

LAPORAN TUGAS KECIL
**Penyelesaian Persoalan 15-Puzzle dengan Algoritma Branch
and Bound**

*Disajikan untuk memenuhi salah satu tugas kecil Mata Kuliah IF2211 Strategi Algoritma yang
diampu oleh:*

Dr. Masayu Leylia Khodra, S.T., M.T.



Disusun oleh:

Nelsen Putra (13520130)

**PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG**

2022

Daftar Isi

| | |
|-----------------------------|----|
| DAFTAR ISI | 2 |
| BAB I Pendahuluan | 3 |
| BAB II Cara Kerja Program | 5 |
| BAB III Source Code Program | 9 |
| BAB IV Pengujian Program | 18 |
| LAMPIRAN | 25 |

BAB 1

Pendahuluan

15-Puzzle adalah salah satu permainan dengan menggunakan papan yang berisikan angka 1 sampai 15 yang tersebar 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.

Salah satu pendekatan yang dapat digunakan untuk menyelesaikan persoalan 15-Puzzle ialah dengan mengimplementasikan algoritma *Branch and Bound*. 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.



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 *solvable*, begitu juga sebaliknya. Apabila susunan ubin *solvable*, 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

Cara Kerja Program

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 pada interface-nya. 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 dapat 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, dapat dilakukan dengan menekan tombol ‘*Show Details*’. Adapun berikut ini merupakan beberapa fungsi yang terdapat pada file gui.py.

| Fungsi | Keterangan |
|-------------|--|
| puzzleLable | Berfungsi untuk memvisualisasikan puzzle yang akan dijadikan sebagai interface untuk output dari GUI |
| solveCaller | Berfungsi untuk mengeksekusi puzzle untuk diselesaikan serta menyimpan informasi lain yang berkaitan dengan output program |
| selectFile | Berfungsi untuk mengambil input file dari folder test yang ada di directory |
| saveFile | Berfungsi untuk menyimpan solusi dari problem 15-Puzzle berupa langkah-langkah penyelesaiannya |
| resetPuzzle | Berfungsi untuk me-reset visualisasi dari langkah-langkah solusi 15-Puzzle yang diberikan |
| visualize | Berfungsi sebagai button untuk memanggil fungsi solveCaller |
| showDetails | Berfungsi sebagai button untuk menampilkan detail dari solusi |

| | |
|----------------|--|
| saveFileButton | Berfungsi sebagai button untuk memanggil fungsi saveFile |
| setting | Berfungsi untuk mengatur setting dari GUI |

Tabel 2.1.1.1 Fungsi pada File gui.py

2.1.2 File puzzle.py

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

| Atribut | Keterangan |
|------------|--|
| puzzle | Menyimpan <i>puzzle</i> yang diambil dari file input |
| solution | Menyimpan urutan gerakan yang berguna untuk menyelesaikan <i>puzzle</i> |
| direction | <i>List</i> yang berisi konstanta untuk arah pergerakan (atas, bawah, kanan, kiri) |
| tempKurang | Menyimpan nilai fungsi kurang tiap urutan iterasi |
| visited | Menyimpan <i>nodes</i> 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

| Method | Keterangan |
|---------------------------|--|
| __init__(self, fileName) | Sebagai konstruktor untuk inisialisasi atribut dari kelas Puzzle |
| getPuzzle(self, fileName) | Sebagai getter puzzle dari masukan file input .txt |
| sumOfKurang(self) | Sebagai getter dari penjumlahan KURANG(i) + valueX pada puzzle |
| sumOfCost(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 Method dari Kelas Puzzle

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 yang menggunakan Algoritma Branch and Bound adalah sebagai berikut.

1. Pertama-tama, fungsi akan mengecek jumlah fungsi kurang dan X awal apakah genap atau tidak.
2. Jika genap, maka puzzle yang di-input oleh user dapat dipastikan memiliki solusi. Apabila memiliki solusi, akan dilakukan pencarian lebih lanjut. Dalam pencariannya, fungsi akan membuat heap minimum yang menyimpan informasi semua simpul, yaitu cost dari simpul hidup ($\text{sumOfCost} + \text{depth}$), 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. Apabila cost dan depth berbeda, fungsi akan membentuk simpul sampai dengan empat simpul, yaitu simpul yang dapat dicapai dari simpul sekarang, dan jika simpul tersebut valid 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 minimumCost 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

Berikut ini merupakan langkah-langkah kerja program yang berjalan sesuai dengan Algoritma Branch and Bound.

