

Laporan Tugas Kecil 3

IF2211 Strategi Algoritma

Algoritma Branch and Bound dengan Aplikasinya



Disusun oleh :

Bariza Haqi 13520018

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2022

#### A. Langkah Algoritma *Branch And Bound*

Untuk langkah awalnya input matriks puzzle terlebih dahulu dengan input manual atau import dari file.

Berikut langkah-langkah algoritma *branch and bound* pada *15-puzzle*:

1. Cari nilai kurang dari tiap bilangan di puzzle yang akan diselesaikan kemudiansan jumlahkan
2. Kemudian tambahkan dengan nilai X
3. Jika hasil yang didapatkan bernilai ganjil maka puzzle tidak dapat diselesaikan
4. Jika hasil yang didapatkan bernilai genap maka puzzle dapat diselesaikan
5. Turunkan beberapa simpul dari puzzle yaitu puzzle jika kotak kosong dipindahkan ke atas, puzzle jika kotak kosong dipindahkan ke kanan, puzzle jika kotak kosong dipindahkan ke kiri dan puzzle jika kotak kosong dipindahkan ke bawah. Cek apakah puzzle tersebut pernah dieksekusi atau sudah ada di dalam list puzzle. Jika tidak, masukan puzzle tersebut ke dalam list puzzle
6. cari nilai cost terkecil dari puzzle di dalam list puzzle dengan cost adalah kedalaman puzzle ditambah banyak kotak yang tidak sesuai dengan tempatnya. Ambil puzzle tersebut dan hilangkan dari list
7. Ulangi Langkah ke-5 sampai banyak kotak yang tidak sesuai dengan tempatnya bernilai 0

#### B. *Source Code* Algoritma *Branch and Bound*

Untuk penggunaan algoritma *branch and bound* sendiri memakai bahasa Python. Berikut adalah source code program untuk menyelesaikan persoalan *15-puzzle* dengan algoritma *branch and bound*.

Program.py

```
import time
import copy

global jumlahSimpul
global waktuEksekusi
global listSolusiPuzzle
jumlahSimpul=0
```

```

waktuEksekusi=0
listSolusiPuzzle=[]

def makePuzzle(file):
    #Membuat puzzle dari input file txt dalam bentuk matriks
    Puzzle=[[0 for j in range (4)] for i in range (4)]
    lines=""
    with open(file) as f:
        while True:
            c = f.read(1)
            if not c:
                break
            else:
                if (c=="\n"):
                    lines+=" "
                else:
                    lines+=c
    index=0
    for i in range (4):
        for j in range (4):
            if (lines[index+1]==" "):
                Puzzle[i][j]=int(lines[index])
                index+=2
            else:
                bil=lines[index]+lines[index+1]
                Puzzle[i][j]=int(bil)
                index+=3
    return Puzzle

def matriksToHuruf(puzzle):
    #mengubah matriks puzzle menjadi string huruf untuk mempercepat algoritma
    agar tidak mengecek elemen matriks satu persatu
    stringHuruf = ""
    Huruf = "abcdefghijklmnop"
    for i in range(4):
        for j in range(4):
            stringHuruf += Huruf[puzzle[i][j]-1]
    return stringHuruf

def sumOfWrongBlock(puzzle):
    #Menghitung jumlah block yang salah atau ongkos mencapai simpul tujuan dari
    simpul puzzle
    count=0
    for i in range (4):
        for j in range (4):

```

```

        if (puzzle[i][j]!=16):
            if puzzle[i][j]!=i*4+j+1:
                count+=1
    return count

def valueX(puzzle):
    #mengembalikan nilai X
    for i in range (4):
        for j in range (4):
            if puzzle[i][j]==16:
                if (i+j)%2==0:
                    return 0
                else: #jika baris+kolom tidak habis dibagi 2 atau berada di
daerah yang diarsir
                    return 1

def indexRowNullBlock(puzzle):
    #mengembalikan baris pada block yang kosong
    for i in range (4):
        for j in range (4):
            if puzzle[i][j]==16:
                return i

def indexColumnNullBlock(puzzle):
    #mengembalikan kolom pada block yang kosong
    for i in range (4):
        for j in range (4):
            if puzzle[i][j]==16:
                return j

def kurang(puzzle,number):
    #mengembalikan nilai kurang dari bilangan number pada puzzle
    sum=number-1
    end=False
    for i in range (4):
        for j in range (4):
            if (sum==0 or puzzle[i][j]==number):
                end=True
                break
            else:
                if (puzzle[i][j]<number):
                    sum-=1
    if (end):
        break
    return sum

