Untuk Memenuhi Tugas Mata Kuliah Kecil 3

Strategi Algoritma (IF 2211)

Disusun Oleh:

Jaya Mangalo Soegeng Rahardjo

NIM : 13520015

TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2021

A. Algoritma Program

Step 0.5: Display

Program mengerluarkan Basic GUI dengan input bar yang bisa digunakan untuk menulis file .txt yang terletak di folder ../test/.

Step 1: Preparation and Validity Checking

Program akan membaca input user dan mencari file dengan nama yang sama serta mengubah isi file menjadi matrix.

Program kemudian akan menggunakan fungsi *isReachable* dan *KurangIplusX* yang menghitung KurangI serta X dan menentukan apakah matriks input memiliki solusi atau tidak.

Step 2: Solving

Jika Puzzle memiliki solusi, maka program akan membuat Node *root* dengan isi matrix awal dan *path* [“none”]. Node tersebut di queue kedalam priorityqueue.

Lalu program akan masuk while-loop yang akan:

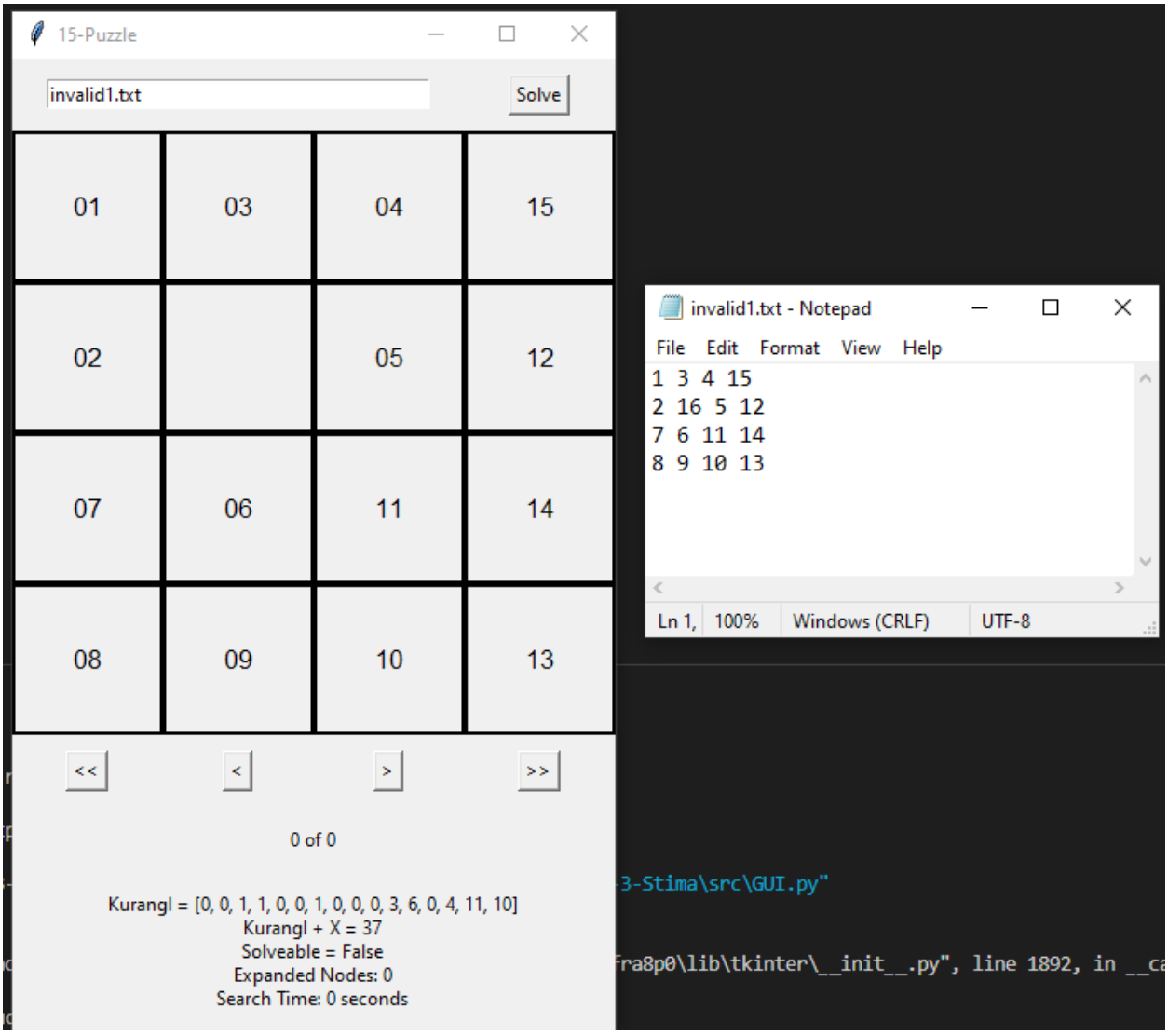
1. Mengecek jika node adalah hasil akhir yang diinginkan, jika iya loop berhenti dan *path* solusi dan *Expanded_Node* akan direturn.
2. Jika tidak, program akan mengattempt untuk menambahkan 4 node (dimana 16 bergerak atas, bawah, kiri, kanan)
   i) Program akan mengecek jika pergerakan directional valid, misalnya jika 16 terletak di baris paling atas, artinya pergerakan ke atas tidak valid.
   ii) Program juga tidak memperbolehkan pergerakan yang berlawanan dengan pergerakan sebelumnya, misalnya jika node sebelumnya bergerak ke atas, berarti pergerakan selanjutnya tidak boleh ke bawah.
   iii) Program juga mengecek jika state tersebut sudah di-expand atau belum, program akan mengubah matrix menjadi key yang kemudian dimasukkan ke hash table. Karena hash table memiliki instant lookup, pengecekan ini sangat cepat dan efisien. JIka state sudah di-expand, maka program tidak akan menambahkan pergerakan tersebut.

