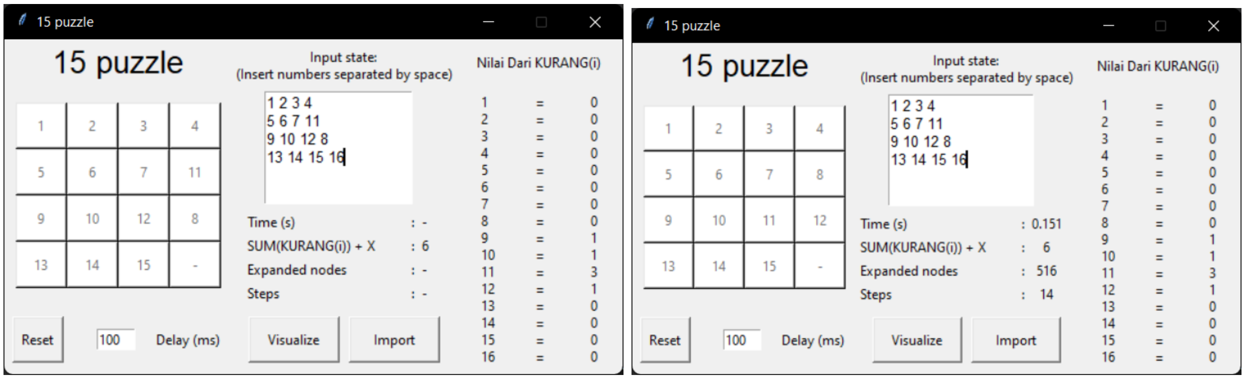
Tugas Kecil 3 IF2211 “15 Puzzle”

Rifqi Naufal Abdjul
13520062 / K02

Tabel Checklist

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	✅	

Contoh Input dan Output Program



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

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

```
Time (s)             : -
SUM(KURANG(i)) + X   : 6
Expanded nodes       : -
Steps                : -
```

Delay (ms): 100

```
1   =   0
2   =   0
3   =   0
4   =   0
5   =   0
6   =   0
7   =   0
8   =   0
9   =   1
10  =   1
11  =   3
12  =   1
13  =   0
14  =   0
15  =   0
16  =   0
```

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

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

```
Time (s)             : 0.151
SUM(KURANG(i)) + X   :   6
Expanded nodes       :   516
Steps                :   14
```

Delay (ms): 100

```
1   =   0
2   =   0
3   =   0
4   =   0
5   =   0
6   =   0
7   =   0
8   =   0
9   =   1
10  =   1
11  =   3
12  =   1
13  =   0
14  =   0
15  =   0
16  =   0
```

Gambar awal program (kiri), setelah visualisasi (kanan)

Algoritma Branch and Bound

Algoritma yang digunakan dalam tugas ini adalah algoritma branch and bound yang mirip dengan algoritma BFS (breadth first search) tetapi dengan menggunakan priority queue (dalam implementasi ini menggunakan heap) sebagai urutan pengecekan dibanding queue biasa pada BFS.

Priority pada priority queue ini, ditentukan dari cost dari suatu state yang dihitung dari jarak state dari state awal ditambahkan dengan estimasi jarak state ke state tujuan (heuristik). Perhitungan ini digambarkan pada persamaan di bawah ini.

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

$\hat{c}(i)$ = ongkos untuk simpul i

$\hat{f}(i)$ = ongkos mencapai simpul i dari akar

$\hat{g}(i)$ = ongkos mencapai simpul tujuan dari simpul i.

Langkah dari algoritma yang diimplementasi adalah sebagai berikut:

1. Inisialisasi queue dengan akar (state awal) dari permasalahan
2. Dequeue queue dan cek apakah sudah dilewati atau belum, jika queue kosong, lanjutkan ke langkah 8
3. Jika sudah, kembali ke langkah 2
4. Jika belum, cek apakah state merupakan state tujuan
5. Jika merupakan state tujuan, buang seluruh state yang mempunyai cost lebih dari state tersebut, lalu kembali ke langkah 2
6. Jika bukan merupakan state tujuan, bangkitkan state dan masukkan seluruh anak yang mempunyai cost lebih kecil dari solusi yang disimpan ke dalam queue
7. Jika queue belum kosong, kembali ke langkah 2
8. Solusi optimal merupakan solusi yang disimpan terakhir kali
9. Untuk mendapatkan rute penyelesaian, kumpulkan seluruh parent lalu putar balik urutannya.

