

TUGAS KECIL 03
PENYELESAIAN PERSOALAN 15-PUZZLE DENGAN
ALGORITMA BRANCH AND BOUND
IF2211 – STRATEGI ALGORITMA



Disusun oleh:

13520034 - Bryan Bernigen

Institut Teknologi Bandung
Jl. Ganesha NO. 10, Bandung 40132
2022

Daftar Isi

Cara Kerja Algoritma Branch and Bound	3
Algoritma Branch and Bound Dengan Visualisasi	4
Source Code	8
PriotizedItem.py.....	8
Gui.py	8
fifteen_puzzle.py.....	10
main.py	15
Hasil Pengujian.....	17
1. Berhasil1.txt	17
2. Berhasil2.txt	17
3. Berhasil3.txt	19
4. Gagal1.txt.....	20
5. Gagal2.txt.....	21
Lampiran	22

Cara Kerja Algoritma Branch and Bound

Part 1: Penghitungan Nilai Pembatas

1. Perhitungan 1:
Evaluasi setiap nilai yang ada (1...16) apakah terdapat nilai lain yang lebih kecil dari nilai yang sedang dievaluasi, namun lokasinya berada pada indeks yang lebih besar. Hitung jumlah seluruh nilai yang lokasinya tidak tepat.
2. Perhitungan 2:
Khusus untuk langkah pertama (akarnya), hitung juga nilai lokasi nilai 16 (kotak kosong) dengan rumus sebagai berikut: 1 jika kotak kosong berada pada indeks $1,3,4,6 + 8i, i=0,1$. Dan 0 jika kotak kosong berada pada indeks $0,2,5,7 + 8i, i=0,1$. Untuk visualisasi lebih jelas silakan mengacu pada bab selanjutnya
3. Jika hasil perhitungan 1 + hasil perhitungan 2 bernilai ganjil, maka puzzle tersebut tidak dapat diselesaikan. Jika hasil perhitungan 1 dan perhitungan 2 bernilai genap, maka puzzle akan dikerjakan. Awali dengan memasukkan akar ke queue prioritas lalu ikuti langkah-langkah pada part 2.

Part 2: Branch and Bound

1. Ambil nilai pertama yang ada pada antrean (nilai diurut berdasarkan nilai terkecil yang didapat pada perhitungan 1 part 1).
2. Bentuk semua pergerakan (atas, bawah, kiri, kanan) yang mungkin dari keadaan tersebut. Pergerakan dibentuk jika memenuhi 2 syarat, yakni:
 - a. Tidak menghasilkan keadaan yang sama dengan ranting pemanggilnya (contoh: command LEFT tidak akan diikuti oleh command RIGHT karena kedua command tersebut hanya akan bolak-balik menghasilkan keadaan yang sama)
 - b. Tidak membuat kotak kosong menjadi out of bounds. (contoh: ketika kotak kosong berada di kanan atas, command UP dan RIGHT tidak akan dipanggil)
 - c. (Opsional untuk optimasi) cek apakah state tersebut sudah ada di hash map atau belu. Jika belum ada, state tersebut akan dibangkitkan dan dibuat true di hash map.
3. Untuk setiap command yang dihasilkan, hitung jumlah nilai yang berada tidak pada tempatnya. Setelah itu, masukkan hasil pergerakan tersebut ke queue prioritas dengan nilai perhitungan ditambah kedalaman node tersebut sebagai prioritasnya (semakin sedikit nilai yang tidak pada tempatnya, semakin besar prioritasnya)
4. Jika hasil perhitungan nilai tidak pada tempatnya = 0, maka solusi ditemukan.
5. Jika solusi belum ditemukan, lakukan langkah 1-5 part 2 sampai solusi ditemukan karena berdasarkan perhitungan, solusi dapat dipastikan ada. (tidak perlu menunggu queue kosong karena solusi dijamin ditemukan menurut teorema)

Algoritma Branch and Bound Dengan Visualisasi

Part 1: Perhitungan Nilai Pembatas

1. Perhitungan 1

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

I	Jumlah Tidak Sesuai
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	1 (7)
9	1 (7)
10	1 (7)
11	0
12	0
13	1 (12)
14	1 (12)
15	1 (12)
16	9 (8,9,10,7,11,13,14,15,12)
Total	15