Step 3: Display

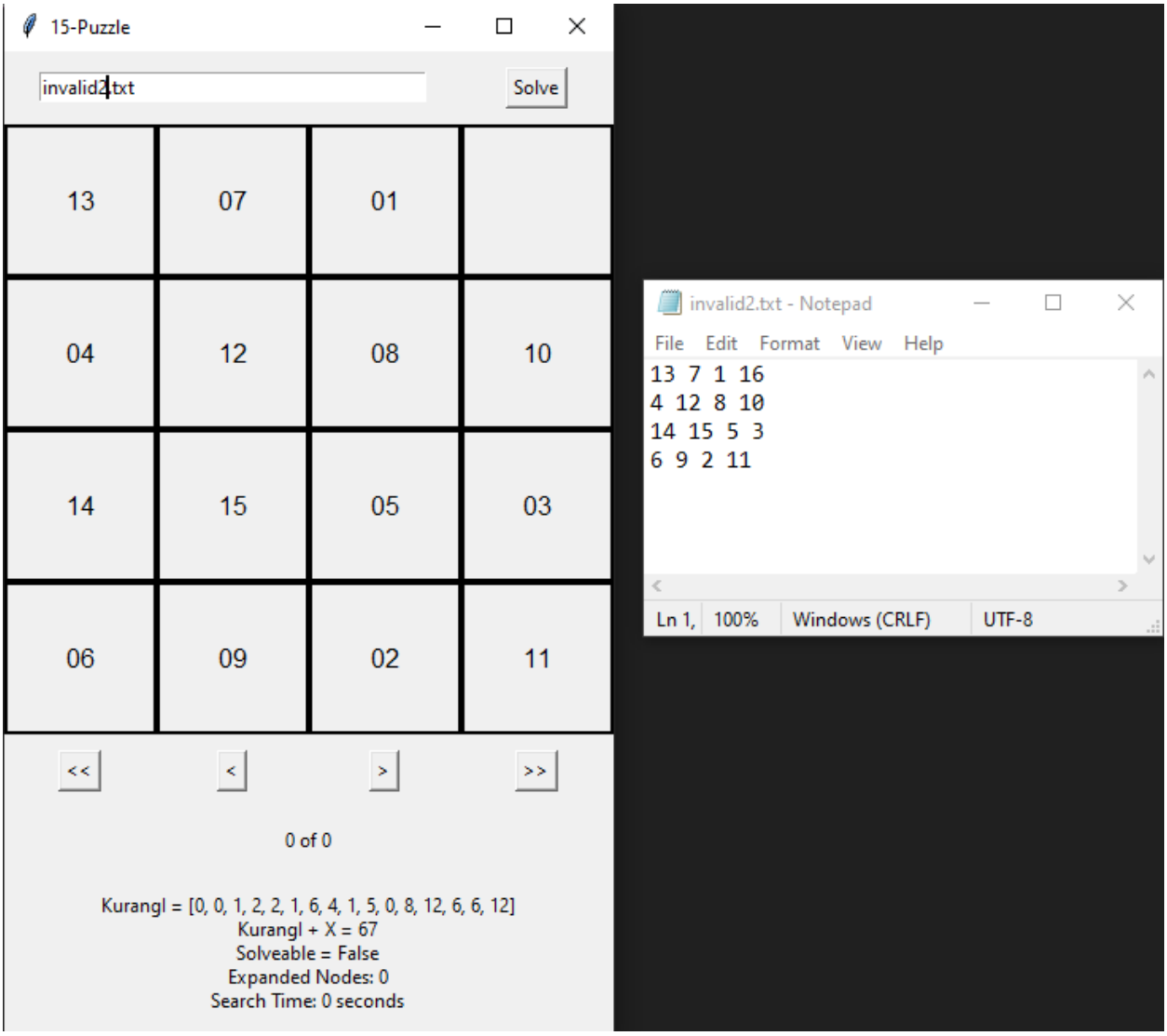
Program men-display interactive 4x4 matrix yang bisa digerakkan dengan input buttons yang disediakan. Program juga men-display informasi relevan seperti KurangI, KurangI+X, Reachable, Expanded_Nodes, SearchTime.

