TUGAS KECIL

Penyelesaian Persoalan 15-Puzzle dengan Algoritma Branch and Bound

LAPORAN

Diajukan sebagai salah satu tugas mata kuliah IF2211 Strategi Algoritma pada

Semester II

Tahun Akademik 2021-2022

oleh

Owen Christian Wijaya 13520124

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2022

DAFTAR ISI

BAB I. ALGORITMA *BRANCH AND BOUND*

Dalam tugas kecil ini, algoritma *branch and bound* digunakan untuk menentukan langkah-langkah optimal untuk menyelesaikan sebuah permasalahan 15-puzzle. Dalam permainan 15-puzzle, pemain harus menyelesaikan sebuah puzzle untuk mencapai posisi akhir yang diinginkan dengan empat buah gerakan. Algoritma *branch and bound* akan melakukan pencarian terhadap semua kemungkinan gerakan, dan menentukan urutan yang diperlukan untuk menyelesaikan permasalahan tersebut.

Saat melakukan eksekusi, program akan menerima input puzzle dari pengguna baik secara langsung atau dari plainteks dan melakukan validasi. Apabila puzzle tersebut sudah mencapai posisi yang diinginkan atau ada yang invalid, maka program tidak akan melakukan pencarian. Sebaliknya, program akan memulai algoritma dengan melakukan perhitungan nilai "kurang" per ubin. Nilai "kurang" menandakan jumlah ubin yang berada pada posisi lebih dari ubin tertentu namun mempunyai nilai lebih kecil. Setelah itu, nilai jumlah ubin ini akan dijumlahkan. Posisi *empty space* (dirujuk sebagai ES lebih lanjut di dokumen ini) akan diperhitungkan juga untuk menentukan apakah puzzle tersebut dapat diselesaikan. Apabila jumlah nilai "kurang" dan nilai posisi ES puzzle bernilai ganjil, maka program tidak akan menyelesaikan puzzle. Sebaliknya, program akan melakukan inisialisasi algoritma.

Program akan menginisialisasikan sebuah *priority queue* untuk menampung *state* puzzle dan *cost* yang dibutuhkan untuk mencapai *state* tersebut. *Cost* diperhitungkan dari kedalaman *state* tersebut dan jumlah ubin yang terletak di posisi yang salah pada puzzle. Setelah itu, *state* tersebut akan di-*enqueue* ke dalam *priority queue* beserta *cost* dan informasi tentang arah yang diambil. *Priority queue* melakukan *enqueue* berdasarkan prioritas *cost. Cost* yang lebih kecil akan diurutkan di posisi yang lebih depan. Setelah semua selesai diperiksa, program akan melakukan *dequeue* untuk melakukan pengecekan terhadap *state* tersebut. Program akan melakukan *enqueue* terhadap *state* yang menggunakan semua kemungkinan arah kecuali arah yang diambil sebelumnya. Misalkan, apabila *state* yang sedang diperiksa diperoleh dari arah UP, maka program tidak akan meng-*enqueue* state DOWN karena akan kembali ke posisi awal. Selain itu, program akan melakukan pengecekan apabila suatu *state* telah pernah diperiksa sebelumnya menggunakan *dictionary*, sehingga *state* yang sudah pernah diperiksa sebelumnya tidak akan diperiksa lagi. Proses ini akan terus dilakukan sampai antara *priority queue* kosong atau ditemukan *state* selesai.

Setelah proses pencarian selesai, program akan menghubungkan *state-state* yang menghasilkan jawaban dengan melihat nilai ID dari *state* yang ada. Setelah itu, program akan menampilkan hasil urutan, jumlah simpul yang dibangkitkan, dan waktu eksekusi. Untuk GUI, program akan menerima *array* berisi *states* yang kemudian di-*update* per 1 detik untuk menampilkan animasi pergerakan.

