Laporan Tugas Kecil 3

IF2211 Strategi Algoritma

Penyelesaian Persoalan 15-Puzzle dengan Algoritma Branch and Bound

Diajukan sebagai salah satu tugas mata kuliah IF2211 Strategi Algoritma pada Semester II Tahun Akademik 2021-2022

Oleh

Malik Akbar Hashemi Rafsanjani 13520105

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2022

A. Cara Kerja Program *Branch and Bound* yang Dibuat

Misalkan diberikan suatu masukan berupa instansiasi persoalan 15-puzzle sebagai berikut.

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

Gambar I Ilustrasi Posisi Awal Persoalan 15-Puzzle

Mula-mula, program akan menentukan terlebih dahulu apakah persoalan 15-puzzle tersebut dapat diselesaikan atau tidak. Untuk menentukannya akan dilakukan perhitungan sesuai dengan teorema berikut. Puzzle dapat diselesaikan jika $\sum_{i=1}^{16} KURANG(i) + X$ bernilai genap.

KURANG(i) adalah banyaknya puzzle bernomor j sedemikian sehingga j < i dan POSISI(j) > POSISI(i). Nilai X adalah nilai dari kolom dan baris dari posisi ubin kosong (didefinisikan sebagai ubin bernomor 16) kemudian dimodulo dengan 2 (X = (BARIS(16) + KOLOM(16)) mod 2).

Berdasarkan masukan diatas, akan didapatkan informasi sebagai berikut.

Tabel I nilai KURANG(i) untuk masing-masing nomor ubin

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

Sehingga didapatkan,

$$\sum_{i=1}^{16} Kurang(i) + X = 15 + 1 = 16$$

Karena 16 adalah angka genap maka persoalan dapat diselesaikan.

Berikutnya, akan dilakukan pembangkitan pohon status pencarian dengan algoritma branch and bound. Pada program, akan digunakan struktur data priority queue untuk menyimpan simpul hidup. Prioritas antrian akan ditentukan oleh nilai cost. Cost simpul P pada 15-puzzle didefinisikan sebagai berikut.

$$\hat{c}(i) = \hat{f}(i) + \hat{g}(i)$$

$f(P)$ adalah panjang lintasan dari simpul akar ke P sedangkan $\hat{g}(P)$ adalah taksiran panjang lintasan terpendek dari P ke simpulan solusi pada upapohon yang akarnya P. Dengan kata lain, $f(P)$ adalah level dari simpul dan $\hat{g}(P)$ adalah jumlah ubin tidak kosong yang tidak terdapat pada susunan akhir.

Susunan akhir dari persoalan 15-puzzle adalah sebagai berikut.

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

Gambar 2 Ilustrasi Susunan Akhir Persoalan 15-puzzle

Pembangkitan simpul didasarkan oleh aksi yang dapat dilakukan pada puzzle. Terdapat 4 aksi, diantaranya:

1. up, ubin kosong ditukar posisinya dengan ubin di sebalah atasnya.
2. right, ubin kosong ditukar posisinya dengan ubin di sebelah kanannya.
3. down, ubin kosong ditukar posisinya dengan ubin di sebelah bawahnya.
4. left, ubin kosong ditukar posisinya dengan ubin di sebelah kirinya.

Adapun batasan aksi diberikan agar setiap langkah selalu valid sebagai berikut.

1. Jika ubin kosong terletak dibaris paling atas atau simpul induknya melakukan aksi down maka aksi up tidak dapat dilakukan.
2. Jika ubin kosong terletak dikolom paling kanan atau simpul induknya melakukan aksi left maka aksi right tidak dapat dilakukan.
3. Jika ubin kosong terletak dibaris paling bawah atau simpul induknya melakukan aksi up maka aksi down tidak dapat dilakukan.
4. Jika ubin kosong terletak dikolom paling kiri atau simpul induknya melakukan aksi right maka aksi left tidak dapat dilakukan.

Berdasarkan penjelasan diatas, setiap simpul akan menyimpan informasi sebagai berikut.

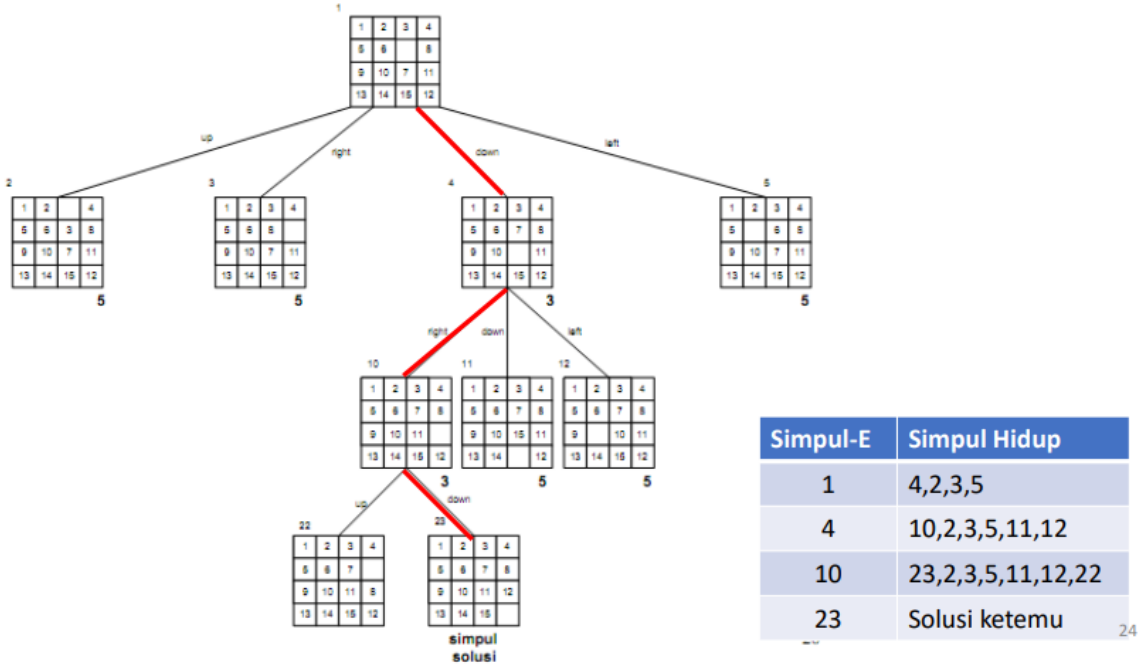
1. Simpul induknya

2. Status dari puzzle yang telah dibangkitkan pada simpul tersebut
3. Level dari simpul tersebut
4. Cost dari simpul tersebut

Selanjutnya akan dijelaskan algoritma branch and bound yang diimplementasikan sebagai berikut.

1. Simpul akar diinisiasi dengan menyimpan status puzzle awal. Simpul akar dimasukkan kedalam antrian simpul hidup.
2. Akan dilakukan perulangan pada antrian simpul hidup. Perulangan akan berhenti ketika antrian simpul hidup kosong atau simpul goal ditemukan.
3. Pada setiap perulangan, satu simpul di-dequeue dari antrian simpul hidup. Simpul tersebut menjadi simpul yang akan diekspansi (simpul ekspan). Pengambilan simpul ekspan diprioritaskan oleh nilai cost yang paling kecil.
4. Dilakukan iterasi sebanyak 4 kali pada simpul ekspan tersebut sesuai dengan aksi yang dapat dilakukan. Jika aksi yang diberikan pada status puzzle pada simpul ekspan tersebut valid, maka akan dibangkitkan simpul baru. Simpul baru tersebut akan menyimpan informasi simpul ekspan sebagai simpul induknya, status puzzle sesuai dengan aksi yang diberikan, level dari simpul (level simpul ekspan + 1), dan cost dari simpul tersebut.
5. Simpul goal ditemukan maka perulangan berhenti. Selanjutnya akan dipanggil fungsi yang mencetak langkah penyelesaian puzzle.

Ilustrasi pohon pencarian yang dibangkitkan adalah sebagai berikut.



Simpul-E	Simpul Hidup
1	4,2,3,5
4	10,2,3,5,11,12
10	23,2,3,5,11,12,22
23	Solusi ketemu

Gambar 3 Ilustrasi Pohon Status Pencarian (Sumber: Slide Algoritma Branch and Bound Bagian 1 Bahan Kuliah IF2211 Strategi Algoritma oleh Rinaldi Munir, Nur Ulfa Maulidevi, Masayu Leylia Khodra)

B. *Source Code* Program

Program ini terdiri atas 12 file berikut.