```

```

def sumOfKurangdanX(puzzle):
    #menghitung jumlah kurang ditambah X pada puzzle
    sum=0
    for i in range (16):
        sum+=kurang(puzzle,i+1)
    X=valueX(puzzle)
    sum+=X
    return sum

def moveUp(puzzle,i,j):
    #mengembalikan puzzle yang block kosongnya dipindahkan ke atas
    newpuzzle=copy.deepcopy(puzzle)
    if (i==0):
        return None
    else:
        temp=newpuzzle[i][j]
        newpuzzle[i][j]=newpuzzle[i-1][j]
        newpuzzle[i-1][j]=temp
        return newpuzzle

def moveRight(puzzle,i,j):
    #mengembalikan puzzle yang block kosongnya dipindahkan ke kanan
    newpuzzle=copy.deepcopy(puzzle)
    if (j==3):
        return None
    else:
        temp=newpuzzle[i][j]
        newpuzzle[i][j]=newpuzzle[i][j+1]
        newpuzzle[i][j+1]=temp
        return newpuzzle

def moveDown(puzzle,i,j):
    #mengembalikan puzzle yang block kosongnya dipindahkan ke bawah
    newpuzzle=copy.deepcopy(puzzle)
    if (i==3):
        return None
    else:
        temp=newpuzzle[i][j]
        newpuzzle[i][j]=newpuzzle[i+1][j]
        newpuzzle[i+1][j]=temp
        return newpuzzle

def moveLeft(puzzle,i,j):
    #mengembalikan puzzle yang block kosongnya dipindahkan ke kiri

```

```

newpuzzle=copy.deepcopy(puzzle)
if (j==0):
    return None
else:
    temp=newpuzzle[i][j]
    newpuzzle[i][j]=newpuzzle[i][j-1]
    newpuzzle[i][j-1]=temp
    return newpuzzle

def sumCost(puzzle,depthOfPuzzle):
    #mengembalikan nilai cost yaitu jumlah block yang salah ditambah kedalaman
    simpul
    return sumOfWrongBlock(puzzle)+depthOfPuzzle

def getIndexSmallestCost(listPuzzle):
    #mengembalikan indeks puzzle dengan nilai cost terkecil
    index=0
    smallestCost=listPuzzle[0][0]
    for i in range (1,len(listPuzzle)):
        if (smallestCost>listPuzzle[i][0]):
            smallestCost=listPuzzle[i][0]
            index=i
    return index

def matrixToList(matrix):
    #mengubah matriks menjadi list
    list=[]
    for i in range (4):
        for j in range (4):
            list.append(matrix[i][j])
    return list

def solvePuzzle(Puzzle):
    global jumlahSimpul
    global waktuEksekusi
    global listSolusiPuzzle

    listSolusiPuzzle=[]
    curNode=1 #simpul awal
    found=False
    mapPuzzle={} #dictionary berisi matriks puzzle dari simpulnya
    depthOfPuzzle={} #dictionary berisi kedalaman dari simpul
    parentNode = {} #dictionary berisi simpul parent dari simpul anaknya
    listString=[] #list berisi stringhuruf dari matriks puzzle
    mapPuzzle[curNode]=Puzzle

```

```

stringHuruf=matriksToHuruf(Puzzle)
listString.append(stringHuruf)
depthOfPuzzle[curNode]=0
cost=sumCost(Puzzle,depthOfPuzzle[curNode])
parentNode[curNode]=None
listPuzzle=[[cost, curNode]]
start_time=time.time()
while(not found):
    index=getIndexSmallestCost(listPuzzle)
    pick=listPuzzle.pop(index)
    node=pick[1]
    i=indexRowNullBlock(mapPuzzle[node])
    j=indexColumnNullBlock(mapPuzzle[node])
    movePuzzle=[]
    movePuzzle.append(moveUp(mapPuzzle[node],i,j))
    movePuzzle.append(moveRight(mapPuzzle[node],i,j))
    movePuzzle.append(moveLeft(mapPuzzle[node],i,j))
    movePuzzle.append(moveDown(mapPuzzle[node],i,j))
    for puzzle in movePuzzle:
        if (puzzle==None): #jika puzzle tidak ada
            continue
        stringHuruf=matriksToHuruf(puzzle)
        if (stringHuruf in listString): #jika puzzle sudah ada di list atau
sudah dieksekusi
            continue
        curNode+=1
        listString.append(stringHuruf)
        mapPuzzle[curNode]=puzzle
        parentNode[curNode]=node
        depthOfPuzzle[curNode]=depthOfPuzzle[node]+1
        cost=sumCost(puzzle,depthOfPuzzle[curNode])
        if (sumOfWrongBlock(puzzle)==0): #jika semua block puzzle sesuai
            found=True
            break
        listPuzzle.append([cost, curNode])
end_time=time.time()
jumlahSimpul=curNode
waktuEksekusi=end_time-start_time
listSolusiPuzzle.append(mapPuzzle[curNode])
while(parentNode[curNode]!=None):
    curNode=parentNode[curNode]
    listSolusiPuzzle.append(mapPuzzle[curNode])
listSolusiPuzzle.reverse() #susun puzzle sesuai dari awal puzzle

