Laporan Tugas Kecil 3 IF2211 Strategi Algoritma

Semester II Tahun Akademik 2021/2022

Penyelesaian Persoalan 15-Puzzle dengan Algoritma Branch and Bound

Disusun Oleh :

Muhammad Gilang Ramadhan

13520137

K02

Program Studi S1 Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2022

BAB 1

Pendahuluan

Algoritma *Branch and Bound* merupakan salah satu strategi algoritma yang dipakai untuk menyelesaikan masalah optimasi. Algoritma ini menggunakan pohon ruang status untuk melakukan pencarian solusi terbaik. Perjalanan dalam pencarian status solusi menggunakan usaha yang paling optimal. Salah satu penerapan algoritma ini adalah dalam menyelesaikan persoalan 15-Puzzle.

15-Puzzle adalah salah satu permainan dengan menggunakan papan yang berisikan angka 1 sampai 15 dalam 16 bagian ubin. Terdapat satu ubin kosong yang dapat digerakkan ke atas, bawah, kiri, dan kanan untuk menggeser ubin lainnya. Tujuan yang dicapai dari permainan ini adalah menyusun angka 1 sampai 15 terurut dari atas ke bawah dengan cara menggeser ubin kosong.

Gambar 1.1 Ilustrasi 15-Puzzle
(Sumber: Wikipedia)

Untuk menyelesaikan permainan tersebut, terdapat teorema yang menyatakan bahwa jika nilai persamaan berikut

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

adalah genap. Adapun KURANG(i) yang dimaksud adalah inversi susunan puzzle, yakni A[i] > A[j] tetapi i < j, sedangkan nilai X menyatakan letak ubin kosong pada susunan awal. Apabila

ditinjau nilai yang mungkin, adapun X bernilai 0 jika nilai i+j genap, dan X bernilai 1 jika nilai i+j ganjil. Jika teorema di atas terpenuhi, maka susunan ubin tersebut solveable, begitu juga sebaliknya. Apabila susunan ubin solveable, maka akan dilakukan proses pencarian simpul tujuan dari simpul utama.

Selain itu juga, pada proses pencarian dengan algoritma *Branch & Bound*, terdapat prinsip *least cost search* yang dipakai pada pencarian simpul anak pada tiap langkahnya. *Least cost search* sendiri dipakai untuk menentukan simpul *child* mana yang akan selanjutnya. Pada tiap pembangkitan simpul *child*, akan dihitung nilai *cost*nya dan dimasukkan ke dalam antrian pemrosesan simpul hidup dengan prioritas utama adalah simpul anak dengan nilai *cost* terkecil. Dengan demikian simpul anak yang dipilih yang memiliki *cost* terkecil dari simpul anak yang lain pada antrian pemrosesan. *Cost* yang digunakan adalah sebagai berikut :

$$c(i) = f(i) + g(i)$$

$c(i)$ = *cost* untuk simpul ke i

$f(i)$ = *cost* atau kedalaman untuk mencapai simpul ke i dari simpul akar

$g(i)$ = cost yang menyatakan banyaknya petak (selain petak kosong) yang berada di posisi yang salah.

BAB 2
Penjelasan Algoritma

2.1 Struktur Program

2.1.1 File Gui.py

File Gui.py berisi beberapa properti dan implementasi dari *Graphical User Interface* (GUI) dari program 15-*Puzzle* solver yang akan dipakai sebagai *interface* utama program. Untuk penggunaan *interface* tersebut, pada bagian inputnya, user diminta untuk menginput file masukan dengan cara menekan tombol "Select file" yang tertera di Interfacenya. Untuk keluaran, user dapat memilih untuk meng*export* file solusi yang berisi langkah-langkah yang diperlukan untuk menyelesaikan 15-*Puzzle* tersebut atau jika user tidak ingin melakukan hal tersebut, user menampilkan solusi tersebut melalui GUI dengan cara menekan tombol *visualize*, yang akan menampilkan pergeseran *cell* kosong sebagai representasi langkah-langkah penyelesaian *puzzle* dari awal sampai akhir. Selain itu, untuk menampilkan rincian fungsi kurang yang dihasilkan beserta jumlahnya, jumlah simpul yang dibangkitkan, dan time execution program. Adapun berikut disajikan beberapa fungsi yang berkaitan dengan file Gui.py.

Nama fungsi	Kegunaan
Puzzle_Labels	Berfungsi untuk memvisualisasikan Puzzle yang dijadikan sebagai intertface untuk output dari GUI
Callsolve	Berfungsi untuk mengeksekusi puzzle untuk diselesaikan serta menyimpan informasi lain yang berkaitan dengan output program
select_file	Berfungsi untuk mengambil input dari file masukan dari folder test yang ada di directory
saveFile	Berfungsi untuk menyimpan solusi dari problem 15-Puzzle berupa langkah-langkah penyelesaiannya
ResetPuzz	Berfungsi untuk mereset visualisasi dari langkah-langkah solusi 15-Puzzle yang diberikan
Visualize	Berfungsi sebagai button untuk memanggil fungsi Callsolve
Details	Berfungsi sebagai button untuk menampilkan detail dari solusi

saveFile_Button	Berfungsi sebagai button untuk memanggil fungsi saveFile
setting_GUI	Berfungsi untuk mengatur setting dari GUI

Tabel 2.1.1.1 Fungsi yang berkaitan dengan Gui.py

2.1.2 File Puzzle.py

Adapun rincian method dan atribut yang diimplementasikan di kelas Puzzle adalah sebagai berikut.

