LAPORAN
TUGAS KECIL 3

“Penyelesaian Persoalan 15-Puzzle dengan Algoritma Branch and Bound”

IF2211 STRATEGI ALGORITMA

Oleh:

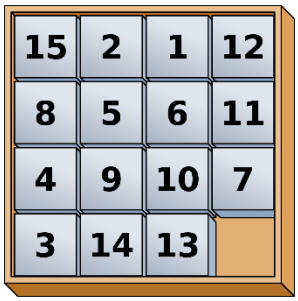
Ng Kyle / 13520040

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

2021

Bagian I
Deskripsi Algoritma dan Kompleksitas Algoritma

1. **Algoritma *Branch and Bound* dalam penyelesaian 15-Puzzle**
   Algoritma *Branch and Bound* sebagaimana menjadi salah satu strategi penyelesaian game 15-Puzzle. Branch akan melakukan pembangkitan terhadap anak simpul ekspan dan Bound akan melakukan *pruning* terhadap jalur yang tidak memenuhi persayaratan (ketika sudah didapat sebuah solusi dan cost halur tersebut ≥ cost solusi sementara). Branch and bound menggunakan struktur data priority queue dalam menenetukan node yang akan dievaluasi/dieskpan. Dalam game 15-Puzzle, tujuan dari game adalah menggeser ubin kosong sehingga konfigurasi papan permainan menjadi:



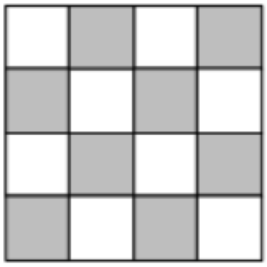
Sumber : https://en.wikipedia.org/wiki/15_puzzle

Konfigurasi awal dari puzzle tersebut menentukan apakah konfigurasi tujuan dapat dicapai maupun tidak. Dalam algoritma yang digunakan, pembangkitan akan dilakukan dengan arah Right, Left, Down, Up yaitu menukar tile kosong dengan tile pada arah tertentu jika dapat dan pantas dilakukan.

2. **Perincian dan Penjelasan Algoritma**
   Keterangan awal: Tile kosong direpresentasikan dengan tile bernilai 16
   Dalam pengaplikasian algoritma *Branch and Bound* persoalan ini berikut merupakan tahap dari algoritma:
   a. Pre-Processing
      Tahap ini melakukan pembacaan dan/atau membangkitkan konfigurasi awal secara random lalu menentukan apakah konfigurasi dapat diselesaikan atau tidak.
      1. Membaca konfigurasi dari file .txt atau mengacak konfigurasi tujuan.
      2. Menentukan apakah konfigurasi dapat diselesaikan yaitu ketika memenuhi persamaan berikut:

$$\sum_{i=1}^{16} KURANG(i) + X \equiv 0 \ (mod\ 2)$$

      Dalam hal ini, KURANG(i) merupakan banyaknya tile j > i dengan POSISI(j) > POSISI(i). X akan bernilai 1 jika tile kosong pada posisi salah satu daerah di arsir:



Sumber: http://www.cs.umsl.edu/~sanjiv/classes/cs5130/lectures/bb.pdf

Program akan menampilkan setiap nilai KURANG(i) beserta nilai dari $\sum_{i=1}^{16} KURANG(i) + X$. Jika memenuhi persamaan, program akan lanjut ke proses pencarian solusi, jika tidak akan menampilkan pesan kesalahan.

b. Proses *Branch and Bound*
Proses *Branch and Bound* dimulai dari state awal dan membuat sebuah objek Node yang mengandung informasi: State (matriks kondisi), Parent (Node parent), f (nilai f yaitu kedalaman pencarian pada state tersebut), g (nilai g yaitu banyaknya tile yang tidak pada posisi seharusnya pada konfigurasi tujuan), c (cost dari node), dir (arah yang dipakai untuk mencapai posisi tersebut yaitu 'U','D','R', atau 'L'), blank (koordinat/posisi dari tile kosong/tile 16).
Lalu, dilakukan proses sebagai berikut:
1. Evaluasi status (apakah status sudah menjadi konfigurasi tujuan) yang berada pada depan priority queue. Jika ya, berhenti. Jika tidak, lanjut ke tahap 2.
2. Ekspan dari status lalu melakukan hash terhadap setiap anaknya.
3. Jika simpul hasil ekspan memiliki nilai hash yang belum terdapat (status belum pernah tercapai), maka simpul tersebut dibuat Node baru dan ditambahkan ke priority queue.
4. Lakukan kembali ke tahap 1.

c. Proses Penelusuran balik solusi
Setelah ditemukannya Node solusi, maka akan dibangkitkan Langkah-langkah yang optimal tersebut dengan menelusuri parent dari Node tersebut hingga Node awal. Solusi merupakan hasil penelusuran tersebut.

Struktur data utama yang digunakan adalah sebagai berikut:
1. Node implementasi abstraksi Linked List dengan atribut Parent mengacu pada state induk. (Implementasi pada node.py)
2. Heapq implementasi dari Priority Queue menyimpan Node, dengan pembanding nilai c dari Node.
3. Dicitionary sebagai implementasi hash map yang menyimpan nilai hash yang pernah ditemui.

Keterangan Tambahan:
- Walaupun secara teoritis konfigurasi yang valid akan selalu dapat diselesaikan dengan program ini, namun karena keterbatasan fungsi heuristik dari g sehingga pencarian kurang efisien dan dapat mengakibatkan pencarian yang lama untuk kedalaman > 10.

Hal ini dapat diperbaiki dengan mengubah fungsi heuristic dari nilai g menjadi manhattan distance.