2. Perhitungan 2

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

Daerah berwarna Abu → Perhitungan 2 bernilai 1

Daerah berwarna Putih → Perhitungan 2 bernilai 0

3. Perhitungan 1 + 2

Dari Perhitungan 1 didapat total = 15
 Dari perhitungan 2 didapat nilai = 1
 Jumlah = 16
 Karena jumlah genap, maka solusi dapat dicari

Part 2: Branch and Bound

1. Ambil node pertama dari priority queue

Queue	Nilai	5	5	6	7	7
	Node ke	1	2	5	3	4

Pada contoh diatas, maka diambil node ke-1 sehingga:

Node yang dipilih = Node ke-1

Queue	Nilai	5	6	7	7
	Node ke	2	5	3	4

2. Bentuk Pergerakan yang Memungkinkan & Hitung nilainya

Misal Node ke-1 merupakan hasil dari command DOWN dari node akar memiliki status sebagai berikut:

NODE 1

Atas : False (bernilai false karena Node 1 dihasilkan dari command DOWN)

Bawah : True

Kiri : True

Kanan : True

Parent : Node 0 (akar)

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

Untuk Node 1, akan digenerate untuk Command DOWN, LEFT, RIGHT sebagai berikut

NODE 6 (Command DOWN dari NODE 1)

Atas : False

Bawah : True

Kiri : True

Kanan : True

Parent : Node 1

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

Nilai = 2 (Depth) + 3 (Salah Lokasi) = 5

NODE 7 (Command LEFT dari NODE 1)

Atas : True

Bawah : True

Kiri : True

Kanan : False

Parent : Node 1

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

Nilai = 2 + 5 = 7

NODE 8 (Command RIGHT dari NODE 1)

Atas : True

Bawah : True

Kiri : False

Kanan : True

Parent : Node 1

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

Nilai = 2 + 5 = 7

Sehingga Setelah Node 1 di cek, maka priority queue menjadi:

Queue	Nilai	5	5	6	7	7	7	7
	Node ke	2	6	5	3	4	7	8

Source Code

PriotizedItem.py

Kode untuk membuat item sebuah priority Queue

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

@dataclass(order=True)
class PrioritizedItem:
    priority: int
    item: Any=field(compare=False)
```

Gui.py

Program yang dijalankan untuk menampilkan GUI

```
import tkinter as tk
from tkinter import messagebox
from fifteen_puzzle import *
import time

class Fifteen_Puzzle:
    def __init__(self):
        for i in range(4):
            for j in range(4):
                entry = tk.Entry(window, width = 5)
                entry.grid(row = i, column = j)
                entry = entry.insert(tk.END, "")

    def show_each_move(self, puzzle, iterasi_ke):
        for i in range(4):
            for j in range(4):
                entry = tk.Entry(window, width = 5)
                entry.grid(row = i, column = j)
                if (puzzle.list[4*i + j] == 16):
                    entry.insert(tk.END, "")
                else:
                    entry.insert(tk.END, puzzle.list[4*i+j])
            if(not puzzle.up):
                move = "Down"
            elif(not puzzle.down):
                move = "Up"
            elif(not puzzle.left):
                move = "Right"
```



```

        elif(not puzzle.right):
            move = "Left"
        else:
            move = "-"
        text = "Langkah ke-" + str(iterasi_ke+1) + ": " + move
        tk.Label(window, text = text).grid(row = mod(iterasi_ke,6), column =
6)

    def showall(self, array_fifteen_puzzle):
        global window
        for i in range(len(array_fifteen_puzzle)-1,0,-1):
            self.show_each_move(array_fifteen_puzzle[i],len(array_fifteen_puzz
le)-i-1)
            time.sleep(1)
            window.update()

def SolveOnClick():
    global array_fifteen_puzzle
    global layouts
    filepath = file_entry.get()
    array = openFile(filepath)
    hasil = solve(array)
    if(len(hasil)==0):
        tk.messagebox.showinfo("Error","Puzzle Tidak Bisa Diselesaikan!")
    else:
        layout.showall(hasil)

window = tk.Tk()
window.geometry("300x150")
window.title("15-Puzzle Solver")

layout = Fifteen_Puzzle()

file_entry = tk.Entry(window, text = "Enter file name")
file_entry.place(x = 0, y = 85)

solve_button = tk.Button(window, text = "Solve", width = 15, command =
SolveOnClick)
solve_button.place(x = 5, y = 110)

window.mainloop()

```

fifteen_puzzle.py

Program yang akan mengitung dan menyelesaikan fifteen puzzle tersebut