Atribut	**Kegunaan**
puzzle	Menyimpan *puzzle* yang diambil dari file inputan
Solution	Menyimpan urutan gerakan yang berguna untuk menyelesaikan *puzzle*
Direction	*List* yang berisi konstanta untuk arah pergerakan (atas, bawah, kanan, kiri)
Tempkurang	Menyimpan nilai fungsi kurang tiap urutan iterasi
visited	Menyimpan *nodes* yang sudah pernah dikunjungi
ValueX	Menyimpan value dari X awal
Total	Menyimpan banyaknya simpul yang dibangkitkan

Tabel 2.1.2.1 Atribut dari kelas Puzzle

Nama Method	**Kegunaan**
__init__(self, FileName)	Sebagai konstruktor untuk mengeset masing-masing atribut dari kelas Puzzle
getPuzzle(sel, FileName)	Sebagai getter puzzle dari masukan file input .txt
SumKurang(self)	Sebagai getter dari penjumlahan KURANG(i) + ValueX pada puzzle
SumCost(self, puzzle)	Sebagai getter dari penjumlahan cost yang dibutuhkan pada puzzle
getZeroPos(self, puzzle)	Sebagai getter dari koordinat sel kosong pada puzzle
solve(self)	Sebagai getter dari sum of fungsi kurang dengan menggunakan Algoritma Branch and Bound

Tabel 2.1.2.2 Fungsi yang berkaitan dengan Puzzle.py

Penjelasan tambahan :

Method solve merupakan method atau fungsi utama yang dipakai untuk menyelesaikan permasalahan 15-Puzzle, dengan keluarannya berupa jumlah fungsi kurang. Adapun cara kerja dari fungsi tersebut menggunakan Algoritma Branch and Bound adalah sebagai berikut.

1. Pertama-tama, fungsi akan mengecek jumlah fungsi kurang dan X awal adalah genap atau tidak.
2. Jika genap, maka puzzle yang diberikan tersebut memiliki solusi. Bila memiliki solusi maka akan dilakukan pencarian lebih lanjut. Dalam pencariannya, fungsi akan membuat heap minimum yang menyimpan informasi semua simpul, yaitu cost dari simpul hidup (SumCost + depth), kondisi puzzle, urutan pergerakan, dan depth simpul dari simpul awal untuk mencapai simpul tersebut.
3. Setelah itu, fungsi tersebut akan terus mencari solusi sampai heap yang dipakai habis, jika heap belum habis, maka fungsi akan mengambil nilai dengan cost terkecil dari heap, kemudian akan menambahkannya ke daftar simpul yang sudah dikunjungi.
4. Adapun jika cost dan depth berbeda, maka fungsi akan membentuk simpul sampai dengan empat simpul, yaitu simpul yang dapat dicapai dari simpul sekarang, dan jika simpul tersebut vaild dan belum dikunjungi, dengan nilai cost dan depth yang baru, maka simpul tersebut akan dimasukkan ke dalam heap. Sebaliknya, jika cost dan depth nilainya sama, maka solusi pun ditemukan, dan nilai mincost akan diupdate dengan cost simpul tersebut, dan loop akan berlanjut ke iterasi berikutnya.
5. Jika heap sudah kosong, maka fungsi akan mengembalikan jumlah nilai fungsi kurang dan ValueX awal. Jika jawaban diperoleh, maka list of Solution akan memuat langkah-langkah yang diperlukan untuk mencapai jawaban, dan jika sebaliknya, maka list of Solution akan kosong.

2.2 Cara Kerja Program

Adapun berikut adalah langkah-langkah kerja program tersebut berjalan sesuai dengan Algoritma Branch and Bound.

1. Program akan menerima masukan dari Interface dalam bentuk GUI
2. Cek state awal puzzle menggunakan fungsi SumKurang. Jika hasil dari SumKurang adalah genap, maka puzzle dapat diselesaikan, jika sebaliknya (ganjil), maka puzzle tidak

dapat diselesaikan (unsolveable). Adapun tahap ini diintegrasikan di fungsi/method solve pada *class* Puzzle

3. *Generates* child node dari simpul awal berdasarkan pergerakan yang mungkin dari puzzle
4. Cek apakah puzzle pernah berada di state yang sama
5. Hitung cost dari masing-masing *move*
6. Masukkan setiap simpul yang digenerates ke dalam container (visited)
7. Bangkitkan simpul selanjutnya dari heap hingga bentuk akhir puzzle (bentuk setelah diselesaikan) ditemukan.

BAB 3

Kode Program

2.1 Puzzle.py

```python
# Nama  : Muhammad Gilang Ramadhan
# NIM   : 13520137
# Puzzle.py
# Containing property of the puzzle and Branch and Bound Algorithm

import heapq as pq
import copy

class Puzzle:
    # Initialate

    # Puzzle as blank arr
    puzzle = []
    # The solution of the puzzle
    Solution = []
    # Direction to move : DOWN, RIGHT, UP, LEFT
    Direction = [(1,0),(0,1),(-1,0),(0,-1)]
    # Kurang Function for each entry
    Tempkurang = [0 for i in range(16)]
    # List of visited node
    visited = []
    # Value X
    ValueX = 0
    # Total
    Total = 0
    # Minimum Cost
    mincost = 10**9 + 7

    def __init__(self, FileName):
        self.puzzle = []
        self.Solution = []
        self.Tempkurang = [0 for i in range(16)]
        self.visited = []
        self.ValueX = 0
        self.total = 0
        self.mincost = 1e9+7
        self.getPuzzle(FileName)

    # Get Puzzle from Input file .txt
    def getPuzzle(self,FileName):
        temp_puzzle = []
        puzzle_set = set({})
        with open(FileName) as f:
            lines = f.readlines()
            if len(lines)==4:
                for line in lines:
                    if len(line.split())==4:
                        temp_puzzle.append([int(i) for i in line.split()])
                        for i in line.split():
                            puzzle_set.add(int(i))
                    else:
                        return
            else:
                return
        if puzzle_set!={i for i in range(16)}:
            return
        self.puzzle = temp_puzzle
```

Gambar 2.1.1 Puzzle.py (line 1-57)

