Tugas Kecil 2 IF2211 Strategi Algoritma

Semester II tahun 2021/2022

Pencarian Convex Hull Menggunakan Algoritma Divide and Conquer

Dipersiapkan oleh:

Rozan Fadhil Al Hafidz

13520039

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung

Jl. Ganesha 10, Bandung 40132

Daftar Isi

1 Algoritma Branch and Bound yang Digunakan

Berikut ini adalah algoritma Branch and Bound yang digunakan dalam menyelesaikan permasalahan 16-puzzle

1.1 Mengecek apakah puzzle dapat diselesaikan

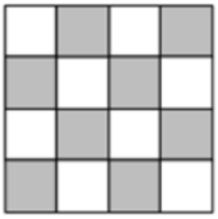
Untuk mengecek apakah puzzle dapat diselesaikan, program yang saya buat menghitung nilai jumlah dari KURANG(i) ditambah X terlebih dahulu

$$\sum_{i=1}^{16} KURANG(i) + X$$

Dengan:

KURANG(i) adalah banyaknya ubin bernomor j sedemikian sehingga j < i dan POSISI(j) > POSISI(i). POSISI(i) = posisi ubin bernomor i pada susunanyang diperiksa.

X bernilai 1 jika kotak kosong berada pada posisi yang diarsir. Jika tidak, X bernilai 0



Misalkan pada puzzle berikut, jumlah KURANG(i) dan X adalah

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

i	Kurang (i)
1	0
2	0
3	1
4	1
5	0
6	0
7	1
8	0
9	0
10	0
11	3
12	6
13	0
14	4
15	11
16	10

X = 1

$$\sum_{i=1}^{16} KURANG(i) + X = 37$$

Karena jumlah KURANG(i) dan X bernilai ganjil, maka puzzle tersebut tidak memiliki solusi

1.2 Memasukkan puzzle awal ke dalam antrian (*queue*).

Saya menerapkan konsep pohon dalam menyelesaikan permasalahan ini. Namun dalam melakukan pencarian solusi, simpul-simpul di dalam pohon tersebut dimasukkan ke dalam antrian untuk memilih simpul mana yang akan dieksekusi terlebih dahulu. Pencarian berhenti jika solusi ditemukan atau simpul pada antrian sudah habis.

1.3 Memilih puzzle dalam antrian yang memiliki biaya terkecil

Dalam melakukan pencarian solusi, program yang saya buat memilih simpul dengan biaya terkecil untuk dieksekusi terlebih dahulu. Biaya dihitung dengan cara berikut

$$\hat{c} = f(P) * 0.05 + \hat{g}(P)$$

Dengan:

$\hat{c}$ adalah biaya
f(P) adalah jumlah langkah yang telah diambil puzzle (kedalaman node)
$\hat{g}(P)$ adalah jumlah ubin yang tidak berada di tempat yang benar

Saya mengalikan f(P) dengan 0.05 karena pada pada percobaan yang saya lakukan, program dapat menyelesaikan puzzle dengan jauh lebih cepat dengan formula tersebut. Oleh karena itu, saya menyimpulkan jumlah ubin di tempat yang benar jauh lebih menentukan ketemunya solusi pada puzzle dibandingkan dengan jumlah langkah yang telah diambil.

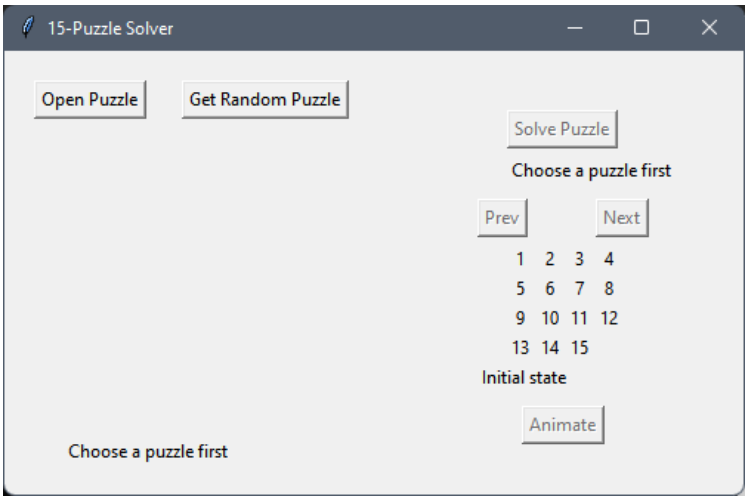
Kelebihan formula tersebut adalah dapat menyelesaikan puzzle dengan waktu yang jauh lebih cepat dari formula sebelumnya. Namun, cara tersebu juga memiliki kekurangan, yaitu solusi yang dihasilkan memiliki jumlah langkah yang lebih banyak.

1.4 Eksekusi puzzle yang dipilih

Cek apakah simpul yang dipilih merupakan simpul solusi. Jika iya, cetak solusi tersebut dan hentikan program. Jika bukan, bangkitkan anak-anak dari simpul tersebut, yaitu keadaan puzzle tersebut jika ubin kosong digerakkan ke atas, kanan, kiri, ataupun bawah jika memungkinkan. Cek apakah susunan puzzle dari setiap anak tersebut sudah pernah ada atau belum. Jika belum, masukan ke dalam antrian. Setelah itu, ulangi langkah ke-3 sampai solusi ketemu.