B. Test Case

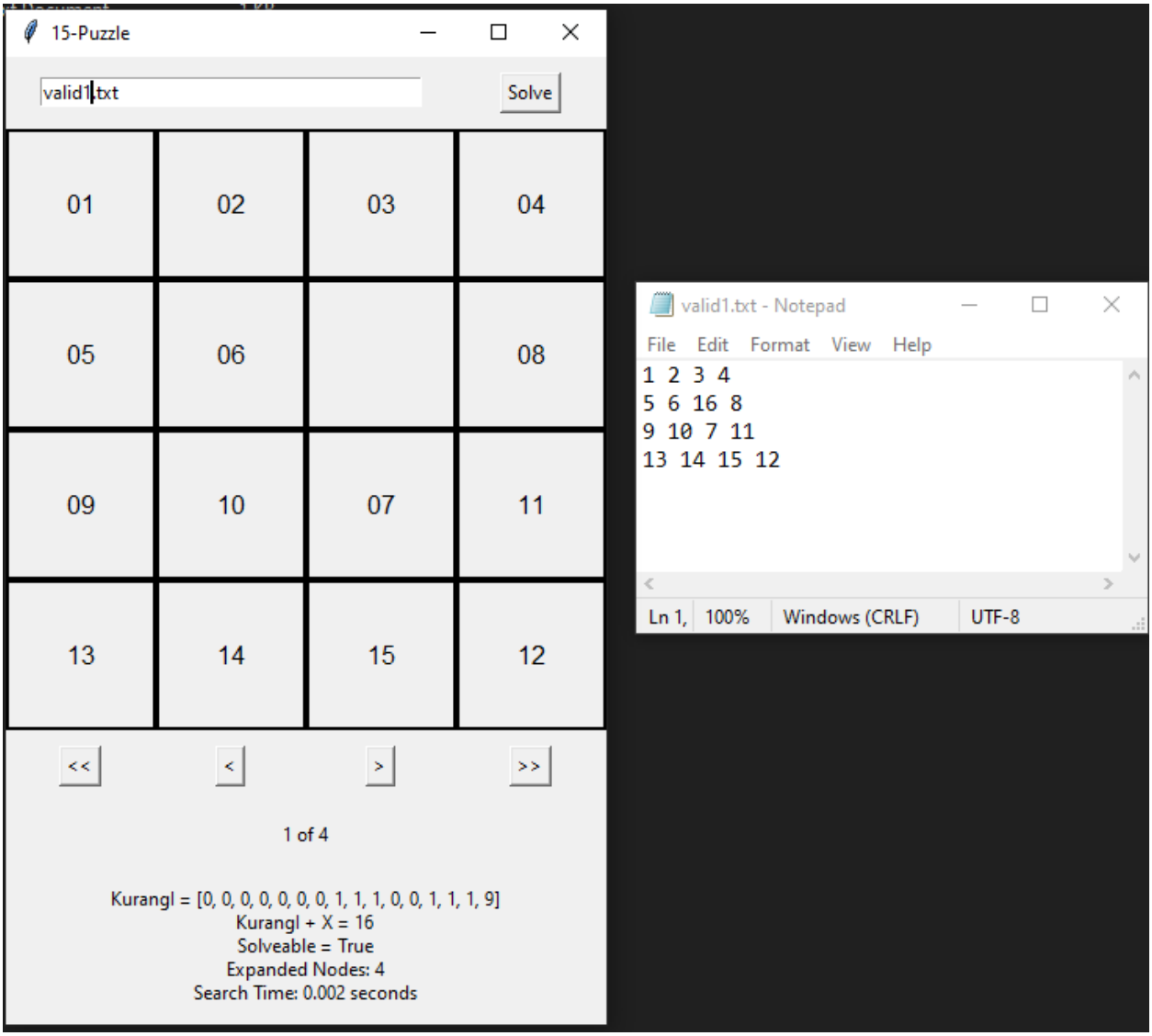
invalid1.txt



invalid2.txt



valid1.txt


15-Puzzle
valid1.txt
Solve
01
02
03
04
05
06
08
09
10
07
11
13
14
15
12
<<
<
>
>>
1 of 4
Kurangl = [0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 9]
Kurangl + X = 16
Solveable = True
Expanded Nodes: 4
Search Time: 0.002 seconds
valid1.txt - Notepad
File Edit Format View Help
1 2 3 4
5 6 16 8
9 10 7 11
13 14 15 12
Ln 1, 100% Windows (CRLF) UTF-8