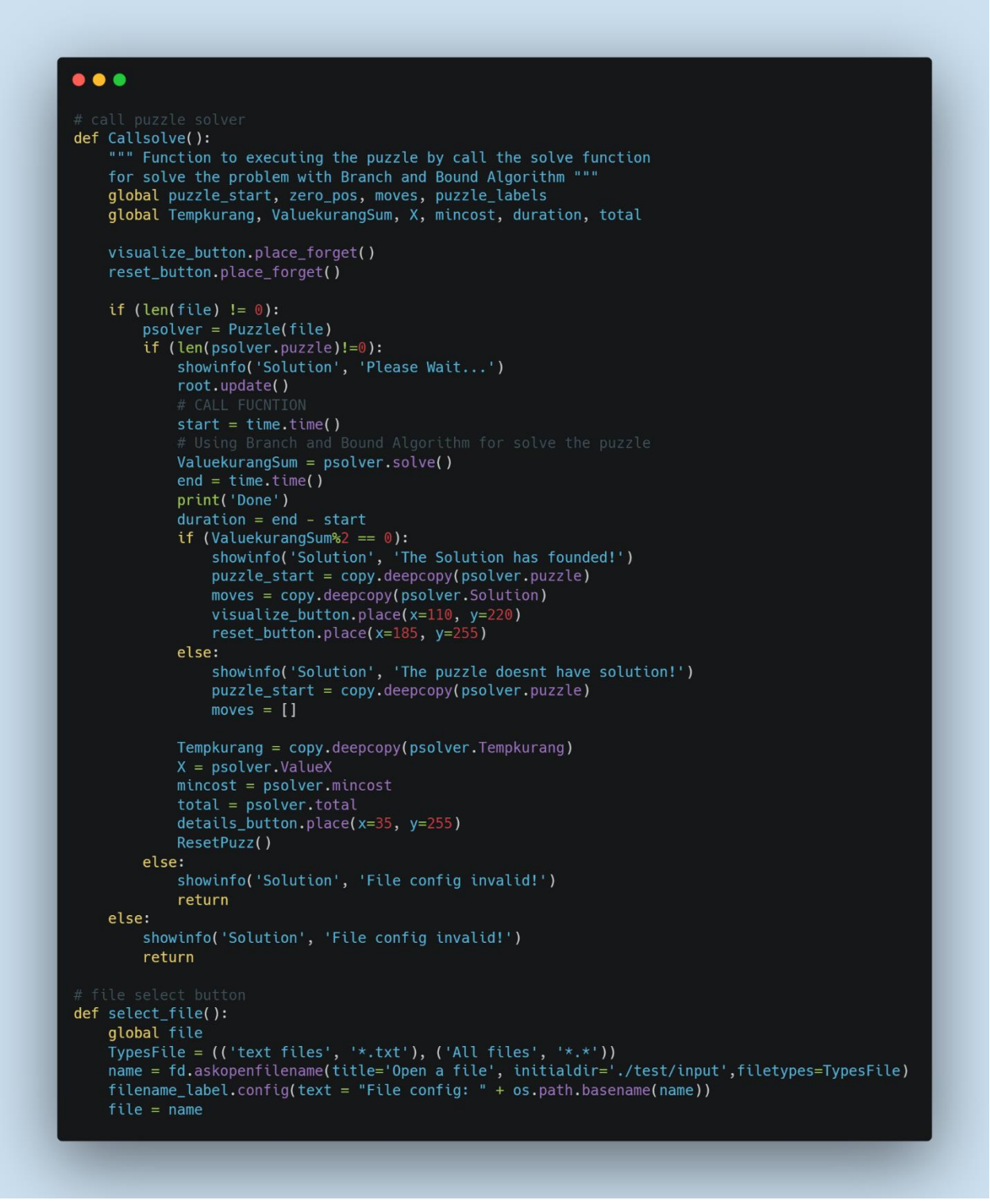
```python
# Calculate sum of KURANG(i) + X
    def SumKurang(self):
        cost = 0
        flat = [i for j in self.puzzle for i in j]
        for i in range(len(flat)):
            temp = 0
            if flat[i]==0 and (((i//4)%2 == 1 and i%2 == 0) or ((i//4)%2 == 0 and i%2==1)):
                self.ValueX = 1
            for j in range(i+1,len(flat)):
                if ((flat[i]>flat[j] or flat[i]==0) and flat[j]!=0):
                    temp += 1
                    cost += 1
            self.Tempkurang[flat[i]] = temp
        return cost + self.ValueX

    # Calculate sum cost
    def SumCost(self, puzzle):
        flat = [i for j in puzzle for i in j]
        cost = 0
        for i in range(len(flat)):
            if ((i+1)%16 != flat[i]):
                cost+=1
        return cost

    # Get coordinat zero value position from the puzzle
    def getZeroPos(self, puzzle):
        for i in range(len(puzzle)):
            for j in range(len(puzzle[0])):
                if puzzle[i][j]==0:
                    return (i,j)
        return (-1,-1)

    # Using Branch and Bound Algorithm to solve the puzzle
    def solve(self):
        kurang = self.SumKurang()
        if kurang%2==0:
            # temp current total cost, depth, puzzle, moves to puzzle
            heap = []
            cost = self.SumCost(self.puzzle)
            pq.heappush(heap,(cost,0,copy.deepcopy(self.puzzle),[]))

            while len(heap)>0:
                curCost, depth, curPuzzle, path = pq.heappop(heap)
                self.visited.append(curPuzzle)
                self.total += 1
                if (curCost<=self.mincost and curCost != depth):
                    # Transitions
                    x0,y0 = self.getZeroPos(curPuzzle)
                    for dx,dy in self.Direction:
                        if (0<=x0+dx<4 and 0<=y0+dy<4): # If the coordinat not valid
                            newPuzzle = copy.deepcopy(curPuzzle)
                            newPuzzle[x0][y0], newPuzzle[x
```

Gambar 2.1.2 Puzzle.py (line 59-124)