```

Main.py

```
import Program as p
import PySimpleGUI as sg
import os

#Buat GUI
menu_column = [ #GUI bagian kanan
    [sg.Text('15 Puzzle Solver', size=(20, 1), justification='center',
font=("Helvetica", 40))],
    [
        sg.Text("File Location"),
        sg.In(size=(30,1), enable_events=True, key="-File-"),
        sg.FileBrowse(initial_folder=os.path.dirname(os.path.abspath(__file__)),
file_types=(("Text Files", "*.txt")),))
    ],
    [
        sg.Button("Solve Puzzle", key="-SOLVE-"),
        sg.Button("Reset", key="-RESET-")
    ],
    [sg.Text(key="-SUM-")],
    [sg.Text(key="-TIME-")],
    [sg.Text(key="-JUMLAH-")]
]

puzzle_column = [#GUI bagian kiri
    [
        sg.Image(key="-PUZZLE1-", size=(118, 118)),
        sg.Image(key="-PUZZLE2-", size=(118, 118)),
        sg.Image(key="-PUZZLE3-", size=(118, 118)),
        sg.Image(key="-PUZZLE4-", size=(118, 118))
    ],
    [
        sg.Image(key="-PUZZLE5-", size=(118, 118)),
        sg.Image(key="-PUZZLE6-", size=(118, 118)),
        sg.Image(key="-PUZZLE7-", size=(118, 118)),
        sg.Image(key="-PUZZLE8-", size=(118, 118))
    ],
    [
        sg.Image(key="-PUZZLE9-", size=(118, 118)),
        sg.Image(key="-PUZZLE10-", size=(118, 118)),
        sg.Image(key="-PUZZLE11-", size=(118, 118)),
        sg.Image(key="-PUZZLE12-", size=(118, 118))
    ],
]
```



```

[
    sg.Image(key="-PUZZLE13-", size=(118, 118)),
    sg.Image(key="-PUZZLE14-", size=(118, 118)),
    sg.Image(key="-PUZZLE15-", size=(118, 118)),
    sg.Image(key="-PUZZLE16-", size=(118, 118))
],
[
    sg.Button("Previous", key="-PREVIOUS-"),
    sg.Button("Next", key="-NEXT-")
],
[
    sg.Text(key="-LANGKAH-")
]
]

layout =[
    [
        sg.Column(puzzle_column,element_justification='c'),
        sg.VSeperator(),
        sg.Column(menu_column,element_justification='c'),
    ]
]

window =sg.Window("Program 15-Puzzle", layout)

puzzle=None
index=0
solved=False
imgFolder=os.path.dirname(os.path.abspath(__file__))+"\\img" #lokasi folder
img
while True:
    event, values = window.read()
    if event == sg.WIN_CLOSED: #keluar dari program
        break
    if event == "-File-": #menekan tombol Browse
        try:
            puzzle = p.makePuzzle(values["-File-"])
            list=p.matrixToList(puzzle)
            for i in range(16):
                if (list[i]==16):
                    window[f"-PUZZLE{i+1}-"].update("", size=(118, 118))
                else:
                    window[f"-PUZZLE{i+1}-"]
                    ].update(imgFolder+"\\img"+str(list[i])+".png")

```

```

except:
    pass

if event=="-SOLVE-":#menekan tombol Solve Puzzle
    if puzzle!=None:
        sumKurangX=p.sumOfKurangdanX(puzzle)
        sumtext="SUM(Kurang)+X: "+str(sumKurangX)
        window["-SUM-"].update(sumtext)
        if (sumKurangX%2==0):
            p.solvePuzzle(puzzle)
            waktu="Waktu Eksekusi: "+str(p.waktuEksekusi) + " s"
            jumlah="Jumlah Simpul: "+str(p.jumlahSimpul)
            window["-TIME-"].update(waktu)
            window["-JUMLAH-"].update(jumlah)
            solved=True
        else: #jika nilai kurang +x ganjil
            text="Puzzle Tidak bisa diselesaikan"
            window["-TIME-"].update(text)
        if (solved):
            window["-LANGKAH-"].update("Langkah ke-1")

if event=="-RESET-":#menekan tombol Reset
    puzzle=None
    window["-SUM-"].update("")
    window["-TIME-"].update("")
    window["-JUMLAH-"].update("")
    window["-LANGKAH-"].update("")
    solved=False
    index=0
    for i in range(16):
        window[f"-PUZZLE{i+1}-"].update("", size=(118, 118))

if event=="-PREVIOUS-": #menekan tombol Previous
    if solved:
        if (index!=0):
            index-=1
            list=p.matrixToList(p.listSolusiPuzzle[index])
            for i in range(16):
                if (list[i]==16):
                    window[f"-PUZZLE{i+1}-"].update("", size=(118, 118))
                else:
                    window[f"-PUZZLE{i+1}-
"].update(imgFolder+"\\ "+str(list[i])+".png")
            window["-LANGKAH-"].update(f"Langkah ke-{index+1}")