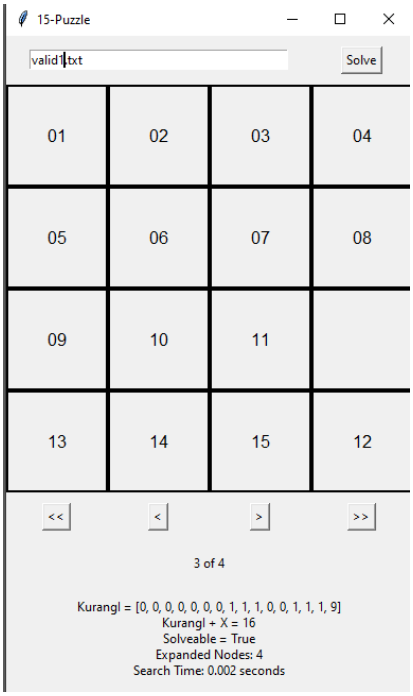
Step 1


15-Puzzle
valid1.txt
Solve
01
02
03
04
05
06
08
09
10
07
11
13
14
15
12
<<
<
>
>>
1 of 4
Kurangl = [0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 9]
Kurangl + X = 16
Solveable = True
Expanded Nodes: 4
Search Time: 0.002 seconds

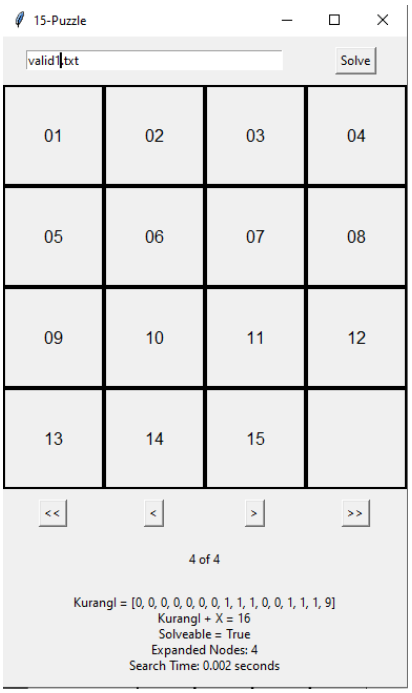
Step 2


15-Puzzle
valid1.txt
Solve
01
02
03
04
05
06
07
08
09
10
11
13
14
15
12
<<
<
>
>>
2 of 4
Kurangl = [0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 9]
Kurangl + X = 16
Solveable = True
Expanded Nodes: 4
Search Time: 0.002 seconds