1. main.py
   File main.py bertanggung jawab sebagai program utama

```
from CLI import CLI
from GUI import GUI


def main():
    """Main program
    """

    print("What interface do you want to use?")
    print("1. GUI")
    print("2. CLI")

    inp = int(input("Input: "))
    while (True):
        if (inp >= 1 and inp <= 2):
            break
        else:
            print("Wrong format, please enter 1 or 2")

    if (inp == 1):
        GUI().mainloop()
    else:
        CLI()


if __name__ == "__main__":
    main()
```

2. GUI.py
   File yang memuat kelas GUI untuk menjalankan Aplikasi GUI. Kelas tersebut diturunkan dari kelas tkinter.Tk

```
from GameManager import GameManager
import tkinter as tk
from tkinter import *
from tkinter import filedialog as fd
```

```
from tkinter.messagebox import showinfo
import enum
from constant import *

class GUI(tk.Tk):
    """A class for running GUI App. Interited from tkinter.Tk

    Attributes
    ----------
    truncated
    """

    class _Status_Enum(enum.Enum):
        """Enum class for Program execution status
        Inherited from enum.Enum
        """

        SOLVED = "Solved"
        UNSOLVABLE = "Unsolvable"
        LOADING = "Loading"
        WAITING_INPUT = "Waiting Input"
        FINISHED = "Finished"

    def __init__(self):
        """Constructor for GUI class
        """

        super().__init__()
        self.title('Puzzle Solver')

        self._arr_btn = [Button(self, text=i+1, padx=40, pady=40,
                                state=DISABLED, width=10,
bg="#ffffff")
                         for i in range(LENGTH)]

        self._from_file_btn = Button(
            self, text="Load from File", padx=40, pady=10, width=10,
command=self._load_file)
        self._random_generated_btn = Button(
            self, text="Random Generated", padx=40, pady=10,
width=10, command=self._random_generated)
```

```
        self._from_file_btn.grid(row=0, column=0)
        self._random_generated_btn.grid(row=0, column=3)

        self._label_status = Label(
            self, text=f"Status:
{self._Status_Enum.WAITING_INPUT.value}")
        self._label_status.grid(row=1, column=0, columnspan=4)

        self._label_generated = Label(self, text=f"Generated Node: ")
        self._label_generated.grid(row=2, column=0, columnspan=4)
        self._label_time = Label(self, text=f"Time Lapse: ms")
        self._label_time.grid(row=3, column=0, columnspan=4)
        self._label_total_move = Label(self, text=f"Total Move: ")
        self._label_total_move.grid(row=4, column=0, columnspan=4)

        for i in range(LENGTH):
            self._arr_btn[i].grid(row=(i//4 + 5), column=(i % 4))

            num = int(self._arr_btn[i]["text"])
            if (num == HOLE):
                self._arr_btn[i].grid_remove()

        self._label_steps = Label(self, text=f"Step Taken: ")
        self._label_steps.grid(row=9, column=0, columnspan=4)
        self._label_misplaced = Label(self, text=f"Total misplaced
puzzle: ")
        self._label_misplaced.grid(row=10, column=0, columnspan=4)
        self._label_kurang = Label(self, text=f"Total kurang puzzle:
")

        self._arr_label_kurang = [
            Label(self, text=f"Kurang ke-{i+1}: ") for i in
range(LENGTH)]

        for i in range(LENGTH):
            self._arr_label_kurang[i].grid(
                row=(11 + i // 2), column=(0 if i % 2 == 0 else 2),
columnspan=2)

        self._label_sum_add_x = Label(self, text=f"Sum of kurang + x:
")
        self._label_sum_add_x.grid(row=20, column=0, columnspan=4)
```

```
        self._set_label(0, 0, 0)
        self._set_label_below(0, [0 for _ in range(LENGTH)], 0)

    def _set_label_below(self, misplaced, arr, sum_x):
        """Method to set label below the tiles

        Args:
            misplaced (int): how many misplaced tiles
            arr (list of int): list of `kurang` calculation per tiles
            sum_x (int): sum of kurang calculation + x (parity)
        """

        self._label_misplaced["text"] = f"Total misplaced puzzle:
{misplaced}"

        for i in range(LENGTH):
            self._arr_label_kurang[i]["text"] = f"Kurang ke-{i+1}:
{arr[i]}"

        self._label_sum_add_x["text"] = f"Sum of kurang + x: {sum_x}"

    def _load_file(self):
        """Method to get file puzzle and run program
        """

        self._label_status["text"] = f"Status:
{self._Status_Enum.LOADING.value}"
        self._random_generated_btn["state"] = DISABLED
        self._from_file_btn["state"] = DISABLED

        self._is_from_file = True
        filetypes = (("Text Files", "*.txt"),)
        self._filename = fd.askopenfilename(
            title='Select puzzle file',
            initialdir='test/',
            filetypes=filetypes)

        self.solve()

    def _random_generated(self):
        """Method to generate random puzzle and run program
```

```
        """

        self._label_status["text"] = f"Status:
{self._Status_Enum.LOADING.value}"
        self._is_from_file = False
        self._random_generated_btn["state"] = DISABLED
        self._from_file_btn["state"] = DISABLED
        self._filename = None
        showinfo(
            title='Warning Random Generated Puzzle',
            message="Random Generated Puzzle tends to be
computational intensive if solveable. This program will most likely
be unresponsive for a long time until puzzle is solved"
        )

        self.solve()

    def solve(self):
        """Main program control to solve the puzzle
        """

        self._gm = GameManager(self._is_from_file, self._filename)
        self.redisplay(self._gm.puzzle().board())

        (solvable, arr_kurang, sum, x, misplaced) =
self._gm.solvable_status()
        self._set_label_below(misplaced, arr_kurang, sum + x)

        if (not solvable):
            self._label_status["text"] = f"Status:
{self._Status_Enum.UNSOLVABLE.value}"
            showinfo(
                title='Unsolvable Puzzle',
                message="Your input puzzle is unsolvable"
            )
            self._label_steps["text"] = "Step Taken: "

            self._enable_btn()
            return

        (arr, count, time_lapse) = self._gm.solve()
```

```
        self._set_label(count, time_lapse, 0 if arr == None else
len(arr))

        self._label_status["text"] = f"Status:
{self._Status_Enum.SOLVED.value}"

        n = len(arr)
        steps = ""
        for i in range(n):
            steps += " " + str(arr[n-i-1].move())
            (lambda x=i: self.loop(n,
                                   x, arr[n-x-1].puzzle().board()))()

        self._label_steps["text"] = f"Step Taken: {steps}"

    def redisplay(self, arr):
        """Method to change tiles position based on current state of
Puzzle

        Args:
            arr (list of int): board of current Puzzle
        """

        for i in range(LENGTH):
            self._arr_btn[i].grid()
            self._arr_btn[i]['text'] = arr[i]

            num = int(arr[i])
            if (num == HOLE):
                self._arr_btn[i].grid_remove()

    def loop(self, n, x, arr):
        """Method to display change tiles based on Node path to solve
the puzzle

        Args:
            n (int): how many step to solve the puzzle
            x (int): index of current path
            arr (list of int): state of current puzzle board
        """

        self.after(SLEEP_TIME * x, lambda: self.redisplay(arr))
```

```
        if (x+1 == n):
            self.after(SLEEP_TIME * x, lambda: self._finish())

    def _enable_btn(self):
        """Method to enable button random generate and load file
        """

        self._random_generated_btn["state"] = "active"
        self._from_file_btn["state"] = "active"

    def _finish(self):
        """Method that to be called when program is finish displaying
the changing tiles
        """

        self._enable_btn()
        self._label_status["text"] = f"Status:
{self._Status_Enum.FINISHED.value}"

    def _set_label(self, node, time_lapse, move):
        """Method to set upper labels text

        Args:
            node (int): How many nodes are generated
            time_lapse (int): How long the program took to solve the
puzzle
            move (int): total move to solve the puzzle
        """

        self._label_generated["text"] = f"Generated Node: {node}"
        self._label_time["text"] = f"Time Lapse: {time_lapse}ms"
        self._label_total_move["text"] = f"Total Move: {move}"
```

3. CLI.py

   File yang memuat kelas CLI untuk menjalankan Aplikasi CLI.

```
from GameManager import GameManager
from constant import LENGTH

class CLI:
    """A class for running CLI App
```

```
    Attributes
    ----------
    _gm : GameManager
        object that controlls the game
    """

    def __init__(self):
        """Constructor for CLI class that also run the program at
once
        """

        print("PUZZLE SOLVER")
        print("Please select input mode for puzzle")
        print("1. From file")
        print("2. Random generator")

        inp = int(input("Input: "))
        while (True):
            if (inp >= 1 and inp <= 2):
                break
            else:
                print("Wrong format, please enter 1 or 2")

        is_from_file = inp == 1
        if (is_from_file):
            print("Please enter file name:")
            file_name = input("File name: ")
            self._gm = GameManager(is_from_file, file_name)
        else:
            self._gm = GameManager(is_from_file)

        (solvable, arr_kurang, sum, x, misplaced) =
self._gm.solvable_status()

        self._gm.puzzle().describe()
        print("Misplaced tiles:", misplaced)
        for i in range(LENGTH):
            print(f"Kurang ke-{i+1}: {arr_kurang[i]}")

        print(f"Sum kurang + x: {sum} + {x} = {sum + x}")
```

```
        if (not solvable):
            print("Puzzle is unsolvable")
            return

        print("Puzzle is solvable")

        (arr, count, time_lapse) = self._gm.solve()

        n = len(arr)
        for i in range(n):
            arr[n-i-1].describe()
            print("<><><><><><><><><><>")

        print("GENERATED:", count)
        print("TIME LAPSE:", time_lapse, "ms")
```