1. Program akan menerima masukan dari user pada interface yang sudah didesain dalam bentuk GUI,
2. Cek state awal puzzle menggunakan fungsi sumOfKurang . Jika hasil dari sumOfKurang adalah genap, maka puzzle dapat diselesaikan, jika sebaliknya (ganjil), maka puzzle tidak

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

3. *Generate* 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 di-generates ke dalam container (visited),
7. Bangkitkan simpul selanjutnya dari heap hingga bentuk akhir puzzle (bentuk setelah diselesaikan) ditemukan.

BAB 3

Source Code Program

2.1 Program puzzle.py

```
1  # puzzle.py
2  # Source code program utama 15-Puzzle
3
4  # IMPORT LIBRARY
5  import heapq as pq
6  import copy
7
8  # KELAS PUZZLE
9  class Puzzle:
10     # Inisialisasi
11     puzzle = [] # Puzzle sebagai array kosong
12     solution = [] # Solusi dari puzzle
13     direction = [(1, 0), (0, 1), (-1, 0), (0, -1)] # Arah perpindahan: kanan, atas, kiri,
    bawah
14     tempKurang = [0 for i in range(16)] # Nilai fungsi kurang untuk tiap iterasi
15     visited = [] # List nodes yang sudah pernah dikunjungi
16     valueX = 0 # Nilai dari X awal
17     total = 0 # Banyaknya simpul yang dibangkitkan
18     minimumCost = (10 ** 9) + 7 # Minimum cost
19
20     '''
21     /* *** __INIT__ *** */
22     ** Konstruktor untuk inisialisasi atribut dari kelas Puzzle
23     ** param: self
24     ** param: fileName (nama file .txt yang menjadi input)
25     '''
26     def __init__(self, fileName):
27         self.puzzle = []
28         self.solution = []
29         self.tempKurang = [0 for i in range(16)]
30         self.visited = []
31         self.valueX = 0
32         self.total = 0
33         self.minimumCost = 1e9 + 7
34         self.getPuzzle(fileName)
35
36     '''
37     /* *** GET PUZZLE *** */
38     ** Getter puzzle dari masukan file input .txt
39     ** param: self
40     ** param: fileName (nama file .txt yang menjadi input)
```

```

41     '''
42     def getPuzzle(self, fileName):
43         tempPuzzle = []
44         puzzleSet = set({})
45         with open(fileName) as f:
46             lines = f.readlines()
47             if (len(lines) == 4):
48                 for line in lines:
49                     if (len(line.split()) == 4):
50                         tempPuzzle.append([int(i) for i in line.split()])
51                         for i in line.split():
52                             puzzleSet.add(int(i))
53                     else:
54                         return
55                 else:
56                     return
57             if (puzzleSet != {i for i in range(16)}):
58                 return
59             self.puzzle = tempPuzzle
60
61     '''
62     /* *** SUM OF KURANG *** */
63     ** Fungsi yang mengembalikan hasil penjumlahan dari KURANG(i) + valueX pada puzzle
64     ** param: self
65     ** return: kurang (hasil penjumlahan dari KURANG(i) + valueX pada puzzle)
66     '''
67     def sumOfKurang(self):
68         cost = 0
69         flat = [i for j in self.puzzle for i in j]
70         for i in range(len(flat)):
71             temp = 0
72             if (((flat[i] == 0) and (((i // 4) % 2 == 1) and (i % 2 == 0)) or (((i // 4) %
2 == 0) and (i % 2 == 1))))):
73                 self.valueX = 1
74                 for j in range(i + 1, len(flat)):
75                     if ((flat[i] > flat[j]) or (flat[i] == 0) and (flat[j] != 0)):
76                         temp += 1
77                         cost += 1
78                 self.tempKurang[flat[i]] = temp
79             kurang = cost + self.valueX
80             return kurang
81
82     '''
83     /* *** SUM OF COST *** */
84     ** Fungsi yang mengembalikan total cost yang diperlukan oleh puzzle
85     ** param: self
86     ** param: puzzle (puzzle yang ingin dihitung cost-nya)