Step 3

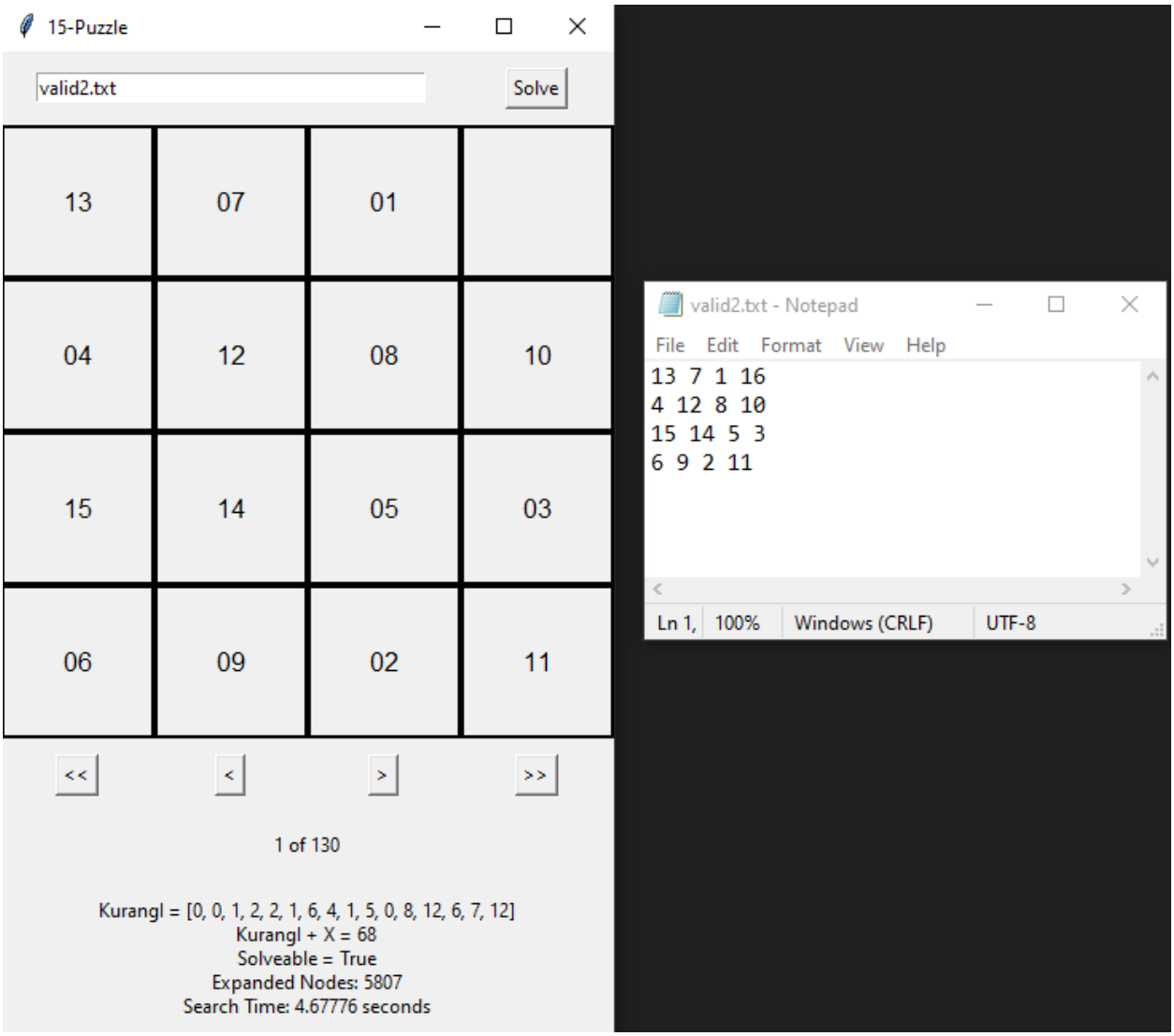

15-Puzzle
valid1.txt
Solve
01
02
03
04
05
06
07
08
09
10
11
13
14
15
12
<<
<
>
>>
3 of 4
Kurangl = [0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 9]
Kurangl + X = 16
Solveable = True
Expanded Nodes: 4
Search Time: 0.002 seconds

Step 4


15-Puzzle
valid1.txt
Solve
01
02
03
04
05
06
07
08
09
10
11
12
13
14
15
<<
<
>
>>
4 of 4
Kurangl = [0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 9]
Kurangl + X = 16
Solveable = True
Expanded Nodes: 4
Search Time: 0.002 seconds