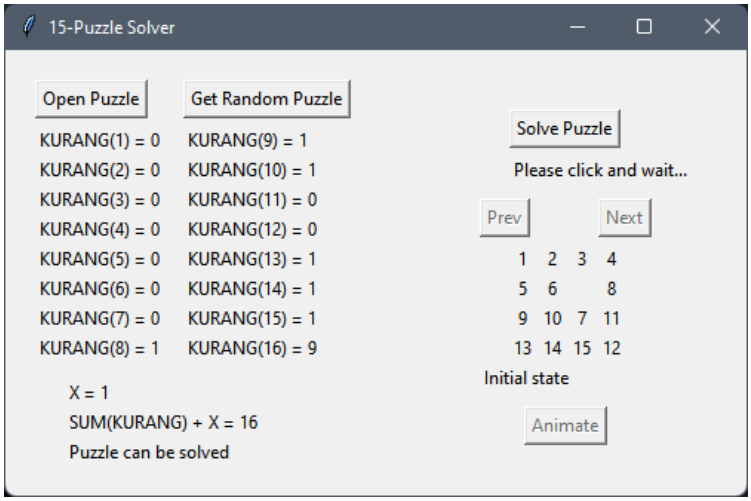
2 Screen-shot Input-output Program

2.1 Tampilan Awal GUI

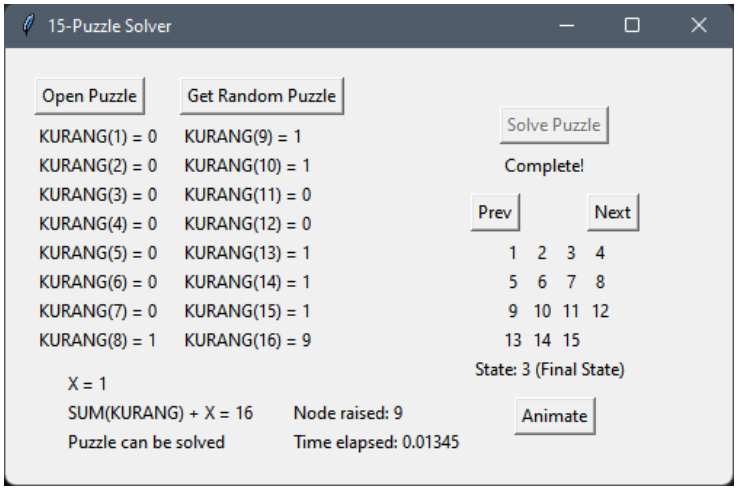


2.2 Puzzle Dari Spek

2.2.1 Input

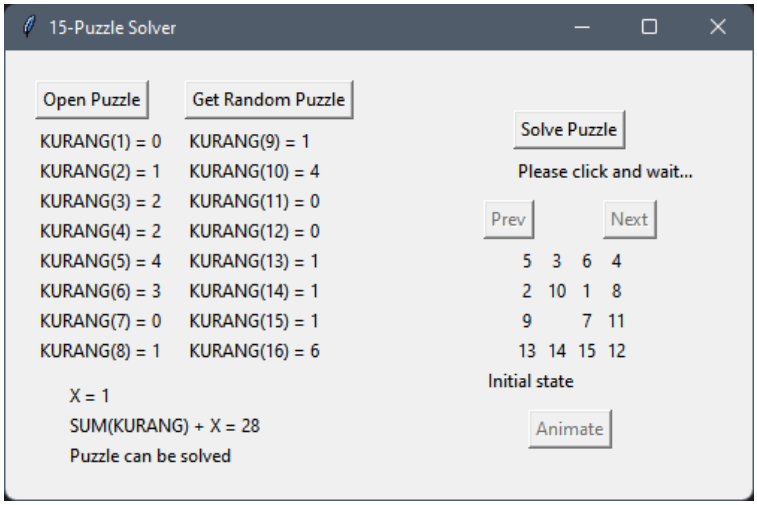


2.2.2 Output

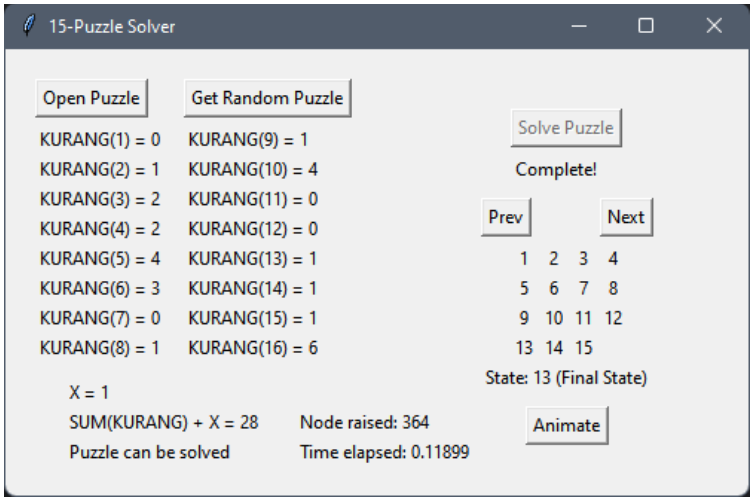


2.3 Puzzle 1

2.3.1 Input

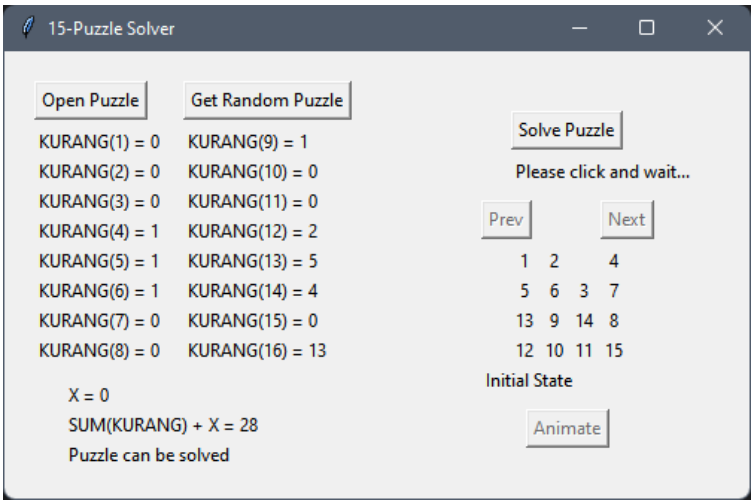