BAB II. *SOURCE PROGRAM* DALAM BAHASA PYTHON

2.1 algo.py

File berisi algoritma *branch and bound* yang digunakan.

```
from queue import PriorityQueue
from puzzle import RED_COLOR, RESET_COLOR, PuzzleItem, InvItem, Puzzle
import timeit as time

def solve(p):
    # displays initial puzzle
    print()
    print("Initial Puzzle:")
    p.show()
    print()

    outputMessage = ""
    res = []
    kurangMessage = ""
    # displays initial invalid values (kurang[i])
    if (not p.isSolved()):
        invalid_pq = PriorityQueue()
        for i in range(16):
            if (p.buffer[int(i/4)][int(i%4)] == "ES"):
                invalid_pq.put(InvItem(16, p.invalidPos(i)))
            else:
                currentValue = int(p.buffer[int(i/4)][int(i%4)])
                invalid_pq.put(InvItem(currentValue, p.invalidPos(i)))

        kurangMessage += "List of Invalid Values: \n"

        while (not invalid_pq.empty()):
            temp = invalid_pq.get()
            if (temp.priority == 16):
                kurangMessage += "Kurang[ES] = {}\n".format(temp.value)
            else:
                kurangMessage += "Kurang[{}] = {}\n".format(temp.priority, temp.value)

        kurangMessage+= "Sum of invalid values: {}\n".format(p.sumOfInvalidPos())
```

```
        kurangMessage += "Sum of invalid values and whether empty space in determined
position: {}\n".format(
            p.sumOfInvalidPos() + p.nullPos())

        print(kurangMessage)

        if ((p.sumOfInvalidPos() + p.nullPos()) % 2 != 0):
            raise Exception(RED_COLOR + "This puzzle cannot be solved!\n" + RESET_COLOR)

        else:
            print("\nSolving puzzle ... ")
            prioqueue = PriorityQueue()
            p.curr_depth = 0
            p.id = 0
            # initializes initial puzzle as first item in queue
            prioqueue.put(PuzzleItem(0, [p, "NONE"]))
            state_dict = {}
            puzzle_arr = []
            curr_id = 0

            # starts searching process
            start_time = time.default_timer()

            while (prioqueue.qsize() != 0):
                # dequeues item with lowest priority
                puzzleItem = prioqueue.get().item
                temp = puzzleItem[0]
                prev_direction = puzzleItem[1]
                # adds that item to the list of solved puzzles
                puzzle_arr.append([temp, prev_direction])
                curr_id += 1

                if temp.isSolved():
                    stop_time = time.default_timer()
                    break

                if (temp.checkDir("UP") and prev_direction != "DOWN"):
                    puzzle_up = Puzzle([x for arr in temp.buffer for x in arr])
                    puzzle_up.shift("UP")
                    if (not puzzle_up.stateExisted(state_dict)):
                        puzzle_up.curr_depth = temp.curr_depth + 1
                        puzzle_up.id = curr_id
                        currCost = puzzle_up.curr_depth + puzzle_up.nonMatchingTile()
                        prioqueue.put(PuzzleItem(currCost, [puzzle_up, "UP"]))
                        state_dict["|".join([x for arr in puzzle_up.buffer for x in
arr])] = True
```

```
                if (temp.checkDir("LEFT") and prev_direction != "RIGHT"):
                    puzzle_left = Puzzle([x for arr in temp.buffer for x in arr])
                    puzzle_left.shift("LEFT")
                    if (not puzzle_left.stateExisted(state_dict)):
                        puzzle_left.curr_depth = temp.curr_depth + 1
                        puzzle_left.id = curr_id
                        currCost = puzzle_left.curr_depth +
puzzle_left.nonMatchingTile()
                        prioqueue.put(PuzzleItem(currCost, [puzzle_left, "LEFT"]))
                        state_dict["|".join([x for arr in puzzle_left.buffer for x in
arr])] = True

                if (temp.checkDir("DOWN") and prev_direction != "UP"):
                    puzzle_down = Puzzle([x for arr in temp.buffer for x in arr])
                    puzzle_down.shift("DOWN")
                    if (not puzzle_down.stateExisted(state_dict)):
                        puzzle_down.curr_depth = temp.curr_depth + 1
                        puzzle_down.id = curr_id
                        currCost = puzzle_down.curr_depth +
puzzle_down.nonMatchingTile()
                        prioqueue.put(PuzzleItem(currCost, [puzzle_down, "DOWN"]))
                        state_dict["|".join([x for arr in puzzle_down.buffer for x in
arr])] = True

                if (temp.checkDir("RIGHT") and prev_direction != "LEFT"):
                    puzzle_right = Puzzle([x for arr in temp.buffer for x in arr])
                    puzzle_right.shift("RIGHT")
                    if (not puzzle_right.stateExisted(state_dict)):
                        puzzle_right.curr_depth = temp.curr_depth + 1
                        puzzle_right.id = curr_id
                        currCost = puzzle_right.curr_depth +
puzzle_right.nonMatchingTile()
                        prioqueue.put(PuzzleItem(currCost, [puzzle_right, "RIGHT"]))
                        state_dict["|".join([x for arr in puzzle_right.buffer for x in
arr])] = True

            # backtracks parent id to get the path
            # of solutions
            res = []
            puzzle_elmt = puzzle_arr[-1]
            while (puzzle_elmt[0].id != 0):
                res = [puzzle_elmt] + res
                puzzle_elmt = puzzle_arr[puzzle_elmt[0].id - 1]
            res = [[p, "NONE"]] + res

            # outputs process information
            outputMessage += "\nPuzzle solved successfully!"
```

```
            outputMessage += "\nElapsed time: " + str("%.11f" % (stop_time -
start_time)) + " seconds"
            outputMessage += "\nRaised nodes: " + str(len(state_dict))
            outputMessage += "\nSteps taken : " + str(len(res) - 1)
            return kurangMessage, res, outputMessage
    else:
        raise Exception(RED_COLOR + "This puzzle is already solved! >:(\n" +
RESET_COLOR)
```