2.2 Gui.py

```python
# Nama  : Muhammad Gilang Ramadhan
# NIM   : 13520137
# Gui.py
# 15-Puzzle Solver with Branch and Bound Algorithm

from tkinter import *
from tkinter import ttk
from tkinter import filedialog as fd
import tkinter
import os
import time
from Puzzle import Puzzle
from Puzzle import *
from tkinter.messagebox import showinfo

# Initialate Global Variables

# puzzle solving details
Tempkurang = []
ValuekurangSum = 0
TempX = []
mincost = 0
duration = 0
total = 0

# file input
file = ""

# puzzle animation config default
puzzle_labels=[[0 for i in range(4)] for j in range(4)]
puzzle_start = [[1,2,3,4],[5,6,7,8],[9,10,11,12],[13,14,15,0]]
zero_pos = (3,3)
Tempmoves = []

# puzzle labels
def Puzzle_Labels():
    for i in range(4):
        for j in range(4):
            puzzle_labels[i][j] = Label(root, text = str(i*4+j+1), font=('calibre',9),
                height= 1, width=2, anchor=CENTER,
                borderwidth=2, relief="solid", bg='yellow')
            puzzle_labels[i][j].place(x=110+20*j, y=120+20*i)
    puzzle_labels[3][3].config(text= " ")
}
```

Gambar 2.2.1 Gui.py (line 1-43)



```python
# call puzzle solver
def Callsolve():
    """ Function to executing the puzzle by call the solve function
    for solve the problem with Branch and Bound Algorithm """
    global puzzle_start, zero_pos, moves, puzzle_labels
    global Tempkurang, ValuekurangSum, X, mincost, duration, total

    visualize_button.place_forget()
    reset_button.place_forget()

    if (len(file) != 0):
        psolver = Puzzle(file)
        if (len(psolver.puzzle)!=0):
            showinfo('Solution', 'Please Wait...')
            root.update()
            # CALL FUCNTION
            start = time.time()
            # Using Branch and Bound Algorithm for solve the puzzle
            ValuekurangSum = psolver.solve()
            end = time.time()
            print('Done')
            duration = end - start
            if (ValuekurangSum%2 == 0):
                showinfo('Solution', 'The Solution has founded!')
                puzzle_start = copy.deepcopy(psolver.puzzle)
                moves = copy.deepcopy(psolver.Solution)
                visualize_button.place(x=110, y=220)
                reset_button.place(x=185, y=255)
            else:
                showinfo('Solution', 'The puzzle doesnt have solution!')
                puzzle_start = copy.deepcopy(psolver.puzzle)
                moves = []

            Tempkurang = copy.deepcopy(psolver.Tempkurang)
            X = psolver.ValueX
            mincost = psolver.mincost
            total = psolver.total
            details_button.place(x=35, y=255)
            ResetPuzz()
        else:
            showinfo('Solution', 'File config invalid!')
            return
    else:
        showinfo('Solution', 'File config invalid!')
        return

# file select button
def select_file():
    global file
    TypesFile = (('text files', '*.txt'), ('All files', '*.*'))
    name = fd.askopenfilename(title='Open a file', initialdir='./test/input',filetypes=TypesFile)
    filename_label.config(text = "File config: " + os.path.basename(name))
    file = name
```

Gambar 2.2.2 Gui.py (line 45-97)

```
# Save File
def saveFile():
    puzzle = puzzle_start
    ZeroPosY = zero_pos[0]
    ZeroPosX = zero_pos[1]
    count = 0
    if (len(moves) > 0):
        f = fd.asksaveasfile(initialfile = 'Untitled.txt', defaultextension=".txt",filetypes=[("All
Files","*.*"),("Text Documents","*.txt")])
        f.write("SOLUTION : \n")
        f.write("\n")
        for dxy in moves:
            count += 1
            f.write(str(count) + " ")
            if (dxy[0] == -1 and dxy[1] == 0):
                f.write("UP\n")
            elif (dxy[0] == 0 and dxy[1] == -1):
                f.write("LEFT\n")
            elif (dxy[0] == 1 and dxy[1] == 0):
                f.write("DOWN\n")
            elif (dxy[0] == 0 and dxy[1] == 1):
                f.write("RIGHT\n")
            f.write("============================\n")
            # SWAP
            temp = puzzle[ZeroPosY][ZeroPosX]
            puzzle[ZeroPosY][ZeroPosX] = puzzle[dxy[0] + ZeroPosY][dxy[1] + ZeroPosX]
            puzzle[dxy[0] + ZeroPosY][dxy[1] + ZeroPosX] = temp
            for i in range(len(puzzle)):
                for j in range(len(puzzle[i])):
                    f.write(str(puzzle[i][j]) + " ")
                f.write("\n")
            f.write("\n")
            ZeroPosY = dxy[0] + ZeroPosY
            ZeroPosX = dxy[1] + ZeroPosX
    else:
        showinfo('Warning', 'Input is invalid')

# Reset Puzzle
def ResetPuzz():
    global zero_pos
    for i in range(4):
        for j in range(4):
            if puzzle_start[i][j]!=0:
                puzzle_labels[i][j].config(text = str(puzzle_start[i][j]))
            else:
                zero_pos = (i,j)
                puzzle_labels[i][j].config(text = "")
    root.update()

# visualize solution
def Visualize():
    global puzzle_labels
    ResetPuzz()
    zero = copy.deepcopy(zero_pos)
    puzzle = copy.deepcopy(puzzle_start)
    for Dxy in moves:
        time.sleep(1)
        dx, dy = Dxy
        zx, zy = zero
        puzzle[zx][zy], puzzle[zx+dx][zy+dy] = puzzle[zx+dx][zy+dy], puzzle[zx][zy]
        puzzle_labels[zx+dx][zy+dy].config(text = "")
        puzzle_labels[zx][zy].config(text = str(puzzle[zx][zy]))
        zero = (zx + dx, zy + dy)
        root.update()
```