2.3.2 Output

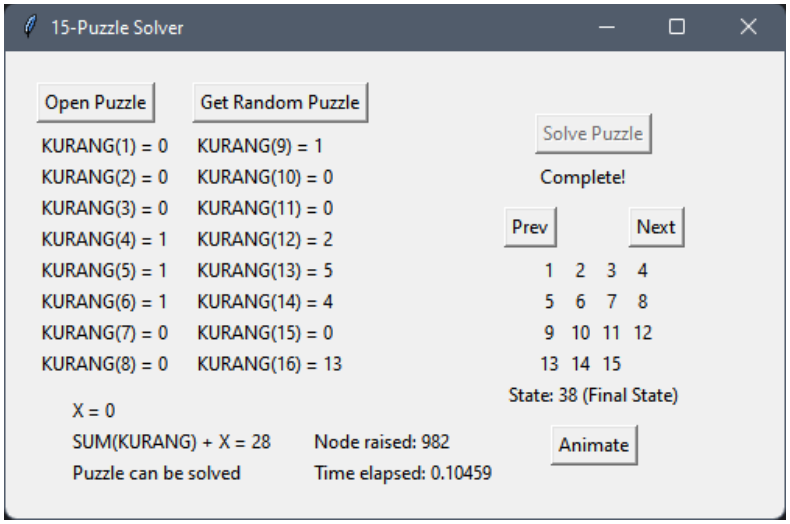


2.4 Puzzle 2

2.4.1 Input

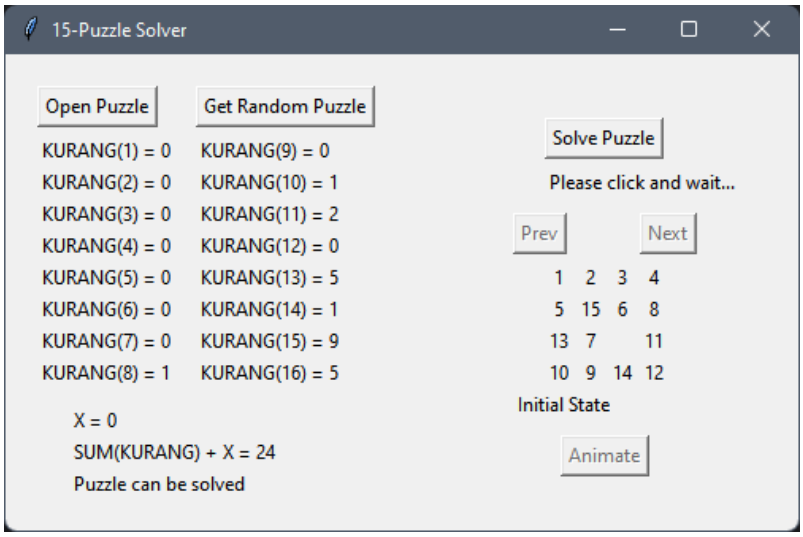


2.4.2 Output

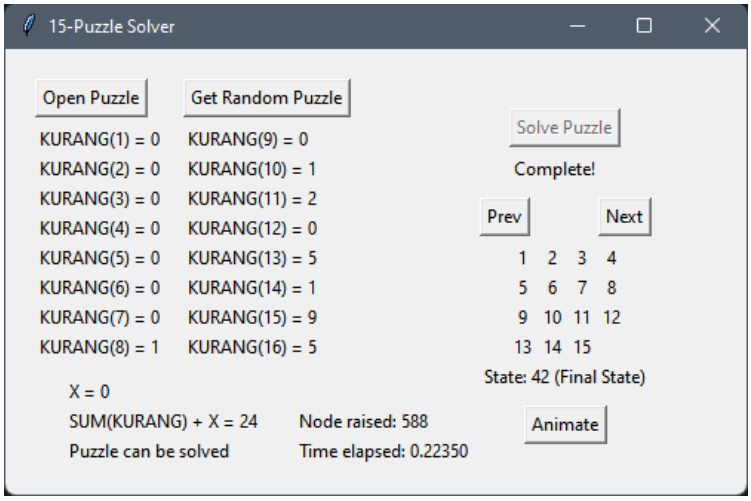


2.5 Puzzle 3

2.5.1 Input

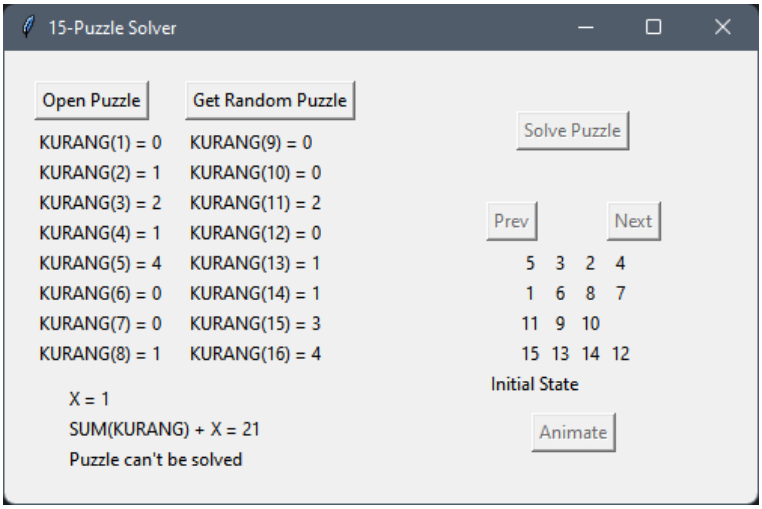


2.5.2 Output



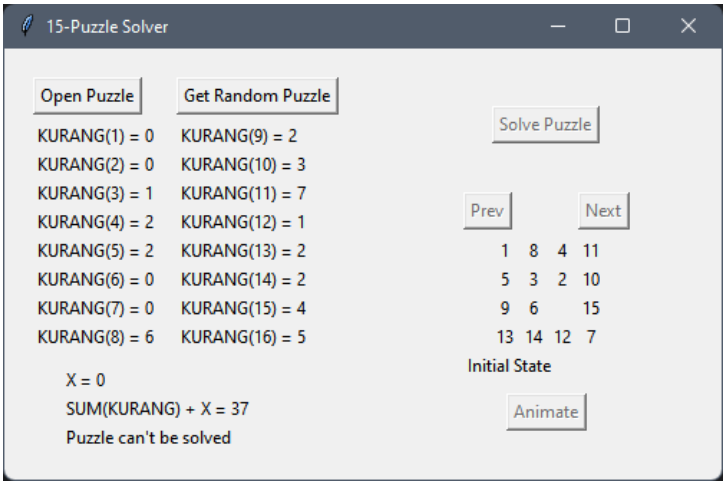
2.6 Puzzle yang Tidak Dapat Diselesaikan 1