2.2 puzzle.py

File berisi *class* Puzzle yang digunakan, beserta *class PuzzleItem* yang digunakan untuk melakukan *enqueue* terhadap *state puzzle*.

```
from dataclasses import dataclass, field
from typing import Any

GREEN_COLOR = "\u001b[32m"
RED_COLOR = "\033[91m"
RESET_COLOR = "\033[0m"
CYAN_COLOR = "\u001b[36m"
@dataclass(order=True)
class PuzzleItem:
    priority: int
    item: Any=field(compare=False)

@dataclass(order=True)
class InvItem:
    priority: int
    value: Any=field(compare=False)

class Puzzle:

    # data members
    ROW_SIZE = 4
    COL_SIZE = 4
    NULL_I = 0
    NULL_J = 0
    curr_depth = 0
    id = 0
    buffer = []

    '''
    Constructor for the puzzle matrix
    '''
```

```
    def __init__(self, puzzle_string):
        self.buffer = [[0 for _ in range(self.COL_SIZE)] for _ in range(self.ROW_SIZE)]
        for i in range(self.ROW_SIZE):
            for j in range(self.COL_SIZE):
                elmt = puzzle_string[i * self.COL_SIZE + j]
                self.buffer[i][j] = elmt
                if (elmt == "ES"):
                    self.NULL_I = i
                    self.NULL_J = j

    '''
    Prints the puzzle matrix in a readable format
    '''
    def show(self):
        for i in range(self.ROW_SIZE):
            for j in range(self.COL_SIZE):
                if (self.buffer[i][j] == "ES"):
                    print(GREEN_COLOR + self.buffer[i][j] + RESET_COLOR, end="")
                else:
                    print(self.buffer[i][j], end = " ")
            print()

    '''
    Checks possible movement directions for the current position
    '''
    def checkDir(self, direction):
        i = self.NULL_I
        j = self.NULL_J
        if (direction == "LEFT"):
            return j != 0 and self.buffer[i][j - 1] != "ES"
        elif (direction == "RIGHT"):
            return (j != self.COL_SIZE - 1) and self.buffer[i][j + 1] != "ES"
        elif (direction == "UP"):
            return i != 0 and self.buffer[i - 1][j] != "ES"
        elif (direction == "DOWN"):
            return (i != self.ROW_SIZE - 1) and self.buffer[i + 1][j] != "ES"

    '''
    Shifts element of the puzzle matrix
    '''
    def shift(self, direction):
        i = self.NULL_I
        j = self.NULL_J
        if (self.checkDir(direction)):
            if (direction == "LEFT"): # NULL goes left
                self.buffer[i][j] = self.buffer[i][j - 1]
                self.buffer[i][j - 1] = "ES"
```

```
                self.NULL_J -= 1
            elif (direction == "RIGHT"): # NULL goes right
                self.buffer[i][j] = self.buffer[i][j + 1]
                self.buffer[i][j + 1] = "ES"
                self.NULL_J += 1
            elif (direction == "UP"): # NULL goes up
                self.buffer[i][j] = self.buffer[i - 1][j]
                self.buffer[i - 1][j] = "ES"
                self.NULL_I -= 1
            elif (direction == "DOWN"): # NULL goes down
                self.buffer[i][j] = self.buffer[i + 1][j]
                self.buffer[i + 1][j] = "ES"
                self.NULL_I += 1

    '''
    Checks if the puzzle is solved
    '''
    def isSolved(self):
        # return False if last element is not NULL
        if (self.buffer[self.ROW_SIZE - 1][self.COL_SIZE - 1] != "ES"):
            return False

        # else, check if all elements are in correct order, except for last element
        flattened_buffer = [x for arr in self.buffer for x in arr]
        for i in range(1, len(flattened_buffer) - 1):
            # return False if any element is not in correct order
            if (int(flattened_buffer[i]) != int(flattened_buffer[i - 1]) + 1):
                return False

        # return solved if all is sorted
        return True

    '''
    Returns 1 if:
    - odd row and even column
    - even row and odd column
    '''
    def nullPos(self):
        return 1 if (self.NULL_I % 2 != self.NULL_J % 2) else 0

    '''
    Counts the appearance of invalid position where
    element with less value than current element appears on a higher position
    '''
    def invalidPos(self, idx):
        count = 0
        flattened_buffer = [x for arr in self.buffer for x in arr]
```

```
        if (flattened_buffer[idx] == "ES"):
            count = self.COL_SIZE * self.ROW_SIZE - idx - 1
        for i in range(idx, len(flattened_buffer)):
            if (flattened_buffer[i] != "ES" and flattened_buffer[idx] != "ES"):
                if (int(flattened_buffer[i]) < int(flattened_buffer[idx]) and i > idx):
                    count += 1
        return count

    '''
    Returns the sum of invalid position
    '''
    def sumOfInvalidPos(self):
        sum = 0
        for i in range(0, self.ROW_SIZE * self.COL_SIZE):
            sum += self.invalidPos(i)
        return sum

    '''
    Counts the appearance of invalid position where
    tile position doesn't match the value of the tile
    '''
    def nonMatchingTile(self):
        count = 0
        flattened_buffer = [x for arr in self.buffer for x in arr]
        for i in range(0, len(flattened_buffer)):
            if (flattened_buffer[i] != "ES" and (int(flattened_buffer[i]) != (i + 1))):
                count += 1
        return count;

    '''
    Checks whether current state of the puzzle has existed before
    '''
    def stateExisted(self, state_dict):
        state = "|".join([x for arr in self.buffer for x in arr])
        return True if state in state_dict else False
```