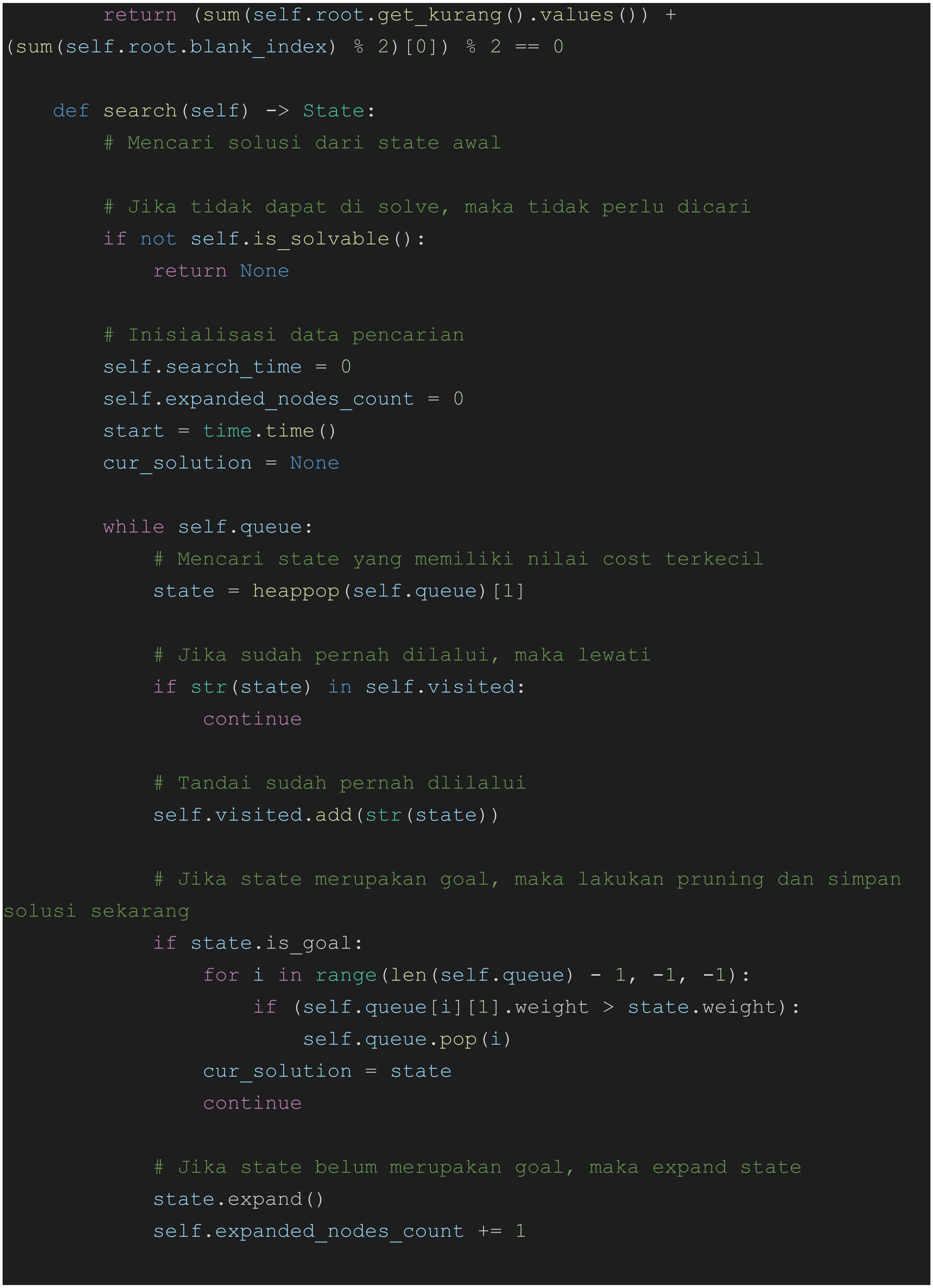
Kode Program

Kelas State

```
class State:
    def __init__(self, board: np.ndarray, parent: 'State' = None,
last_dir: tuple[int, int] = (0, 0)):
        # Inisialisasi state awal
        self.board = board
        self.parent = parent
        self.blank_index = np.where(self.board == 16)
        self.is_expanded = False
        self.children = []
        self.last_dir = last_dir

        # Mendapatkan kedalaman state dari state awal
        self.depth = 0 if parent is None else parent.depth + 1

        # Mendapatkan nilai cost dari state
        self.weight = self._calculate_weight()
        # Apakah state merupakan goal
        if self.weight == self.depth:
            self.is_goal = True
        else:
            self.is_goal = False

    def get_kurang(self) -> dict[int, int]:
        kurang = {}
        for i in range(4):
            for j in range(4):
                for key in kurang:
                    if key > self.board[i][j]:
                        kurang[key] += 1
                kurang[self.board[i][j]] = 0
        return kurang

    def _calculate_weight(self) -> int:
        # Menghitung nilai cost dari state
        heuristic = 0
        for i in range(4):
            for j in range(4):
                if self.board[i][j] != 4*i + j + 1 and self.board[i][j] != 16:
                    heuristic += 1
        return heuristic + self.depth

    def expand(self) -> None:
        # Expand state untuk mendapatkan semua state yang dapat dijalankan dari state
        if (self.is_expanded):
            # Jika state sudah di expand, maka tidak perlu di expand lagi
            return
        self.is_expanded = True

        # Seluruh arah yang memungkinkan selain arah sebelum
        directions = {(1, 0), (0, 1), (-1, 0), (0, -1)} - \
            {(self.last_dir[0] * -1, self.last_dir[1] * -1)}

        children = []
        for direction in directions:
            # Jika posisi blank tidak out of bounds jika ditambah dengan
jarak, lewati
            if (self.blank_index[0] + direction[0]) < 0 or
(self.blank_index[0] + direction[0]) > 3 or (self.blank_index[1] +
direction[1]) < 0 or (self.blank_index[1] + direction[1]) > 3:
                continue
            new_board = self.board.copy()
            new_board[self.blank_index] = new_board[(
                self.blank_index[0] + direction[0], self.blank_index[1] +
direction[1])]
            new_board[(self.blank_index[0] + direction[0],
                       self.blank_index[1] + direction[1])] = 16
            children.append(State(new_board, self, direction))
        self.children = children

    def __lt__(self, other):
        # Untuk mengurutkan state berdasarkan nilai cost
        return self.weight < other.weight

    def __str__(self):
        # Untuk menampilkan state
        return str(self.board)
```