4. GameManager.py
   File yang memuat kelas GameManager yang berperan sebagai kelas yang mengkontrol perilaku dari eksekusi program

```
from FileManager import FileManager
from Puzzle import Puzzle
from PrioQueue import PrioQueue
from Node import Node
from Timer import Timer
from PuzzleGenerator import PuzzleGenerator
from VisitedPuzzle import VisitedPuzzle


class GameManager():
    """A class that control behaviour of program execution

    Attributes
    ----------
    _pz : Puzzle
        Puzzle input, either from file or PuzzleGenerator
    """

    def __init__(self, is_from_file, path=None):
        """Constructor for GameManager class

        Args:
```

```
            is_from_file (bool): flag whether puzzle input is from
file
            path (string | None, optional): Path of puzzle input
file. Defaults to None.
        """

        if (is_from_file):
            self._pz = Puzzle(FileManager(path).arr())
        else:
            self._pz = Puzzle(PuzzleGenerator().generate())

        self._pz.count_misplaced_tiles()
        self._pz.find_empty()

    def puzzle(self):
        """Getter for Puzzle attribute

        Returns:
            Puzzle: puzzle input that will be solved
        """

        return self._pz

    def solvable_status(self):
        """Method that return status of solvability of the puzzle

        Returns:
            (boolean, list of int, int, int, int): status of
solvability of the puzzle
        """

        return self._pz.solvable_status()

    def solve(self):
        """Main control program to solve the puzzle

        Returns:
            (list of Node, int, int): Snapshot of Node path to solve
the puzzle,
                how many Nodes are generated, time lapse taken to
solve the puzzle
        """
```

```
        t = Timer()
        sol_node = None
        pq = PrioQueue()
        vp = VisitedPuzzle()

        n = Node(self._pz)
        pq.push(n)
        vp.insert(n)
        count = 1

        while (not pq.empty()):
            cur_node = pq.pop()

            if (cur_node.is_solution()):
                sol_node = cur_node
                break

            children = cur_node.generate_children()
            for child in children:
                if (not vp.contain(child)):
                    vp.insert(child)
                    pq.push(child)
                    count += 1

        time_lapse = t.stop()

        arr = sol_node.path_solving()

        return (arr, count, time_lapse)
```

5. Node.py

   File yang memuat kelas Node yang berperan sebagai kelas *wrapper* untuk menyimpan informasi *state* dari puzzle, kedalaman, induk Node, dan gerakan yang dilakukan dalam instantiasi Node tersebut.

```
class Node:
    """A wrapper class for store information about
    state of puzzle, depth, parent, and move that is used

    Attributes
    ----------
```

```
    _moves_units (static) : list of tuple
        list of operator move that will be used to generated new
Puzzle
    _moves_names (static) : list of string
        label for _moves_units
    _puzzle : Puzzle
        Puzzle object that store state of puzzle
    _parent : Node
        Node object that store parent of current Node
    _depth : int
        depth of current Node
    _move : string
        label for operator move that is used to generate current Node
    """

    _moves_units = [(0, 1), (1, 0), (0, -1), (-1, 0)]
    _moves_names = ["Right", "Down", "Left", "Up"]

    def __init__(self, puzzle, parent=None, depth=0, move=None):
        """Constructor for Node class

        Args:
            puzzle (Puzzle): Puzzle object that store state of puzzle
            parent (Node, optional): Node object that store parent of
                current Node. Defaults to None.
            depth (int, optional): depth of current Node. Defaults to
0.
            move (string, optional): label for operator move that is
                used to generate current Node. Defaults to None.
        """

        self._puzzle = puzzle
        self._parent = parent
        self._depth = depth
        self._move = move

    def path_solving(self):
        """Generate path from current Node to root Node

        Returns:
            list of Node: path from current Node to root Node
        """
```

```
        temp = self
        arr = []

        while (True):
            if (temp._move == None):
                return arr

            arr.append(temp)
            temp = temp._parent

    def depth(self):
        """Getter for depth attribute

        Returns:
            int: depth of current Node
        """
        return self._depth

    def puzzle(self):
        """Getter for puzzle attribute

        Returns:
            Puzzle: state puzzle of current Node
        """
        return self._puzzle

    def is_solution(self):
        """Check if current Node is solution

        Returns:
            boolean: flag whether current Node is solution
        """

        return self._puzzle.weight() == 0

    def weight(self):
        """Getter for weight attribute

        Returns:
            int: weight of current Node
        """
```

```
        return self._depth + self._puzzle.weight()

    def generate_children(self):
        """Generate children Node of current Node.
        This method generate if move is possible
        and not opposite move to parent

        Returns:
            list of Node: children of current Node
        """

        arr = []

        for i in range(len(Node._moves_units)):
            opp_move = Node._moves_names[(i+2) % 4]
            if (opp_move != self._move):
                (d_row, d_col) = Node._moves_units[i]
                new_puzzle, success = self._puzzle.move(d_row, d_col)
                if (success):
                    arr.append(
                        Node(new_puzzle, self, self._depth+1,
Node._moves_names[i]))

        return arr

    def describe(self):
        """Describe the current Node as weight, depth, move,
        and puzzle state
        """

        print("weight:", self.weight())
        print("depth:", self._depth)
        print("move:", self._move)
        print("puzzle:")
        self._puzzle.describe()

    def __lt__(self, next):
        """Operator overloading for less than operator based on
        weight of Node. If weight is same, compare depth

        Args:
            next (Node): Right hand side of less than operation
```

```
        Returns:
            boolean: flag whether current Node is less than next Node
        """

        if self.weight() == next.weight():
            return self.depth() >= next.depth()

        return self.weight() < next.weight()

    def move(self):
        """Getter for move attribute

        Returns:
            string: move attribute of current Node
        """

        return self._move
```

6. Puzzle.py
   File yang memuat kelas Puzzle yang berperan sebagai kelas yang menyimpan informasi *state* dari puzzle

```
from constant import HOLE, SIZE, LENGTH


class Puzzle:
    """A class to store puzzle state

    Attributes:
    ----------------
    _board: list of int
        1D list of puzzle state
    _weight: int
        number of misplaced tiles
    _idx_empty: int
        index of empty tile
    """

    def __init__(self, arr, weight=-1, idx_empty=-1):
        """Constructor for Puzzle class
```

```
        Args:
            arr (list of int): 1D list of puzzle state
            weight (int, optional): number of misplaced tiles.
                Defaults to -1.
            idx_empty (int, optional): index of empty tile.
                Defaults to -1.
        """

        self._board = arr
        self._weight = weight
        self._idx_empty = idx_empty

    def find_empty(self):
        """Find the index of empty tile and store it in class
attribute
        """

        for i in range(len(self._board)):
            if self._board[i] == HOLE:
                self._idx_empty = i

    def _get_row_col(self, idx):
        """Get row and column of a tile based on given index

        Args:
            idx (int): given index to calculate row and column

        Returns:
            (int, int): tuple of row and column
        """

        return (idx // SIZE, idx % SIZE)

    def solvable_status(self):
        """Calculate `kurang` per tiles and return status solvability
of the puzzle

        Returns:
            (boolean, list of int, int, int, int): status of
solvability of the puzzle
        """
```

```
        (row, col) = self._get_row_col(self._idx_empty)
        x = (row+col) % 2

        arr = [-1 for _ in range(LENGTH)]

        sum = 0
        for i in range(len(self._board)):
            arr[self._board[i] - 1] = self._kurang(i)
            sum += arr[self._board[i] - 1]

        return ((sum + x) % 2 == 0, arr, sum, x, self._weight)

    def _kurang(self, i):
        """Helper method to calculate inversion of given index

        Args:
            i (int): given index to calculate inversion

        Returns:
            int: inversion of given index
        """

        count = 0
        for idx in range(i, len(self._board)):
            if (self._board[idx] < self._board[i]):
                count += 1

        return count

    def move(self, d_row, d_col):
        """Generate new puzzle by moving empty tile if possible

        Args:
            d_row (int): change of row
            d_col (int): change of col

        Returns:
            (Puzzle | None, boolean): tuple of new Puzzle
                and boolean flag whether move is success
        """
```

```
        (row, col) = self._get_row_col(self._idx_empty)

        if (row+d_row < 0 or row+d_row >= SIZE or col+d_col < 0 or
col+d_col >= SIZE):
            return None, False

        idx1 = row*SIZE + col
        idx2 = (row+d_row)*SIZE + (col+d_col)

        new_weight = self._weight
        if (self._board[idx2] == idx1 + 1):
            new_weight -= 1
        elif (self._board[idx2] == idx2 + 1):
            new_weight += 1

        new_puzzle = Puzzle(self._copy_board(), new_weight, idx2)
        new_puzzle._board[idx1], new_puzzle._board[idx2] =
new_puzzle._board[idx2], new_puzzle._board[idx1]

        return new_puzzle, True

    def describe(self):
        """Describe current puzzle state
        """

        for i in range(SIZE):
            for j in range(SIZE):
                cur = self._board[i*SIZE + j]
                print("#" if cur == HOLE else cur, end=" ")
            print()

    def _copy_board(self):
        """Copy current board and return it

        Returns:
            1D list of int: copy of current board
        """
        return [x for x in self._board]

    def count_misplaced_tiles(self):
        """Count misplaced tiles in current puzzle.
        To be called on first Puzzle that being instantiated
```

```
        """

        count = 0
        for i in range(LENGTH):
            if (self._board[i] != HOLE and self._board[i] != i+1):
                count += 1

        self._weight = count

    def weight(self):
        """Getter for weight attribute

        Returns:
            int: weight of current puzzle
        """

        return self._weight

    def board(self):
        """Getter for board attribute

        Returns:
            1D list of int: board of current puzzle
        """

        return self._board
```