```

```

87     ** return: cost (total cost untuk puzzle)
88     '''
89     def sumOfCost(self, puzzle):
90         flat = [i for j in puzzle for i in j]
91         cost = 0
92         for i in range(len(flat)):
93             if ((i + 1) % 16 != flat[i]):
94                 cost += 1
95         return cost
96
97     '''
98     /* *** GET ZERO POSITION *** */
99     ** Fungsi yang mengembalikan koordinat dari sel kosong (nilai elemen = 0) pada puzzle
100    ** param: self
101    ** param: puzzle (puzzle yang ingin dicari koordinat sel kosongnya)
102    ** return: koordinat sel kosong pada puzzle
103    '''
104    def getZeroPos(self, puzzle):
105        for i in range(len(puzzle)):
106            for j in range(len(puzzle[0])):
107                if (puzzle[i][j] == 0):
108                    return (i, j)
109        return (-1, -1)
110
111    '''
112    /* *** SOLVE *** */
113    ** Fungsi yang mengembalikan sum of fungsi kurang, memecahkan puzzle dengan algoritma
    Branch and Bound
114    ** param: self
115    ** return: kurang (sum of kurang function, solution)
116    '''
117    def solve(self):
118        kurang = self.sumOfKurang()
119        if ((kurang % 2) == 0):
120            heap = []
121            cost = self.sumOfCost(self.puzzle)
122            pq.heappush(heap, (cost, 0, copy.deepcopy(self.puzzle), []))
123            while (len(heap) > 0):
124                currentCost, depth, currentPuzzle, path = pq.heappop(heap)
125                self.visited.append(currentPuzzle)
126                self.total += 1
127                if ((currentCost <= self.minimumCost) and (currentCost != depth)):
128                    # Transisi
129                    x0, y0 = self.getZeroPos(currentPuzzle)
130                    for dx, dy in self.direction:
131                        if ((0 <= x0+dx <4) and (0 <= y0+dy <4)): # Kasus koordinat tidak
valid

```

```

132             newPuzzle = copy.deepcopy(currentPuzzle)
133             newPuzzle[x0][y0], newPuzzle[x0+dx][y0+dy] =
    newPuzzle[x0+dx][y0+dy], newPuzzle[x0][y0]
134             newPath = copy.deepcopy(path)
135             newPath.append((dx, dy))
136             cost = self.sumOfCost(newPuzzle)
137             if (not newPuzzle in self.visited):
138                 pq.heappush(heap, (cost + depth + 1, depth + 1, newPuzzle,
    newPath))
139             else:
140                 if (currentCost > self.minimumCost): # Bound
141                     continue
142                 if (currentCost == depth): # Solusi ditemukan
143                     if (currentCost < self.minimumCost):
144                         self.minimumCost = currentCost
145                         self.solution = path
146                     continue
147             return kurang

```

2.2 Program gui.py

```

# gui.py
# Source code program untuk Graphical User Interface (GUI)

# IMPORT LIBRARY DAN PUZZLE
from tkinter import *
from tkinter import ttk
from tkinter import filedialog as fd
import tkinter
import os
import time
from puzzle import *
from tkinter.messagebox import showinfo

# INISIALISASI VARIABEL GLOBAL
# Detail untuk memecahkan puzzle
tempKurang = []
valueKurang = 0
tempX = []
minimumCost = 0
duration = 0
total = 0

# File input
file = ""

# Default config untuk animasi puzzle

```

```

puzzleLabels = [[0 for i in range(4)] for j in range(4)]
puzzleInit = [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 0]]
zeroPos = (3, 3)
tempMoves = []

'''
/* *** PUZZLE LABEL *** */
** Visualizer puzzle yang akan dijadikan sebagai interface untuk output dari GUI
'''
def puzzleLable():
    for i in range(4):
        for j in range(4):
            puzzleLabels[i][j] = Label(root, text = str(i * 4 + j + 1), font = ('Arial', 9),
                height = 1, width = 2, anchor = CENTER,
                borderwidth = 2, relief = "solid", bg = 'yellow')
            puzzleLabels[i][j].place(x = 110 + 20 * j, y = 120 + 20 * i)
            puzzleLabels[3][3].config(text = " ")

'''
/* *** SOLVE CALLER *** */
** Fungsi untuk mengeksekusi puzzle dengan memanggil fungsi solve yang dapat memecahkan
** puzzle dengan menggunakan algoritma Branch and Bound
'''
def solveCaller():
    global puzzleInit, zeroPos, moves, puzzleLabels
    global tempKurang, valueKurang, X, minimumCost, 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()
            start = time.time()
            # Memanggil fungsi solve untuk memecahkan puzzle menggunakan algoritma Branch and
Bound
            valueKurang = psolver.solve()
            end = time.time()
            print('Done')
            duration = end - start
            if ((valueKurang % 2) == 0):
                showinfo('Solution', 'The solution has been found!')
                puzzleInit = 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 doesn\'t have any solution!')
        puzzleInit = copy.deepcopy(psolver.puzzle)
        moves = []
        tempKurang = copy.deepcopy(psolver.tempKurang)
        X = psolver.valueX
        minimumCost = psolver.minimumCost
        total = psolver.total
        details_button.place(x = 35, y = 255)
        resetPuzzle()

    else:
        showinfo('Solution', 'File config invalid!')
        return

else:
    showinfo('Solution', 'File config invalid!')
    return

'''
/* *** SELECT FILE *** */
** Fungsi untuk mengambil input file dari folder test yang ada di directory
'''
def selectFile():
    global file
    fileType = (('text files', '*.txt'), ('All files', '*.*'))
    name = fd.askopenfilename(title = 'Open a file', initialdir = './test/input', filetypes =
fileType)
    filename_label.config(text = "File config: " + os.path.basename(name))
    file = name

'''
/* *** SAVE FILE *** */
** Fungsi untuk menyimpan hasil solusi dari problem 15-Puzzle berupa langkah-langkah
penyelesaiannya
'''
def saveFile():
    puzzle = puzzleInit
    zeroPosY = zeroPos[0]
    zeroPosX = zeroPos[1]
    count = 0
    if (len(moves) > 0):
        f = fd.asksaveasfile(initialfile = 'Untitled.txt', defaultextension = ".txt", filetypes
= [("All Files", "*.*"), ("Text Documents", "*.txt")])
        f.write("SOLUTION STEPS: \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 *** */
** Fungsi untuk me-reset hasil visualisasi dari langkah-langkah penyelesaian 15-Puzzle
'''

def resetPuzzle():
    global zeroPos
    for i in range(4):
        for j in range(4):
            if (puzzleInit[i][j] != 0):
                puzzleLabels[i][j].config(text = str(puzzleInit[i][j]))
            else:
                zeroPos = (i, j)
                puzzleLabels[i][j].config(text = "")
    root.update()

'''
/* *** VISUALIZE *** */
** Button untuk melakukan visualisasi dengan memanggil fungsi solveCaller
'''

def visualize():
    global puzzleLabels
    resetPuzzle()
    zero = copy.deepcopy(zeroPos)