Kelas BNB Tree

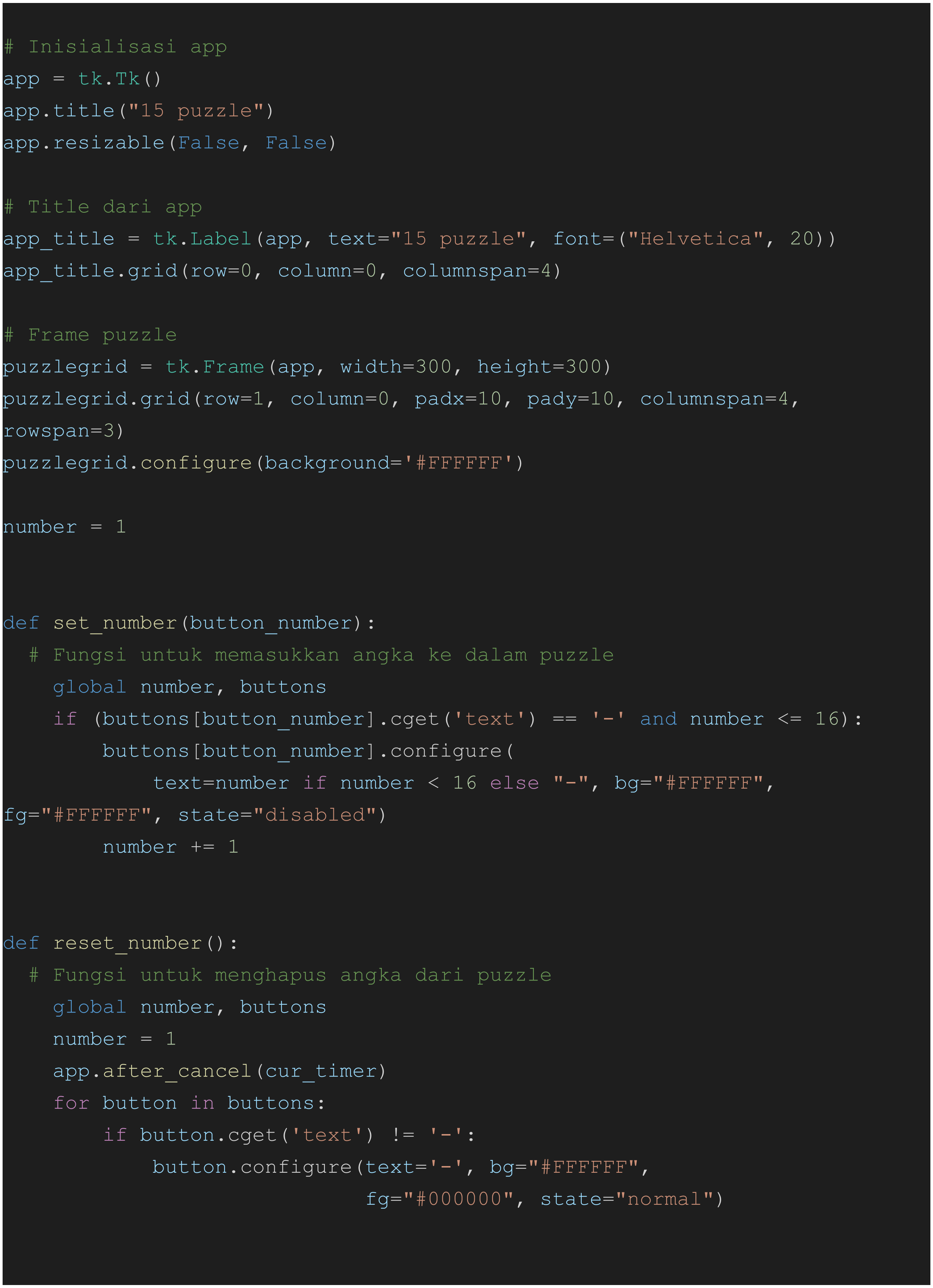
```
class BNBTree:
    def __init__(self, root: State):
        # Inisialisasi state pohon awal
        self.root = root
        self.queue = []
        heappush(self.queue, (0, root))
        self.visited = set()
        self.solution = None
        self.search_time = None
        self.expanded_nodes_count = None
        self.route = []

    def is_solvable(self) -> bool:
```