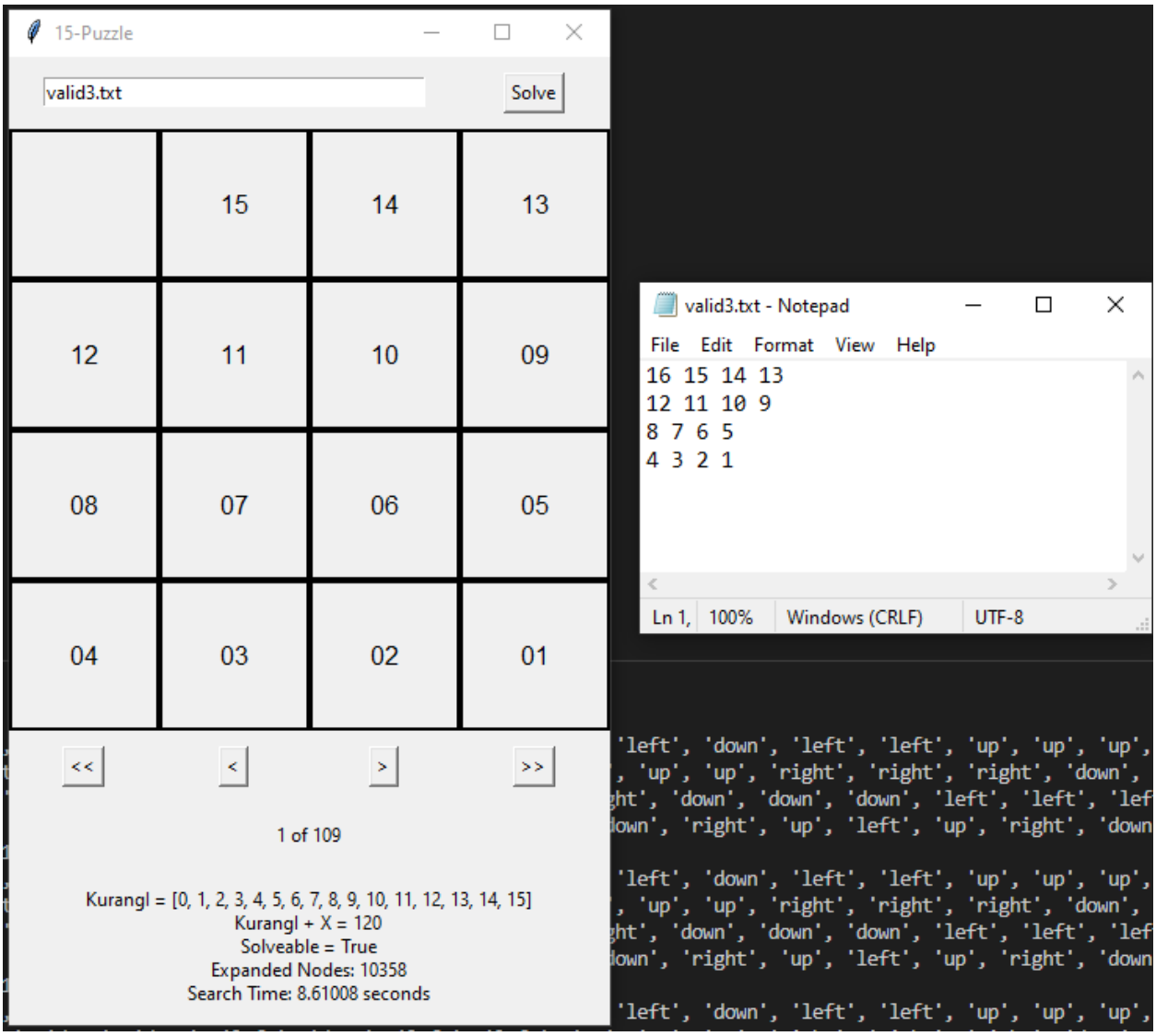
valid2.txt

Step tidak ditunukkan karena terlalu besar



valid3.txt

Step tidak ditunukkan karena terlalu besar



C. Source Code

BranchAndBound.py

src > BranchAndBound.py > ...

```python
from queue import PriorityQueue
import os
import copy

#Tree Nodes
#data is matrix of current state, path is the path to reach that state
#up,right,down,left are children
class Node:
    def __init__(self,data,path):
        self.data = data
        self.path = path

    def __lt__(self, other):
        return True;

    def __le__(self, other):
        return True;

#reads a file named text from ../test/ and returns inside as a 4x4 matrix of int, invalid inputs will cause the program to fail
def read(text):
    os.chdir("../test")
    f = open(text, "r")

    mat = []
    for x in f:
        arr = (x.strip("\n").split(" ")[:4])
        arr2 = []
        for x in arr:
            xInt = int(x)
            arr2.append(xInt)
        mat.append(arr2)
    f.close()
    return mat;

#turns the 4x4 matrix into an array of 16
def mergelist(mat):
    arr = []
    for l in mat:
        arr += l
    return arr

#returns a list consisting of Kurang(I)
def KurangI(arr):
    arr_KurangI = []
    for i in range(1,17):
        count = 0
        for j in range(1,i):
            if(arr.index(j) > arr.index(i)):
                count+=1
        arr_KurangI += [count]
    return arr_KurangI;

```

```python
#totals and returns KurangI + X
def KurangIplusX(arr):
    count = sum(KurangI(arr))
    if(arr.index(16) in [1,3,4,6,9,11,12,14]):
        count+=1
    return count

#returns true if count is event
def isReachable(count):
    if(count%2 == 0):
        return True
    return False

#returns the weight or g(x) of the matrix
def getweight(mat):
    count = 0;
    for i in range(4):
        for j in range(4):
            if(mat[i][j] != 4*i + j + 1):
                count+=1
    return count

#returns the weight or f(x) of the node
def getdepth(Current_Node):
    return 0.001*len(Current_Node.path)
    # return 0

#returns i,j coordinates of 16 in the 4x4 matrix
def Find16(mat):
    for i in range(4):
        for j in range(4):
            if(mat[i][j] == 16):
                return i,j

#turns the matrix into a string such as 01020304..... to be used as a key in a hash table
def matrixtokey(mat):
    key = ""
    for i in range(4):
        for j in range(4):
            key += "{:02d}".format(mat[i][j])
    return key