```

```

puzzle = copy.deepcopy(puzzleInit)
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]
    puzzleLabels[zx+dx][zy+dy].config(text = "")
    puzzleLabels[zx][zy].config(text = str(puzzle[zx][zy]))
    zero = (zx + dx, zy + dy)
    root.update()

'''
/* *** SHOW DETAILS *** */
** Button untuk menampilkan detail dari solusi
'''

def showDetails():
    INFO = "==== KURANG(i) =====\n"
    for i in range(1, 17):
        if (i < 16):
            INFO += "Value of KURANG(" + str(i) + ")" + ": " + str(tempKurang[i]) + "\n"
        else:
            INFO += "Value of KURANG(" + str(i) + ")" + ": " + str(tempKurang[0]) + "\n"
    INFO += "Sum of KURANG(i): " + str(valueKurang) + "\n"
    INFO += "Nodes Generated: " + str(total) + "\n"
    INFO += "Time Execution: " + str(duration * 1000) + " ms\n"
    showinfo(title = 'Details Solution', message = INFO)

'''
/* *** SAVE FILE BUTTON *** */
** Button untuk memanggil fungsi saveFile
'''

def saveFileButton():
    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)

'''
/* *** SETTING *** */
** Fungsi untuk melakukan pengaturan terhadap setting dari GUI
'''

def setting():
    # Title and size
    root.title("15-Puzzle Solver by nelsenputra")
    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__":
    # Inisialisasi GUI
    root = Tk()
    root.geometry("1000x1000")
    root['background'] = 'blue'

    puzzleLabels = [[0 for i in range(4)] for j in range(4)]
    puzzleLable()
    prompt_label = Label(root, text = 'Please input your puzzle:', font = ('Arial', 9), bg =
'blue')
    open_button = ttk.Button(root, text = 'Select File', command = selectFile)
    filename_label = Label(root, text = 'File has not been selected', font = ('Arial', 9), bg =
'blue')
    solve_button = ttk.Button(root, text = 'Solve', command = solveCaller)
    kurang_label = Label(root, text = '', font = ('Arial', 8), bg = 'blue')
    visualize_button = ttk.Button(root, text = 'Visualize', command = visualize)
    reset_button = ttk.Button(root, text = 'Reset', command = resetPuzzle)
    details_button = ttk.Button(root, text = 'Show Details', command = showDetails)

    setting()

    saveFileButton()

    root.mainloop()

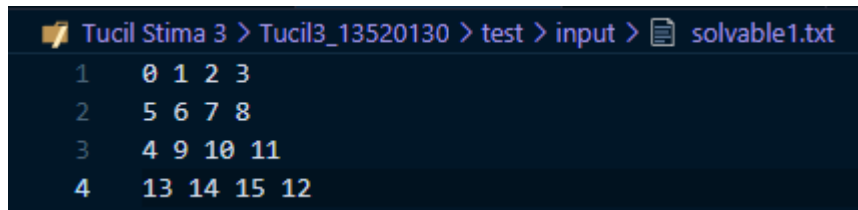
```