7. PrioQueue.py
   File yang memuat kelas PrioQueue yang berperan sebagai kelas yang menyimpan Node yang akan dicek dalam suatu *priority queue*.

```
import heapq


class PrioQueue:
    """A class for store Nodes that will be checked as priority queue

    Attributes:
    ---------------
    _queue: heap of Node
        Heap data structure for storing Nodes to fasten
        insert and delete first operation
```

```
    """

    def __init__(self):
        """Constructor for PrioQueue class
        """

        self._queue = []
        heapq.heapify(self._queue)
        self._size = 0

    def empty(self):
        """Check if current PrioQueue is empty

        Returns:
            boolean: flag whether current PrioQueue is empty
        """

        return self._size == 0

    def push(self, item):
        """Insert new item into heap

        Args:
            item (Node): new item that will be inserted
                into priority queue
        """
        heapq.heappush(self._queue, item)
        self._size += 1

    def pop(self):
        """Remove first item from priority queue and return it

        Returns:
            Node: first item in priority queue
        """

        self._size -= 1
        return heapq.heappop(self._queue)

    def size(self):
        """Getter for size attribute
```

```
        Returns:
            int: size of current PrioQueue
        """

        return self._size

    def describe(self):
        """Descibe all Nodes that being stored in
        current PrioQueue (unordered)
        """

        for node in self._queue:
            node.describe()
            print("================")
```

8. VisitedPuzzle.py
   File yang memuat kelas VisitedPuzzle yang berperan sebagai kelas yang menyimpan puzzle mana saja yang telah dikunjungi agar puzzle yang telah dikunjungi tidak perlu dicek kembali.

```
class VisitedPuzzle:
    """A class for storing puzzles that have been visited

    Attributes:
    ----------------
    _map: set of tuple of int
        set data structure to store the puzzle that has been visited
    """

    def __init__(self):
        """Constructor for VisitedPuzzle class
        """

        self._map = set()

    def contain(self, item):
        """Check whether given Node's puzzle board is in visited set

        Args:
            item (Node): Node that being checked

        Returns:
```

```
            boolean: flag whether puzzle of Node is in visited set
        """

        return tuple(item.puzzle().board()) in self._map

    def insert(self, item):
        """Insert puzzle board of Node into visited set

        Args:
            item (Node): Node that being inserted into visited set
        """

        self._map.add(tuple(item.puzzle().board()))
```

9. PuzzleGenerator.py
   File yang memuat kelas PuzzleGenerator yang berperan sebagai kelas untuk membangkitkan puzzle board baru secara acak.

```
from numpy import random
import numpy as np
from constant import SIZE


class PuzzleGenerator:
    """A class for generating new puzzle board
    """

    def generate(self):
        """Generate new puzzle board randomly

        Returns:
            1D list of int: random puzzle board
        """

        arr = np.array(range(1, SIZE*SIZE+1))
        random.shuffle(arr)
        return list(arr)
```

10. Timer.py
    File yang memuat kelas Timer yang berperan sebagai kelas untuk menghitung waktu eksekusi program

```
from time import time


class Timer:
    """A class for measuring time lapse

    Attributes:
    ----------------
    _timer: int
        time of current Timer instantiated
    """

    def __init__(self):
        """Constructor for Timer class
        """
        self._timer = int(time() * 1000)

    def stop(self):
        """Return time lapse of current Timer instantiated

        Returns:
            int: time lapse of current Timer instantiated
        """

        return int(time() * 1000) - self._timer
```

11. FileManager.py

    File yang memuat kelas FileManager yang berperan sebagai kelas untuk membaca masukan puzzle dari file.

```
from constant import SIZE, LENGTH


class FileManager:
    """A class for reading input puzzle from file

    Attributes:
    ----------------
    _matrix: list of list of int
        A 2D matrix to store the puzzle that being read from file
    """
```

```
    def __init__(self, path):
        """Constructor for FileManager class

        Args:
            path (string): path of file that will be read from
        """

        self._matrix = [[-1 for _ in range(SIZE)] for _ in
range(SIZE)]

        f = open(path, 'r')
        for i, line in enumerate(f):
            arr = line.split()
            for j in range(SIZE):
                self._matrix[i][j] = int(arr[j])

    def _flatten(self, matrix):
        """Convert 2D matrix to 1D list

        Args:
            matrix (list of list of int): matrix that will be
converted to 1D list

        Returns:
            list of int: flattened matrix
        """

        arr = [-1 for _ in range(LENGTH)]

        for i in range(SIZE):
            for j in range(SIZE):
                arr[i*SIZE + j] = matrix[i][j]

        return arr

    def arr(self):
        """Getter for puzzle that being read by FileManager as 1D
list

        Returns:
            list of int: puzzle file reading result
        """
```

```
        return self._flatten(self._matrix)
```

12. constant.py

    File yang memuat nilai-nilai konfigurasi konstan yang digunakan oleh file-file lainnya

```
"""A file to store constant values that being used in other files
"""

SIZE = 4
HOLE = SIZE**2
LENGTH = SIZE**2
SLEEP_TIME = 1000
```

C. Screenshot Input dan Output

Untuk menguji kebenaran program, diberikan 5 buah instantiasi persoalan 15-puzzle, dengan 3 kasus dapat diselesaikan dan 2 kasus tidak dapat diselesaikan.