2.6.1 Input/Output



2.7 Puzzle yang Tidak Dapat Diselesaikan 2

2.7.1 Input/Output



3 Checklist Program

Checklist sebagai berikut:

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	√	

4 Kode Program dalam Bahasa Python

4.1 Ubin.py

```
# Kelas untuk menyimpan ubin
class Ubin:
    def __init__(self, value):
        self.value = value
```

4.2 Puzzle.py

```
from Ubin import Ubin

# Kelas untuk menyimpan puzzle
class Puzzle:
    # Mencari index dari ubin dengan nilai yang diinginkan
    def findUbinLocation(self, value):
        for i in range(16):
            if self.ubin[i].value == value:
                return i
        return -1

    # Mencari nilai dari KURANG(i)
    def kurang(self, i):
        count = 0
        idx = self.findUbinLocation(i+1)
        for j in range(idx, 16):
            if self.ubin[idx].value > self.ubin[j].value:
                count += 1
        return count

    # Mencari jumlah dari seluruh KURANG(i)
    def findSumOfKurang(self):
        sum = 0
        for i in range(16):
            kurang_i = self.kurang(i)
            print(f"KURANG({i+1}) = {kurang_i}")
            sum += kurang_i
        return sum

    # Mengembalikan setiap dilai dari KURANG(i) dalam bentuk list
    def getKurangList(self):
        kurangList = []
        self.sumOfKurang = 0
        for i in range(16):
            kurang_i = self.kurang(i)
            kurangList.append(kurang_i)
            self.sumOfKurang += kurang_i
        return kurangList

    # Mengecek nilai dari X sesuai lokasi ubin kosong
    def checkX(self):
        if self.blankUbin == 0:
            self.X = 0
        if self.blankUbin == 1:
```

```
            self.X = 1
        if self.blankUbin == 2:
            self.X = 0
        if self.blankUbin == 3:
            self.X = 1
        if self.blankUbin == 4:
            self.X = 1
        if self.blankUbin == 5:
            self.X = 0
        if self.blankUbin == 6:
            self.X = 1
        if self.blankUbin == 7:
            self.X = 0
        if self.blankUbin == 8:
            self.X = 0
        if self.blankUbin == 9:
            self.X = 1
        if self.blankUbin == 10:
            self.X = 0
        if self.blankUbin == 11:
            self.X = 1
        if self.blankUbin == 12:
            self.X = 1
        if self.blankUbin == 13:
            self.X = 0
        if self.blankUbin == 14:
            self.X = 1
        if self.blankUbin == 15:
            self.X = 0
        return self.X

    # ctor
    def __init__(self, listUbin):
        self.ubin = [Ubin(i) for i in listUbin]     # Isi dari Ubin
        self.steps = []         # Langkah yang dilakukan
        self.stepsCount = 0     # Jumlah langkah yang dilakukan

        # Mencari lokasi ubin kosong
        for i in range(16):
            if listUbin[i] == 16 :
                self.blankUbin = i
                break
```

```
    # Menginisialisasi CLI
    def initializeCLI(self):
        print("Posisi awal:")
        self.print()

        self.sumOfKurang = self.findSumOfKurang()
        self.checkX()
        print(f"X = {self.X}")

        print(f"Sum of KURANG(i) + X = {self.sumOfKurang + self.X}")
        print()


    # Mendapatkan langkah terakhir yang dilakukan
    def getLastStep(self):
        return self.steps[-1] if len(self.steps) > 0 else "No steps"



    # Menukar 2 ubin
    def swap(self, i, j):
        temp = self.ubin[i]
        self.ubin[i] = self.ubin[j]
        self.ubin[j] = temp

    # Mencetak isi puzzle ke layar
    def print(self):
        for i in range(16):
            if i % 4 == 0:
                print('|', end='')
            if self.ubin[i].value == 16:
                print("  ", end='|')
            else:
                print(str(self.ubin[i].value).rjust(2, ' '), end='|')
            if i % 4 == 3:
                print()
        print()

    # Menghitung ubin yang tidak berada pada tempat seharusnya
    def countNotInRightPlace(self):
        count = 0
        for i in range(16):
            if self.ubin[i].value != i+1:
                count += 1
        return count
```

```
    # Menggeser ubin kosong ke atas
    def swapUp(self):
        topUbin = self.blankUbin - 4
        self.swap(topUbin, self.blankUbin)
        self.blankUbin = topUbin
        self.steps.append("U")
        self.stepsCount += 1

    # Menggeser ubin kosong ke bawah
    def swapDown(self):
        bottomUbin = self.blankUbin + 4
        self.swap(bottomUbin, self.blankUbin)
        self.blankUbin = bottomUbin
        self.steps.append("D")
        self.stepsCount += 1

    # Menggeser ubin kosong ke kiri
    def swapLeft(self):
        leftUbin = self.blankUbin - 1
        self.swap(leftUbin, self.blankUbin)
        self.blankUbin = leftUbin
        self.steps.append("L")
        self.stepsCount += 1

    # Menggeser ubin kosong ke kanan
    def swapRight(self):
        rightUbin = self.blankUbin + 1
        self.swap(rightUbin, self.blankUbin)
        self.blankUbin = rightUbin
        self.steps.append("R")
        self.stepsCount += 1

    # Mengembalikan isi dari puzzle dalam bentuk list of string
    def toStringList(self):
        list = []
        for i in range(16):
            list.append(str(self.ubin[i].value if self.ubin[i].value != 16 else
" ").rjust(2, ' '))
        print(list)
        return list

    def toPuzzleId(self):
        id = ""
        for i in range(16):
```

```
            id += str(self.ubin[i].value)
        return id
```

4.3 puzzleRandomizer.py

```
import random
from Puzzle import Puzzle

# Membuat puzzle secara acak
def getRandomPuzzle():
    # Definisikan puzzle yang sudah diselesaikan
    p = Puzzle([i+1 for i in range(16)])
    count = 0

    # Acak acak puzzle
    for i in range(50):
        r = random.randint(1,6)
        if (r == 1 or r == 5) and p.blankUbin > 3 and p.getLastStep() != "U":
            p.swapUp()
            count += 1
        if r == 2 and p.blankUbin < 12 and p.getLastStep() != "D":
            p.swapDown()
            count += 1
        if r == 3 and p.blankUbin % 4 != 3 and p.getLastStep() != "R":
            try:
                p.swapRight()
                count += 1
            except:
                pass
        if (r == 4 or r == 6) and p.blankUbin % 4 != 0 and p.getLastStep() !=
"L":
            try:
                p.swapLeft()
                count += 1
            except:
                pass
        p.print()

    # Masukkan puzzle ke list
    arr = []
    for i in range(16):
        arr.append(p.ubin[i].value)
    return arr
```