```

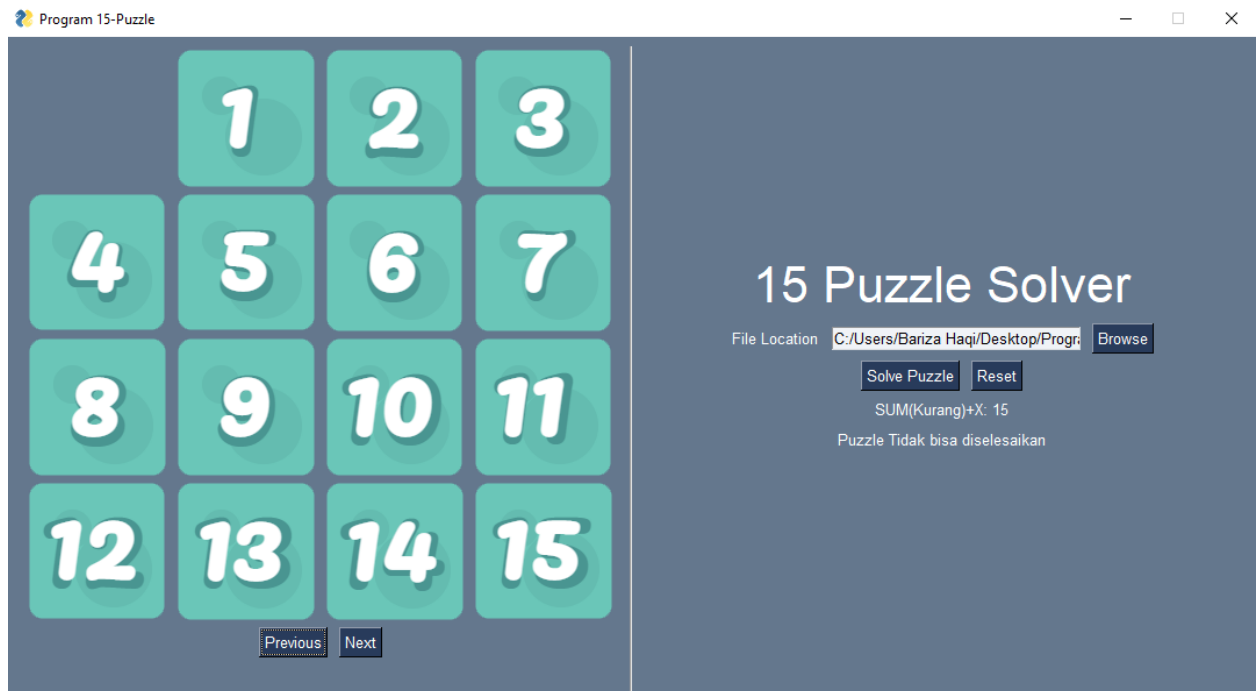
```

if event=="-NEXT-": #menekan tombol Next
    if solved:
        if (index!=len(p.listSolusiPuzzle)-1):
            index+=1
            list=p.matrixToList(p.listSolusiPuzzle[index])
            for i in range(16):
                if (list[i]==16):
                    window[f"-PUZZLE{i+1}-"].update("", size=(118, 118))
                else:
                    window[f"-PUZZLE{i+1}-
"].update(imgFolder+"\\ "+str(list[i])+".png")
                    window["-LANGKAH-"].update(f"Langkah ke-{index+1}")

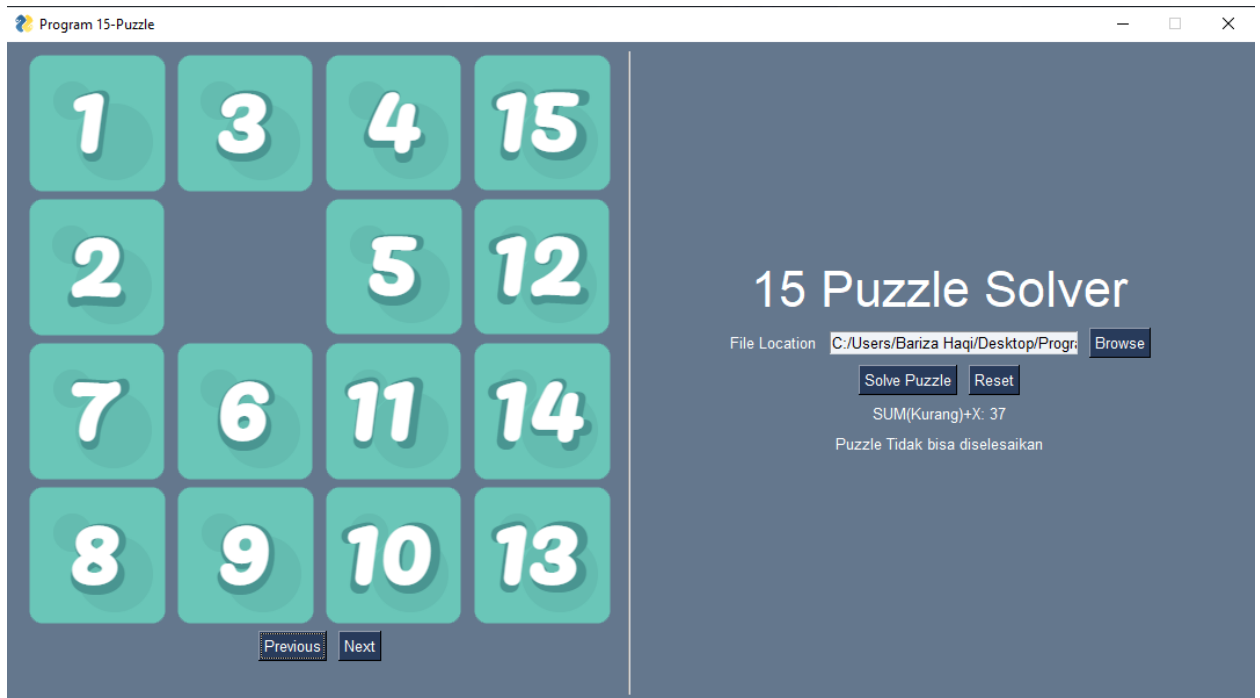
```