```
from operator import mod
from queue import PriorityQueue
from PriotizedItem import PrioritizedItem
import time
import os

class fifteen_puzzle:
    def __init__(self, list, up, down, left, right):
        self.list = list
        self.up = up
        self.down = down
        self.left = left
        self.right = right
        self.lokasi_kosong = 0
        self.nilai = 0
        self.kedalaman = 0
        self.id_parent = 0

    #Hitung total i yang lokasinya salah
    def hitungan_awal(self):
        hitunngan_awal = 0
        #iterasi untuk mencari posisi yang salah dari nilai 1 - 16
        for i in range(16):
            #cari posisi nilai ke-i di list
            for j in range(16):
                if(self.list[j] == i+1):
                    break

            #hitung banyaknya nilai k yang lebih kecil dari i tapi posisi
            #lebih besar dari i
            for k in range(16-j-1):
                if(self.list[k+j+1] < i+1):
                    hitunngan_awal += 1
        return hitunngan_awal

    #Hitung apakah nilai kosong(nilai ke-16) ada di lokasi kosong
    def arsir_atau_tidak(self):
        for i in range(16):
            if(self.list[i] == 16):
                break
        i = mod(i,8)
        if(i==1 or i==3 or i==4 or i==6):
            return 1
        return 0

    #Menyatakan apakah puzzle bisa di selesaikan atau tidak
```

```

def solveable(self):
    nilai = self.hitungan_awal() + self.arsir_atau_tidak()
    if(mod(nilai,2)==0):
        return True
    return False

def salah_posisi(self):
    salah=0
    for i in range (16):
        if(self.list[i] != i+1):
            salah+=1
        if(self.list[i]==16):
            self.lokasi_kosong = i
    return salah

#Menggerakan puzzle
def move(self,queue,id_parent, hashmap):
    if(self.salah_posisi()==0):
        print("Puzzle berhasil diselesaikan!!")
        return False
    if(self.up):
        if(self.lokasi_kosong>3):
            self.moveup(queue,id_parent,hashmap)
    if(self.down):
        if(self.lokasi_kosong<12):
            self.movedown(queue,id_parent,hashmap)
    if(self.left):
        if(self.lokasi_kosong%4!=0):
            self.moveleft(queue,id_parent,hashmap)
    if(self.right):
        if(self.lokasi_kosong%4!=3):
            self.moveright(queue,id_parent,hashmap)
    return True

#Menggerakan puzzle ke atas
def moveup(self,queue,id_parent,hashmap):
    list = self.list[:]
    list[self.lokasi_kosong], list[self.lokasi_kosong-4] =
list[self.lokasi_kosong-4], list[self.lokasi_kosong]
    list = fifteen_puzzle(list, False, True, True, True)
    list.kedalaman=self.kedalaman+1
    cek_hash, key = list.in_hashmap(hashmap)
    if(cek_hash):
        return
    else:
        list.nilai+=list.kedalaman + list.salah_posisi()
        list.id_parent = id_parent
        queue.put(PrioritizedItem(list.nilai, list))

```

```

        hashmap[key] = True

    #Menggerakan puzzle ke bawah
    def movedown(self,queue,id_parent,hashmap):
        list = self.list[:]
        list[self.lokasi_kosong], list[self.lokasi_kosong+4] =
list[self.lokasi_kosong+4], list[self.lokasi_kosong]
        list = fifteen_puzzle(list, False, True, True, True)
        cek_hash, key = list.in_hashmap(hashmap)
        if(cek_hash):
            return
        else:
            list.kedalaman=self.kedalaman+1
            list.nilai+=list.kedalaman + list.salah_posisi()
            list.id_parent = id_parent
            queue.put(PrioritizedItem(list.nilai, list))
            hashmap[key] = True

    #Menggerakan puzzle ke kiri
    def moveleft(self,queue,id_parent,hashmap):
        list = self.list[:]
        list[self.lokasi_kosong], list[self.lokasi_kosong-1] =
list[self.lokasi_kosong-1], list[self.lokasi_kosong]
        list = fifteen_puzzle(list, True, True, True, False)
        cek_hash, key = list.in_hashmap(hashmap)
        if(cek_hash):
            return
        else:
            list.kedalaman=self.kedalaman+1
            list.nilai+=list.kedalaman+list.salah_posisi()
            list.id_parent = id_parent
            queue.put(PrioritizedItem(list.nilai, list))
            hashmap[key] = True