3. **Elemen Branch and Bound QuickHull**
   Branch: Membangkitkan simpul anak (ekspan simpul) pada tahap b.1
   Bound: Melakukan *pruning* terhadap jalur yang memiliki cost lebih besar dari simpul solusi sementara.
   Fungsi Pembatas dan Cost: nilai f berupa kedalaman simpul dan g berupa heuristic banyaknya tile yang tidak sesuai dengan posisi pada konfigurasi tujuan.

4. **Analisis Kompleksitas Algoritma**
   Pada setiap ekspan, dilakukan ekspansi rata-rata 3 buah simpul baru. Sehingga kompleksitas waktu yaitu $O(3^m)$ dan kompleksitas ruangan $O(3^m)$ dengan m merupakan kedalaman maksimum dari pencarian.

Bagian II
Source Program

Program dibuat menjadi 2 versi yaitu Console Application dan GUI Application. Console Application diimplementasikan pada file **console.py** sedangkan GUI diimplementasikan dalam file **program.py**. Implementasi dari branch and bound disebar menjadi 3 buah file, masing-masing yaitu node.py, puzzle.py, dan fpsolver.py.

1. node.py
   File implementasi class Node yaitu struktur data menyimpan state dan cost dari simpul beserta parent dari simpul tersebut.

```
class Node(object):
    def __init__(self, state, parent, f, dir, blankpos):
        self.state = state
        self.parent = parent
        self.f = f
        self.g = self.calcg()
        self.c = f + self.g
        self.dir = dir
        self.blank = blankpos

    def calcg(self):
        temp =0
        for i in range(4):
            for j in range(4):
                refer = self.state[i][j]
                if (refer != 16 and i*4+j+1 != refer):
                    temp += 1
        return temp

    def __lt__(self, other):
        return self.c < other.c
```

2. puzzle.py
   File implementasi pembacaan file dan randomizer konfigurasi. Selain itu implementasi dari validasi konfigurasi file valid atau tidak (BUKAN solveable atau tidak) dan print puzzle pada console.
   a. Read from file

```
import numpy as np
from colorama import Fore, Style

def read(filepath):
    f = open(filepath, 'r')
    mat = [[int(val) for val in line.split()] for line in f]
    validate(mat)
    return mat
```

b. Randomize Puzzle

```
def random():
    base = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16]
    np.random.shuffle(base)
    base = np.reshape(base,(4,4))
    base = base.tolist()
    return base
```

c. Validate Konfigurasi

```
def validate(mat):
    tester = [False for i in range(16)]
    for i in range(4):
        for j in range(4):
            tester[mat[i][j]-1] = True
    for status in tester:
        if (not(status)):
            raise Exception("Configuration Not Valid!")
```

d. Print Puzzle
Menggunakan colorama dalam print warna

```
# unicode for draw puzzle in command promt or terminal
left_down_angle = '\u2514'
right_down_angle = '\u2518'
right_up_angle = '\u2510'
left_up_angle = '\u250C'

middle_junction = '\u253C'
top_junction = '\u252C'
bottom_junction = '\u2534'
right_junction = '\u2524'
left_junction = '\u251C'

#bar color
bar = Style.BRIGHT + Fore.CYAN + '\u2502' + Fore.RESET + Style.
RESET_ALL
dash = '\u2500'
dashes = dash + dash + dash + dash
first_line = Style.BRIGHT + Fore.CYAN + left_up_angle + dashes
 + top_junction + dashes + top_junction + dashes+ top_junction
 + dashes+  right_up_angle + Fore.RESET + Style.RESET_ALL
middle_line = Style.BRIGHT + Fore.CYAN + left_junction + dashes
 + middle_junction + dashes + middle_junction + dashes +
middle_junction + dashes + right_junction + Fore.RESET + Style.
RESET_ALL
last_line = Style.BRIGHT + Fore.CYAN + left_down_angle + dashes
 + bottom_junction + dashes + bottom_junction + dashes+
bottom_junction + dashes+  right_down_angle + Fore.RESET + Style
.RESET_ALL

#puzzle print function
def print_puzzle(array):
    print(first_line)
    for a in range(len(array)):
        for i in array[a]:
            if (i < 16):
                if (i < 10):
                    print(bar + ' ', end = '')
                else:
                    print(bar, end='')
                print(' ' + str(i), end=' ')
            else:
                print(bar + '   ', end = ' ')

        print(bar)
        if a == 3:
            print(last_line)
        else:
            print(middle_line)
```

3. fpsolver.py
   File implementasi branch and bound dari 15 puzzle.
   a. Definisi variable global dan import

```
import heapq
from node import *
from copy import deepcopy
from heapq import *
from puzzle import *
directions = {'R' : [0,1], 'L' : [0,-1], 'D' : [1,0], 'U' : [-1,0]}
nodes = []
goalState = [[1,2,3,4], [5,6,7,8],[9,10,11,12],[13,14,15,16]]
hashes = {}
isFound = False
comparisons = 0
resultNode : Node
```

   b. Hash state
      Melakukan hashing pada state (matriks 4 x 4)

```
def hashState(mat):
    idx = 0
    for i in range(16):
        row = i // 4
        col = i %4
        val = mat[row][col]
        idx |= i << (val * 4)
    return idx

hashgoal = hashState(goalState)
```

   c. Solve (Branch and Bound)
      Fungsi melakukan branch and bound (pruning dan ekspan)

```
def solve(mat):
    global resultNode
    global hashes
    global isFound
    global comparisons
    global nodes
    nodes = []
    comparisons = 0
    isFound = False
    hashes = {}
    x,y = findBlank(mat)
    newNode = Node(mat,None,0,None,[x,y])
    heappush(nodes,newNode )
    if(hashState(mat) != hashgoal):
        while(len(nodes) > 0):
            if(isFound):

#Bound any further search if goal is found and head of priority wor
se than resultNode
                if(nodes[0].c >= resultNode.c):
                    break
            evalNode = heappop(nodes)
            evalChilds(evalNode)
        path = getPath(resultNode)
    else:
        path = [newNode]

    return path, comparisons
```