BAB 4

Pengujian Program

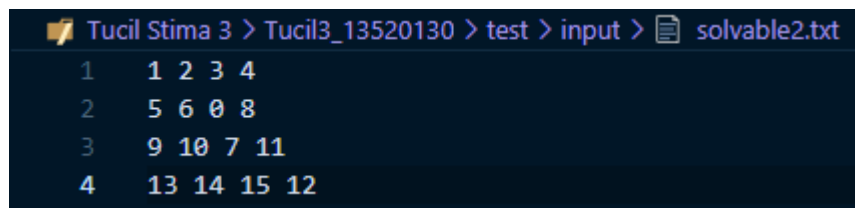
4.1 Test Case yang Solvable

4.1.1 Input



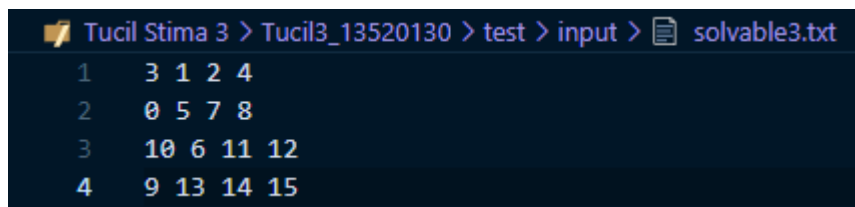
```
Tucil Stima 3 > Tucil3_13520130 > test > input > solvable1.txt
1  0 1 2 3
2  5 6 7 8
3  4 9 10 11
4  13 14 15 12
```

Gambar 4.1.1.1 File Input solvable1.txt



```
Tucil Stima 3 > Tucil3_13520130 > test > input > solvable2.txt
1  1 2 3 4
2  5 6 0 8
3  9 10 7 11
4  13 14 15 12
```

Gambar 4.1.1.2 File Input solvable2.txt



```
Tucil Stima 3 > Tucil3_13520130 > test > input > solvable3.txt
1  3 1 2 4
2  0 5 7 8
3  10 6 11 12
4  9 13 14 15
```

Gambar 4.1.1.3 File Input solvable3.txt

4.1.2 Output

```

SOLUTION STEPS:
1 DOWN
=====
5 1 2 3
0 6 7 8
4 9 10 11
13 14 15 12

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

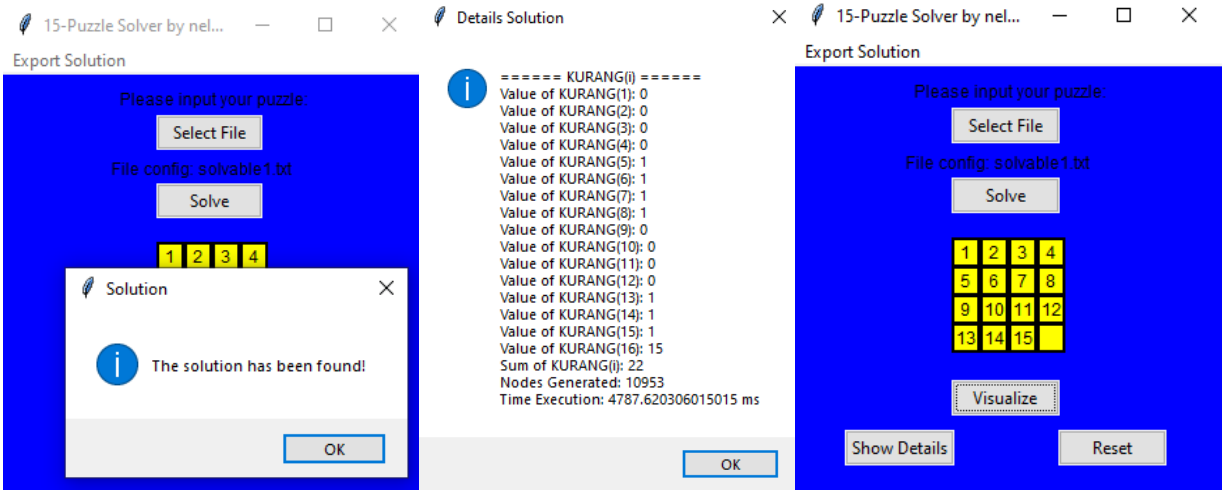
18 DOWN
=====
1 2 3 4
5 6 7 0
9 10 11 8
13 14 15 12

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

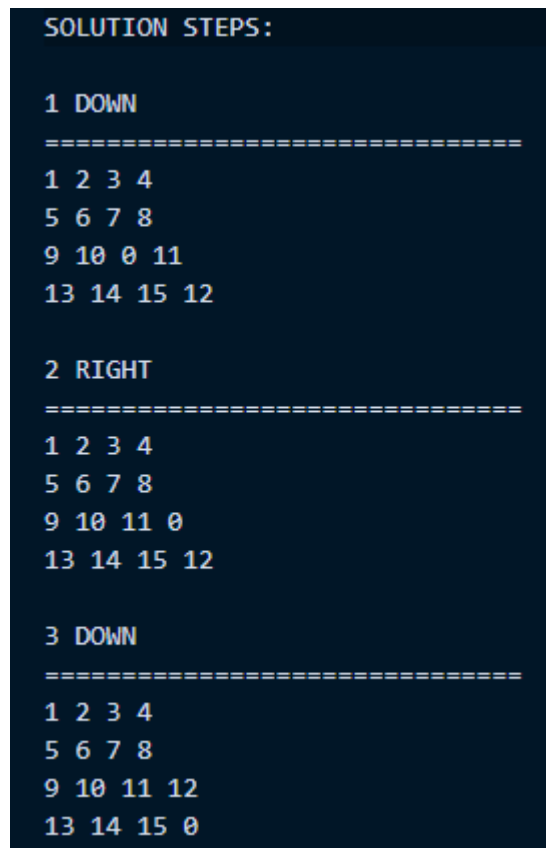
20 DOWN
=====
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 0