```

```python
#checks and updates the hash table
#gets the matrix mat, hashes it into a key
#if key not exist or key exist but weight (from parameter) is lower than key's value in hash table, returns true and insert/update hash table
#if key exist and key's value is lower than weight (from parameter), returns false
def CheckAndUpdateHashTable(mat,weight):
    key = matrixtokey(mat)
    global dict
    if(key not in dict.keys()):
        dict[key] = weight
        return True
    else:
        # return False
        if(dict[key] < weight):
            return False
        else:
            dict[key] = weight
            return True

#create a new branch and enqueues it (if valid from checking hash table)
def AddBranch(mat,path,i,j,direction):
    newmat = copy.deepcopy(mat)
    path2 = copy.deepcopy(path)
    if(direction == "up" and path[-1] != "down" and i!=0):
        newmat[i][j],newmat[i-1][j] = newmat[i-1][j],newmat[i][j]
        newpath = path2 + ["up"]
        Weight = getdepth(Current_Node) + getweight(newmat)
        if(CheckAndUpdateHashTable(newmat,Weight)):
            NewNode = Node(newmat,newpath)
            q.put((Weight,NewNode))
    elif(direction == "down" and path[-1] != "up" and i!=3):
        newmat[i][j],newmat[i+1][j] = newmat[i+1][j],newmat[i][j]
        newpath = path2 + ["down"]
        Weight = getdepth(Current_Node) + getweight(newmat)
        if(CheckAndUpdateHashTable(newmat,Weight)):
            NewNode = Node(newmat,newpath)
            q.put((Weight,NewNode))
    elif(direction == "left" and path[-1] != "right" and j!=0):
        newmat[i][j],newmat[i][j-1] = newmat[i][j-1],newmat[i][j]
        newpath = path2 + ["left"]
        Weight = getdepth(Current_Node) + getweight(newmat)
        if(CheckAndUpdateHashTable(newmat,Weight)):
            NewNode = Node(newmat,newpath)
            q.put((Weight,NewNode))
    elif(direction == "right" and path[-1] != "left" and j!=3):
        newmat[i][j],newmat[i][j+1] = newmat[i][j+1],newmat[i][j]
        newpath = path2 + ["right"]
        Weight = getdepth(Current_Node) + getweight(newmat)
        if(CheckAndUpdateHashTable(newmat,Weight)):
            NewNode = Node(newmat,newpath)
            q.put((Weight,NewNode))
```

```python
#solves the matrix and returns path(list of "left","right",etc) and number of nodes expanded
def solve(mat):
    global dict
    global q
    global Current_Node

    dict = {}
    q = PriorityQueue()
    Expanded_Nodes = 0

    root = Node(mat,["none"])

    q.put((0, root))

    while not q.empty():
        Current_Node = q.get()[1]
        Expanded_Nodes +=1
        mat = Current_Node.data
        path = Current_Node.path

        print(mat)
        print(path)

        weight = getweight(mat)
        i,j = Find16(mat)

        if(weight == 0 ):
            break
        AddBranch(mat,path,i,j,"up")
        AddBranch(mat,path,i,j,"down")
        AddBranch(mat,path,i,j,"left")
        AddBranch(mat,path,i,j,"right")

    return Current_Node.path,Expanded_Nodes

#uses the base matrix and a list of paths to return a matrix in step number no_state
def getstate(mat,path,no_state):
    newmat = copy.deepcopy(mat)
    i,j = Find16(newmat)

    for no in range(no_state):
        if (path[no+1] == "up"):
            newmat[i][j],newmat[i-1][j] = newmat[i-1][j],newmat[i][j]
            i = i-1
        elif(path[no+1] == "down"):
            newmat[i][j],newmat[i+1][j] = newmat[i+1][j],newmat[i][j]
            i = i+1
        elif(path[no+1] == "left"):
            newmat[i][j],newmat[i][j-1] = newmat[i][j-1],newmat[i][j]
            j = j-1
        elif(path[no+1] == "right"):
            newmat[i][j],newmat[i][j+1] = newmat[i][j+1],newmat[i][j]
            j = j+1

    return newmat