d. Get path
Mendapatkan path solusi (setelah ditemukannya node solusi)

```
def getPath(CurNode):
    path = []
    while(CurNode.parent != None):
        path.append(CurNode)
        CurNode = CurNode.parent
    path.append(CurNode)
    return path
```

e. Find Blank

```
def findBlank(mat):
    for i in range(4):
        for j in range(4):
            if (mat[i][j] == 16):
                return i,j
```

f. Eval Childs (Ekspan dan periksa solusi)
Melakukan ekspan pada sebuah node, melakukan hashing dan sekaligus memeriksa apakah sudah mencapai solusi. Jika tidak, menambahkan pada priority queue.

```
def evalChilds(curnode):
    global hashes
    global resultNode
    global isFound
    global comparisons
    for dir in directions:
        if (validMove(curnode,dir)):
            newMat = swap(curnode,dir)
            # hashstates to avoid duplicates
            hashval = hashState(newMat)
            if(hashval == hashgoal):
                isFound = True
                resultNode = Node(newMat, curnode, curnode.f + 1,
dir,[curnode.blank[0] + directions[dir][0], curnode.blank[1] +
directions[dir][1]])
                comparisons +=1
                break
            elif(hashval not in hashes):
                hashes[hashval] = True
                comparisons += 1
                newNode = Node(newMat, curnode, curnode.f + 1, dir
,[curnode.blank[0] + directions[dir][0], curnode.blank[1] +
directions[dir][1]])
                heappush(nodes,newNode )
    return
```

g. Valid Move
Memeriksa apakah move dari node saat ini valid atau tidak

```
def validMove(curnode : Node, dir):
    newblankpos = [curnode.blank[0] + directions[dir][0], curnode.
blank[1] + directions[dir][1]]
    return (newblankpos[0] in range(0,4) and newblankpos[1] in
range(0,4))
```

h. Swap
Melakukan pemindahan tile kosong sesuai direction

```
def swap(curnode: Node, dir):
    newmat = deepcopy(curnode.state)
    x = curnode.blank[0]
    y = curnode.blank[1]
    newx = x + directions[dir][0]
    newy = y + directions[dir][1]
    newmat[x][y] = newmat[newx][newy]
    newmat[newx][newy] = 16
    return newmat
```

i. Solvable
Melakukan perhitungan jumlah kurang serta memeriksa apakah konfigurasi awal dapat di-solve atau tidak.

```
def solvable(mat):
    kurang,blankidx = countKurang(mat)
    tot = 0
    for row in mat:
        for col in row:
            tot += kurang[col-1]
    issolvable = ((blankidx + tot) % 2 == 0)
    return kurang, tot, blankidx, issolvable
```

j. Count Kurang
Menghitung nilai kurang dari konfigurasi

```python
def countKurang(mat):
    kurang = [0 for i in range(16)]
    for i in range(16):
        row = i // 4
        col = i % 4
        refer = mat[row][col]
        if (refer == 16):
            blankidx = row+col
        for j in range(i,16):
            row = j // 4
            col = j % 4
            check = mat[row][col]
            if (check < refer):
                kurang[refer-1] += 1
    return kurang, blankidx
```

4. console.py
   Program utama console

```
import fpsolver as s
import puzzle as p
import os
from time import time

if __name__ == '__main__':
    print('Current Working Directory: '+ os.getcwd() )
    #Membaca konfigurasi awal

    while (True):
        mode = input('Input from File? (y/n): ')
        if (mode == 'y' or mode == 'n'):
            break
    if(mode =='y'):
        while (True):
            try:
                filepath = input(
'Input file path (ex: test/in1.txt): ')
                initial_state = p.read(filepath)
                p.validate(initial_state)
                break
            except Exception as e:
                print(e)
    else:
        initial_state = p.random()

    # Initial State
    p.print_puzzle(initial_state)

    # Hitung Nilai Kurang State Matriks
    kurang,tot, blankidx, issolveable = s.solvable(initial_state)
    print("===============NILAI KURANG(i)===============")
    for i in range(16):
        print('KURANG('+str(i+1) + ') = ' + str(kurang[i]))
    print("============================")
    print("\u03a3(Kurang(i)) + X =", tot, '+', (blankidx%2),'=',
tot+(blankidx%2))

    #Melakukan Solve jika dapat disolve
    if (not issolveable):
        print('\nConfiguration Can\'t Be Solved!')
    else:
        begin = time()
        (path, comp) = s.solve(initial_state)
        end = time()
        path.reverse()
        print('\n========Initial State========')
        for mat in path:
            if (mat.dir != None):
                print('========MOVE '+ mat.dir+ '========')
            p.print_puzzle(mat.state)
        print('Total Gerakan =', len(path)-1)
        print("Total simpul dibangkitkan =", comp)
        print("Elapsed time =", end-begin)
```