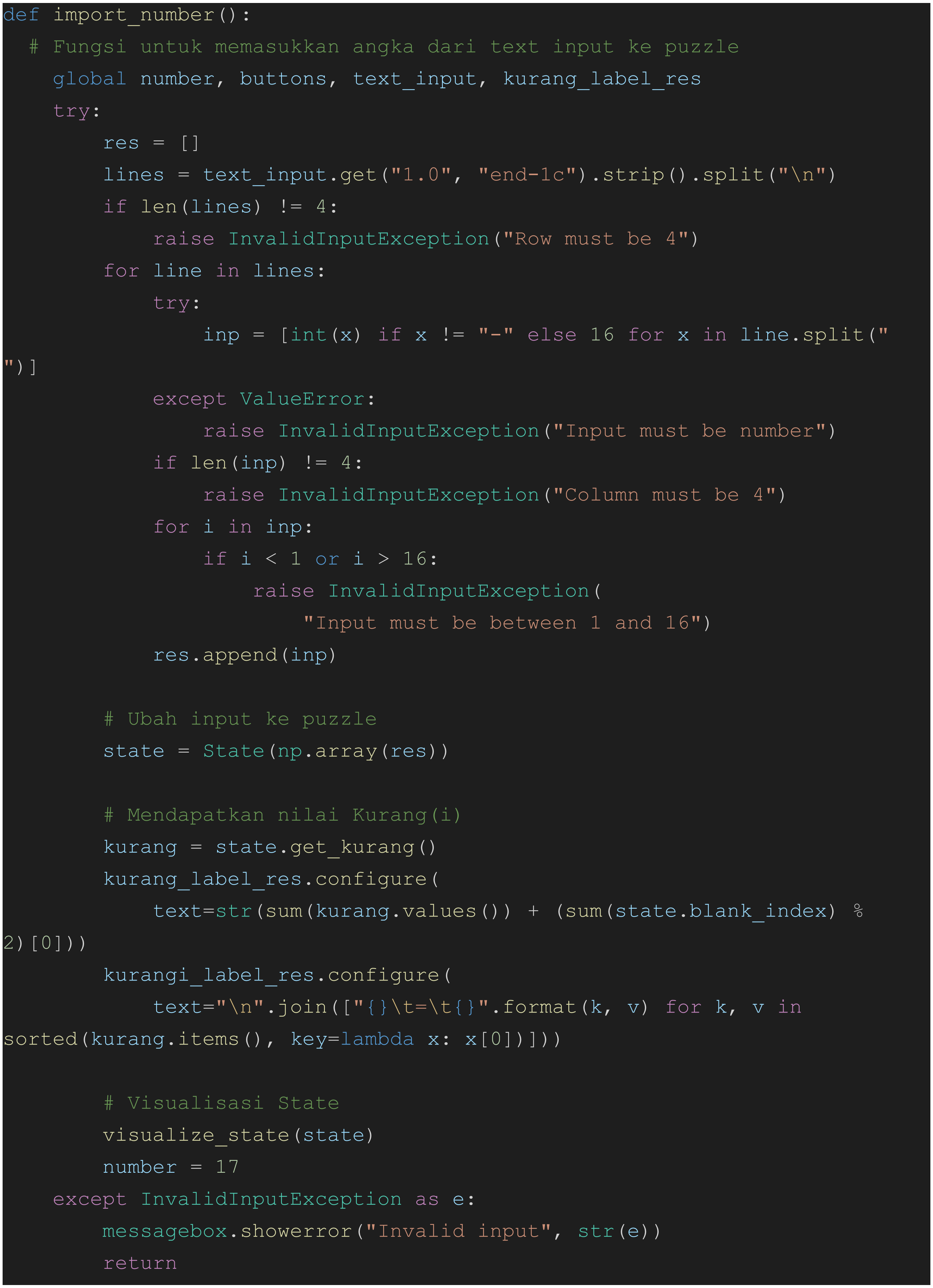
```
        return (sum(self.root.get_kurang().values()) +
(sum(self.root.blank_index) % 2)[0]) % 2 == 0

    def search(self) -> State:
        # Mencari solusi dari state awal

        # Jika tidak dapat di solve, maka tidak perlu dicari
        if not self.is_solvable():
            return None

        # Inisialisasi data pencarian
        self.search_time = 0
        self.expanded_nodes_count = 0
        start = time.time()
        cur_solution = None

        while self.queue:
            # Mencari state yang memiliki nilai cost terkecil
            state = heappop(self.queue)[1]

            # Jika sudah pernah dilalui, maka lewati
            if str(state) in self.visited:
                continue

            # Tandai sudah pernah dlilalui
            self.visited.add(str(state))

            # Jika state merupakan goal, maka lakukan pruning dan simpan
solusi sekarang
            if state.is_goal:
                for i in range(len(self.queue) - 1, -1, -1):
                    if (self.queue[i][1].weight > state.weight):
                        self.queue.pop(i)
                cur_solution = state
                continue

            # Jika state belum merupakan goal, maka expand state
            state.expand()
            self.expanded_nodes_count += 1
            # Masukkan semua anak state yang sudah di expand ke dalam queue
            for child in state.children:
                if (str(child) not in self.visited) and (child.weight < cur_solution.weight if cur_solution is not None else True):
                    heappush(self.queue, (child.weight, child))

        # Selesaikan pencarian
        self.search_time = time.time() - start
        state = cur_solution
        self.solution = state

        # Mendapatkan jalur dari solusi
        self.route = []
        self.route.append(self.solution)
        while state.parent is not None:
            self.route.append(state.parent)
            state = state.parent
        self.route.append(state)

        # Mengembalikan solusi
        return self.solution
```

GUI

```
import tkinter as tk
from tkinter import messagebox
from puzzle15 import State, BNBTree
import numpy as np

# Kelas Exception untuk mempermudah validasi input


class InvalidInputException(Exception):
    def __init__(self, message):
        self.message = message

    def __str__(self):
        return self.message