4.4 fileReader.py

```
# Fungsi untuk membaca input dari file
def readPuzzleFromFile(fileName):
    f = open(fileName, "r")
    puzzle = []
    for line in f:
        line = line.strip()
        if line == "":
            continue
        puzzle.append(line)
    f.close()

    # Jika diletakkan dalam 1 baris
    if len(puzzle) == 1:
        puzzle = puzzle[0].split(" ")

    # Jika diletakkan dalam bentuk matriks
    elif len(puzzle) == 4:
        puzzle = puzzle[0].split(" ") + puzzle[1].split(" ") +
puzzle[2].split(" ") + puzzle[3].split(" ")

    if len(puzzle) != 16:
        print("Puzzle tidak valid")
        return
    for i in range(16):
        puzzle[i] = int(puzzle[i])
    return puzzle
```

4.5 puzzleSolver.py

```
import copy
from Puzzle import Puzzle
from SolvedPuzzle import SolvedPuzzle
import time

puzzleList = []
puzzleHistoryMap = {}
found = False
result = None
nodeCount = 0

# Mengecek apakah dua ubin bernilai sama
def checkIfUbinIsEqual(ubin1, ubin2):
```

```
    for i in range(16):
        if ubin1[i].value != ubin2[i].value:
            return False
    return True

# Mengecek apakah puzzle sudah pernah ada
def nodeIdIsInHistoryMap(puzzle):
    global puzzleHistoryMap
    id = puzzle.toPuzzleId()
    return id in puzzleHistoryMap

# Mengeksekusi node
def executeNode(puzzle):
    global puzzleList
    global puzzleHistoryMap
    global found
    global result
    global nodeCount
    if puzzle.countNotInRightPlace() == 0:  # Jika sudah ketemu
        found = True
        result = puzzle
    else:   # Bangkitkan simpul anak
        if puzzle.blankUbin % 4 != 3 and puzzle.getLastStep() != "R":
            swapRightPuzzle = copy.deepcopy(puzzle)
            swapRightPuzzle.swapRight()
            if not nodeIdIsInHistoryMap(swapRightPuzzle):
                puzzleList.append(swapRightPuzzle)
                puzzleHistoryMap[swapRightPuzzle.toPuzzleId()] = True
                nodeCount += 1
        if puzzle.blankUbin % 4 != 0 and puzzle.getLastStep() != "L":
            swapLeftPuzzle = copy.deepcopy(puzzle)
            swapLeftPuzzle.swapLeft()
            if not nodeIdIsInHistoryMap(swapLeftPuzzle):
                puzzleList.append(swapLeftPuzzle)
                puzzleHistoryMap[swapLeftPuzzle.toPuzzleId()] = True
                nodeCount += 1
        if puzzle.blankUbin < 12 and puzzle.getLastStep() != "D":
            swapDownPuzzle = copy.deepcopy(puzzle)
            swapDownPuzzle.swapDown()
            if not nodeIdIsInHistoryMap(swapDownPuzzle):
                puzzleList.append(swapDownPuzzle)
                puzzleHistoryMap[swapDownPuzzle.toPuzzleId()] = True
                nodeCount += 1
        if puzzle.blankUbin > 3 and puzzle.getLastStep() != "U":
            swapUpPuzzle = copy.deepcopy(puzzle)
```

```
            swapUpPuzzle.swapUp()
            if not nodeIdIsInHistoryMap(swapUpPuzzle):
                puzzleList.append(swapUpPuzzle)
                puzzleHistoryMap[swapUpPuzzle.toPuzzleId()] = True
                nodeCount += 1


# Mencari index node dengan jarak terpendek
def findShortestDistanceIndex():
    global puzzleList
    min = puzzleList[0].countNotInRightPlace() + puzzleList[0].stepsCount *
0.05
    index = 0
    for i in range(1, len(puzzleList)):
        currCost = puzzleList[i].countNotInRightPlace() +
puzzleList[i].stepsCount * 0.05
        if currCost < min:
            min = currCost
            index = i
    return index

# Membersihkan variabel global
def clearGlobalVariable():
    global found
    global result
    global puzzleList
    global puzzleHistoryMap
    puzzleList = []
    puzzleHistoryMap = {}
    found = False
    result = None


# Fungsi utama untuk mencari solusi puzzle
# Digunakan untuk CLI
def solvePuzzle(puzzleAwal):
    global found
    global result
    global puzzleList
    global nodeCount
    clearGlobalVariable()
    before = time.time()
    puzzle = Puzzle(puzzleAwal)
    puzzle.initializeCLI()
```

```
    # Cek apakah puzzle bisa diseelsaikan
    if (puzzle.sumOfKurang + puzzle.X) % 2 == 1:
        print("Sum of Kurang + X bernilai ganjil")
        print("Puzzle tidak dapat diselesaikan")
    else:
        print("Please wait...")

        # Masukkan puzzle awal ke dalam antrian
        puzzleList = [puzzle]

        # Eksekusi node sampai ketemu
        while len(puzzleList) > 0 and not found:
            idx = findShortestDistanceIndex()
            executeNode(puzzleList[idx])
            puzzleList.pop(idx)
        after = time.time()

        sp = SolvedPuzzle(puzzleAwal, result.steps)
        sp.printStepByStep()
        print("Node count:", nodeCount)
        print("Step count:", result.stepsCount)
        print("Time elapsed:",after-before, "seconds")

# Fungsi utama untuk mendapatkan puzzle yang sudah dipecahkan
# Digunakan untuk GUI
def getSolvedPuzzle(puzzleAwal):
    global found
    global result
    global puzzleList
    clearGlobalVariable()
    before = time.time()
    puzzle = Puzzle(puzzleAwal)
    puzzle.initializeCLI()

    # Cek apakah puzzle bisa diseelsaikan
    if (puzzle.sumOfKurang + puzzle.X) % 2 == 1:
        print("Sum of Kurang + X bernilai ganjil")
        print("Puzzle tidak dapat diselesaikan")
    else:
        print("Please wait...")

        # Masukkan puzzle awal ke dalam antrian
        puzzleList = [puzzle]

        # Eksekusi node sampai ketemu
```

```
        while len(puzzleList) > 0 and not found:
            idx = findShortestDistanceIndex()
            executeNode(puzzleList[idx])
            puzzleList.pop(idx)

        after = time.time()

        sp = SolvedPuzzle(puzzleAwal, result.steps)
        return sp, after-before, nodeCount
```