    #Menggerakan puzzle ke kanan
    def moveright(self,queue,id_parent,hashmap):
        list = self.list[:]
        list[self.lokasi_kosong], list[self.lokasi_kosong+1] =
list[self.lokasi_kosong+1], list[self.lokasi_kosong]
        list = fifteen_puzzle(list, True, True, False, True)
        cek_hash, key = list.in_hashmap(hashmap)
        if(cek_hash):
            return
        else:
            list.kedalaman=self.kedalaman+1
            list.nilai+=list.kedalaman+list.salah_posisi()
            list.id_parent = id_parent
            queue.put(PrioritizedItem(list.nilai, list))

```

```

        hashmap[key] = True

#Mengecek sate puzzle dalam hash table
def in_hashmap(self,hash_table):
    key=''
    for i in range(16):
        key+=str(self.list[i])+'- '
    if(key in hash_table):
        return True,"-"
    else:
        return False,key

#Print Puzzle
def print_puzzle(self):
    for i in range (4):
        print("    ",end="")
        for j in range (4):
            if(self.list[i*4+j] == 16):
                print("    ", end=" ")
            else:
                print("%2d" % self.list[i*4+j], end=" ")
        print("")

#Fungsi Tambahan Untuk GUI
#Baca data dari file
def openFile(nama_file):
    curr_dir = os.path.dirname(os.path.abspath(__file__))
    try:
        f = open(curr_dir+"../test/"+nama_file, "r")
    except:
        print("File tidak ditemukan")
        exit()
    array=[]
    for x in f:
        x = x.split(' ')
        array.append((x[0]))
        array.append((x[1]))
        array.append((x[2]))
        array.append((x[3]))
    for i in range(16):
        if array[i]=='- ':
            array[i]=16
        else:
            array[i]=int(array[i])
    f.close()
    return array

def solve(array):

```

```

#inisialisasi variabel
puzzle = fifteen_puzzle(array,True,True,True,True);
selesai = False
id_dikerjakan = 0
queue = PriorityQueue()
urutan_pengerjaan = []
hasil_print = []
puzzle.salah_posisi()
puzzle_awal = puzzle
hashmap = {}

#memasukkan akar ke queue
start_time = time.time()
queue.put(PrioritizedItem(puzzle.nilai, puzzle))
while(not queue.empty() and not selesai):
    puzzle = queue.get().item
    urutan_pengerjaan.append(puzzle)
    id_dikerjakan += 1;
    if(not puzzle.move(queue, id_dikerjakan,hashmap)):
        stop_time = time.time()
        break

#Menampilkan Hasil
while(not queue.empty()):
    queue.get()
    id_dikerjakan+=1

# Mencari dan Mencetak Path yang menghasilkan tujuan
hasil_print.append(puzzle_awal)
while(puzzle.id_parent != 0):
    hasil_print.append(puzzle)
    puzzle = urutan_pengerjaan[puzzle.id_parent-1]
return hasil_print

```

main.py

Program utama yang dijalankan. Program akan membaca data dari file dan akan memanggil fifteen_puzzle.py. Setelah selesai, program juga akan menampilkan path menuju solusi

```
# DISCLAIMER!!!
# Semua Puzzle yang bernilai Genap dapat diselesaikan dengan
# Heuristik sesuai Spek, Namun tidak semua dapat diselesaikan
# Dengan cukup cepat sehingga jika puzzle kompleks (>30an langkah)
# Maka ada kemungkinan python akan mengalami error timeout sebelum hasil
ditemukan

import queue
from fifteen_puzzle import *
from queue import PriorityQueue
from PriotizedItem import PrioritizedItem

nama_file = input("Masukkan nama file yang ada di Folder Test: ")
print("=====")
array = openFile(nama_file)
#=====
#Solve 15 Puzzle
puzzle = fifteen_puzzle(array, True, True, True, True)
if(not puzzle.solveable()):
    print("Puzzle tidak bisa di selesaikan")
    print("karena puzzle bernilai ganjil yakni:
",puzzle.hitungan_awal()+puzzle.arsir_atau_tidak())
    print("=====")
    puzzle.print_puzzle()
    print("=====")
else:
    #inisialisasi variabel
    selesai = False
    id_dikerjakan = 0
    queue = PriorityQueue()
    urutan_pengerjaan = []
    hasil_print = []
    puzzle.salah_posisi()
    puzzle_awal = puzzle
    hashmap = {}