2.3 main.py

File berisi *command-line interface* untuk penggunaan di terminal.

```
import fparser as fp
import puzzle as pc
import algo

def title():
```

```
    print(" ____  _     ____  ____  _____ ____  ____  ____  _           _  ____ ")
    print("/  __\\/ \\ /\\/_   \\/_   \\/  __//  _ \\/  __\\/   _\\/ \\ /|      / \\/
___\\")
    print("|  \\/|| | || /   / /   /|  \\  | / \\||  \\/||  /  | |_||_____ | ||    \\")
    print("|  __/| \_/|/   /_/   /_|  /_ | |-|||    /|  \_ | | ||\____\| |\___ |")
    print("\_/   \____/\____/\____/\_____\\_/ \|\_/\_/\\____/\_/ \|      \_/\____/")
    print()
    print(pc.GREEN_COLOR + "A(nother) 15-Puzzle Solver" + pc.RESET_COLOR)

def main():

    while (True):
        print("\nSelect your desired input method: ")
        print("[1] Text file")
        print("[2] Input by user")
        print("[0] Exit")

        try:

            option = int(input("| >> "))

            if (option == 1):
                print(pc.CYAN_COLOR + "\n[SELECTED] Text file" + pc.RESET_COLOR)
                print("Input your filename (without .txt)|")
                print("[IMPORTANT] File must be included in the test folder!")
                fname = input("| >> ")
                buffer = fp.parseText(fname)
            elif (option == 2):
                print(pc.CYAN_COLOR + "\n[SELECTED] Input by user" + pc.RESET_COLOR)
                print("Input the desired matrix in a 4 x 4 grid style!")
                print("Fill the empty space character with '-'!")
                buffer = fp.parseInput()
            elif (option == 0):
                print("Exiting program...\n")
                break

            p = pc.Puzzle(buffer)
            _, res, outputMessage = algo.solve(p)
            for i in range(len(res)):
                res[i][0].show()
                print("Step {} | Command: {} \n".format(i, pc.GREEN_COLOR + res[i][1] +
pc.RESET_COLOR))
            print(outputMessage)

        except Exception as e:
            print(e)
            continue
```

```
title()
main()
```

2.4 gui.py

File berisi konfigurasi *GUI* menggunakan *library* TKinter.

```
from tkinter import *
from tkinter import messagebox
from algo import solve
from puzzle import *
from fparser import *
import time

puzzle_arr = []

def solveClick():

    '''
    Function to solve the puzzle from the GUI
    '''
    global puzzle_arr
    filepath = fname_entry.get()
    try:
        if (len(filepath) != 0):
            p = Puzzle(parseText(filepath))
        else:
            p = Puzzle(parseGUI(layout.getBuf()))
        kurangMsg, res, outputMsg = solve(p)
        ans_text.configure(text = outputMsg)
        kurang_label.configure(text = kurangMsg)

        delay_time = 0.5
        if (time_entry.get()) != "":
            delay_time = float(time_entry.get())
        layout.renderAll(res, delay_time)

    except Exception as e:
        messagebox.showerror("[ERROR]", e)

class GUIPuzzle:
    '''
    Initialize table for the puzzle
    '''
    def __init__(self):
        for i in range(4):
```

```
            for j in range(4):
                self.e = Entry(frame, width = 4, font = ('Arial', 20))
                self.e.grid(row = i, column = j)
                self.e.insert(END, "")

    '''
    Gets buffer value from GUI to be parsed
    '''
    def getBuf(self):
        buffer = ""
        for i in range(4):
            for j in range(4):
                buffer += frame.grid_slaves(row = i, column = j)[0].get() + " "

        return buffer


    '''
    Clears table and resets background color
    '''
    def clear(self):
        kurang_label.configure(text = "")
        steps_label.configure(text = "")
        ans_text.configure(text = "Waiting for search to begin...")
        for i in range(4):
            for j in range(4):
                frame.grid_slaves(row = i, column = j)[0].config({"background":
"white"})
                frame.grid_slaves(row = i, column = j)[0].delete(0, END)

    '''
    Renders a puzzle to the GUI
    '''
    def render(self, puzzle):
        for i in range(4):
            for j in range(4):
                self.e = Entry(frame, width = 4, font = ('Arial', 20))
                self.e.grid(row = i, column = j)
                if (puzzle.buffer[i][j] == "ES"):
                    self.e.insert(END, "")
                    self.e.config({"background": "gray"})
                else:
                    self.e.insert(END, puzzle.buffer[i][j])

    '''
    Renders all puzzles in an array to the GUI
    with delay time
```

```
    '''
    def renderAll(self, puzzle_arr, delay_time):
        global frame
        for i in range(len(puzzle_arr)):
            self.render(puzzle_arr[i][0])
            steps_label.configure(text = "Step " + str(i + 1) + ": " + puzzle_arr[i][1])
            time.sleep(delay_time)
            window.update()
begin_coord = 150
'''
GUI components
'''
window = Tk()
window.geometry("500x400")
window.minsize(500, 400)
window.maxsize(500, 400)
window.title("Puzzearch-15 Puzzle Solver")

frame = Frame(window)
frame.pack(fill= BOTH, expand= True, padx= 20, pady=20)

fname_entry = Entry(frame, text = "Input file name (without *.txt)", font = ('Arial',
10), width = 20)
fname_entry.place(x = 0, y = begin_coord + 40)

time_entry = Entry(frame, text = "Time limit (in seconds)", font = ('Arial', 10), width
= 6)
time_entry.place(x = 200, y = begin_coord + 40 )

time_label = Label(frame, text = "Delay\ntime", font = ('Arial', 8))
time_label.place(x = 170, y = begin_coord + 30)

layout = GUIPuzzle()

steps_label = Label(frame, font = ("Arial", 8))
steps_label.place(x = 0, y = begin_coord)

fname_label = Label(frame, font = ("Arial, 8"), text = "Input file name (without
*.txt)")
fname_label.place(x = 0, y = begin_coord + 20)

kurang_label = Label(frame, font = ("Arial, 8"), wraplength = 150)
kurang_label.place(x = 300, y = 0)

solve_button = Button(frame, text = "Solve", width = 16, command = solveClick)
solve_button.place(x = 0, y = begin_coord + 70)
```

```
clear_button = Button(frame, text = "Clear", width = 16, command = layout.clear)
clear_button.place(x = 128, y = begin_coord + 70)

ans_text = Label(frame, font = ("Arial", 8), text = "Waiting for search to begin...")
ans_text.place(x = 25, y = begin_coord + 100)

window.mainloop()
```