4.6 SolvedPuzzle.py

```
from Ubin import Ubin

# Kelas untuk menyimpan puzzle yang sudah dipecahkan
class SolvedPuzzle:
    def __init__(self, listUbin, steps):
        self.steps = steps
        self.ubin = [Ubin(i) for i in listUbin]
        self.ubinAwal = [Ubin(i) for i in listUbin]

        for i in range(16):
            if listUbin[i] == 16 :
                self.blankUbin = i
                self.blankUbinAwal = i
                break



        self.stepsCount = len(self.steps)
        self.state = 0

        self.print()

    # Menukar 2 ubin
    def swap(self, i, j):
        temp = self.ubin[i]
        self.ubin[i] = self.ubin[j]
        self.ubin[j] = temp

    # Menggeser nilai dari ubin kosong ke atas
    def swapUp(self):
        topUbin = self.blankUbin - 4
        self.swap(topUbin, self.blankUbin)
```

```
        self.blankUbin = topUbin

    # Menggeser ubin kosong ke bawah
    def swapDown(self):
        bottomUbin = self.blankUbin + 4
        self.swap(bottomUbin, self.blankUbin)
        self.blankUbin = bottomUbin

    # Menggeser ubin kosong ke kiri
    def swapLeft(self):
        leftUbin = self.blankUbin - 1
        self.swap(leftUbin, self.blankUbin)
        self.blankUbin = leftUbin

    # Menggeser ubin kosong ke kanan
    def swapRight(self):
        rightUbin = self.blankUbin + 1
        self.swap(rightUbin, self.blankUbin)
        self.blankUbin = rightUbin

    # Mencetak puzzle langkah demi langkah
    # Digunakan untuk CLI
    def printStepByStep(self):
        for step in self.steps:
            if step == "U":
                self.swapUp()
                print("UP")
            elif step == "D":
                self.swapDown()
                print("DOWN")
            elif step == "L":
                self.swapLeft()
                print("LEFT")
            elif step == "R":
                self.swapRight()
                print("RIGHT")
            self.print()

    # Mencetak isi puzzle ke layar
    def print(self):
        for i in range(16):
            if i % 4 == 0:
                print('|', end='')
            if self.ubin[i].value == 16:
                print("  ", end='|')
```

```
            else:
                print(str(self.ubin[i].value).rjust(2, ' '), end='|')
            if i % 4 == 3:
                print()
        print()

    # Mengembalikan isi dari puzzle dalam bentuk list of string
    def toStringList(self):
        list = []
        for i in range(16):
            list.append(str(self.ubin[i].value if self.ubin[i].value != 16 else
" ").rjust(2, ' '))
        return list

    # Pergi ke state selanjutnya
    def nextStep(self):
        if self.state < self.stepsCount:
            if self.steps[self.state] == "U":
                self.swapUp()
            elif self.steps[self.state] == "D":
                self.swapDown()
            elif self.steps[self.state] == "L":
                self.swapLeft()
            elif self.steps[self.state] == "R":
                self.swapRight()
            self.state += 1

    # Pergi ke state sebelumnya
    def prevStep(self):
        if self.state > 0:
            if self.steps[self.state-1] == "U":
                self.swapDown()
            elif self.steps[self.state-1] == "D":
                self.swapUp()
            elif self.steps[self.state-1] == "L":
                self.swapRight()
            elif self.steps[self.state-1] == "R":
                self.swapLeft()
            self.state -= 1
```

4.7 main.py

```
from fileReader import readPuzzleFromFile
```

```
from puzzleSolver import solvePuzzle
from puzzleRandomizer import getRandomPuzzle
import sys
import os

# Pindah direktori ke parent-nya
os.chdir("..")

# Buat string path ke folder test
path = os.getcwd() + "\\test\\"

# Cek apakah tidak terdapat argumen
if len(sys.argv) < 2:
    print("1. Input dari file")
    print("2. Input dari CLI")
    print("3. Dapatkan puzzle secara acak")
    pilihan = input("Pilihan: ")
    print()

    if pilihan == "1": # Input dari file
        print("(File harus berada di dalam folder test)")
        fileName = input("Nama file: ")
        p = readPuzzleFromFile(path + fileName)

    elif pilihan == "2": # Input dari CLI
        p = []
        for i in range(16):
            p.append(int(input(f"Ubin {i+1}: ")))

    elif pilihan == "3": # Dapatkan puzzle secara acak
        p = getRandomPuzzle()

    else: # Input tidak valid
        print("Pilihan tidak valid")
        exit()

else:
    p = readPuzzleFromFile(path + sys.argv[1])

solvePuzzle(p)
```