5. program.py
   Program utama GUI

```
from cgitb import enable
from tkinter import *
from tkinter import messagebox
from tkinter import filedialog
from puzzle import *
from fpsolver import *
from time import time
root = Tk()
root.title("15 Puzzle")

path = []
idx = 0

def showprev():
    global idx
    if (idx == 1):
        leftbutton['state'] = DISABLED
    rightbutton['state'] = NORMAL
    idx -= 1
    setEntries(path[idx].state)
    navVal.set('Puzzle Solved\nSteps:\n' + str(idx) + '/' + str(len(path)-1))

def shownext():
    global idx
    if (idx == len(path) - 2):
        rightbutton['state'] = DISABLED
    leftbutton["state"] = NORMAL
    idx += 1
    setEntries(path[idx].state)
    navVal.set('Puzzle Solved\nSteps:\n' + str(idx) + '/' + str(len(path)-1))

def disableEntries():
    for e in grids:
        e.config(state=DISABLED)

def enableEntries():
    for e in grids:
        e.config(state=NORMAL)

def openfile():
    filepath = filedialog.askopenfilename(initialdir = "/",title = "Select file",filetypes = (("text files","*.txt"),("all files","*.*"
)))
    try:
        mat = read(filepath)
        setEntries(mat)
    except:
        messagebox.showerror("Error","Invalid input")
        return
def help():
    messagebox.showinfo("Help","Enter number 1-15 in the grid or press random to generate a random puzzle.\n\n
Press solve to solve the puzzle.\n\nPress Open File to use configuration from existing file\n\nPress reset to reset the puzzle.\n\n
Use arrow to navigate the puzzle.")

def randomize():
    mat = random()
    setEntries(mat)

def getEntries():
    mat = [[0 for _ in range(4)] for _ in range(4)]
    for i in range(4):
        for j in range(4):
            mat[i][j] = getVal(entries[i][j])
    return mat
```

```
def getVal(entry):
    if(entry.get() == ''):
        return 16
    return int(entry.get())

def setKurang():
    for i in range(4):
        for j in range(4):
            kurangval = kurang[i*4+j]
            kurangString = kuranglist[i][j]
            kurangString.set('KURANG('+str(i*4+j+1) + ') = ' + str(kurangval))
    kuranghead.set('Nilai KURANG(i)')
    kurangstat.set('\u03a3(Kurang(i)) + X = ' + str(tot)+ ' + '+ str(blankidx%2) +' = ' + str(tot+(blankidx%2)))

def solvePuzzle():
    try:
        global path
        global idx
        global kurang, tot, blankidx, issolveable
        idx = 0
        mat = getEntries()
        kurang,tot, blankidx, issolveable = solvable(mat)
        setKurang()
        validate(mat)
        disableEntries()
        solvebutton['state'] = DISABLED
        randombutton['state'] = DISABLED
        openfilebutton['state'] = DISABLED
    except:
        enableEntries()
        messagebox.showerror("Error","Invalid input")
        solvebutton['state'] = NORMAL
        randombutton['state'] = NORMAL

    if (issolveable):
        begin = time()
        (path, built) = solve(mat)
        end = time()
        path.reverse()
        timeVal.set( str(end-begin) + ' ms')
        nodesVal.set( str(built))
        if(len(path) > 1):
            rightbutton['state'] = NORMAL
        navVal.set('Puzzle Solved\nSteps:\n' + '0/' + str(len(path)-1))
    else:
        messagebox.showerror("Error","Configuration can't be solved")
```

```
def reset():
    setEntries([[i*4+j+1 for j in range(4)] for i in range(4)])
    enableEntries()
    timeVal.set('')
    nodesVal.set('')
    solvebutton['state'] = NORMAL
    randombutton['state'] = NORMAL
    openfilebutton['state'] = NORMAL
    leftbutton['state'] = DISABLED
    rightbutton['state'] = DISABLED
    for i in range(4):
        for j in range(4):
            kuranglist[i][j].set('')
    kuranghead.set('')
    kurangstat.set('')
    navVal.set('')

def setEntries(mat):
    for i in range(4):
        for j in range(4):
            if (mat[i][j] == 16):
                mat[i][j] = ''
    for i in range(4):
        for j in range(4):
            entry = entries[i][j]
            entry.set(mat[i][j])



mainlabel = Label(root, text="Configuration", font=("Helvetica", 10),justify=CENTER)
mainlabel.grid(row=0, column=0, columnspan=3)

#Time Labels
timelable = Label(root, text="Time", font=("Helvetica", 10),justify=CENTER)
timelable.grid(row=4, column=4)
timeVal = StringVar()
timeVallabel = Label(root, textvariable=timeVal, font=("Helvetica", 10),justify=CENTER)
timeVallabel.grid(row=4, column=5)

#Nodes Created Label
nodeslabel = Label(root, text="Nodes Created", font=("Helvetica", 10),justify=CENTER)
nodeslabel.grid(row=5, column=4)
nodesVal = StringVar()
nodesVallabel = Label(root, textvariable=nodesVal, font=("Helvetica", 10),justify=CENTER)
nodesVallabel.grid(row=5, column=5)
```

```
#Kurang Labels
kuranghead = StringVar()
kurangheadlabel = Label(root, textvariable=kuranghead, font=("Helvetica", 10),justify=CENTER)
kurangheadlabel.grid(row=0, column=7, columnspan=2)
kurangstat = StringVar()
kurangstatlabel = Label(root, textvariable=kurangstat, font=("Helvetica", 10),justify=CENTER)
kurangstatlabel.grid(row=5, column=6, columnspan=4)

kurang1 = StringVar()
kurang1label = Label(root,textvariable= kurang1 , font=("Helvetica", 10),justify=CENTER)
kurang2 = StringVar()
kurang2label = Label(root,textvariable= kurang2 , font=("Helvetica", 10),justify=CENTER)
kurang3 = StringVar()
kurang3label = Label(root,textvariable= kurang3 , font=("Helvetica", 10),justify=CENTER)
kurang4 = StringVar()
kurang4label = Label(root,textvariable= kurang4 , font=("Helvetica", 10),justify=CENTER)

kurang5 = StringVar()
kurang5label = Label(root,textvariable= kurang5 , font=("Helvetica", 10),justify=CENTER)
kurang6 = StringVar()
kurang6label = Label(root,textvariable= kurang6 , font=("Helvetica", 10),justify=CENTER)
kurang7 = StringVar()
kurang7label = Label(root,textvariable= kurang7 , font=("Helvetica", 10),justify=CENTER)
kurang8 = StringVar()
kurang8label = Label(root,textvariable= kurang8 , font=("Helvetica", 10),justify=CENTER)

kurang9 = StringVar()
kurang9label = Label(root,textvariable= kurang9 , font=("Helvetica", 10),justify=CENTER)
kurang10 = StringVar()
kurang10label = Label(root,textvariable= kurang10 , font=("Helvetica", 10),justify=CENTER)
kurang11 = StringVar()
kurang11label = Label(root,textvariable= kurang11 , font=("Helvetica", 10),justify=CENTER)
kurang12 = StringVar()
kurang12label = Label(root,textvariable= kurang12 , font=("Helvetica", 10),justify=CENTER)

kurang13 = StringVar()
kurang13label = Label(root,textvariable= kurang13 , font=("Helvetica", 10),justify=CENTER)
kurang14 = StringVar()
kurang14label = Label(root,textvariable= kurang14 , font=("Helvetica", 10),justify=CENTER)
kurang15 = StringVar()
kurang15label = Label(root,textvariable= kurang15 , font=("Helvetica", 10),justify=CENTER)
kurang16 = StringVar()
kurang16label = Label(root,textvariable= kurang16 , font=("Helvetica", 10),justify=CENTER)

kuranglist = [[kurang1,kurang2,kurang3,kurang4],[kurang5,kurang6,kurang7,kurang8],[kurang9,
kurang10,kurang11,kurang12],[kurang13,kurang14,kurang15,kurang16]]
kuranglabels = [[kurang1label, kurang2label, kurang3label, kurang4label],[kurang5label,
kurang6label, kurang7label, kurang8label],[kurang9label, kurang10label, kurang11label,
kurang12label],[kurang13label, kurang14label, kurang15label, kurang16label]]
# set grid for all kurang
for i in range(4):
    for j in range(4):
        kuranglist[i][j].set('')
        kuranglabels[i][j].grid(row=i+1, column=j+6)
```