2.5 fparser.py

File berisi *script* untuk melakukan *parsing* puzzle.

```
import os

def checkValid(arr):
    temp = [x for x in (arr)]
    temp.remove("ES")
    temp.append("16")
    temp = [int(x) for x in temp]
    temp.sort()
    for i in range(len(temp)):
        if (int(temp[i]) != i + 1):
            raise Exception("[INVALID] Input is not valid!")
    return True

def parseText(fname):
    dirname = os.path.dirname(__file__)
    path = os.path.join(dirname, '../test/')
    if (os.path.exists(path + fname + ".txt")):
        file = open(path + fname + ".txt", "r")
        arr = file.read().replace("-", "ES").replace("\n", " ").split(" ")
        return arr if checkValid(arr) else None
    else:
        raise Exception("[INVALID] File doesn't exist! Make sure it is stored in the
'test' folder and the filename is correct! (without .txt)")

def parseInput():
    print("\n[SELECTED] Input by user")
    print("Input the desired matrix in a 4 x 4 grid style!")
    print("Fill the empty space character with '-'!")
    buffer = [[0 for _ in range(4)] for _ in range(4)]
    for i in range(4):
        print("[ROW {}] | >> ".format(i + 1), end = " ")
        buffer[i] = list(map(str, input().split()))

    flattened_buffer = ' '.join([x for arr in buffer for x in arr])
    arr = flattened_buffer.replace("-", "ES").replace("\n", " ").split(" ")
```

```
    return arr if checkValid(arr) else None

def parseGUI(buffer):
    arr = buffer.rstrip().replace("-", "ES").replace("\n", " ").split(" ")
    return arr if checkValid(arr) else None
```

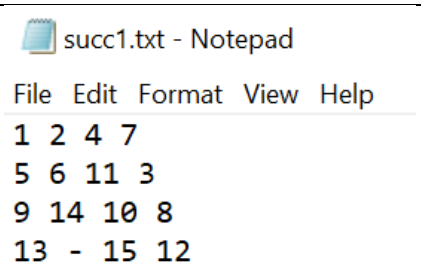
BAB III. PENGUJIAN

Pengujian dilakukan menggunakan CPU i5-9300H 2.40 GHz, sehingga kemungkinan ada perbedaan waktu eksekusi saat pengujian di komputer lain. Perlu diperhatikan juga bahwa karena pembatasan heuristik pada jumlah ubin yang tidak cocok, program ini akan memakan waktu lebih lama untuk menyelesaikan permasalahan yang membutuhkan penyelesaian di atas 20 langkah. *Test cases* yang diujikan mengambil waktu dari < 1 detik hingga 30 detik untuk penyelesaian 30 langkah.

Untuk pengujian GUI, dilakukan pengujian dari input file (succ1.txt) dan input pengguna. Animasi pada GUI dapat dilihat pada *repository* (link dilampirkan di Bab 4).