Gambar 2.2.3 Gui.py (line 99-161)

```python
# Details Process
def Details():
    INFO = ">>>>>>KURANG(i)<<<<<<\n"
    for i in range(1,17):
        if (i < 16):
            INFO += "Value KURANG(" + str(i) + ") "  +": " + str(Tempkurang[i]) + "\n"
        else:
            INFO += "Value KURANG(" + str(i) + ") "  +": " + str(Tempkurang[0]) + "\n"
    INFO += "Sum of KURANG(i) : " + str(ValuekurangSum) + "\n"
    INFO += "Nodes Generates : " + str(total) + "\n"
    INFO += "Time Execution : "+str(duration*1000)+" ms\n"
    showinfo(title='Details Solution', message=INFO)

# save button
def saveFile_Button():
    menu = tkinter.Menu(root)
    root.config(menu=menu)
    fileMenu = tkinter.Menu(menu, tearoff=0)
    menu.add_cascade(label='Export Solution', menu=fileMenu)
    fileMenu.add_command(label='Save Puzzle', command=saveFile)

# Gui Setting
def setting_GUI():
    # Title and size
    root.title("15-Puzzle Solver | Gilang")
    root.geometry("300x300")

    # label
    prompt_label.place(x=81, y=7)

    # open button
    open_button.place(x=110, y=29)

    # filename
    filename_label.place(x=75, y=57)

    # Solve button
    solve_button.place(x=110, y=78)

    # Kurang_label
    kurang_label.place(x=20, y=105)

if __name__ == "__main__":
    # Initialate GUI
    root=Tk()
    root.geometry("1000x1000")
    root['background'] = 'green'

    puzzle_labels=[[0 for i in range(4)] for j in range(4)]
    Puzzle_Labels()
    prompt_label = Label(root, text = 'Please input your puzzle :', font=('calibre', 9), bg='green')
    open_button = ttk.Button(root, text='Select file', command=select_file)
    filename_label = Label(root, text = 'File config has not selected', font=('calibre', 9), bg='green')
    solve_button = ttk.Button(root, text='Solve', command=Callsolve)
    kurang_label = Label(root, text = '', font=('calibre',8), bg='green')
    visualize_button = ttk.Button(root, text='Visualize', command=Visualize)
    reset_button = ttk.Button(root, text='Reset', command=ResetPuzz)
    details_button = ttk.Button(root, text='Details', command=Details)

    setting_GUI()

    saveFile_Button()

    root.mainloop()
```

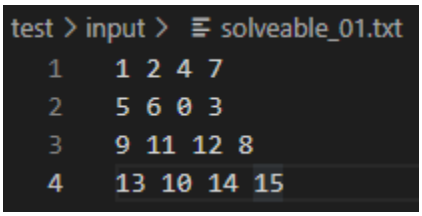
Gambar 2.2.4 Gui.py (line 163-226)