```

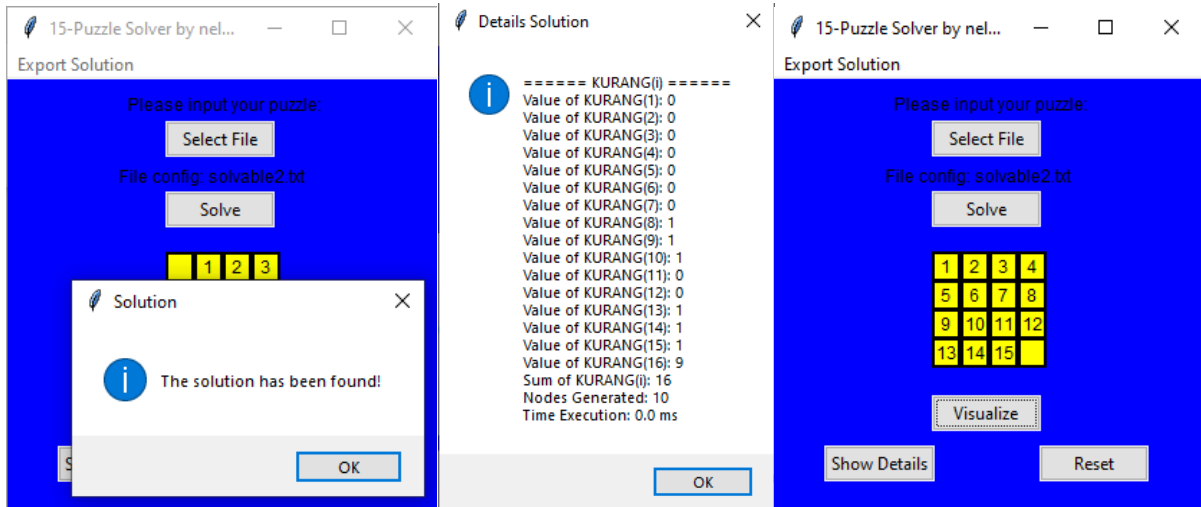
Gambar 4.1.2.1 Solusi Step Penyelesaian Puzzle solvable1.txt