3.1 Pengujian 10 Langkah

Berkas succ1.txt



Daftar nilai Kurang(I) dan total Kurang(I) + X

```
[SELECTED] Text file
Input your filename (without .txt)|
[IMPORTANT] File must be included in the test folder!
| >> succ1

Initial Puzzle:
1 2 4 7
5 6 11 3
9 14 10 8
13 ES 15 12

List of Invalid Values:
Kurang[1] = 0
Kurang[2] = 0
Kurang[3] = 0
Kurang[4] = 1
Kurang[5] = 1
Kurang[6] = 1
Kurang[7] = 3
Kurang[8] = 0
Kurang[9] = 1
Kurang[10] = 1
Kurang[11] = 4
Kurang[12] = 0
Kurang[13] = 1
Kurang[14] = 4
Kurang[15] = 1
Kurang[ES] = 2
Sum of invalid values: 20
Sum of invalid values and whether empty space in determined position: 20
```

Hasil eksekusi (1)	Hasil Eksekusi (2)

```
Solving puzzle ...
1 2 4 7
5 6 11 3
9 14 10 8
13 ES 15 12
Step 0 | Command: NONE

1 2 4 7
5 6 11 3
9 ES 10 8
13 14 15 12
Step 1 | Command: UP

1 2 4 7
5 6 11 3
9 10 ES 8
13 14 15 12
Step 2 | Command: RIGHT

1 2 4 7
5 6 ES 3
9 10 11 8
13 14 15 12
Step 3 | Command: UP

1 2 4 7
5 6 3 ES
9 10 11 8
13 14 15 12
Step 4 | Command: RIGHT
```

```
1 2 4 ES
5 6 3 7
9 10 11 8
13 14 15 12
Step 5 | Command: UP

1 2 ES 4
5 6 3 7
9 10 11 8
13 14 15 12
Step 6 | Command: LEFT

1 2 3 4
5 6 ES 7
9 10 11 8
13 14 15 12
Step 7 | Command: DOWN

1 2 3 4
5 6 7 ES
9 10 11 8
13 14 15 12
Step 8 | Command: RIGHT

1 2 3 4
5 6 7 8
9 10 11 ES
13 14 15 12
Step 9 | Command: DOWN

1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 ES
Step 10 | Command: DOWN
```

Waktu eksekusi dan banyak *node* yang dibangkitkan

```
Puzzle solved successfully!
Elapsed time: 0.00284720000 seconds
Raised nodes: 48
Steps taken : 10
```

3.2 Pengujian 15 Langkah

Berkas succ2.txt

succ2.txt - Notepad

File Edit Format View Help

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

Daftar nilai Kurang(I) dan total Kurang(I) + X

```
[SELECTED] Text file
Input your filename (without .txt)|
[IMPORTANT] File must be included in the test folder!
| >> succ2

Initial Puzzle:
2 ES 6 3
1 10 7 4
5 11 12 8
9 13 14 15

List of Invalid Values:
Kurang[1] = 0
Kurang[2] = 1
Kurang[3] = 1
Kurang[4] = 0
Kurang[5] = 0
Kurang[6] = 4
Kurang[7] = 2
Kurang[8] = 0
Kurang[9] = 0
Kurang[10] = 5
Kurang[11] = 2
Kurang[12] = 2
Kurang[13] = 0
Kurang[14] = 0
Kurang[15] = 0
Kurang[ES] = 14
Sum of invalid values: 31
Sum of invalid values and whether empty space in determined position: 32
```

Hasil eksekusi (1)

```
Solving puzzle ...
2 ES 6 3
1 10 7 4
5 11 12 8
9 13 14 15
Step 0 | Command: NONE

2 6 ES 3
1 10 7 4
5 11 12 8
9 13 14 15
Step 1 | Command: RIGHT

2 6 3 ES
1 10 7 4
5 11 12 8
9 13 14 15
Step 2 | Command: RIGHT

2 6 3 4
1 10 7 ES
5 11 12 8
9 13 14 15
Step 3 | Command: DOWN

2 6 3 4
1 10 7 8
5 11 12 ES
9 13 14 15
Step 4 | Command: DOWN

2 6 3 4
1 10 7 8
5 11 ES 12
9 13 14 15
Step 5 | Command: LEFT
```

Hasil eksekusi (2)

```
2 6 3 4
1 10 7 8
5 ES 11 12
9 13 14 15
Step 6 | Command: LEFT

2 6 3 4
1 ES 7 8
5 10 11 12
9 13 14 15
Step 7 | Command: UP

2 ES 3 4
1 6 7 8
5 10 11 12
9 13 14 15
Step 8 | Command: UP

ES 2 3 4
1 6 7 8
5 10 11 12
9 13 14 15
Step 9 | Command: LEFT

1 2 3 4
ES 6 7 8
5 10 11 12
9 13 14 15
Step 10 | Command: DOWN
```

Hasil eksekusi (3)

```
1 2 3 4
5 6 7 8
ES 10 11 12
9 13 14 15
Step 11 | Command: DOWN

1 2 3 4
5 6 7 8
9 10 11 12
ES 13 14 15
Step 12 | Command: DOWN

1 2 3 4
5 6 7 8
9 10 11 12
13 ES 14 15
Step 13 | Command: RIGHT

1 2 3 4
5 6 7 8
9 10 11 12
13 14 ES 15
Step 14 | Command: RIGHT

1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 ES
Step 15 | Command: RIGHT
```