```

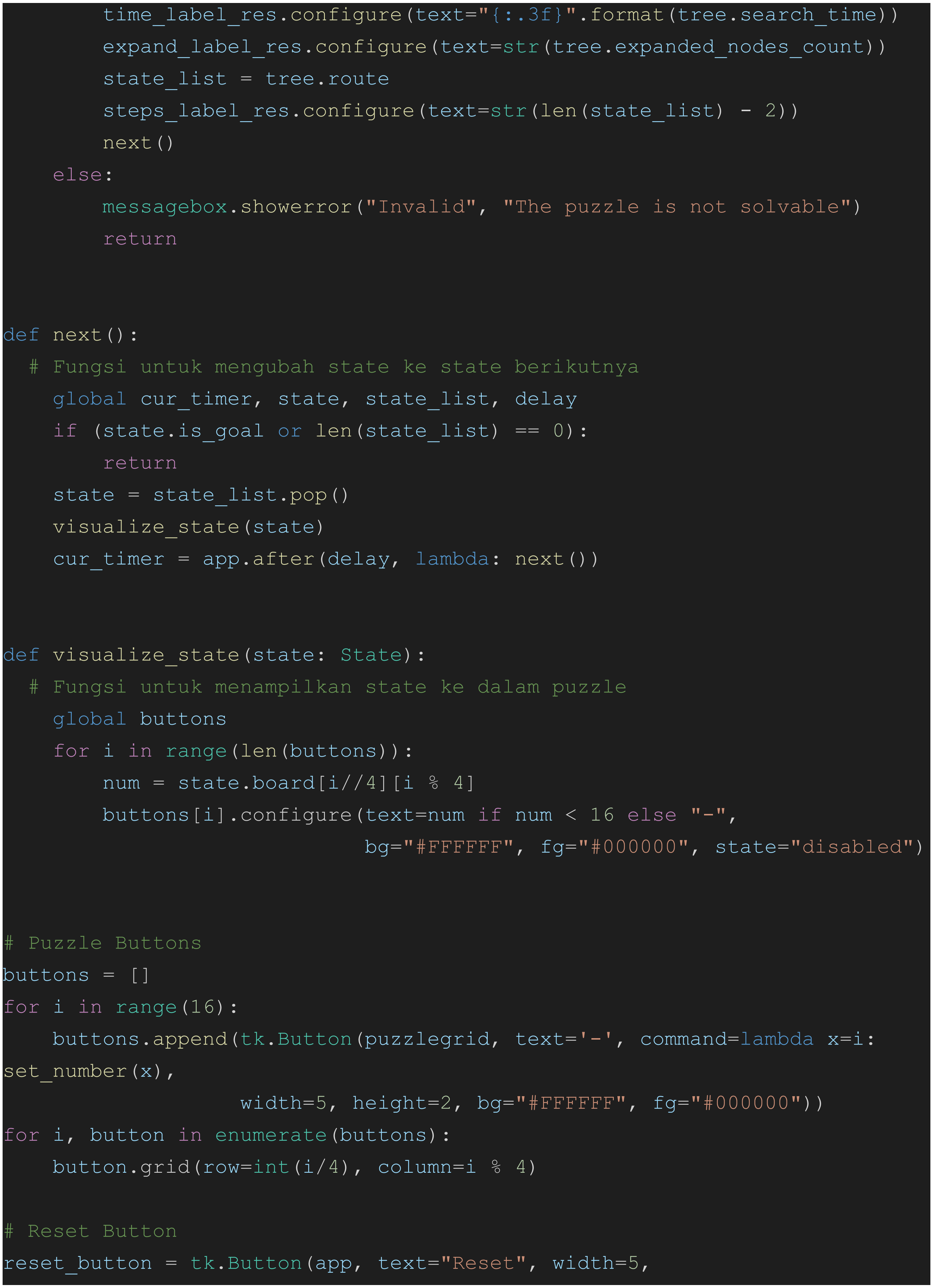


```
# Inisialisasi app
app = tk.Tk()
app.title("15 puzzle")
app.resizable(False, False)

# Title dari app
app_title = tk.Label(app, text="15 puzzle", font=("Helvetica", 20))
app_title.grid(row=0, column=0, columnspan=4)

# Frame puzzle
puzzlegrid = tk.Frame(app, width=300, height=300)
puzzlegrid.grid(row=1, column=0, padx=10, pady=10, columnspan=4,
rowspan=3)
puzzlegrid.configure(background='#FFFFFF')

number = 1


def set_number(button_number):
  # Fungsi untuk memasukkan angka ke dalam puzzle
    global number, buttons
    if (buttons[button_number].cget('text') == '-' and number <= 16):
        buttons[button_number].configure(
            text=number if number < 16 else "-", bg="#FFFFFF",
fg="#FFFFFF", state="disabled")
        number += 1


def reset_number():
  # Fungsi untuk menghapus angka dari puzzle
    global number, buttons
    number = 1
    app.after_cancel(cur_timer)
    for button in buttons:
        if button.cget('text') != '-':
            button.configure(text='-', bg="#FFFFFF",
                              fg="#000000", state="normal")
```



```
def import_number():
  # Fungsi untuk memasukkan angka dari text input ke puzzle
    global number, buttons, text_input, kurang_label_res
    try:
        res = []
        lines = text_input.get("1.0", "end-1c").strip().split("\n")
        if len(lines) != 4:
            raise InvalidInputException("Row must be 4")
        for line in lines:
            try:
                inp = [int(x) if x != "-" else 16 for x in line.split("
")]
            except ValueError:
                raise InvalidInputException("Input must be number")
            if len(inp) != 4:
                raise InvalidInputException("Column must be 4")
            for i in inp:
                if i < 1 or i > 16:
                    raise InvalidInputException(
                        "Input must be between 1 and 16")
            res.append(inp)

        # Ubah input ke puzzle
        state = State(np.array(res))

        # Mendapatkan nilai Kurang(i)
        kurang = state.get_kurang()
        kurang_label_res.configure(
            text=str(sum(kurang.values()) + (sum(state.blank_index) %
2)[0]))
        kurangi_label_res.configure(
            text="\n".join(["{}\t=\t{}".format(k, v) for k, v in
sorted(kurang.items(), key=lambda x: x[0])]))

        # Visualisasi State
        visualize_state(state)
        number = 17
    except InvalidInputException as e:
        messagebox.showerror("Invalid input", str(e))
        return
```

```python
# Type casting state untuk visualisasi
tree = BNBTree
state = State
state_list = list
cur_timer = str
delay = int


def start_visualize():
  # Fungsi untuk memulai visualisasi
    global tree, state, buttons, state_list, cur_timer, number, delay, delay_input, time_label_res, expand_label_res, steps_label_res

    # Validasi Input
    delay = delay_input.get()
    if not delay.isdigit():
        messagebox.showerror("Invalid input", "Delay must be number")
        return
    delay = int(delay)
    if number < 17:
        messagebox.showerror("Invalid input", "Input must be full")
        return

    # Mendapatkan input dari puzzle
    board = [[], [], [], []]
    for (i, button) in enumerate(buttons):
        txt = button.cget('text')
        board[i//4].append(int(txt if txt.isdigit() else '16'))

    # Memasukkan input ke dalam state dan tree
    state = State(np.array(board))
    tree = BNBTree(state)

    # Mengecek apakah state dapat diselesaikan
    if tree.is_solvable():
        app.after_cancel(cur_timer)
        # Melakukan Search
        tree.search()
```