### C. Screenshot Input dan Output Program

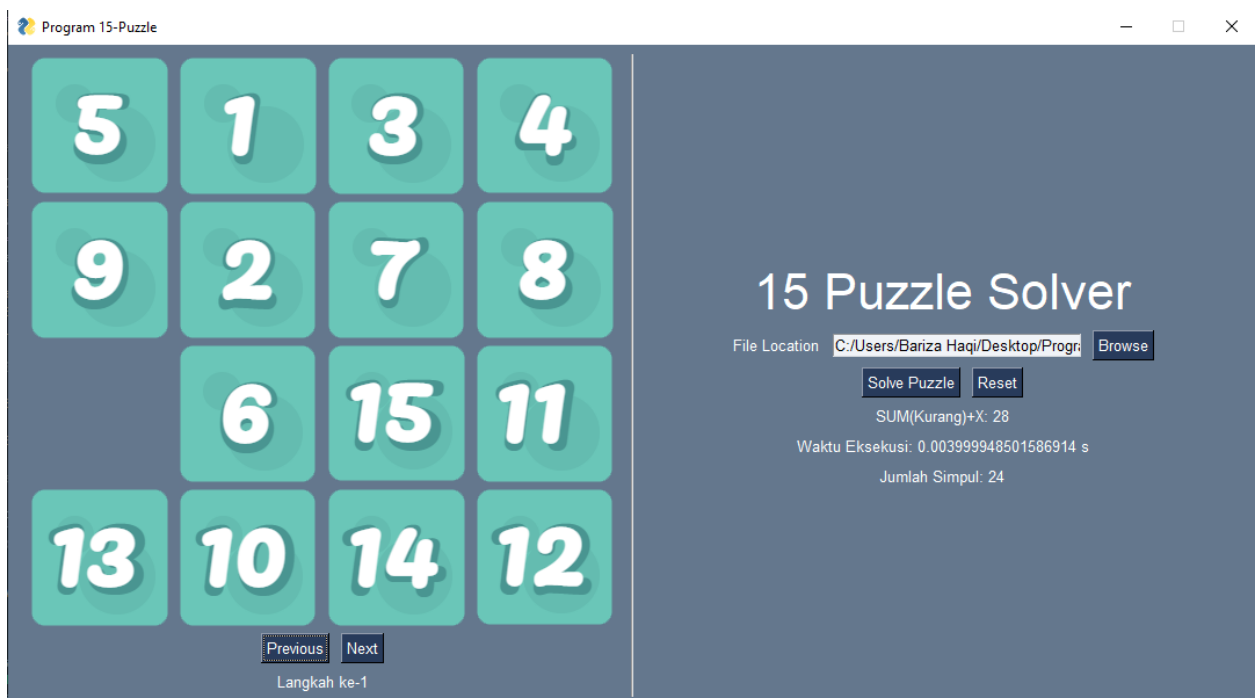
#### a. Test1

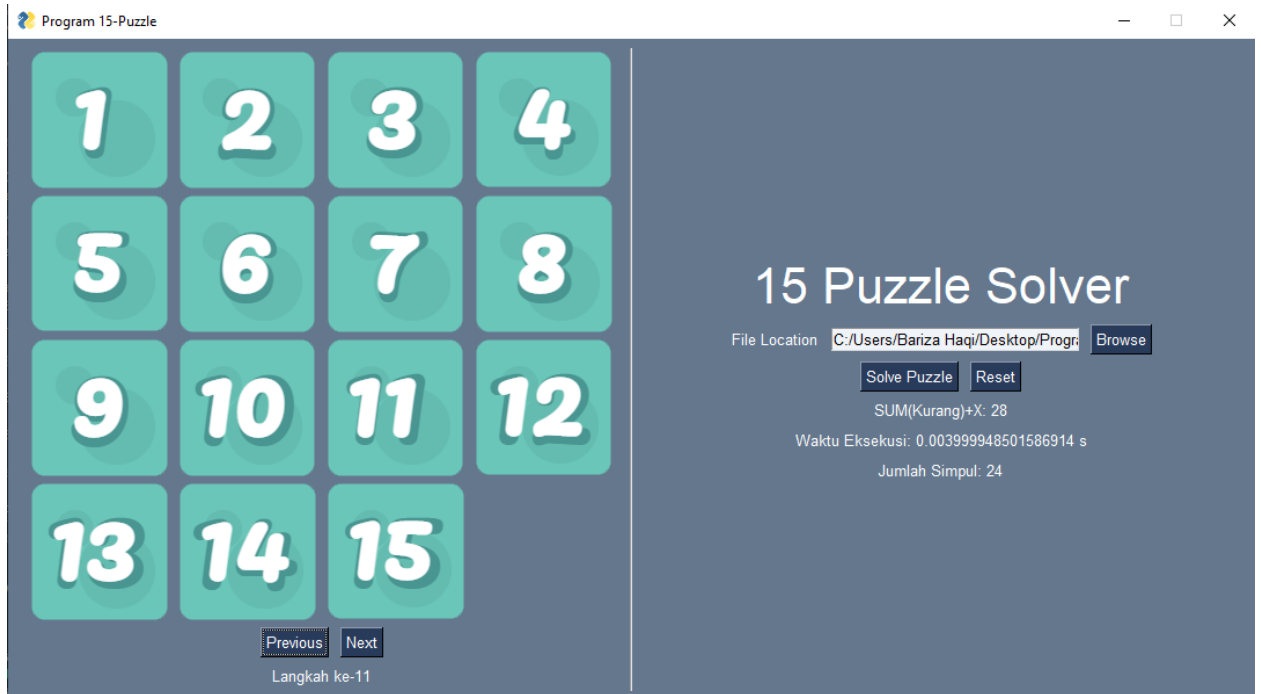


