tekan Panah Atas (Stop Auto Move)



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

Last Move up · Next Move right · Auto Move right

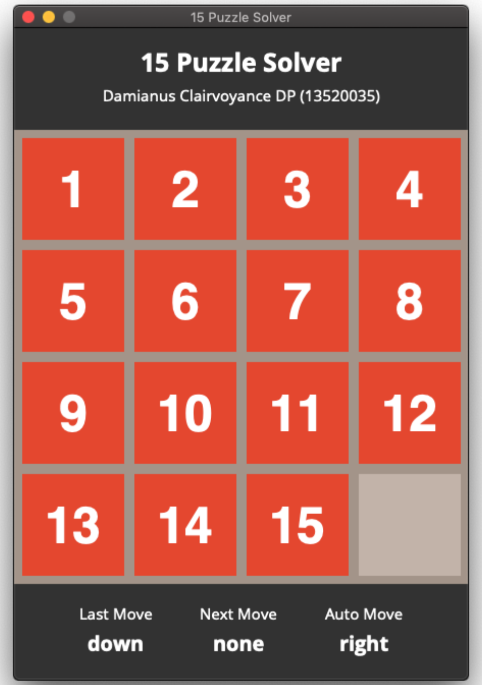


1	2	3	4
5	6	7	8
9	10	11	
13	14	15	12

Last Move down · Next Move down · Auto Move right

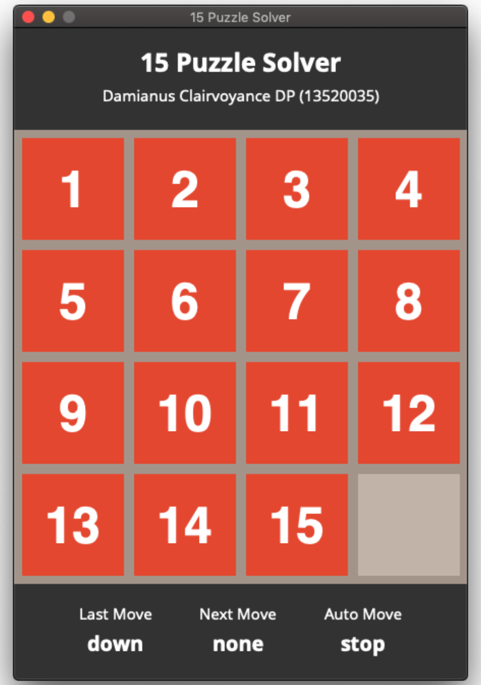
Tekan Panah Atas (Next Move per 1 s)

Do Nothing



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

Last Move down · Next Move none · Auto Move right



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

Last Move down · Next Move none · Auto Move stop

Do Nothing

Do Nothing (Sampai Solved: Stop)

Gambar III.12. Proses *Test Case* 1 pada GUI

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.	√	

Tabel III.1. Tabel Penilaian Mandiri

BAB IV
AKSES *REPOSITORY*

Seluruh fail tugas kecil ini dapat diunduh melalui repository GitHub pada tautan berikut.
https://github.com/dclairvoyance/Tucil3Stima

Struktur repository tersebut ialah sebagai berikut.

- Folder **src**, berisi *source code* program dan *test case* (instansiasi persoalan). *Source code* terdiri atas sebuah fail program utama (app.py) dan 5 implementasi kelas (gui.py, inputfile.py, livenodequeue.py, node.py, dan puzzle.py), beserta sebuah file konfigurasi GUI (configurations.py). *Test case* terdiri atas 5 fail teks, berupa 2 kasus yang tidak dapat diselesaikan dan 3 kasus yang dapat diselesaikan.
- Folder **doc**, berisi laporan tugas kecil Tucil3_13520035.pdf.
- **README**, berisi tata cara penggunaan program, yaitu deskripsi singkat, *requirement*, langkah kompilasi, dan identitas pemrogram.

Catatan : Sesuai arahan di QnA, karena program dirancang dalam bahasa Python, folder bin tidak diperlukan (tidak ada fail *executable*). Sesuai arahan milis yang tidak secara eksplisit menyatakan direktori *test case*, test case diletakkan dalam src demi kemudahan.

BAB V
KESIMPULAN DAN SARAN

15-Puzzle dapat diselesaikan dengan algoritma Branch and Bound. Akan tetapi, meskipun mampu menyelesaikan 15-Puzzle secara efektif, algoritma Branch and Bound dengan metode ini tidak menyelesaikannya secara efisien, terlihat dari lamanya waktu proses. Pemilihan fungsi pembatas yang lebih baik akan menyelesaikan 15-Puzzle dengan lebih efisien.

Sebagai saran pengembangan (bukan fungsionalitas), dapat ditambahkan komponen lain pada GUI, seperti *import puzzle*, sehingga GUI dapat berjalan secara independen tanpa CLI. Tujuannya, GUI tidak hanya berperan sebagai visualisasi penyelesaian.

REFERENSI

Slide Bahan Kuliah IF2211 Strategi Algoritma – Algoritma Branch and Bound (Bagian 1)