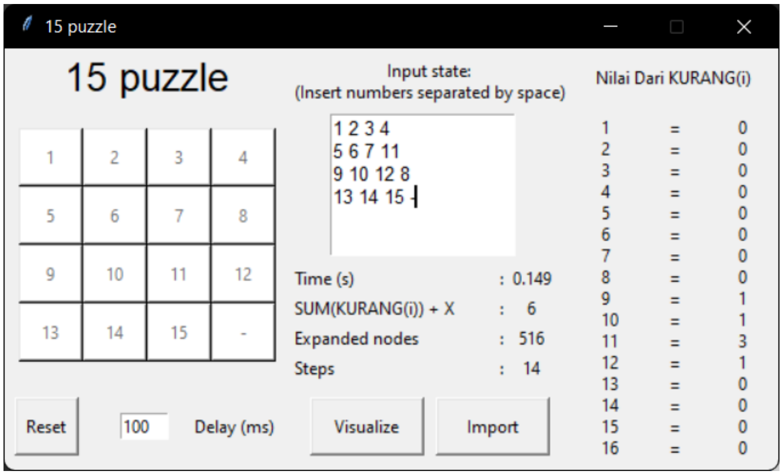
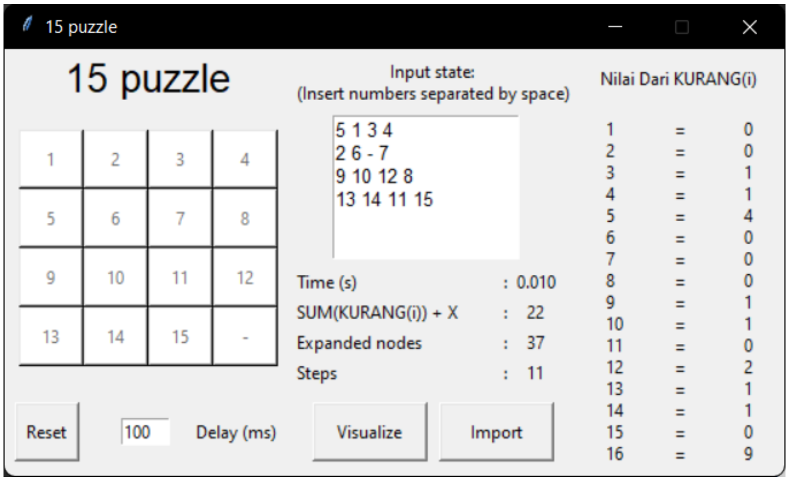