1. Persoalan Kasus dapat Diselesaikan 1

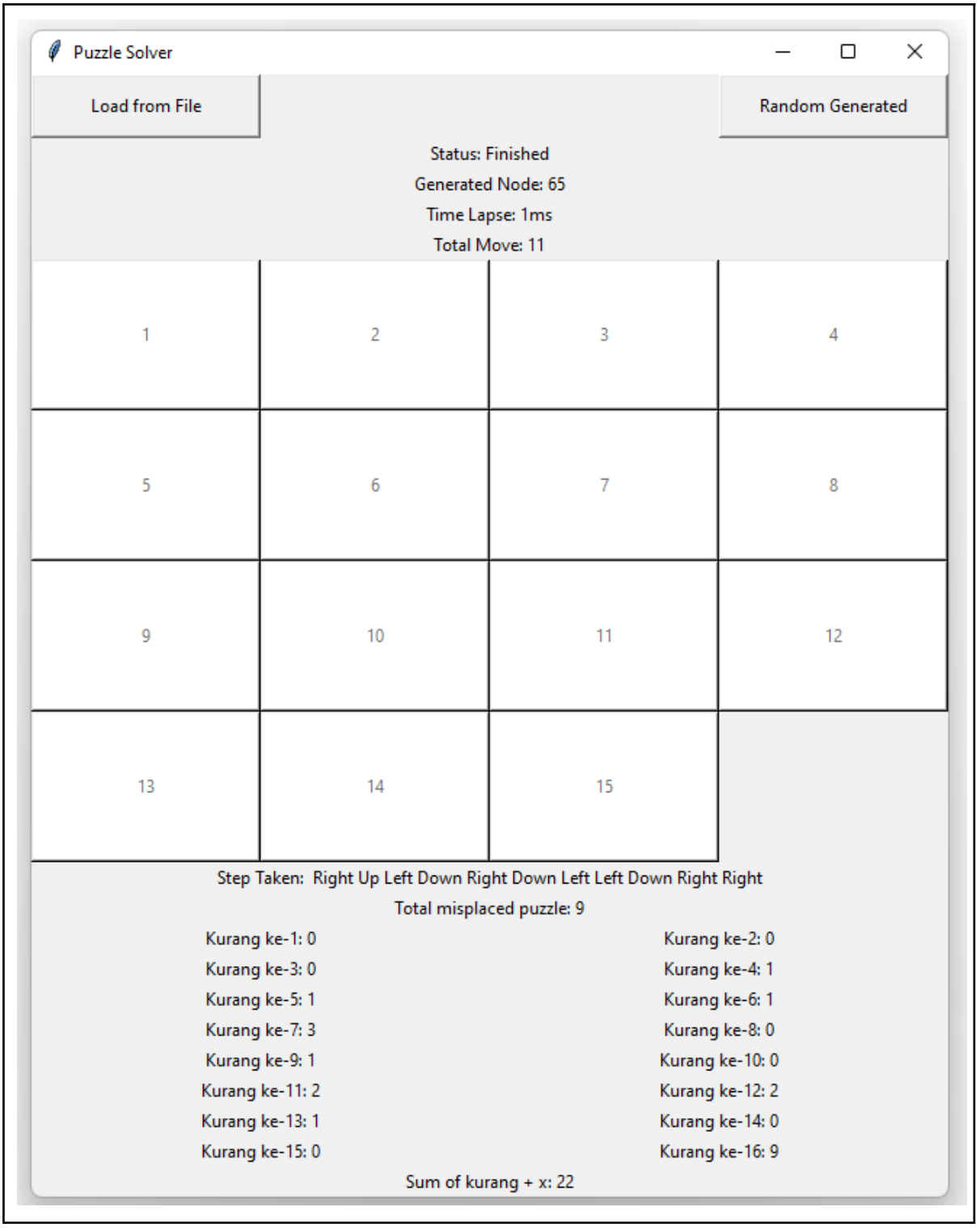
Input file: solveable-1.txt

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

Jawaban:

```
D:\KULIAH-AKBAR\SEMESTER-4\STIMA\tucil3\mirror_subm
itted>py src/main.py
What interface do you want to use?
1. GUI
2. CLI
Input: 2
PUZZLE SOLVER
Please select input mode for puzzle
1. From file
2. Random generator
Input: 1
Please enter file name:
File name: test/solveable-1.txt
1 2 4 7
5 6 # 3
9 11 12 8
13 10 14 15
Misplaced tiles: 9
Kurang ke-1: 0
Kurang ke-2: 0
Kurang ke-3: 0
Kurang ke-4: 1
Kurang ke-5: 1
Kurang ke-6: 1
Kurang ke-7: 3
Kurang ke-8: 0
Kurang ke-9: 1
Kurang ke-10: 0
Kurang ke-11: 2
Kurang ke-12: 2
Kurang ke-13: 1
Kurang ke-14: 0
Kurang ke-15: 0
Kurang ke-16: 9
Sum kurang + x: 21 + 1 = 22
Puzzle is solvable
weight: 10
depth: 1
move: Right
puzzle:
1 2 4 7
5 6 3 #
9 11 12 8
13 10 14 15
<><><><><><><><><>
weight: 11
depth: 2
move: Up
puzzle:
1 2 4 #
5 6 3 7
9 11 12 8
13 10 14 15
<><><><><><><><><>
<><><><><><><><><>
weight: 11
depth: 3
move: Left
puzzle:
1 2 # 4
5 6 3 7
9 11 12 8
13 10 14 15
<><><><><><><><><>
weight: 11
depth: 4
move: Down
puzzle:
1 2 3 4
5 6 # 7
9 11 12 8
13 10 14 15
<><><><><><><><><>
weight: 11
depth: 5
move: Right
puzzle:
1 2 3 4
5 6 7 #
9 11 12 8
13 10 14 15
<><><><><><><><><>
weight: 11
depth: 6
move: Down
puzzle:
1 2 3 4
5 6 7 8
9 11 12 #
13 10 14 15
<><><><><><><><><>
weight: 11
depth: 7
move: Left
puzzle:
1 2 3 4
5 6 7 8
9 11 # 12
13 10 14 15
<><><><><><><><><>
weight: 11
depth: 8
move: Left
puzzle:
1 2 3 4
5 6 7 8
9 # 11 12
13 10 14 15
```

```
<><><><><><><><><>
weight: 11
depth: 9
move: Down
puzzle:
1 2 3 4
5 6 7 8
9 10 11 12
13 # 14 15
<><><><><><><><><>
weight: 11
depth: 10
move: Right
puzzle:
1 2 3 4
5 6 7 8
9 10 11 12
13 14 # 15
<><><><><><><><><>
weight: 11
depth: 11
move: Right
puzzle:
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 #
<><><><><><><><><>
GENERATED: 65
TIME LAPSE: 1 ms
```

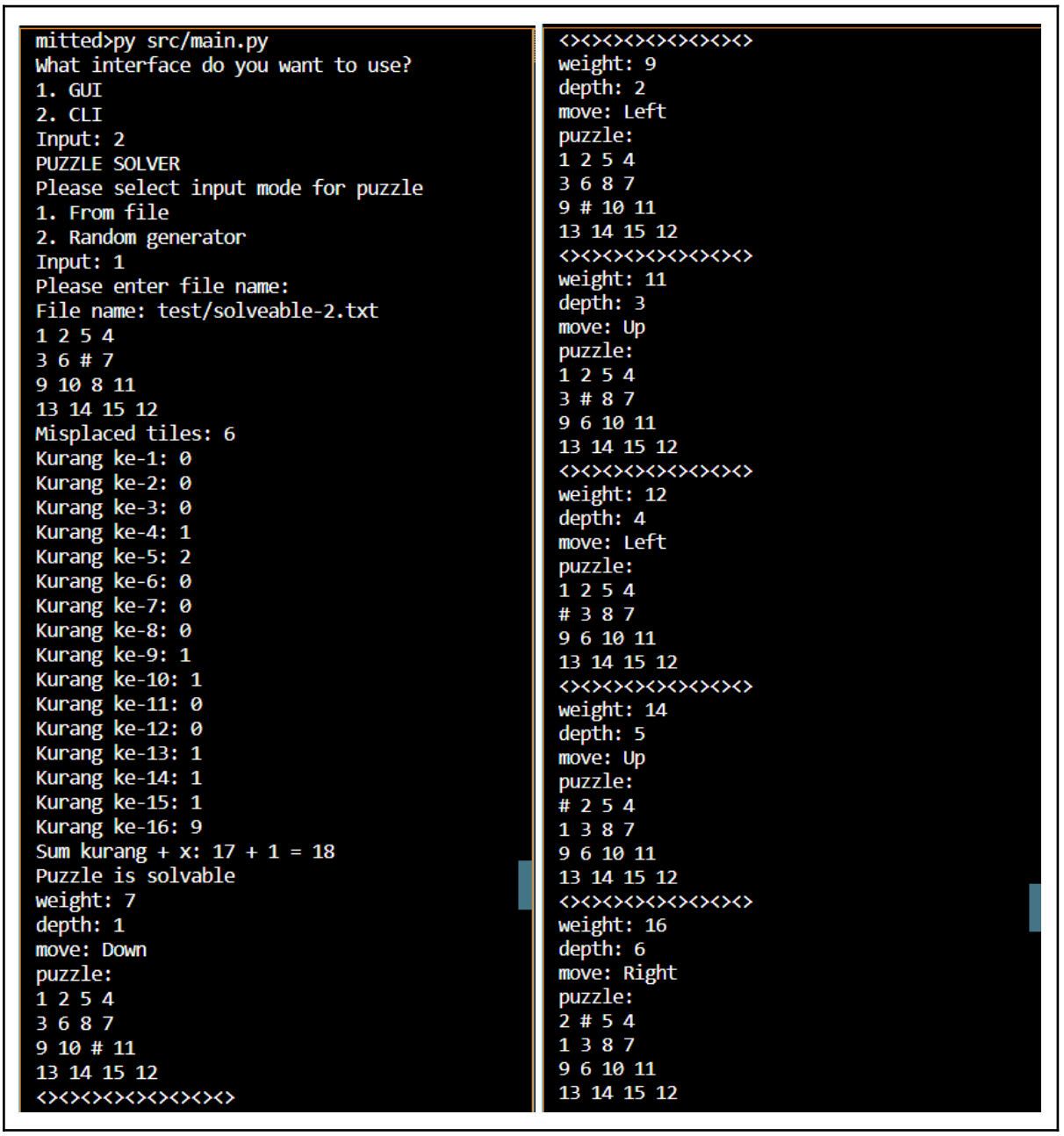


2. Persoalan Kasus dapat Diselesaikan 2

Input file: solveable-2.txt

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

Jawaban:

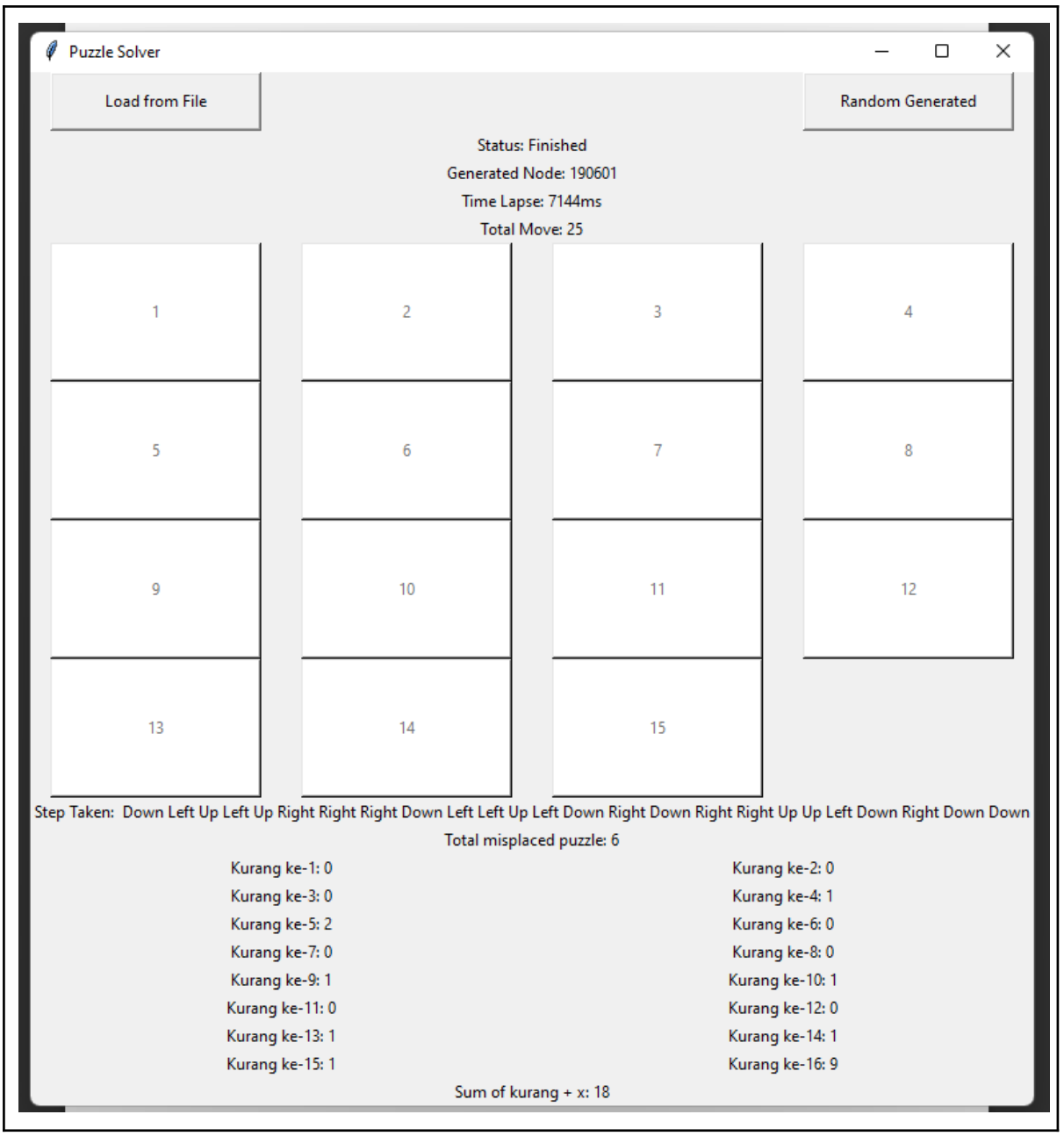

mitted>py src/main.py
What interface do you want to use?
1. GUI
2. CLI
Input: 2
PUZZLE SOLVER
Please select input mode for puzzle
1. From file
2. Random generator
Input: 1
Please enter file name:
File name: test/solveable-2.txt
1 2 5 4
3 6 # 7
9 10 8 11
13 14 15 12
Misplaced tiles: 6
Kurang ke-1: 0
Kurang ke-2: 0
Kurang ke-3: 0
Kurang ke-4: 1
Kurang ke-5: 2
Kurang ke-6: 0
Kurang ke-7: 0
Kurang ke-8: 0
Kurang ke-9: 1
Kurang ke-10: 1
Kurang ke-11: 0
Kurang ke-12: 0
Kurang ke-13: 1
Kurang ke-14: 1
Kurang ke-15: 1
Kurang ke-16: 9
Sum kurang + x: 17 + 1 = 18
Puzzle is solvable
weight: 7
depth: 1
move: Down
puzzle:
1 2 5 4
3 6 8 7
9 10 # 11
13 14 15 12
<><><><><><><><>
<><><><><><><><>
weight: 9
depth: 2
move: Left
puzzle:
1 2 5 4
3 6 8 7
9 # 10 11
13 14 15 12
<><><><><><><><>
weight: 11
depth: 3
move: Up
puzzle:
1 2 5 4
3 # 8 7
9 6 10 11
13 14 15 12
<><><><><><><><>
weight: 12
depth: 4
move: Left
puzzle:
1 2 5 4
3 8 7
9 6 10 11
13 14 15 12
<><><><><><><><>
weight: 14
depth: 5
move: Up
puzzle:
2 5 4
1 3 8 7
9 6 10 11
13 14 15 12
<><><><><><><><>
weight: 16
depth: 6
move: Right
puzzle:
2 # 5 4
1 3 8 7
9 6 10 11
13 14 15 12

```
<><><><><><><><><>
weight: 17
depth: 7
move: Right
puzzle:
2 5 # 4
1 3 8 7
9 6 10 11
13 14 15 12
<><><><><><><><><>
weight: 19
depth: 8
move: Right
puzzle:
2 5 4 #
1 3 8 7
9 6 10 11
13 14 15 12
<><><><><><><><><>
weight: 20
depth: 9
move: Down
puzzle:
2 5 4 7
1 3 8 #
9 6 10 11
13 14 15 12
<><><><><><><><><>
weight: 20
depth: 10
move: Left
puzzle:
2 5 4 7
1 3 # 8
9 6 10 11
13 14 15 12
<><><><><><><><><>
weight: 21
depth: 11
move: Left
puzzle:
2 5 4 7
1 # 3 8
9 6 10 11
13 14 15 12
```

```
<><><><><><><><><>
weight: 22
depth: 12
move: Up
puzzle:
2 # 4 7
1 5 3 8
9 6 10 11
13 14 15 12
<><><><><><><><><>
weight: 22
depth: 13
move: Left
puzzle:
# 2 4 7
1 5 3 8
9 6 10 11
13 14 15 12
<><><><><><><><><>
weight: 22
depth: 14
move: Down
puzzle:
1 2 4 7
# 5 3 8
9 6 10 11
13 14 15 12
<><><><><><><><><>
weight: 22
depth: 15
move: Right
puzzle:
1 2 4 7
5 # 3 8
9 6 10 11
13 14 15 12
<><><><><><><><><>
weight: 22
depth: 16
move: Down
puzzle:
1 2 4 7
5 6 3 8
9 # 10 11
13 14 15 12
```

```
weight: 22
depth: 17
move: Right
puzzle:
1 2 4 7
5 6 3 8
9 10 # 11
13 14 15 12
<><><><><><><><>
weight: 22
depth: 18
move: Right
puzzle:
1 2 4 7
5 6 3 8
9 10 11 #
13 14 15 12
<><><><><><><><>
weight: 24
depth: 19
move: Up
puzzle:
1 2 4 7
5 6 3 #
9 10 11 8
13 14 15 12
<><><><><><><><>
weight: 25
depth: 20
move: Up
puzzle:
1 2 4 #
5 6 3 7
9 10 11 8
13 14 15 12
<><><><><><><><>
weight: 25
depth: 21
move: Left
puzzle:
1 2 # 4
5 6 3 7
9 10 11 8
13 14 15 12
<><><><><><><><>
```

```
puzzle:
1 2 # 4
5 6 3 7
9 10 11 8
13 14 15 12
<><><><><><><><>
weight: 25
depth: 22
move: Down
puzzle:
1 2 3 4
5 6 # 7
9 10 11 8
13 14 15 12
<><><><><><><><>
weight: 25
depth: 23
move: Right
puzzle:
1 2 3 4
5 6 7 #
9 10 11 8
13 14 15 12
<><><><><><><><>
weight: 25
depth: 24
move: Down
puzzle:
1 2 3 4
5 6 7 8
9 10 11 #
13 14 15 12
<><><><><><><><>
weight: 25
depth: 25
move: Down
puzzle:
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 #
<><><><><><><><>
GENERATED: 190601
TIME LAPSE: 5638 ms
```



3. Persoalan Kasus dapat Diselesaikan 3

Input file: solveable-3.txt

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

Jawaban:

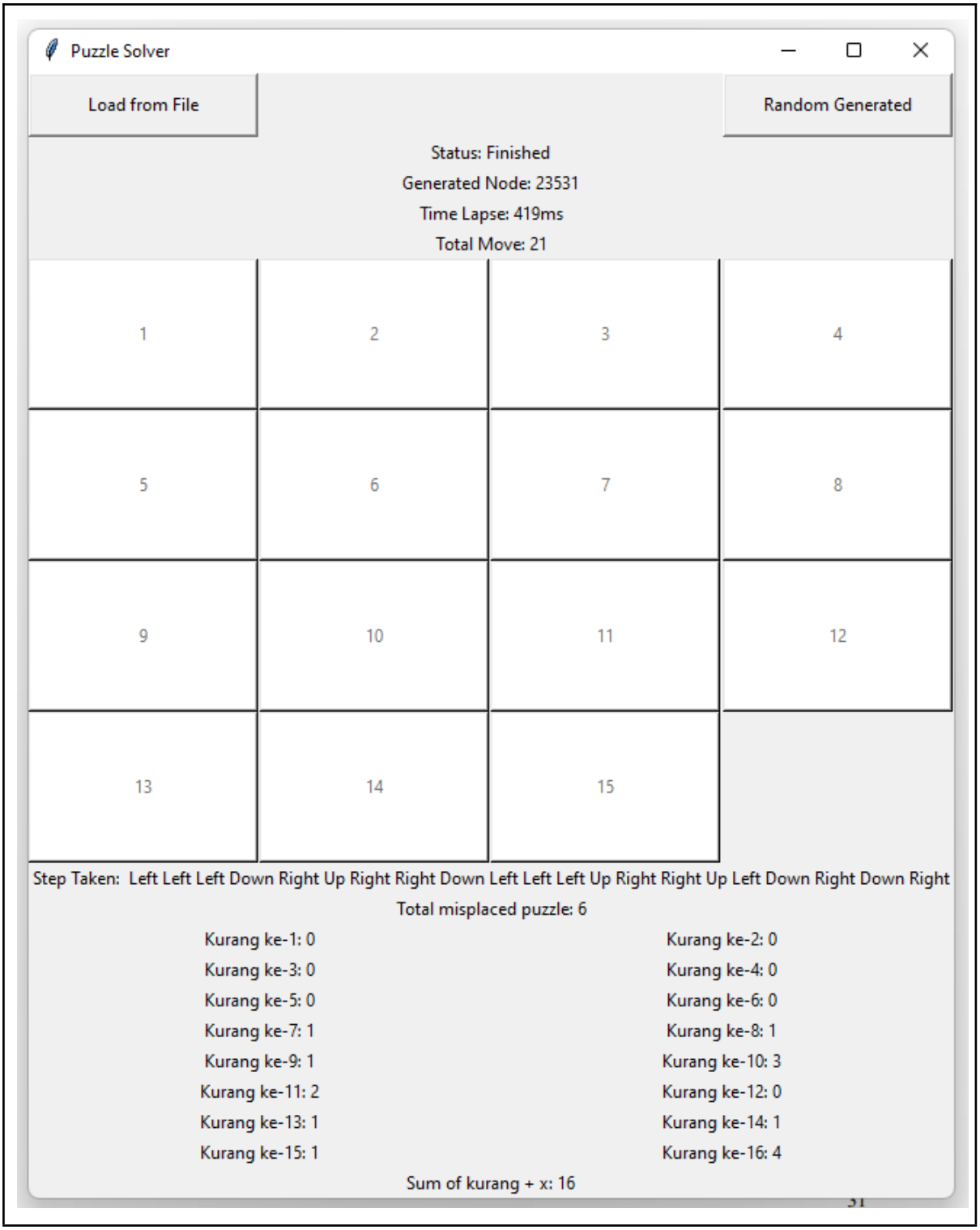
```
D:\KULIAH-AKBAR\SEMESTER-4\STIMA\tucil3\mir
ror_submitted>py src/main.py
What interface do you want to use?
1. GUI
2. CLI
Input: 2
PUZZLE SOLVER
Please select input mode for puzzle
1. From file
2. Random generator
Input: 1
Please enter file name:
File name: test/solveable-3.txt
1 2 3 4
5 7 10 8
11 9 6 #
13 14 15 12
Misplaced tiles: 6
Kurang ke-1: 0
Kurang ke-2: 0
Kurang ke-3: 0
Kurang ke-4: 0
Kurang ke-5: 0
Kurang ke-6: 0
Kurang ke-7: 1
Kurang ke-8: 1
Kurang ke-9: 1
Kurang ke-10: 3
Kurang ke-11: 2
Kurang ke-12: 0
Kurang ke-13: 1
Kurang ke-14: 1
Kurang ke-15: 1
Kurang ke-16: 4
Sum kurang + x: 15 + 1 = 16
Puzzle is solvable
weight: 7
depth: 1
move: Left
puzzle:
1 2 3 4
5 7 10 8
11 9 # 6
13 14 15 12
<><><><><><><><><>
<><><><><><><><><>
weight: 8
depth: 2
move: Left
puzzle:
1 2 3 4
5 7 10 8
11 # 9 6
13 14 15 12
<><><><><><><><><>
weight: 9
depth: 3
move: Left
puzzle:
1 2 3 4
5 7 10 8
# 11 9 6
13 14 15 12
<><><><><><><><><>
weight: 11
depth: 4
move: Down
puzzle:
1 2 3 4
5 7 10 8
13 11 9 6
# 14 15 12
<><><><><><><><><>
weight: 13
depth: 5
move: Right
puzzle:
1 2 3 4
5 7 10 8
13 11 9 6
14 # 15 12
<><><><><><><><><>
weight: 14
depth: 6
move: Up
puzzle:
1 2 3 4
5 7 10 8
13 # 9 6
14 11 15 12
```

```
<><><><><><><><>
weight: 15
depth: 7
move: Right
puzzle:
1 2 3 4
5 7 10 8
13 9 # 6
14 11 15 12
<><><><><><><><>
weight: 16
depth: 8
move: Right
puzzle:
1 2 3 4
5 7 10 8
13 9 6 #
14 11 15 12
<><><><><><><><>
weight: 16
depth: 9
move: Down
puzzle:
1 2 3 4
5 7 10 8
13 9 6 12
14 11 15 #
<><><><><><><><>
weight: 18
depth: 10
move: Left
puzzle:
1 2 3 4
5 7 10 8
13 9 6 12
14 11 # 15
<><><><><><><><>
weight: 19
depth: 11
move: Left
puzzle:
1 2 3 4
5 7 10 8
13 9 6 12
14 # 11 15
```

```
weight: 19
depth: 12
move: Left
puzzle:
1 2 3 4
5 7 10 8
13 9 6 12
# 14 11 15
<><><><><><><><>
weight: 19
depth: 13
move: Up
puzzle:
1 2 3 4
5 7 10 8
# 9 6 12
13 14 11 15
<><><><><><><><>
weight: 19
depth: 14
move: Right
puzzle:
1 2 3 4
5 7 10 8
9 # 6 12
13 14 11 15
<><><><><><><><>
weight: 20
depth: 15
move: Right
puzzle:
1 2 3 4
5 7 10 8
9 6 # 12
13 14 11 15
<><><><><><><><>
weight: 21
depth: 16
move: Up
puzzle:
1 2 3 4
5 7 # 8
9 6 10 12
13 14 11 15
<><><><><><><><>
```

```
<><><><><><><><><>
weight: 21
depth: 17
move: Left
puzzle:
1 2 3 4
5 # 7 8
9 6 10 12
13 14 11 15
<><><><><><><><><>
weight: 21
depth: 18
move: Down
puzzle:
1 2 3 4
5 6 7 8
9 # 10 12
13 14 11 15
<><><><><><><><><>
weight: 21
depth: 19
move: Right
puzzle:
1 2 3 4
5 6 7 8
9 10 # 12
13 14 11 15
```

```
<><><><><><><><><>
weight: 21
depth: 20
move: Down
puzzle:
1 2 3 4
5 6 7 8
9 10 11 12
13 14 # 15
<><><><><><><><><>
weight: 21
depth: 21
move: Right
puzzle:
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 #
<><><><><><><><><>
GENERATED: 23531
TIME LAPSE: 720 ms
```