    #memasukkan akar ke queue
    print("Searching.....")
    start_time = time.time()
    queue.put(PrioritizedItem(puzzle.nilai, puzzle))
    while(not queue.empty() and not selesai):
        puzzle = queue.get().item
        urutan_pengerjaan.append(puzzle)
```

```

        id_dikerjakan += 1;
        if(not puzzle.move(queue, id_dikerjakan,hashmap)):
            stop_time = time.time()
            break

#Menampilkan Hasil
print("Total Pengecekan: ",id_dikerjakan-1)
while(not queue.empty()):
    queue.get()
    id_dikerjakan+=1
print("Total Pembangkitan: ",id_dikerjakan-1)

# Mencari dan Mencetak Path yang menghasilkan tujuan
while(puzzle.id_parent != 0):
    hasil_print.append(puzzle)
    puzzle = urutan_pengerjaan[puzzle.id_parent-1]
hasil_print.append(puzzle_awal)
print("Puzzle diselesaikan dalam:",end=" ")
print(len(hasil_print)-1,"langkah")
print("Puzzle diselesaikan dalam:",end=" ")
print("%.10s detik" % (stop_time - start_time))
print("=====")
for i in range (len(hasil_print)-1,-1,-1):
    hasil_print[i].print_puzzle()
    print("Command: ",end="")
    if(not hasil_print[i-1].up):
        print("Down")
    elif(not hasil_print[i-1].down):
        print("Up")
    elif(not hasil_print[i-1].left):
        print("Right")
    elif(not hasil_print[i-1].right):
        print("Left")
    else:
        print("-")
print("=====")

```


Hasil Pengujian

1. Berhasil1.txt

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

> python main.py
Masukkan nama file yang ada di Folder Test: berhasil1.txt
=====
Searching.....
Puzzle berhasil diselesaikan!!
Total Pengecekan: 3
Total Pembangkitan: 9
Puzzle diselesaikan dalam: 3 langkah
Puzzle diselesaikan dalam: 0.00099229 detik
=====
  1  2  3  4
  5  6      8
  9 10  7 11
 13 14 15 12
Command: Down
=====
  1  2  3  4
  5  6  7  8
  9 10      11
 13 14 15 12
Command: Right
=====
  1  2  3  4
  5  6  7  8
  9 10 11
 13 14 15 12
Command: Down
=====
  1  2  3  4
  5  6  7  8
  9 10 11 12
 13 14 15
Command: -
=====
```

15-Puzzle Solver

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

berhasil1.txt

Solve

Langkah ke-1: Down
Langkah ke-2: Right
Langkah ke-3: Down


2. Berhasil2.txt

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

```

> python main.py
Masukkan nama file yang ada di Folder Test: berhasil2.txt
=====
Searching.....
Puzzle berhasil diselesaikan!!
Total Pengecekan: 6
Total Pembangkitan: 11
Puzzle diselesaikan dalam: 6 langkah
Puzzle diselesaikan dalam: 0.00099945 detik
=====
      1  2  3
    5  6  7  4
    9 10 11  8
   13 14 15 12
Command: Right
=====
      1      2  3
    5  6  7  4
    9 10 11  8
   13 14 15 12
Command: Right
=====
      1  2      3
    5  6  7  4
    9 10 11  8
   13 14 15 12
Command: Right
=====
      1  2  3
    5  6  7  4
    9 10 11  8
   13 14 15 12
Command: Down
=====
      1  2  3  4
    5  6  7
    9 10 11  8
   13 14 15 12
Command: Down
=====
      1  2  3  4
    5  6  7  8
    9 10 11
   13 14 15 12
Command: Down
=====
      1  2  3  4
    5  6  7  8
    9 10 11 12
   13 14 15
Command: -
=====

```


15-Puzzle Solver

1	2	3	4	Langkah ke-1: Right
5	6	7	8	Langkah ke-2: Right
9	10	11	12	Langkah ke-3: Right
13	14	15		Langkah ke-4: Down
berhasil2.txt				Langkah ke-5: Down
				Langkah ke-6: Down