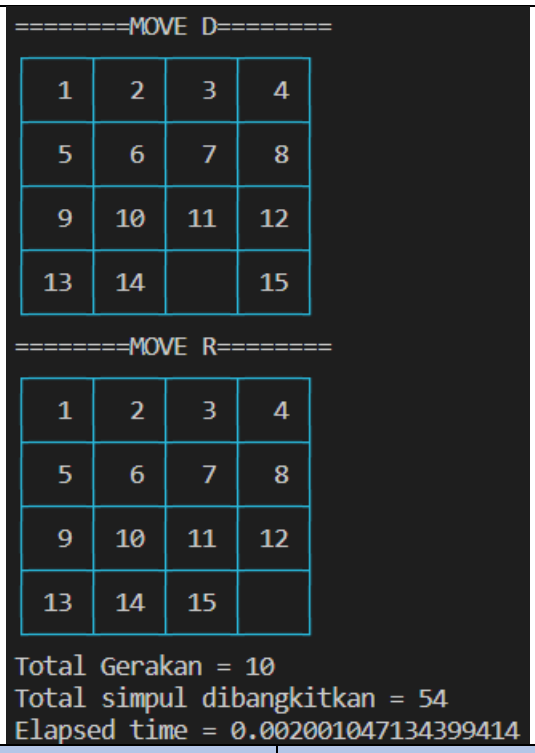
```
#Navigation Label
navVal = StringVar()
navlabel = Label(root, textvariable=navVal, font=("Helvetica", 10),justify=CENTER)
navlabel.grid(row=6, column=0, columnspan=4)

#Buttons
helpbutton = Button(root, text="Help", command=help)
helpbutton.grid(row=0, column=3)
leftbutton = Button(root, text="<", command=showprev, width=3, state=DISABLED)
leftbutton.grid(row=5, column=0)
resetbutton = Button(root, text="Reset", command=reset, width=10)
resetbutton.grid(row=5, column=1,columnspan=2)
rightbutton = Button(root, text=">", command=shownext, width=3, state=DISABLED)
rightbutton.grid(row=5, column=3)
randombutton = Button(root, text="Random", command=randomize, height=2, width = 12)
randombutton.grid(row=0, column=4, rowspan=2)
solvebutton = Button(root, text="Solve", command=solvePuzzle, height=2, width = 12)
solvebutton.grid(row=1, column=4, rowspan=2)
openfilebutton = Button(root, text="Open File", command=openfile, height=2, width = 12)
openfilebutton.grid(row=2, column=4, rowspan=2)

#set StringVar for each entry
entry1 = StringVar()
entry2 = StringVar()
entry3 = StringVar()
entry4 = StringVar()
entry5 = StringVar()
entry6 = StringVar()
entry7 = StringVar()
entry8 = StringVar()
entry9 = StringVar()
entry10 = StringVar()
entry11 = StringVar()
entry12 = StringVar()
entry13 = StringVar()
entry14 = StringVar()
entry15 = StringVar()
entry16 = StringVar()
entries = [[entry1,entry2,entry3,entry4],[entry5,entry6,entry7,entry8],[entry9,entry10,
entry11,entry12],[entry13,entry14,entry15,entry16]]
```

```
#Create Grid of 3x3 aligned center, filled 1-15
e1 = Entry(root, width=5,borderwidth= 5, justify=CENTER, textvariable= entry1)
e2 = Entry(root, width=5, borderwidth= 5, justify=CENTER, textvariable= entry2)
e3 = Entry(root, width=5, borderwidth= 5, justify=CENTER,textvariable= entry3)
e4 = Entry(root, width=5, borderwidth= 5, justify=CENTER,textvariable= entry4)

e5 = Entry(root, width=5, borderwidth= 5, justify=CENTER,textvariable= entry5)
e6 = Entry(root, width=5, borderwidth= 5, justify=CENTER,textvariable= entry6)
e7 = Entry(root, width=5, borderwidth= 5, justify=CENTER,textvariable= entry7)
e8 = Entry(root, width=5, borderwidth= 5, justify=CENTER,textvariable= entry8)

e9 = Entry(root, width=5, borderwidth= 5, justify=CENTER,textvariable= entry9)
e10 = Entry(root, width=5, borderwidth= 5, justify=CENTER,textvariable= entry10)
e11 = Entry(root, width=5, borderwidth= 5, justify=CENTER,textvariable= entry11)
e12 = Entry(root, width=5, borderwidth= 5, justify=CENTER,textvariable= entry12)

e13 = Entry(root, width=5, borderwidth= 5, justify=CENTER,textvariable= entry13)
e14 = Entry(root, width=5, borderwidth= 5, justify=CENTER,textvariable= entry14)
e15 = Entry(root, width=5, borderwidth= 5, justify=CENTER,textvariable= entry15)
e16 = Entry(root, width=5, borderwidth= 5, justify=CENTER,textvariable= entry16)

grids = [e1,e2,e3,e4,e5,e6,e7,e8,e9,e10,e11,e12,e13,e14,e15,e16]
#set grid for all entry
for i in range(16):
    e = grids[i]
    e.grid(row=(i//4)+1, column=i%4, ipady=5)

# Set each grid to numbers 1-15 and blank
setEntries([[i*4+j+1 for j in range(4)] for i in range(4)])
e1.focus_force()
root.mainloop()
```