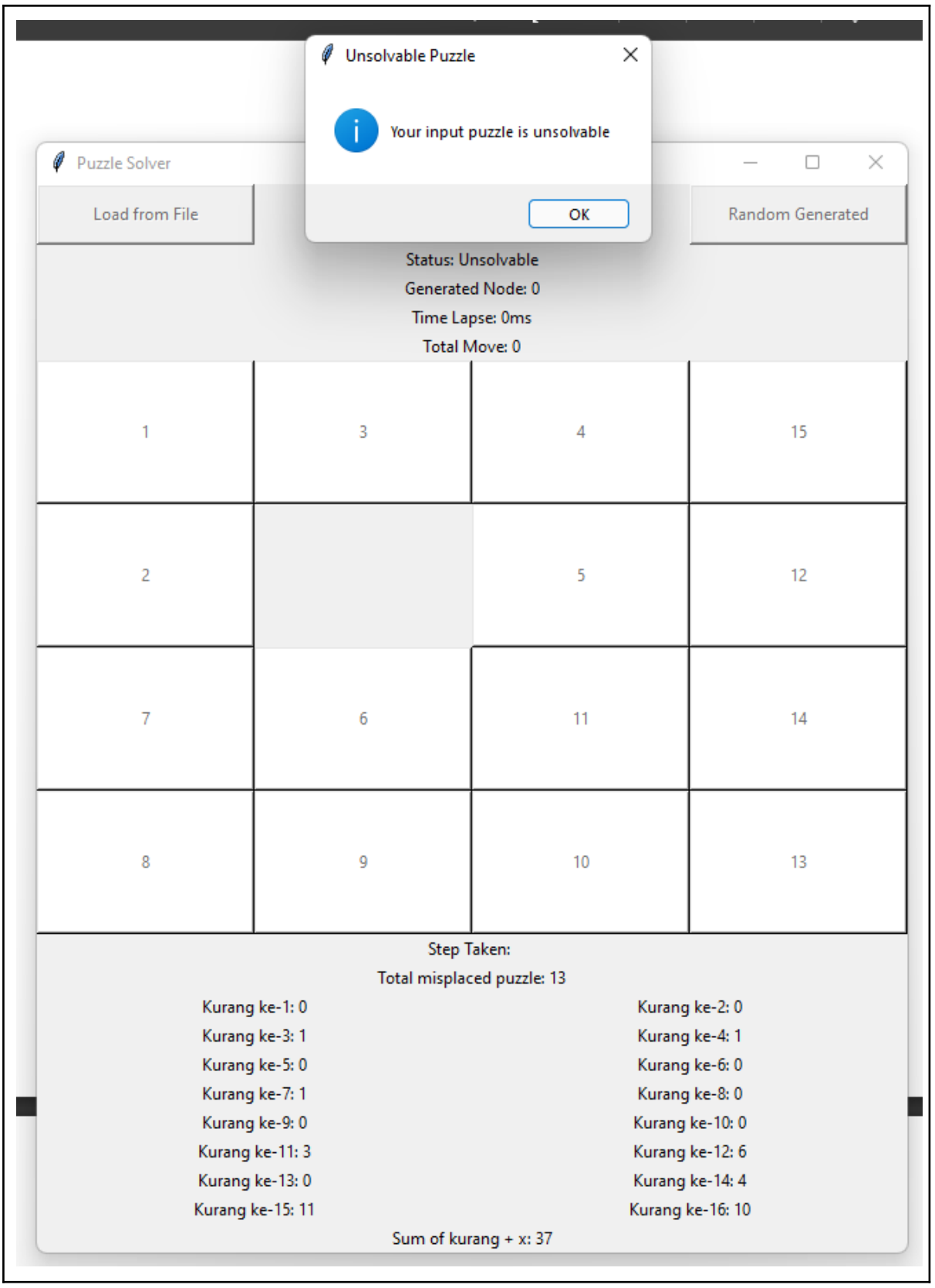
4. Persoalan Kasus tidak dapat Diselesaikan 1

Input file: unsolveable-1.txt

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

Jawaban:

```
D:\KULIAH-AKBAR\SEMESTER-4\STIMA\tucil3\mir
ror_submitted>py src/main.py
What interface do you want to use?
1. GUI
2. CLI
Input: 2
PUZZLE SOLVER
Please select input mode for puzzle
1. From file
2. Random generator
Input: 1
Please enter file name:
File name: test/unsolveable-1.txt
1 3 4 15
2 # 5 12
7 6 11 14
8 9 10 13
Misplaced tiles: 13
Kurang ke-1: 0
Kurang ke-2: 0
Kurang ke-3: 1
Kurang ke-4: 1
Kurang ke-5: 0
Kurang ke-6: 0
Kurang ke-7: 1
Kurang ke-8: 0
Kurang ke-9: 0
Kurang ke-10: 0
Kurang ke-11: 3
Kurang ke-12: 6
Kurang ke-13: 0
Kurang ke-14: 4
Kurang ke-15: 11
Kurang ke-16: 10
Sum kurang + x: 37 + 0 = 37
Puzzle is unsolvable
```



5. Persoalan Kasus tidak dapat Diselesaikan 2

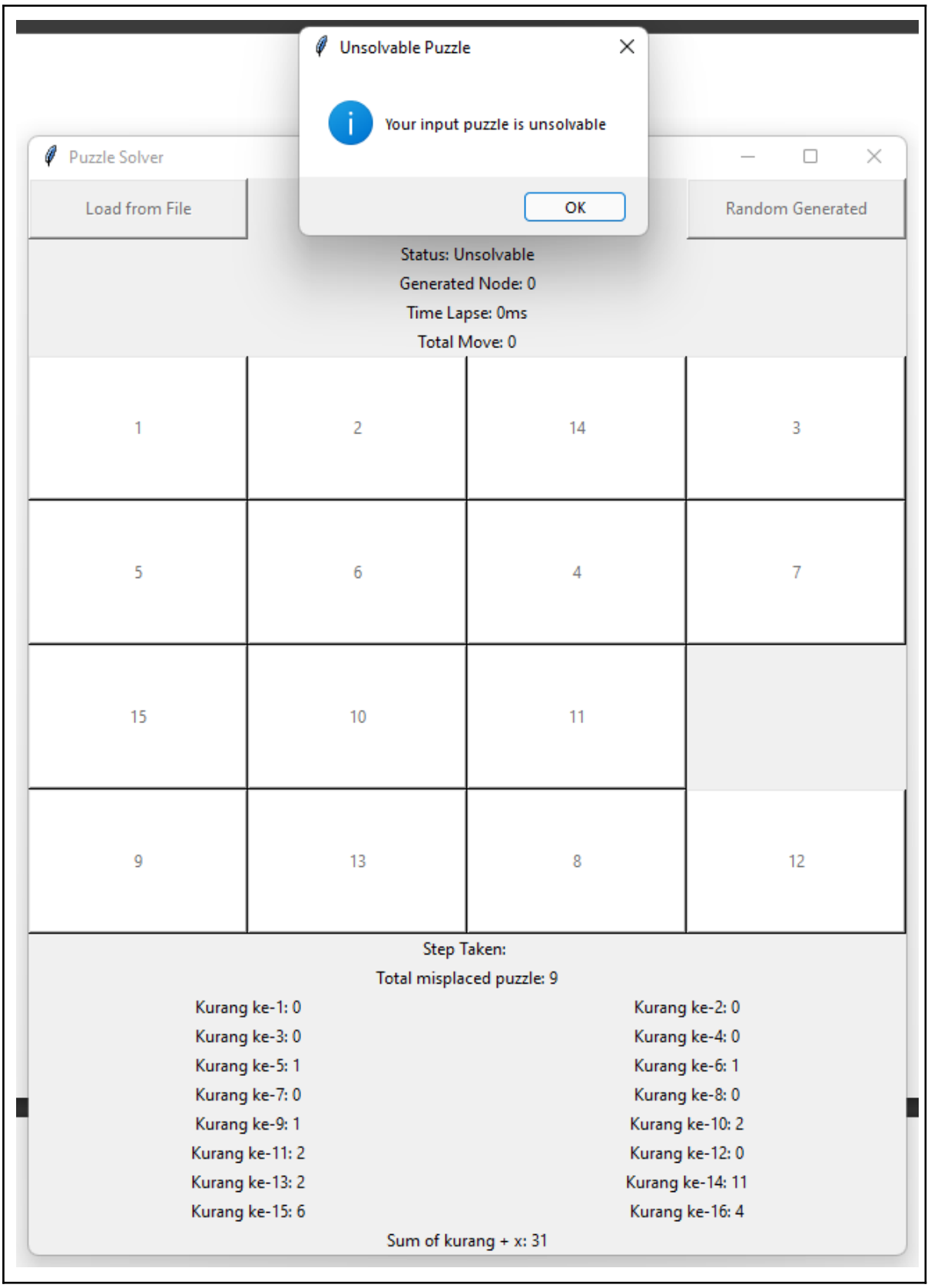
Input file: unsolveable-2.txt

```
1 2 14 3
```

```
5 6 4 7
15 10 11 16
9 13 8 12
```

Jawaban:

```
D:\KULIAH-AKBAR\SEMESTER-4\STIMA\tucil3\mir
ror_submitted>py src/main.py
What interface do you want to use?
1. GUI
2. CLI
Input: 2
PUZZLE SOLVER
Please select input mode for puzzle
1. From file
2. Random generator
Input: 1
Please enter file name:
File name: test/unsolveable-2.txt
1 2 14 3
5 6 4 7
15 10 11 #
9 13 8 12
Misplaced tiles: 9
Kurang ke-1: 0
Kurang ke-2: 0
Kurang ke-3: 0
Kurang ke-4: 0
Kurang ke-5: 1
Kurang ke-6: 1
Kurang ke-7: 0
Kurang ke-8: 0
Kurang ke-9: 1
Kurang ke-10: 2
Kurang ke-11: 2
Kurang ke-12: 0
Kurang ke-13: 2
Kurang ke-14: 11
Kurang ke-15: 6
Kurang ke-16: 4
Sum kurang + x: 30 + 1 = 31
Puzzle is unsolvable
```



D. Alamat Repository Kode Program

Github: https://github.com/malikrafsan/15-Puzzle-Solver

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	✓	