#### b. Test2

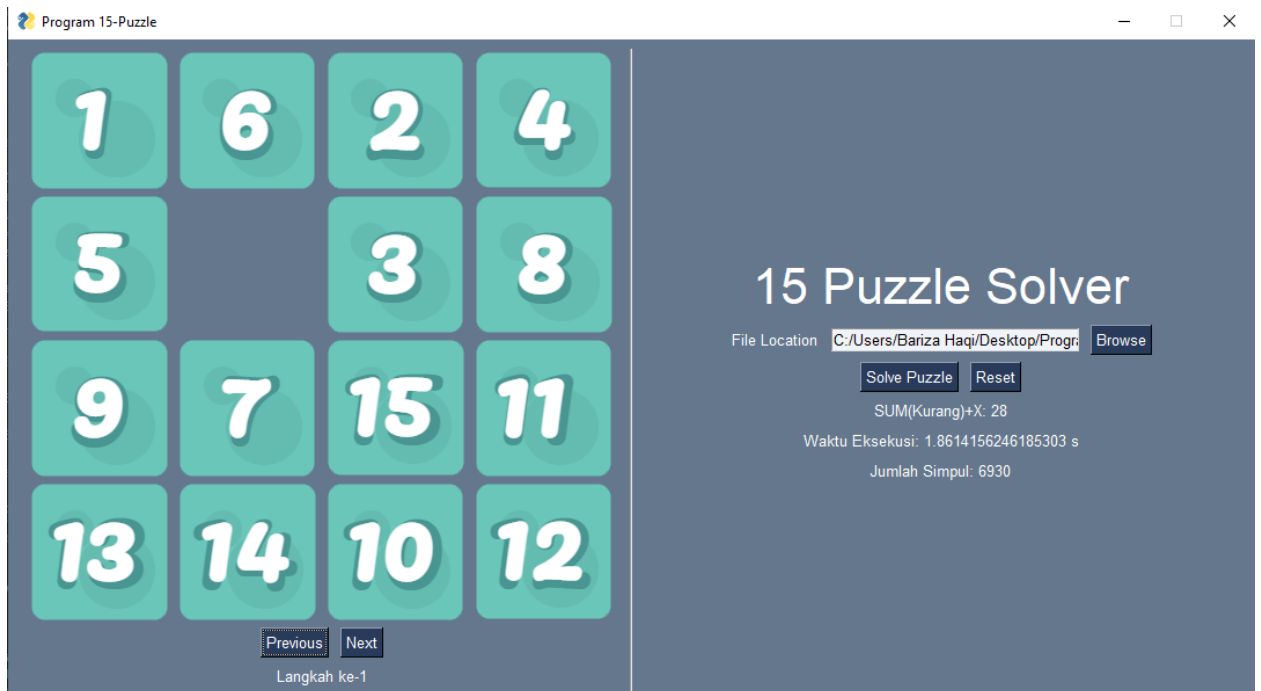


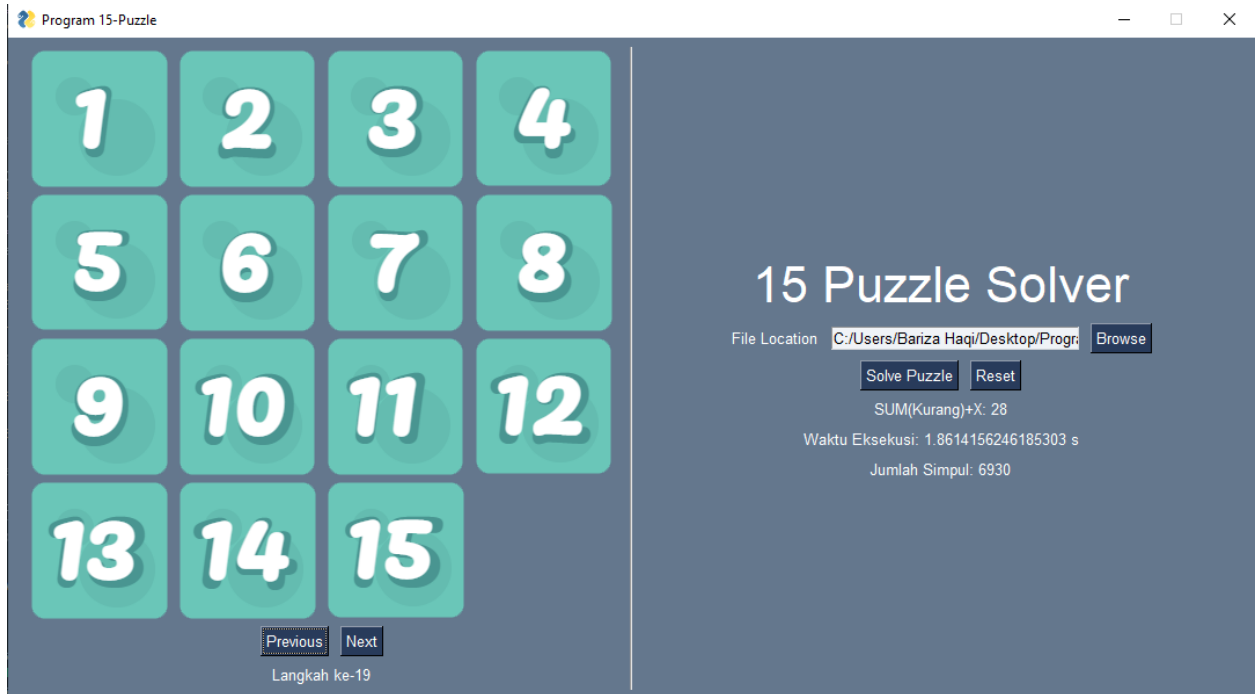
c. Test3