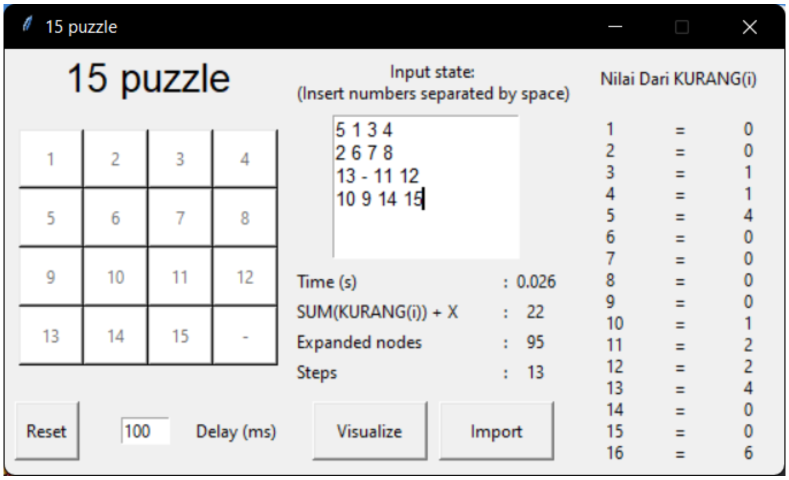
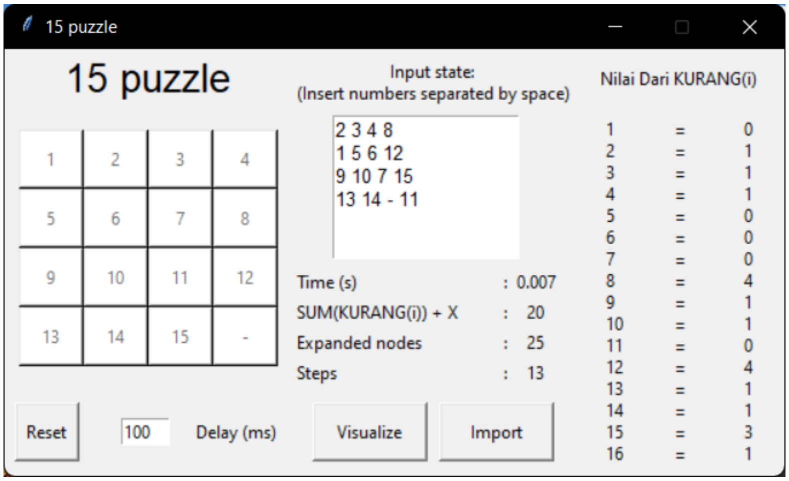
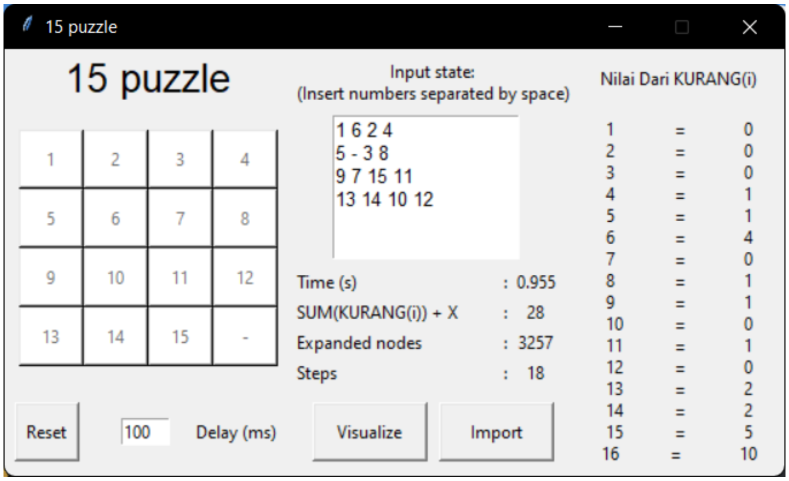
```
        time_label_res.configure(text="{:.3f}".format(tree.search_time))
        expand_label_res.configure(text=str(tree.expanded_nodes_count))
        state_list = tree.route
        steps_label_res.configure(text=str(len(state_list) - 2))
        next()
    else:
        messagebox.showerror("Invalid", "The puzzle is not solvable")
        return


def next():
  # Fungsi untuk mengubah state ke state berikutnya
    global cur_timer, state, state_list, delay
    if (state.is_goal or len(state_list) == 0):
        return
    state = state_list.pop()
    visualize_state(state)
    cur_timer = app.after(delay, lambda: next())


def visualize_state(state: State):
  # Fungsi untuk menampilkan state ke dalam puzzle
    global buttons
    for i in range(len(buttons)):
        num = state.board[i//4][i % 4]
        buttons[i].configure(text=num if num < 16 else "-",
                             bg="#FFFFFF", fg="#000000", state="disabled")


# Puzzle Buttons
buttons = []
for i in range(16):
    buttons.append(tk.Button(puzzlegrid, text='-', command=lambda x=i:
set_number(x),
                   width=5, height=2, bg="#FFFFFF", fg="#000000"))
for i, button in enumerate(buttons):
    button.grid(row=int(i/4), column=i % 4)

# Reset Button
reset_button = tk.Button(app, text="Reset", width=5,
                         height=2, command=reset_number)
reset_button.grid(row=4, column=0, pady=10)

# Delay Input
delay_input = tk.Entry(app, width=5, )
delay_input.grid(row=4, column=2, pady=10)
delay_input.insert(-1, "100")

# Delay Label
delay_label = tk.Label(app, text="Delay (ms)")
delay_label.grid(row=4, column=3, pady=10)

# State Input Label
input_label = tk.Label(
    app, text="Input state:\n(Insert numbers separated by space)")
input_label.grid(row=0, column=4, pady=(5, 0))

# State Input
text_input = tk.Text(app, width=18, height=6)
text_input.configure(font=("Helvetica", 10))
text_input.grid(row=1, column=4, rowspan=1, pady=(5, 0), padx=(0, 10))

# Result Frame
result_frame = tk.Frame(app)
result_frame.grid(row=2, column=4, rowspan=1, pady=(5, 0), padx=(0, 10))

# Time Label
time_label = tk.Label(result_frame, text="Time (s)\t\t\t:")
time_label.grid(row=0, column=0)

# Time Result
time_label_res = tk.Label(result_frame, text="-")
time_label_res.grid(row=0, column=1)

# Kurang Label
kurang_label = tk.Label(result_frame, text="SUM(KURANG(i)) + X\t:")
kurang_label.grid(row=1, column=0)

# Kurang Result
kurang_label_res = tk.Label(result_frame, text="-")
```

```
kurang_label_res.grid(row=1, column=1)

# Expanded Label
expand_label = tk.Label(result_frame, text="Expanded nodes\t\t:")
expand_label.grid(row=2, column=0)

# Expanded Result
expand_label_res = tk.Label(result_frame, text="-")
expand_label_res.grid(row=2, column=1)

# Steps Label
steps_label = tk.Label(result_frame, text="Steps\t\t\t:")
steps_label.grid(row=3, column=0)

# Steps Result
steps_label_res = tk.Label(result_frame, text="-")
steps_label_res.grid(row=3, column=1)

# Button Frame
button_frame = tk.Frame(app)
button_frame.grid(row=4, column=4)

# Visualize Button
visualize_button = tk.Button(
    button_frame, text="Visualize", width=10, height=2,
command=start_visualize)
visualize_button.grid(row=4, column=4, pady=10, padx=4)

# Import Button
import_button = tk.Button(button_frame, text="Import", width=10,
                          height=2, command=import_number)
import_button.grid(row=4, column=5, pady=10, padx=4)

# Kurang(i) Label
kurangi_label = tk.Label(app, text="Nilai Dari KURANG(i)", width=20)
kurangi_label.grid(row=0, column=5)

# Kurang(i) Result
kurangi_label_res = tk.Label(app, text="-")
kurangi_label_res.grid(row=1, column=5, rowspan=5)
```

```
app.mainloop()
```