BAB 4

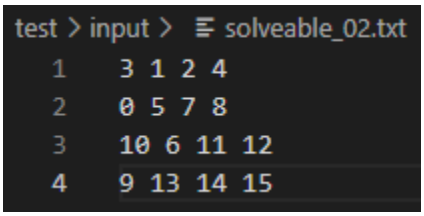
Pengujian Program

3.1 Test case yang *solveable*

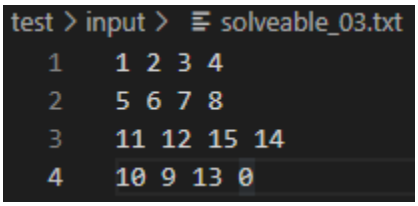
3.1.1 Input



Gambar 3.1.1.1 Input solveable_01

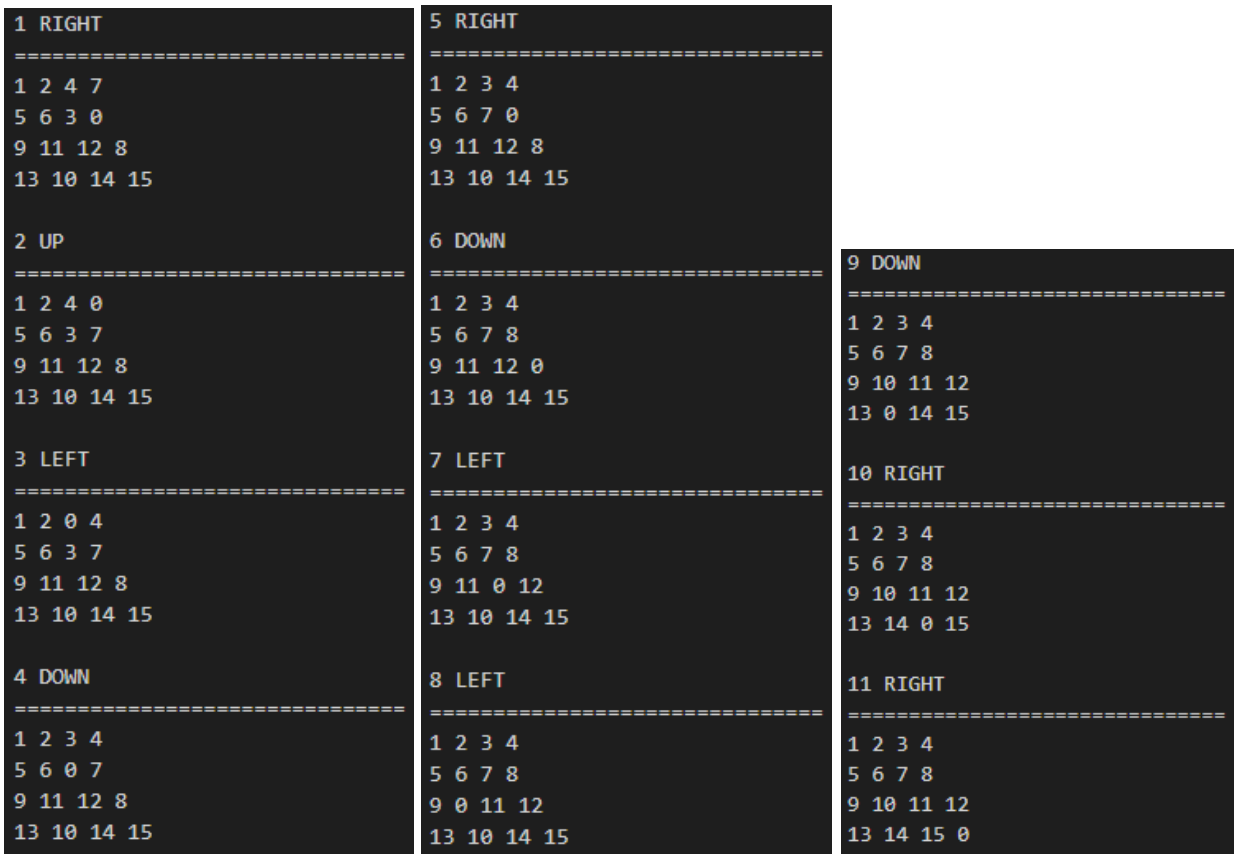


Gambar 3.1.1.2 Input solveable_02

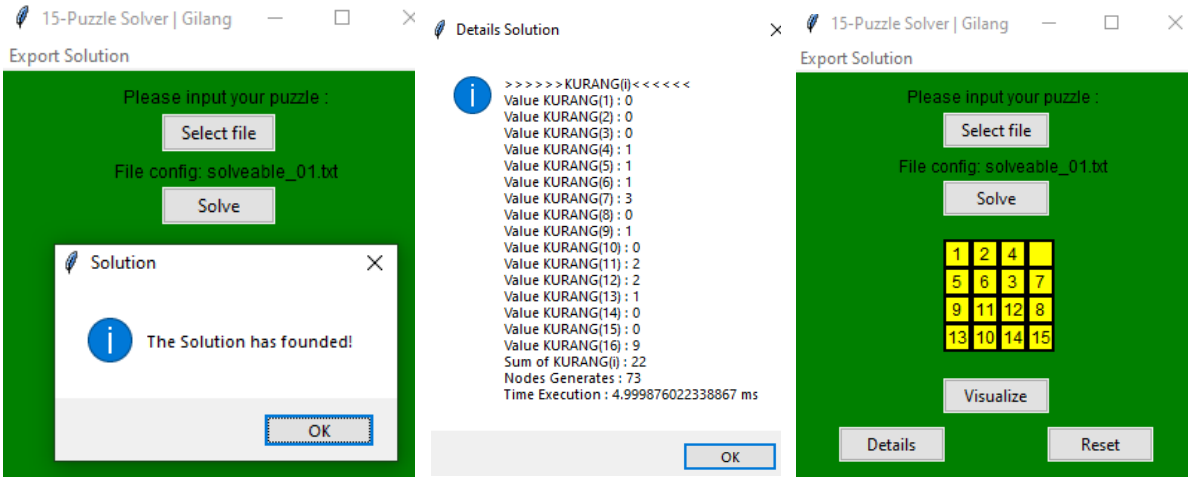