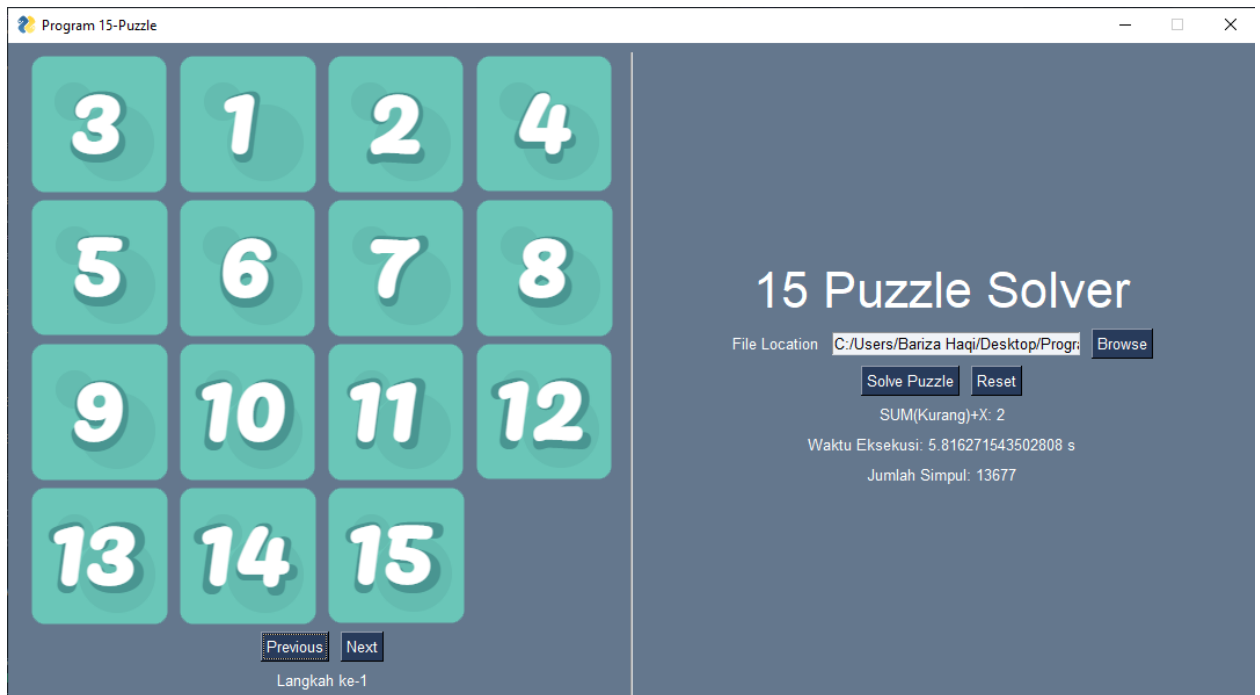


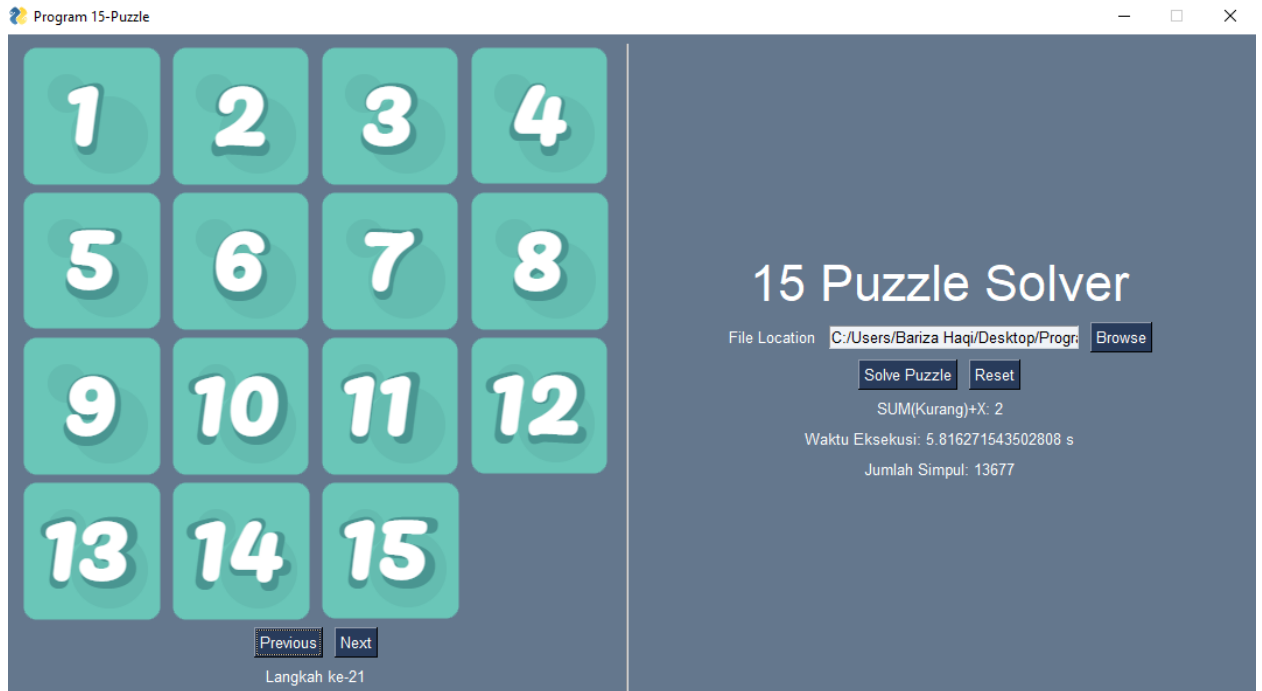
d. Test4





e. Test5





D. Alamat drive

<https://github.com/barizahaqi/Tucil-3-Strategi-Algoritma>

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	✓	