Waktu eksekusi dan banyak *node* yang dibangkitkan

```
Puzzle solved successfully!
Elapsed time: 0.00506660000 seconds
Raised nodes: 80
Steps taken : 15
```

3.3 Pengujian 20 Langkah

Berkas succ3.txt

succ3.txt - Notepad

File Edit Format View Help

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

Daftar nilai Kurang(I) dan total Kurang(I) + X

```
[SELECTED] Text file
Input your filename (without .txt)|
[IMPORTANT] File must be included in the test folder!
| >> succ3

Initial Puzzle:
2 6 7 3
5 1 15 4
9 14 10 8
13 ES 12 11

List of Invalid Values:
Kurang[1] = 0
Kurang[2] = 1
Kurang[3] = 1
Kurang[4] = 0
Kurang[5] = 2
Kurang[6] = 4
Kurang[7] = 4
Kurang[8] = 0
Kurang[9] = 1
Kurang[10] = 1
Kurang[11] = 0
Kurang[12] = 1
Kurang[13] = 2
Kurang[14] = 5
Kurang[15] = 8
Kurang[ES] = 2
Sum of invalid values: 32
Sum of invalid values and whether empty space in determined position: 32
```

Hasil eksekusi (1)

```
Solving puzzle ...
2 6 7 3
5 1 15 4
9 14 10 8
13 ES 12 11
Step 0 | Command: NONE

2 6 7 3
5 1 15 4
9 ES 10 8
13 14 12 11
Step 1 | Command: UP

2 6 7 3
5 1 15 4
ES 9 10 8
13 14 12 11
Step 2 | Command: LEFT

2 6 7 3
ES 1 15 4
5 9 10 8
13 14 12 11
Step 3 | Command: UP

2 6 7 3
1 ES 15 4
5 9 10 8
13 14 12 11
Step 4 | Command: RIGHT

2 ES 7 3
1 6 15 4
5 9 10 8
13 14 12 11
Step 5 | Command: UP
```

Hasil eksekusi (2)

```
ES 2 7 3
1 6 15 4
5 9 10 8
13 14 12 11
Step 6 | Command: LEFT

1 2 7 3
ES 6 15 4
5 9 10 8
13 14 12 11
Step 7 | Command: DOWN

1 2 7 3
5 6 15 4
ES 9 10 8
13 14 12 11
Step 8 | Command: DOWN

1 2 7 3
5 6 15 4
9 ES 10 8
13 14 12 11
Step 9 | Command: RIGHT

1 2 7 3
5 6 15 4
9 10 ES 8
13 14 12 11
Step 10 | Command: RIGHT
```

Hasil eksekusi (3)

```
1 2 7 3
5 6 ES 4
9 10 15 8
13 14 12 11
Step 11 | Command: UP

1 2 ES 3
5 6 7 4
9 10 15 8
13 14 12 11
Step 12 | Command: UP

1 2 3 ES
5 6 7 4
9 10 15 8
13 14 12 11
Step 13 | Command: RIGHT

1 2 3 4
5 6 7 ES
9 10 15 8
13 14 12 11
Step 14 | Command: DOWN

1 2 3 4
5 6 7 8
9 10 15 ES
13 14 12 11
Step 15 | Command: DOWN
```

Hasil eksekusi (4)

```
1 2 3 4
5 6 7 8
9 10 15 11
13 14 12 ES
Step 16 | Command: DOWN

1 2 3 4
5 6 7 8
9 10 15 11
13 14 ES 12
Step 17 | Command: LEFT

1 2 3 4
5 6 7 8
9 10 ES 11
13 14 15 12
Step 18 | Command: UP

1 2 3 4
5 6 7 8
9 10 11 ES
13 14 15 12
Step 19 | Command: RIGHT

1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 ES
Step 20 | Command: DOWN
```

Waktu eksekusi dan banyak *node* yang dibangkitkan

```
Puzzle solved successfully!
Elapsed time: 0.17648880000 seconds
Raised nodes: 3882
Steps taken : 20
```

3.4 Pengujian File Gagal 1

Berkas fail1.txt

fail1.txt - Notepad

File Edit Format View Help

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

Hasil pengujian (gagal karena nilai Kurang(i) + X ganjil)

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

List of Invalid Values:
Kurang[1] = 0
Kurang[2] = 0
Kurang[3] = 0
Kurang[4] = 0
Kurang[5] = 1
Kurang[6] = 1
Kurang[7] = 1
Kurang[8] = 1
Kurang[9] = 1
Kurang[10] = 0
Kurang[11] = 1
Kurang[12] = 3
Kurang[13] = 9
Kurang[14] = 3
Kurang[15] = 4
Kurang[ES] = 0
Sum of invalid values: 25
Sum of invalid values and whether empty space in determined position: 25