Gambar 4.1.2.2 Visualisasi GUI untuk Input solvable1.txt



Gambar 4.1.2.3 Solusi Step Penyelesaian Puzzle solvable2.txt

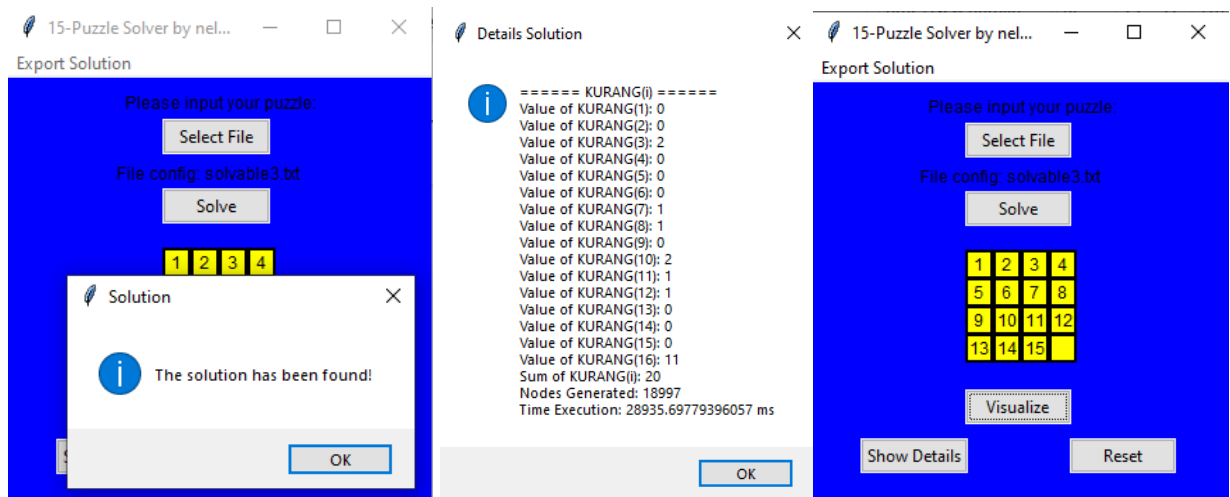


Gambar 4.1.2.4 Visualisasi GUI untuk Input solvable2.txt

| | | |
|------------------------|----------------|-----------------|
| SOLUTION STEPS: | | |
| 1 UP | 5 LEFT | 9 DOWN |
| ===== | ===== | ===== |
| 0 1 2 4 | 1 2 7 4 | 2 3 7 4 |
| 3 5 7 8 | 3 0 5 8 | 1 0 5 8 |
| 10 6 11 12 | 10 6 11 12 | 10 6 11 12 |
| 9 13 14 15 | 9 13 14 15 | 9 13 14 15 |
| 2 RIGHT | 6 LEFT | 10 RIGHT |
| ===== | ===== | ===== |
| 1 0 2 4 | 1 2 7 4 | 2 3 7 4 |
| 3 5 7 8 | 0 3 5 8 | 1 0 5 8 |
| 10 6 11 12 | 10 6 11 12 | 10 6 11 12 |
| 9 13 14 15 | 9 13 14 15 | 9 13 14 15 |
| 3 RIGHT | 7 UP | 11 UP |
| ===== | ===== | ===== |
| 1 2 0 4 | 0 2 7 4 | 2 3 0 4 |
| 3 5 7 8 | 1 3 5 8 | 1 5 7 8 |
| 10 6 11 12 | 10 6 11 12 | 10 6 11 12 |
| 9 13 14 15 | 9 13 14 15 | 9 13 14 15 |
| 4 DOWN | 8 RIGHT | 12 LEFT |
| ===== | ===== | ===== |
| 1 2 7 4 | 2 0 7 4 | 2 0 3 4 |
| 3 5 0 8 | 1 3 5 8 | 1 5 7 8 |
| 10 6 11 12 | 10 6 11 12 | 10 6 11 12 |
| 9 13 14 15 | 9 13 14 15 | 9 13 14 15 |

| | |
|--|--|
| <p>13 LEFT</p> <p>=====</p> <p>0 2 3 4</p> <p>1 5 7 8</p> <p>10 6 11 12</p> <p>9 13 14 15</p> | <p>17 LEFT</p> <p>=====</p> <p>1 2 3 4</p> <p>5 6 7 8</p> <p>0 10 11 12</p> <p>9 13 14 15</p> |
| <p>14 DOWN</p> <p>=====</p> <p>1 2 3 4</p> <p>0 5 7 8</p> <p>10 6 11 12</p> <p>9 13 14 15</p> | <p>18 DOWN</p> <p>=====</p> <p>1 2 3 4</p> <p>5 6 7 8</p> <p>9 10 11 12</p> <p>0 13 14 15</p> |
| <p>15 RIGHT</p> <p>=====</p> <p>1 2 3 4</p> <p>5 0 7 8</p> <p>10 6 11 12</p> <p>9 13 14 15</p> | <p>19 RIGHT</p> <p>=====</p> <p>1 2 3 4</p> <p>5 6 7 8</p> <p>9 10 11 12</p> <p>13 0 14 15</p> |
| <p>16 DOWN</p> <p>=====</p> <p>1 2 3 4</p> <p>5 6 7 8</p> <p>10 0 11 12</p> <p>9 13 14 15</p> | <p>20 RIGHT</p> <p>=====</p> <p>1 2 3 4</p> <p>5 6 7 8</p> <p>9 10 11 12</p> <p>13 14 0 15</p> |
| <p>21 RIGHT</p> <p>=====</p> <p>1 2 3 4</p> <p>5 6 7 8</p> <p>9 10 11 12</p> <p>13 14 15 0</p> | |