Bagian III
Screenshoot Program Testing

Berikut merupakan kumpulan hasil penjalanan program beserta text instansiasi 15-puzzle yang digunakan:

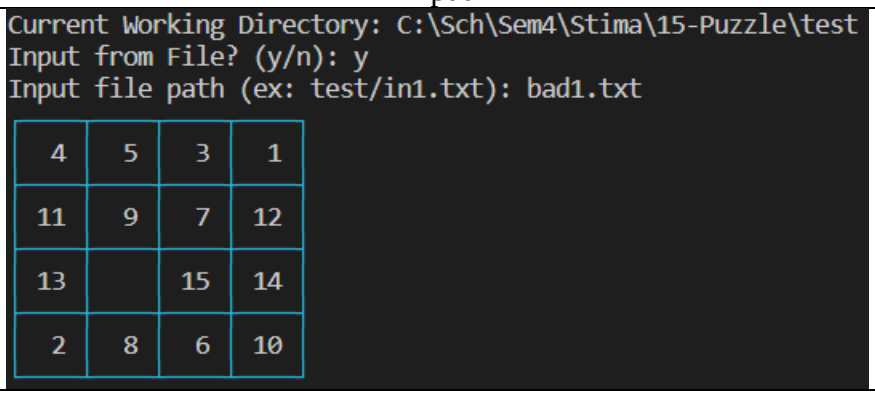
<table>
<tr><td>Configuration File</td><td>good1.txt</td></tr>
<tr><td colspan="2">5 1 16 4
2 6 3 8
9 10 7 12
13 14 11 15</td></tr>
<tr><td colspan="2">Input</td></tr>
<tr><td colspan="2">
<pre>Current Working Directory: C:\Sch\Sem4\Stima\15-Puzzle\test
Input from File? (y/n): y
Input file path (ex: test/in1.txt): good1.txt</pre>
<table>
<tr><td>5</td><td>1</td><td></td><td>4</td></tr>
<tr><td>2</td><td>6</td><td>3</td><td>8</td></tr>
<tr><td>9</td><td>10</td><td>7</td><td>12</td></tr>
<tr><td>13</td><td>14</td><td>11</td><td>15</td></tr>
</table>
</td></tr>
<tr><td colspan="2">Output</td></tr>
<tr><td colspan="2">
<pre>===============NILAI KURANG(i)===============
KURANG(1) = 0
KURANG(2) = 0
KURANG(3) = 0
KURANG(4) = 2
KURANG(5) = 4
KURANG(6) = 1
KURANG(7) = 0
KURANG(8) = 1
KURANG(9) = 1
KURANG(10) = 1
KURANG(11) = 0
KURANG(12) = 1
KURANG(13) = 1
KURANG(14) = 1
KURANG(15) = 0
KURANG(16) = 13
============================
Σ(Kurang(i)) + X = 26 + 0 = 26</pre>
</td></tr>
</table>



Configuration File	bad1.txt

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

Input



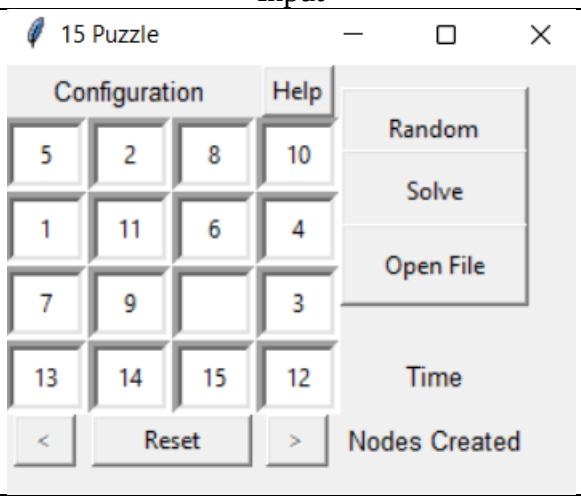
Output

```
===============NILAI KURANG(i)===============
KURANG(1) = 0
KURANG(2) = 0
KURANG(3) = 2
KURANG(4) = 3
KURANG(5) = 3
KURANG(6) = 0
KURANG(7) = 2
KURANG(8) = 1
KURANG(9) = 4
KURANG(10) = 0
KURANG(11) = 6
KURANG(12) = 4
KURANG(13) = 4
KURANG(14) = 4
KURANG(15) = 5
KURANG(16) = 6
============================
Σ(Kurang(i)) + X = 44 + 1 = 45

Configuration Can't Be Solved!
```