This puzzle cannot be solved!
```

3.5 Pengujian File Gagal 2

Berkas fail2.txt

fail2.txt - Notepad

File Edit Format View Help

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

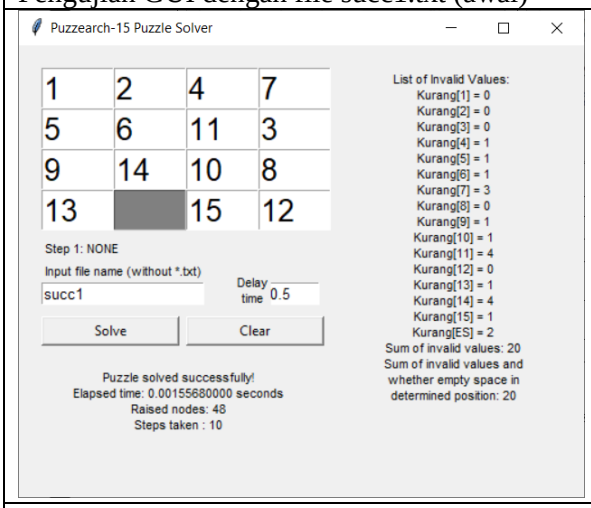
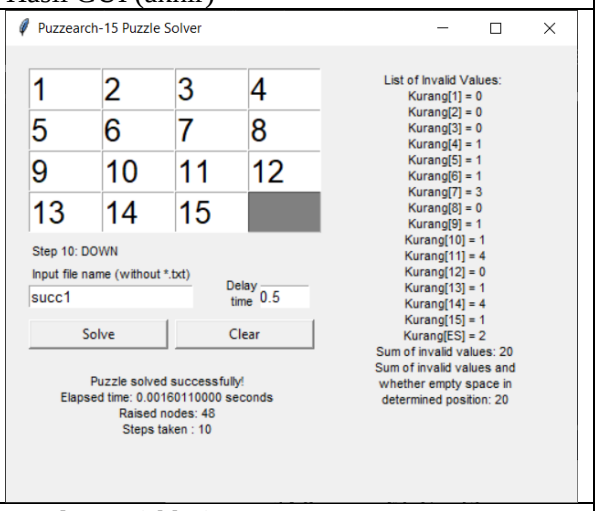
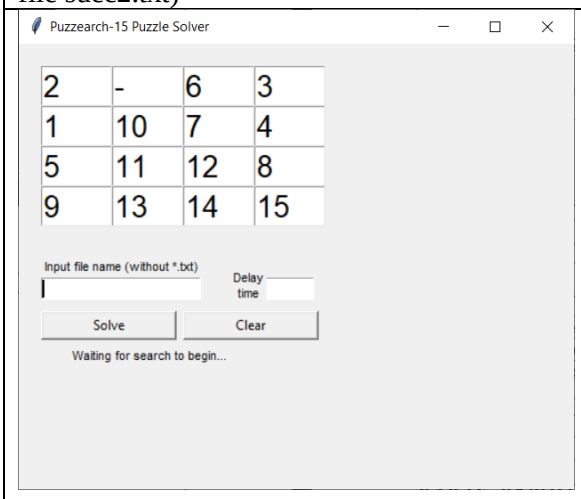
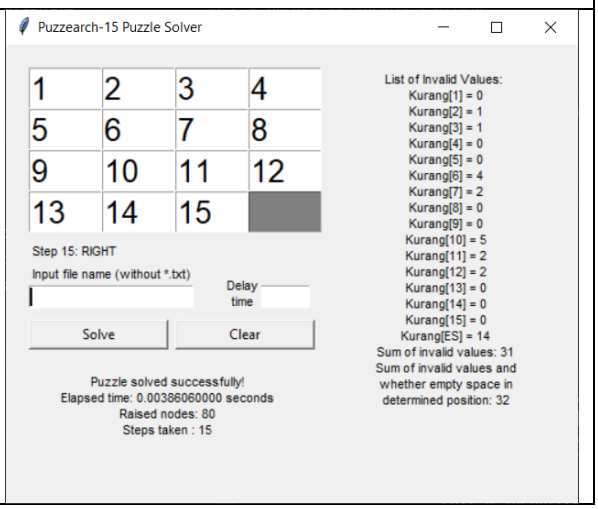
Hasil pengujian (gagal karena nilai Kurang(i) + X ganjil)

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

List of Invalid Values:
Kurang[1] = 0
Kurang[2] = 0
Kurang[3] = 2
Kurang[4] = 0
Kurang[5] = 0
Kurang[6] = 0
Kurang[7] = 0
Kurang[8] = 1
Kurang[9] = 0
Kurang[10] = 2
Kurang[11] = 1
Kurang[12] = 1
Kurang[13] = 0
Kurang[14] = 0
Kurang[15] = 0
Kurang[ES] = 11
Sum of invalid values: 18
Sum of invalid values and whether empty space in determined position: 19

This puzzle cannot be solved!
```

3.6 Penggunaan GUI

Pengujian GUI dengan file succ1.txt (awal)	Hasil GUI (akhir)
	
Pengujian GUI dengan input sendiri (mengikuti file succ2.txt)	Hasil GUI (akhir)
	

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	✓	

BAB IV. *REPOSITORY*

Repository dapat diakses via `https://github.com/clumsyyyy/Tucil3_13520124`