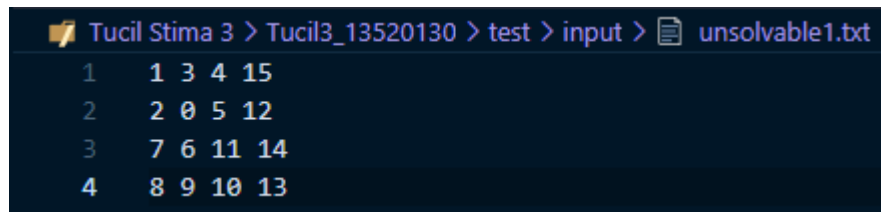
Gambar 4.1.2.5 Solusi Step Penyelesaian Puzzle solvable3.txt



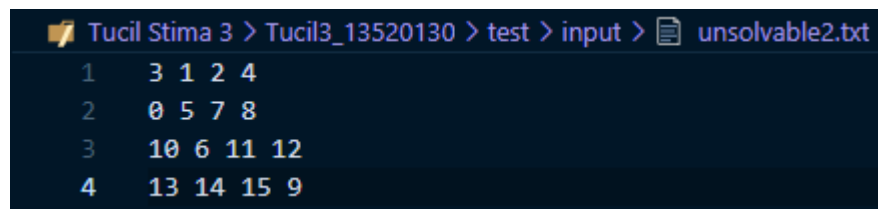
Gambar 4.1.2.6 Visualisasi GUI untuk Input solvable3.txt

4.2 Test Case yang Unsolvable

4.2.1 Input

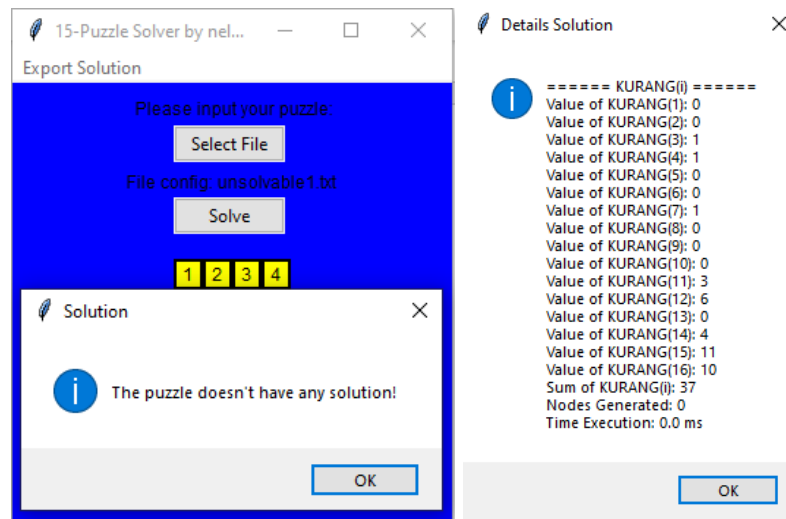


Gambar 4.2.1.1 File Input unsolvable1.txt

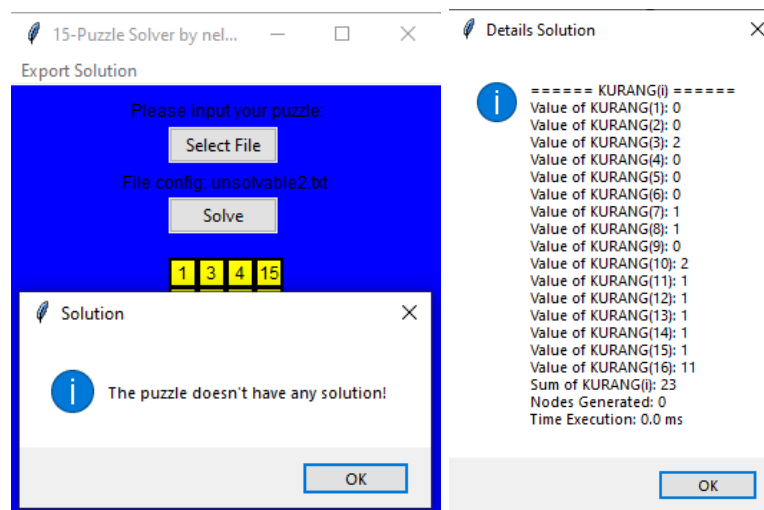


Gambar 4.2.1.2 File Input unsolvable2.txt

4.2.2 Output



Gambar 4.2.2.1 Visualisasi GUI untuk Input unsolvable1.txt



Gambar 4.2.2.1 Visualisasi GUI untuk Input unsolvable2.txt

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/nelsenputra/Tucil3_13520130