Gambar 3.1.1.3 Input solveable_03

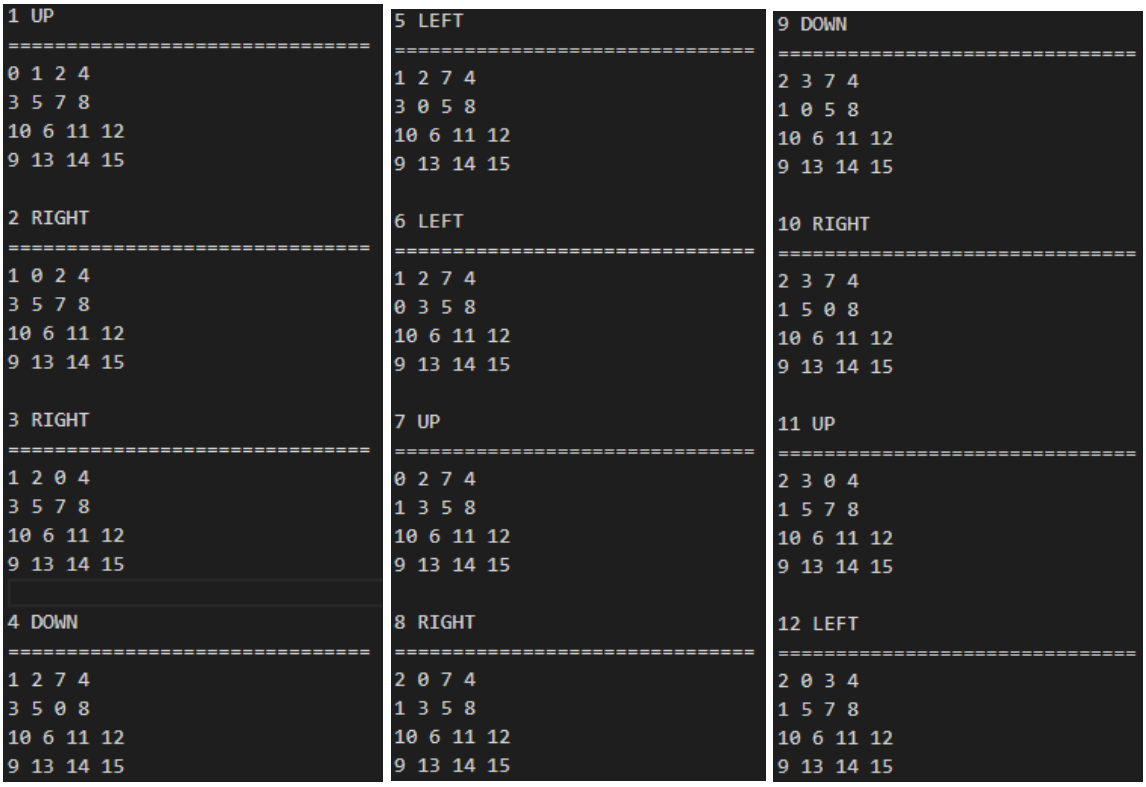
3.1.2 Output



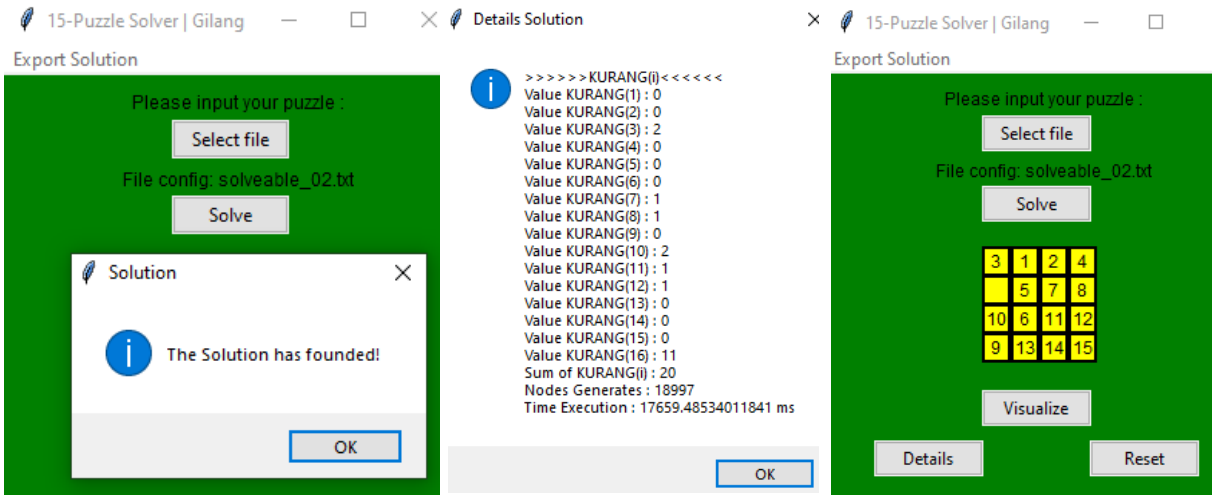
Gambar3.1.2.1 Solusi step penyelesaian puzzle solveable_01



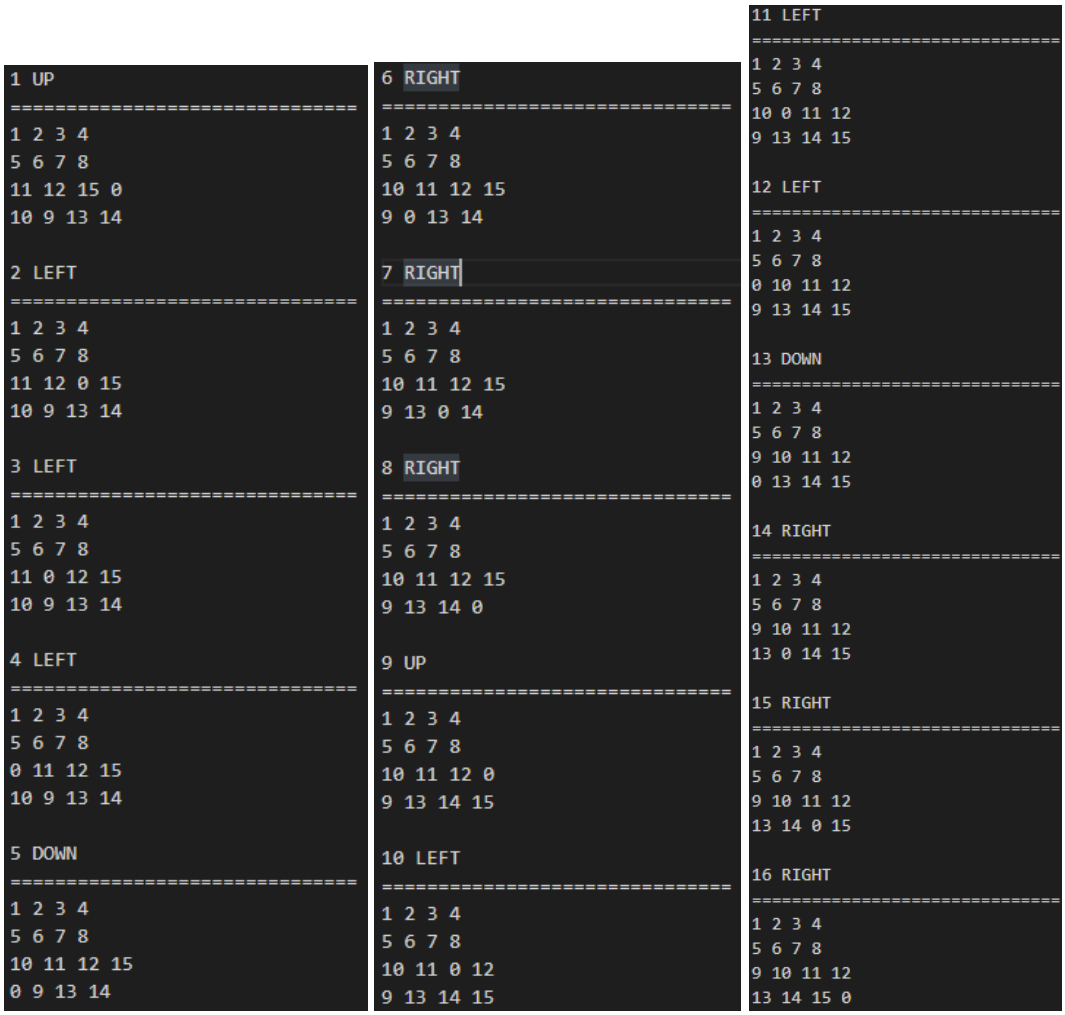
Gambar 3.1.2.2 Visualisasi Gui Input solveable_01