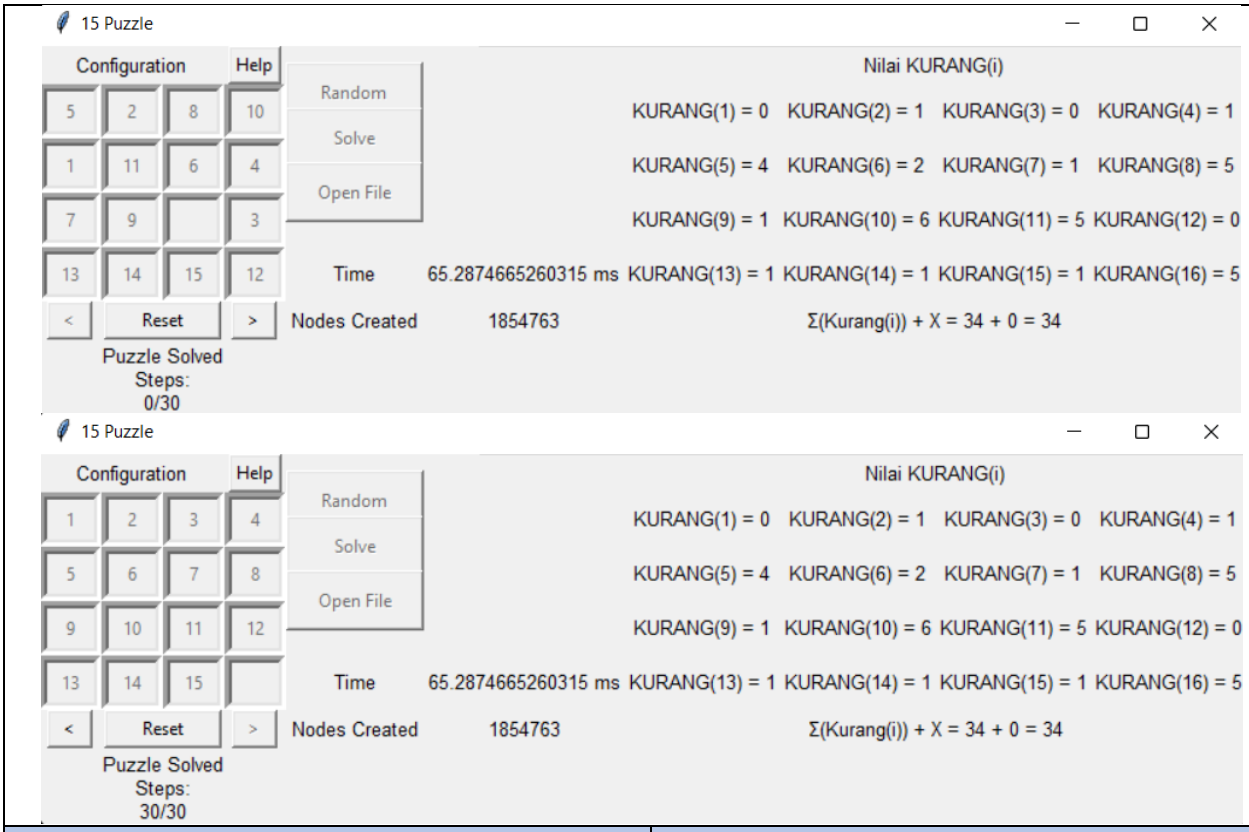
Configuration File	good2.txt

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

Input



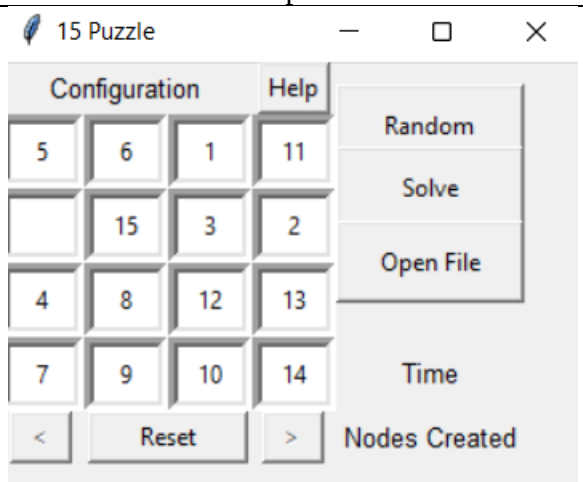
Output



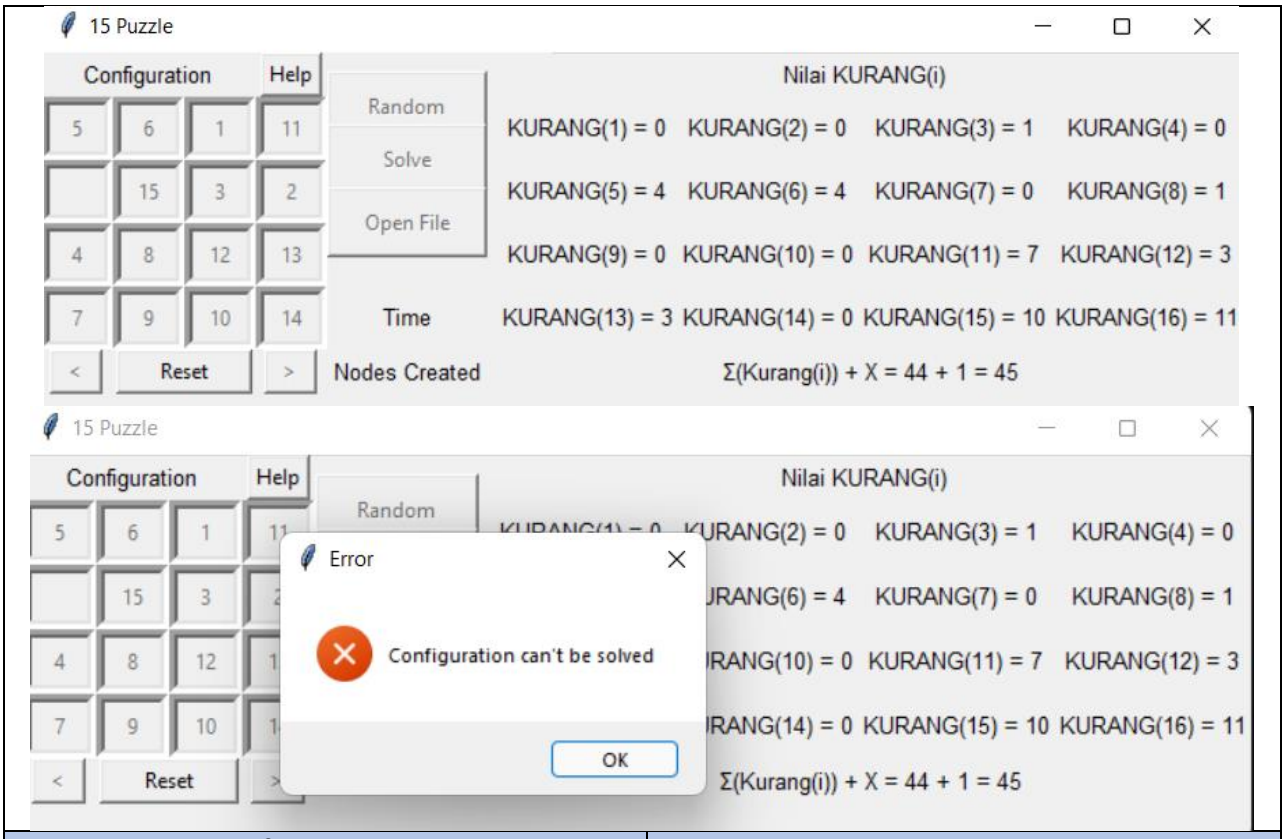
Configuration File	bad2.txt

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

Input



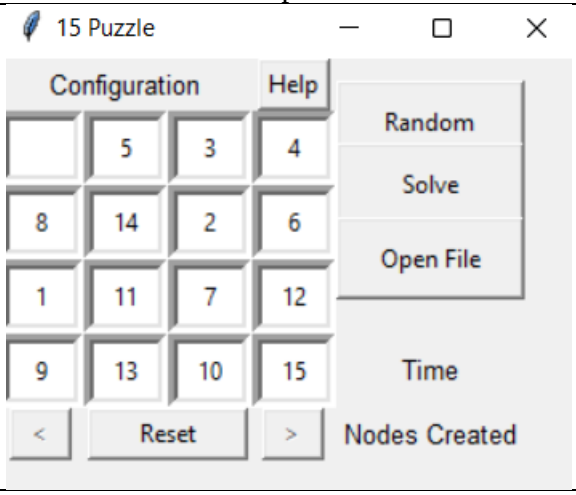
Output



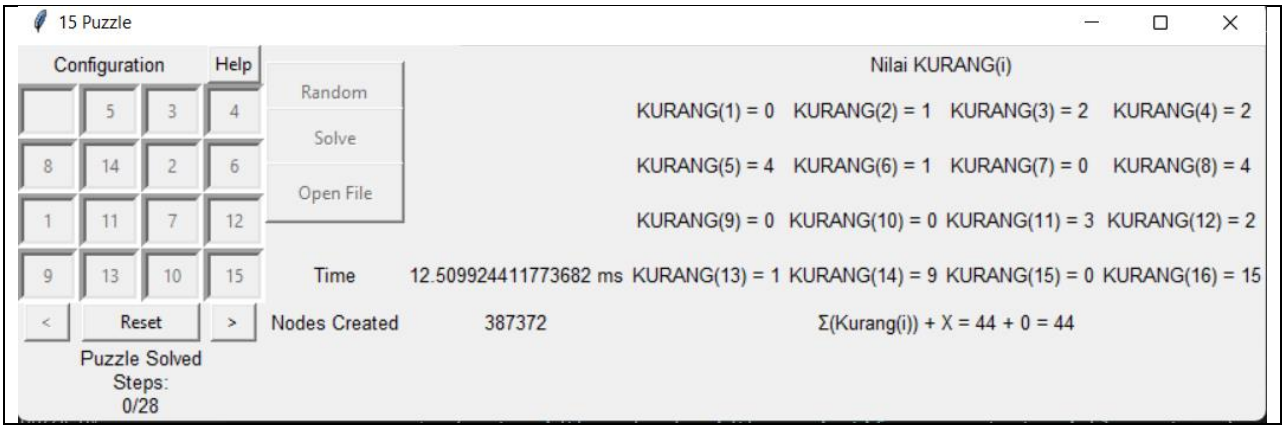
Configuration File	good3.txt

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

Input



Output


15 Puzzle
Configuration
Help
Random
Solve
Open File
5
3
4
8
14
2
6
1
11
7
12
9
13
10
15
<
Reset
>
Puzzle Solved
Steps:
0/28
Time
12.509924411773682 ms
Nodes Created
387372
Nilai KURANG(i)
KURANG(1) = 0
KURANG(2) = 1
KURANG(3) = 2
KURANG(4) = 2
KURANG(5) = 4
KURANG(6) = 1
KURANG(7) = 0
KURANG(8) = 4
KURANG(9) = 0
KURANG(10) = 0
KURANG(11) = 3
KURANG(12) = 2
KURANG(13) = 1
KURANG(14) = 9
KURANG(15) = 0
KURANG(16) = 15
Σ(Kurang(i)) + X = 44 + 0 = 44

Bagian IV
Link Source Code / Github

Berikut terlampir link drive source code serta isinya sesuai dengan spesifikasi :
https://drive.google.com/drive/u/1/folders/111DbcWby1qEFbqSt1Nol8JlgdMQDOWpP

Berikut Repository Github:
https://github.com/Nk-Kyle/15-Puzzle

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	✓	