Solve

3. Berhasil3.txt

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

```
> python main.py
Masukkan nama file yang ada di Folder Test: berhasil3.txt
=====
Searching.....
Puzzle berhasil diselesaikan!!
Total Pengecekan: 4530
Total Pembangkitan: 8272
Puzzle diselesaikan dalam: 21 langkah
Puzzle diselesaikan dalam: 0.32460808 detik
=====
  3  1  2  4
    5  7  8
 10  6 11 12
  9 13 14 15
Command: Down
=====
    1  2  4
  3  5  7  8
10  6 11 12
  9 13 14 15
Command: Right
=====
  1  2  4
  3  5  7  8
10  6 11 12
  9 13 14 15
Command: Right
=====
  1  2  4
  3  5  7  8
10  6 11 12
  9 13 14 15
Command: Down
=====
```

...

```

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

```

15-Puzzle Solver

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

berhasil3.txt

Solve

Langkah ke-19: Right
 Langkah ke-20: Right
 Langkah ke-21: Right
 Langkah ke-16: Down
 Langkah ke-17: Left
 Langkah ke-18: Down

4. Gagal1.txt

```

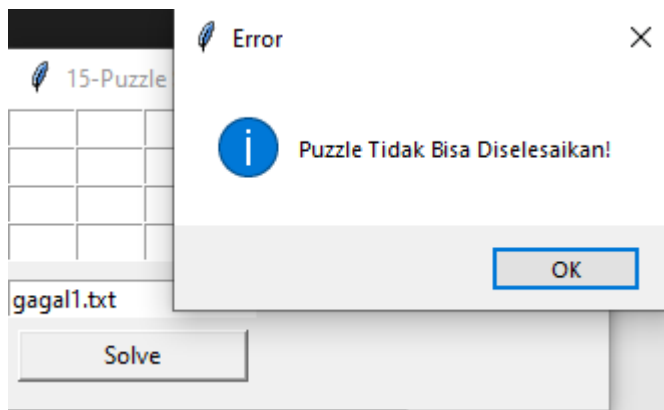
1 3 4 15
2 - 5 12
7 6 11 14
8 9 10 13

```

```

> python -u "d:\git\ITB\Sem4\Stima\Tucil3_13520034\src\main.py"
Masukkan nama file yang ada di Folder Test: gagal1.txt
=====
Puzzle tidak bisa di selesaikan
karena puzzle bernilai ganjil yakni: 37
=====
  1  3  4 15
  2   5 12
  7  6 11 14
  8  9 10 13
=====

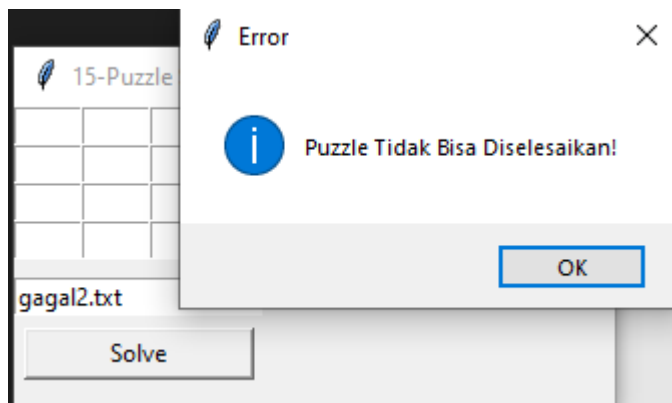
```



5. Gagal2.txt

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

```
> python -u "d:\git\ITB\Sem4\Stima\Tucil3_13520034\src\main.py"
Masukkan nama file yang ada di Folder Test: gagal2.txt
=====
Puzzle tidak bisa di selesaikan
karena puzzle bernilai ganjil yakni: 23
=====
  3  1  2  4
    5  7  8
10  6 11 12
13 14 15  9
=====
```



Lampiran

Poin	Ya	Tidak
1. Program berhasil dikompilasi	√	
2. Program berhasil <i>running</i>	√	
3. Program dapat menerima input dan menuliskan output.	√	
4. Luaran sudah benar untuk semua data uji	√	
5. Bonus dibuat	√	

Link Drive:

https://drive.google.com/drive/folders/1C_vXKN0rep4xt0j0xMcm7B-HoZUgxGih?usp=sharing