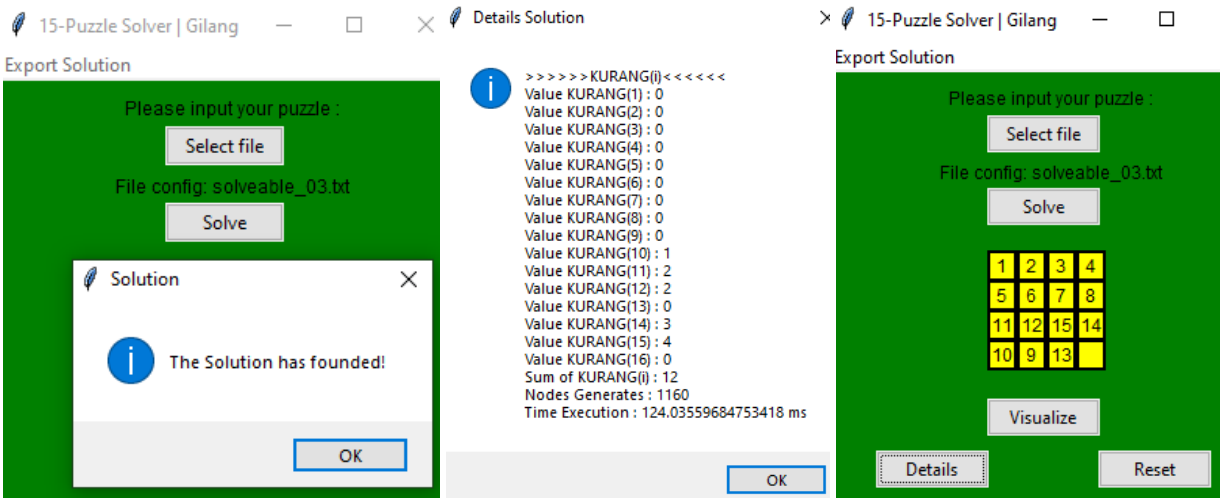
Gambar 3.1.2.3 Solusi step penyelesaian puzzle solveable_02



Gambar 3.1.2.4 Visualisasi Gui Input solveable_02



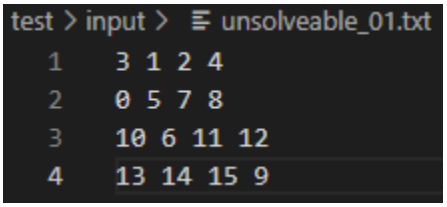
Gambar 3.1.2.5 Solusi step penyelesaian puzzle solveable_03



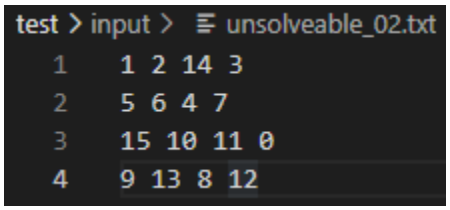
Gambar 3.1.2.6 Visualisasi Gui Input solveable_03

3.2 Test case yang *unsolveable*

3.2.1 Input

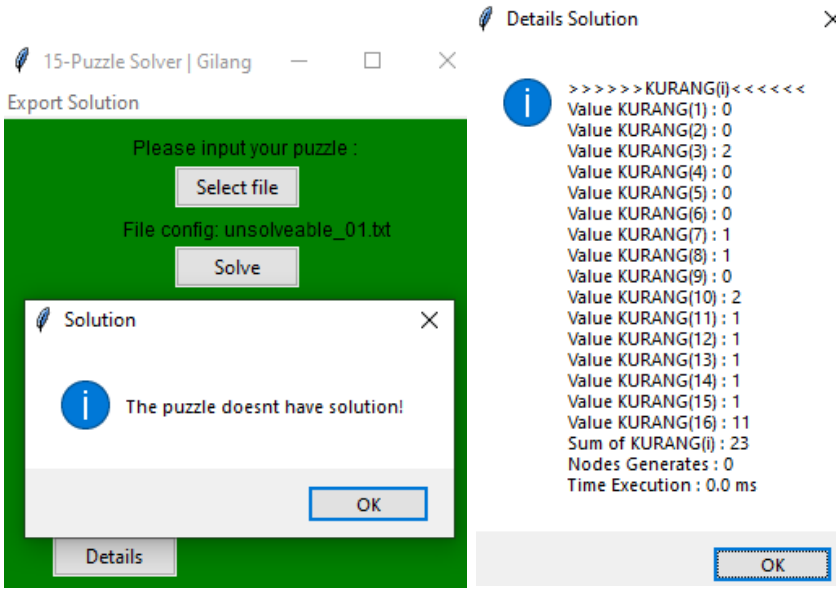


Gambar 3.2.1.1 Input unsolveable_01

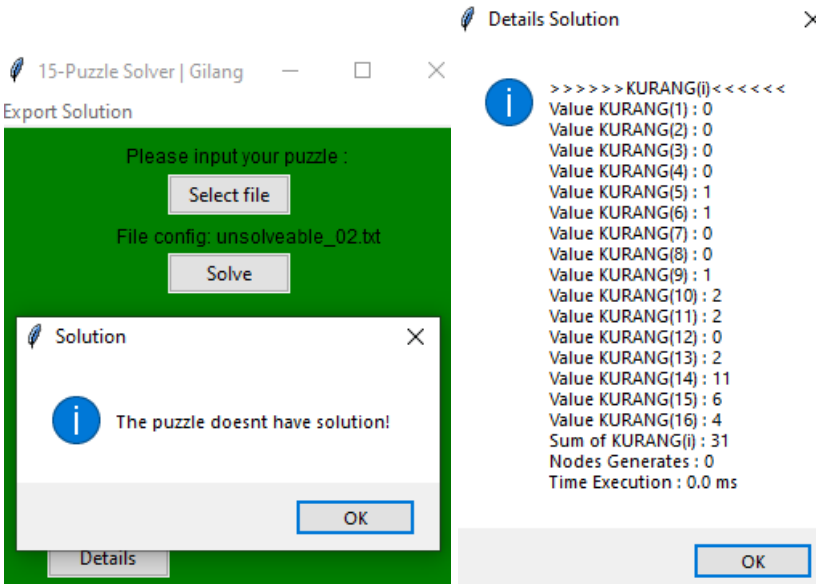


Gambar 3.2.1.2 Input unsolveable_02

3.2.2 Output



Gambar 3.2.2.1 Visualisasi Gui Input unsolveable_01



Gambar 3.2.2.2 Visualisasi Gui Input unsolveable_02

LAMPIRAN

Lampiran 1

Checklist penilaian :

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	✓	

Lampiran 2

Link *Repository* Github : https://github.com/gilanglahat22/Tucil3_13520137