4.8 mainGUI.py

```
from tkinter import *
from SolvedPuzzle import SolvedPuzzle
from fileReader import readPuzzleFromFile
from puzzleSolver import getSolvedPuzzle, solvePuzzle
from Puzzle import Puzzle
from puzzleRandomizer import getRandomPuzzle
from tkinter import filedialog as fd

# Deklarasi variabel global
mainGUI = Tk()
mainGUI.title("15-Puzzle Solver")
mainGUI.geometry("500x300")
mainGUI.minsize(500,300)
mainGUI.maxsize(500,300)
p = None
puzzle = None
timeElapsed = 0
nodeCount = 0


# Inisiasi GUI setelah menerima puzzle
def initialize(puzzleInput):
    global puzzle
    puzzle = None
    puzzle = puzzleInput
    global p
    puzzleObject = None
    puzzleObject = Puzzle(puzzle)
    arr = puzzleObject.toStringList()
    label1.config(text = arr[0])
    label2.config(text = arr[1])
    label3.config(text = arr[2])
    label4.config(text = arr[3])
    label5.config(text = arr[4])
    label6.config(text = arr[5])
    label7.config(text = arr[6])
    label8.config(text = arr[7])
    label9.config(text = arr[8])
    label10.config(text = arr[9])
    label11.config(text = arr[10])
    label12.config(text = arr[11])
    label13.config(text = arr[12])
    label14.config(text = arr[13])
    label15.config(text = arr[14])
    label16.config(text = arr[15])
```

```
    kurangList = puzzleObject.getKurangList()
    kurang_1.config(text = f"KURANG(1) = {kurangList[0]}")
    kurang_2.config(text = f"KURANG(2) = {kurangList[1]}")
    kurang_3.config(text = f"KURANG(3) = {kurangList[2]}")
    kurang_4.config(text = f"KURANG(4) = {kurangList[3]}")
    kurang_5.config(text = f"KURANG(5) = {kurangList[4]}")
    kurang_6.config(text = f"KURANG(6) = {kurangList[5]}")
    kurang_7.config(text = f"KURANG(7) = {kurangList[6]}")
    kurang_8.config(text = f"KURANG(8) = {kurangList[7]}")
    kurang_9.config(text = f"KURANG(9) = {kurangList[8]}")
    kurang_10.config(text = f"KURANG(10) = {kurangList[9]}")
    kurang_11.config(text = f"KURANG(11) = {kurangList[10]}")
    kurang_12.config(text = f"KURANG(12) = {kurangList[11]}")
    kurang_13.config(text = f"KURANG(13) = {kurangList[12]}")
    kurang_14.config(text = f"KURANG(14) = {kurangList[13]}")
    kurang_15.config(text = f"KURANG(15) = {kurangList[14]}")
    kurang_16.config(text = f"KURANG(16) = {kurangList[15]}")
    X = puzzleObject.checkX()
    X_label.config(text = f"X = {X}")
    sumOfKurangValue = puzzleObject.sumOfKurang
    sum_of_kurang.config(text = f"SUM(KURANG) + X = {sumOfKurangValue + X}")
    if (sumOfKurangValue + X) % 2 == 0:
        bSolvePuzzle.config(state=ACTIVE)
        lPuzzleCanBeSolved.config(text = "Puzzle can be solved")
        lPleaseWait.config(text = "Please click and wait...")
    else :
        bSolvePuzzle.config(state=DISABLED)
        lPuzzleCanBeSolved.config(text = "Puzzle can't be solved")
        lPleaseWait.config(text = "")
    lTimeElapsed.config(text = "")
    lNodeCount.config(text = "")
    stateLabel.config(text = "Initial State")



# Memilih file untuk puzzle
def chooseFile():
    global p
    global puzzle
    b1.config(state=DISABLED)
    b2.config(state=DISABLED)
    bSolvePuzzle.config(state=DISABLED)
    p = None
```

```
    puzzle = None
    filename = None
    filename = fd.askopenfilename()
    initialize(readPuzzleFromFile(filename))

    bAnimate.config(state=DISABLED)

# Mendapatkan puzzle random
def getRandom():
    global p
    global puzzle
    b1.config(state=DISABLED)
    b2.config(state=DISABLED)
    bSolvePuzzle.config(state=DISABLED)
    p = None
    puzzle = None
    initialize(getRandomPuzzle())
    bSolvePuzzle.config(state=ACTIVE)
    bAnimate.config(state=DISABLED)


# Menyelesaikan puzzle
def solvePuzzle():
    global p
    global puzzle
    global timeElapsed
    global nodeCount
    p = None
    p, timeElapsed, nodeCount = getSolvedPuzzle(puzzle)
    b1.config(state=ACTIVE)
    b2.config(state=ACTIVE)
    bAnimate.config(state=ACTIVE)
    lPleaseWait.config(text = "Complete!")
    lTimeElapsed.config(text = "Time elapsed: {:.5f}".format(timeElapsed))
    lNodeCount.config(text = "Node raised: {}".format(nodeCount))
    bSolvePuzzle.config(state=DISABLED)
    stateLabel.config(text = "Initial State")

# Menempelkan puzzle ke label
def putInLabel():
    global p
    arr = p.toStringList()
    label1.config(text = arr[0])
    label2.config(text = arr[1])
    label3.config(text = arr[2])
```

```
    label4.config(text = arr[3])
    label5.config(text = arr[4])
    label6.config(text = arr[5])
    label7.config(text = arr[6])
    label8.config(text = arr[7])
    label9.config(text = arr[8])
    label10.config(text = arr[9])
    label11.config(text = arr[10])
    label12.config(text = arr[11])
    label13.config(text = arr[12])
    label14.config(text = arr[13])
    label15.config(text = arr[14])
    label16.config(text = arr[15])

# Maju ke langkah selanjutnya
def next():
    global p
    p.nextStep()
    arr = p.toStringList()
    label1.config(text = arr[0])
    label2.config(text = arr[1])
    label3.config(text = arr[2])
    label4.config(text = arr[3])
    label5.config(text = arr[4])
    label6.config(text = arr[5])
    label7.config(text = arr[6])
    label8.config(text = arr[7])
    label9.config(text = arr[8])
    label10.config(text = arr[9])
    label11.config(text = arr[10])
    label12.config(text = arr[11])
    label13.config(text = arr[12])
    label14.config(text = arr[13])
    label15.config(text = arr[14])
    label16.config(text = arr[15])
    if p.state == p.stepsCount:
        stateLabel.config(text = f"State: {p.state} (Final State)")
    else:
        stateLabel.config(text = "State: " + str(p.state))

# Mundur ke langkah sebelumnya
def prev():
    global p
    p.prevStep()
    arr = p.toStringList()
```

```
    label1.config(text = arr[0])
    label2.config(text = arr[1])
    label3.config(text = arr[2])
    label4.config(text = arr[3])
    label5.config(text = arr[4])
    label6.config(text = arr[5])
    label7.config(text = arr[6])
    label8.config(text = arr[7])
    label9.config(text = arr[8])
    label10.config(text = arr[9])
    label11.config(text = arr[10])
    label12.config(text = arr[11])
    label13.config(text = arr[12])
    label14.config(text = arr[13])
    label15.config(text = arr[14])
    label16.config(text = arr[15])
    if p.state == 0:
        stateLabel.config(text = "State: 0 (Initial State)")
    else:
        stateLabel.config(text = "State: " + str(p.state))

# Animasikan puzzle
def animatePuzzle():
    global p
    p = SolvedPuzzle(puzzle, p.steps)
    for i in range(p.stepsCount + 1):
        if i == 0:
            mainGUI.after(i * 500, lambda: putInLabel())
        else:
            mainGUI.after(i * 500, lambda: next())



puzzleFrame = Frame(mainGUI, width = 200, height = 250)
puzzleFrame.place(x = 300, y = 90)
label1 = Label(puzzleFrame, text = " 1")
label2 = Label(puzzleFrame, text = " 2")
label3 = Label(puzzleFrame, text = " 3")
label4 = Label(puzzleFrame, text = " 4")
label5 = Label(puzzleFrame, text = " 5")
label6 = Label(puzzleFrame, text = " 6")
label7 = Label(puzzleFrame, text = " 7")
label8 = Label(puzzleFrame, text = " 8")
label9 = Label(puzzleFrame, text = " 9")
label10 = Label(puzzleFrame, text = "10")
```

```
label11 = Label(puzzleFrame, text = "11")
label12 = Label(puzzleFrame, text = "12")
label13 = Label(puzzleFrame, text = "13")
label14 = Label(puzzleFrame, text = "14")
label15 = Label(puzzleFrame, text = "15")
label16 = Label(puzzleFrame, text = "   ")



label1.place(x = 40, y = 40)
label2.place(x = 60, y = 40)
label3.place(x = 80, y = 40)
label4.place(x = 100, y = 40)
label5.place(x = 40, y = 60)
label6.place(x = 60, y = 60)
label7.place(x = 80, y = 60)
label8.place(x = 100, y = 60)
label9.place(x = 40, y = 80)
label10.place(x = 60, y = 80)
label11.place(x = 80, y = 80)
label12.place(x = 100, y = 80)
label13.place(x = 40, y = 100)
label14.place(x = 60, y = 100)
label15.place(x = 80, y = 100)
label16.place(x = 100, y = 100)

b1 = Button(puzzleFrame, text = "Next", command = next)
b1.place(x = 100, y = 10)
b1.config(state = DISABLED)

b2 = Button(puzzleFrame, text = "Prev", command = prev)
b2.place(x = 20, y = 10)
b2.config(state = DISABLED)

stateLabel = Label(puzzleFrame, text = "Initial State")
stateLabel.place(x = 20, y = 120)


leftFrame = Frame(mainGUI, width = 300, height = 260)
leftFrame.place(x = 20, y = 20)

bOpenPuzzle = Button(leftFrame, text = "Open Puzzle", command = chooseFile)
bGetRandom = Button(leftFrame, text = "Get Random Puzzle", command = getRandom)
bOpenPuzzle.place(x = 0, y = 0)
bGetRandom.place(x = 100, y = 0)
```

```
kurang_1 = Label(leftFrame, text = " ")
kurang_1.place(x = 0, y = 30)
kurang_2 = Label(leftFrame, text = " ")
kurang_2.place(x = 0, y = 50)
kurang_3 = Label(leftFrame, text = " ")
kurang_3.place(x = 0, y = 70)
kurang_4 = Label(leftFrame, text = " ")
kurang_4.place(x = 0, y = 90)
kurang_5 = Label(leftFrame, text = " ")
kurang_5.place(x = 0, y = 110)
kurang_6 = Label(leftFrame, text = " ")
kurang_6.place(x = 0, y = 130)
kurang_7 = Label(leftFrame, text = " ")
kurang_7.place(x = 0, y = 150)
kurang_8 = Label(leftFrame, text = " ")
kurang_8.place(x = 0, y = 170)
kurang_9 = Label(leftFrame, text = " ")
kurang_9.place(x = 100, y = 30)
kurang_10 = Label(leftFrame, text = " ")
kurang_10.place(x = 100, y = 50)
kurang_11 = Label(leftFrame, text = " ")
kurang_11.place(x = 100, y = 70)
kurang_12 = Label(leftFrame, text = " ")
kurang_12.place(x = 100, y = 90)
kurang_13 = Label(leftFrame, text = " ")
kurang_13.place(x = 100, y = 110)
kurang_14 = Label(leftFrame, text = " ")
kurang_14.place(x = 100, y = 130)
kurang_15 = Label(leftFrame, text = " ")
kurang_15.place(x = 100, y = 150)
kurang_16 = Label(leftFrame, text = " ")
kurang_16.place(x = 100, y = 170)
X_label = Label(leftFrame, text = " ")
X_label.place(x = 20, y = 200)
sum_of_kurang = Label(leftFrame, text = " ")
sum_of_kurang.place(x = 20, y = 220)

lPuzzleCanBeSolved = Label(leftFrame, text = "Choose a puzzle first")
lPuzzleCanBeSolved.place(x = 20, y = 240)

bSolvePuzzle = Button(mainGUI, text = "Solve Puzzle", command = solvePuzzle,
state=DISABLED)
bSolvePuzzle.place(x = 340, y = 40)
```

```
bAnimate = Button(mainGUI, text = "Animate", command = animatePuzzle,
state=DISABLED)
bAnimate.place(x = 350, y = 240)

lPleaseWait = Label(mainGUI, text = "Choose a puzzle first")
lPleaseWait.place(x = 340, y = 70)


lTimeElapsed = Label(leftFrame, text = "")
lTimeElapsed.place(x = 175, y = 240)

lNodeCount = Label(leftFrame, text = "")
lNodeCount.place(x = 175, y = 220)

mainGUI.mainloop()
```

5 Contoh Persoalan 15-Puzzle

5.1 Puzzle 1

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

5.2 Puzzle 2

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

5.3 Puzzle 3

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

5.4 Puzzle yang Tidak Dapat Diselesaikan 1

```
5 3 2 4
1 6 8 7
```

11 9 10 16 15 13 14 12

5.5 Puzzle yang Tidak Dapat Diselesaikan 1

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

6 Alamat Github Kode Program

https://github.com/rozanfa/TucilStima3-BnB