Contoh Persoalan

No.	Teks	Hasil
1.	1 2 3 4 5 6 7 11 9 10 12 8 13 14 15 -	 15 puzzle 15 puzzle Input state: (Insert numbers separated by space) 1 2 3 4 5 6 7 11 9 10 12 8 13 14 15 - Time (s) : 0.149 SUM(KURANG(i)) + X : 6 Expanded nodes : 516 Steps : 14 Nilai Dari KURANG(i) 1 = 0 2 = 0 3 = 0 4 = 0 5 = 0 6 = 0 7 = 0 8 = 0 9 = 1 10 = 1 11 = 3 12 = 1 13 = 0 14 = 0 15 = 0 16 = 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 - Reset 100 Delay (ms) Visualize Import
2.	5 1 3 4 2 6 - 7 9 10 12 8 13 14 11 15	 15 puzzle 15 puzzle Input state: (Insert numbers separated by space) 5 1 3 4 2 6 - 7 9 10 12 8 13 14 11 15 Time (s) : 0.010 SUM(KURANG(i)) + X : 22 Expanded nodes : 37 Steps : 11 Nilai Dari KURANG(i) 1 = 0 2 = 0 3 = 1 4 = 1 5 = 4 6 = 0 7 = 0 8 = 0 9 = 1 10 = 1 11 = 0 12 = 2 13 = 1 14 = 1 15 = 0 16 = 9 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 - Reset 100 Delay (ms) Visualize Import

3.	5 1 3 4 2 6 7 8 13 - 11 12 10 9 14 15	 15 puzzle 15 puzzle 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 - Input state: (Insert numbers separated by space) 5 1 3 4 2 6 7 8 13 - 11 12 10 9 14 15 Time (s) : 0.026 SUM(KURANG(i)) + X : 22 Expanded nodes : 95 Steps : 13 Reset 100 Delay (ms) Visualize Import Nilai Dari KURANG(i) 1 = 0 2 = 0 3 = 1 4 = 1 5 = 4 6 = 0 7 = 0 8 = 0 9 = 0 10 = 1 11 = 2 12 = 2 13 = 4 14 = 0 15 = 0 16 = 6
4.	2 3 4 8 1 5 6 12 9 10 7 15 13 14 - 11	 15 puzzle 15 puzzle 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 - Input state: (Insert numbers separated by space) 2 3 4 8 1 5 6 12 9 10 7 15 13 14 - 11 Time (s) : 0.007 SUM(KURANG(i)) + X : 20 Expanded nodes : 25 Steps : 13 Reset 100 Delay (ms) Visualize Import Nilai Dari KURANG(i) 1 = 0 2 = 1 3 = 1 4 = 1 5 = 0 6 = 0 7 = 0 8 = 4 9 = 1 10 = 1 11 = 0 12 = 4 13 = 1 14 = 1 15 = 3 16 = 1
5.	1 6 2 4 5 - 3 8 9 7 15 11 13 14 10 12	 15 puzzle 15 puzzle 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 - Input state: (Insert numbers separated by space) 1 6 2 4 5 - 3 8 9 7 15 11 13 14 10 12 Time (s) : 0.955 SUM(KURANG(i)) + X : 28 Expanded nodes : 3257 Steps : 18 Reset 100 Delay (ms) Visualize Import Nilai Dari KURANG(i) 1 = 0 2 = 0 3 = 0 4 = 1 5 = 1 6 = 4 7 = 0 8 = 1 9 = 1 10 = 0 11 = 1 12 = 0 13 = 2 14 = 2 15 = 5 16 = 10

Link Penting

Github

https://github.com/rifqi2320/15puzzle

Google Drive

https://drive.google.com/drive/folders/1e7LvRgUkn-4erfekU-dehcAMusI9N7bs?usp=sharing