```

GUI.py

```python
from tkinter import *
from BranchAndBound import *
import time


#arranges the labels to match mat
def display(mat):
    for i in range(4):
        for j in range(4):
            Labels[int(mat[i][j])-1].grid(row=i+1,column=j)

#updates the display based on the command
def play(command):
    global path
    global total_steps
    global state
    global myLabel17

    if(path == None):
        return
    if(command == "first"):
        state = 1;
    elif(command == "back"):
        state = max(1,state-1)
    elif(command == "next"):
        state = min(total_steps,state+1)
    elif(command == "last"):
        state = total_steps
    newmat = getstate(mat,path,state-1)
    myLabel17["text"] = str(state) +" of " + str(total_steps)
    display(newmat)
    return
```

```
#solve the matrix
def run(myEntry):
    global mat
    global state
    global total_steps
    global path
    global statslabel

    text = myEntry.get()
    mat = read(text)

    arr = mergelist(mat)
    ListKurangI = KurangI(arr)
    Kurangiplusx = KurangIplusX(arr)
    if(isReachable(Kurangiplusx)):
        start_time = time.time()
        path,expanded = solve(mat)
        runtime = round((time.time() - start_time),5)

        total_steps = len(path)         #1 to total_steps, not 0 to total
        state = 1;
    else:
        expanded = 0
        runtime = 0
        state = 0
        total_steps = 0

    stats1 ="KurangI = "+str(ListKurangI)+ "\nKurangI + X = " + str(Kurangiplusx) + "\nSolveable = "+str(isReachable(Kurangiplusx))
    stats2 = "\nExpanded Nodes: "+str(expanded)+"\nSearch Time: "+str(runtime) + " seconds"

    stats = stats1 + stats2
    display(mat)
    myLabel17["text"] = str(state) +" of " + str(total_steps)
    statslabel["text"] = stats
```

```
#gui interface
def main():
    root = Tk()
    root.title("15-Puzzle")
    myEntry = Entry(root,text ="Enter Filename here",width=40)
    myEntry.grid(row=0,column=0,pady=10,columnspan = 3)

    myButton = Button(root,text = "Solve",command= lambda: run(myEntry))
    myButton.grid(row=0,column=3,pady=10)

    myLabel1 = Label(root,text ="01",height=5,width=10,borderwidth=2, relief="solid",font=(30))
    myLabel2 = Label(root,text ="02",height=5,width=10,borderwidth=2, relief="solid",font=(30))
    myLabel3 = Label(root,text ="03",height=5,width=10,borderwidth=2, relief="solid",font=(30))
    myLabel4 = Label(root,text ="04",height=5,width=10,borderwidth=2, relief="solid",font=(30))

    myLabel5 = Label(root,text ="05",height=5,width=10,borderwidth=2, relief="solid",font=(30))
    myLabel6 = Label(root,text ="06",height=5,width=10,borderwidth=2, relief="solid",font=(30))
    myLabel7 = Label(root,text ="07",height=5,width=10,borderwidth=2, relief="solid",font=(30))
    myLabel8 = Label(root,text ="08",height=5,width=10,borderwidth=2, relief="solid",font=(30))

    myLabel9 = Label(root,text ="09",height=5,width=10,borderwidth=2, relief="solid",font=(30))
    myLabel10 = Label(root,text ="10",height=5,width=10,borderwidth=2, relief="solid",font=(30))
    myLabel11 = Label(root,text ="11",height=5,width=10,borderwidth=2, relief="solid",font=(30))
    myLabel12 = Label(root,text ="12",height=5,width=10,borderwidth=2, relief="solid",font=(30))

    myLabel13 = Label(root,text ="13",height=5,width=10,borderwidth=2, relief="solid",font=(30))
    myLabel14 = Label(root,text ="14",height=5,width=10,borderwidth=2, relief="solid",font=(30))
    myLabel15 = Label(root,text ="15",height=5,width=10,borderwidth=2, relief="solid",font=(30))
    myLabel16 = Label(root,text ="  ",height=5,width=10,borderwidth=2, relief="solid",font=(30))

    Button_first = Button(root,text="<<",command = lambda: play("first"))
    Button_back = Button(root,text="<",command = lambda: play("back"))
    Button_next = Button(root,text=">",command = lambda: play("next"))
    Button_last = Button(root,text=">>",command = lambda: play("last"))

    Button_first.grid(row=5,column=0,pady=10,padx=20)
    Button_back.grid(row=5,column=1,pady=10,padx=20)
    Button_next.grid(row=5,column=2,pady=10,padx=20)
    Button_last.grid(row=5,column=3,pady=10,padx=20)

    global path
    path = None
    state = 0;
    total_steps = 0;
    mat = [[1,2,3,4],[5,6,7,8],[9,10,11,12],[13,14,15,16]]
    stats = ""

    global myLabel17
    myLabel17 = Label(root,text=str(state) +" of " + str(total_steps))
    myLabel17.grid(row=6,column=0,pady=10,columnspan=4)

    global Labels
    Labels = [myLabel1,myLabel2,myLabel3,myLabel4,myLabel5,myLabel6,myLabel7,myLabel8,myLabel9,myLabel10,myLabel11,myLabel12,myLabel13,myLabel14,myLabel15,myLabel16]


    display(mat)

    global statslabel
    statslabel = Label(root,text = stats)
    statslabel.grid(row = 7,column=0,pady=10,padx=20,columnspan=4)
    root.mainloop()


#starts the gui
if __name__ == "__main__":
    main()
```

D. Test Case Files

valid1.txt

1 2 3 4

5 6 16 8

9 10 7 11

13 14 15 12

valid2.txt

13 7 1 16

4 12 8 10

15 14 5 3

6 9 2 11

valid3.txt

16 15 14 13

12 11 10 9

8 7 6 5

4 3 2 1

invalid1.txt

1 3 4 15

2 16 5 12

7 6 11 14

8 9 10 13

invalid2.txt

13 7 1 16

4 12 8 10

14 15 5 3

6 9 2 11

E. 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	√	

F. Github Repository

https://github.com/JayaMangalo/Tucil-3-Stima