LAPORAN TUGAS KECIL 3

"Penyelesaian Persoalan 15-Puzzle dengan Algoritma *Branch and Bound*"

Mata Kuliah Strategi Algoritma (IF2211)

KELAS 03

Dosen : Dr. Ir. Rinaldi, M.T.

DISUSUN OLEH:

Rheza Rizqullah Ecaldy (13520060)

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

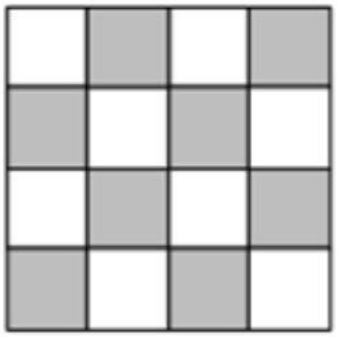
SEMESTER GENAP TAHUN 2021-2022

A. Cara Kerja Program

Program yang saya buat ini terdiri atas 5 modul, yaitu puzzleBoard.py, priorityQueue.py, searchState.py, bnbFunction.py, dan main.py. Modul puzzleBoard berisi implementasi sebuah kelas bernama puzzleBoard yang memiliki atribut berupa matriks bernama board yang gunanya menyimpan posisi angka pada puzzle. Selain itu, puzzleBoard juga menyimpan method yang berhubungan dengan matriks board. Modul priorityQueue berisi implementasi kelas PriorityQueue yang nantinya akan membantu pada proses *Branch and Bound*. Modul searchState berisi sebuah kelas bernama searchState yang mereprentasikan state yang sedang ditelusuri dalam pencarian dan memiliki 4 atribut, yaitu root berupa matriks state puzzle sekarang, prevState berupa searchState sebelumnya, prevMove berupa string nama move sebelumnya, dan level berupa integer yang menggambarkan level kedalaman state dari root awal. Modul bnbFunction berisi implementasi fungsi cost dan solusi dari algoritma. Modul main berisi proses input/output dan badan utama dari algoritma *Branch and Bound*.

Algoritma yang saya gunakan dalam tucil ini memanfaatkan konsep *Branch and Bound*. Secara umum, cara kerja program ini adalah sebagai berikut :

1. Program meminta user untuk memilih apakah puzzle akan di-*generate* secara acak atau diinput sebagai text. Kemudian, state awal puzzle akan disimpan.
2. Kemudian, dihitung nilai total dari fungsi Kurang[i] untuk seluruh 16 ubin pada matriks puzzle dengan i adalah indeks dari matriks setelah diubah menjadi bentuk 1 dimensi. Kurang[i] adalah banyaknya ubin bernomor j sedemikian sehingga j < i dan POSISI(j) > POSISI(i). POSISI(i) = posisi ubin bernomor i pada susunan yang diperiksa.
3. Kemudian, dihitung nilai X. X bernilai 1 jika sel kosong berada pada posisi yang diarsir. Sebaliknya, X bernilai 0.



4. Apabila nilai total fungsi Kurang[i] ditambah X bernilai genap, puzzle dapat dicapai solusinya. Sebaliknya, tidak bisa.
5. Nilai level suatu state ditambah Fungsi g(i) dijadikan fungsi cost dari PriorityQueue. Nilai level merupakan nilai kedalaman dari state dari root. Nilai fungsi g(i) adalah banyaknya ubin yang tidak berada pada posisi seharusnya. State awal puzzle kemudian di push ke dalam priority queue.

6. Kemudian, penelusuran akan dilakukan dengan cara mengepop priorityQueue. Jika state yang dipop telah merupakan state solusi, maka pencarian selesai. Jika bukan merupakan solusi, akan dipush beberapa kemungkinan state puzzle apabila dikenakan command Up, Down, Left, atau Right. Proses 6 ini akan dilakukan terus menerus hingga priorityQueue kosong atau solusi ditemukan.
7. Ditampilkan langkah-langkah yang ditempuh hingga mencapai state solusi, jumlah langkah, jumlah node yang dibangkitkan, dan waktu berjalannya program.

B. Source Code dalam python

priorityQueue.py

```
class PriorityQueue: # Implementasi dari priority queue

    # Konstruktor dengan input fungsi prioritas dari queue
    def __init__(self, priority_function):
        self.queue = []
        self.func = priority_function

    # Mengecek apakah queue kosong
    def isEmpty(self):
        return len(self.queue) == 0
    # Melihat elemen pertama
    def first(self):
        return self.queue[0]
    # Melihat elemen terakhir
    def last(self):
        return self.queue[len(self.queue) - 1]
    # Memasukkan element ke dalam priority queue
    def push(self, element):
        pos = 0
        found = False
        while(not found and pos < len(self.queue)):
            if(self.func(element, self.queue[pos])):
                found = True
            else:
                pos+=1

        self.queue.insert(pos, element)
    # Menghapus elemen terdepan
    def pop(self):
        self.queue.pop(0)
```

puzzleBoard.py

```
import numpy as np
import copy
from priorityQueue import PriorityQueue

class PuzzleBoard:
    # Konstruktor dengan input berupa path testcase
    def __init__(self, path, cc):
        self.board = [[0, 0, 0, 0],[0, 0, 0, 0],[0, 0, 0, 0],[0, 0, 0, 0]]
        if(cc == 1) :
            mat = np.arange(1,17)
```

```
            np.random.shuffle(mat)
            self.board = np.reshape(mat, (4, 4))
            for i in range(4):
                for j in range(4):
                    tmp = self.board[i][j]
                    self.board[i][j] = int(tmp)
        else :
            f = open(path, "r")
            rowCount = 0
            for line in f:
                row = line.split()
                for i in range(4):
                    self.board[rowCount][i] = int(row[i])
                rowCount+=1
    # Mengubah board 2D jadi 1D
    def board1D(self):
        arr = [0 for i in range(16)]
        idx = 0
        for i in range(4):
            for j in range(4):
                arr[idx] = self.board[i][j]
                idx+=1
        return arr
    # Mencari index slot yang kosong
    def findBlankIdx(self):
        for i in range(4) :
            for j in range(4) :
                if(self.board[i][j] == 16) :
                    return (i, j)
    # Menghitung total Kurang[i]
    def checkKurang(self):
        arr = self.board1D()
        kurang = [0 for i in range(16)]
        sum = 0
        for i in range(16):
            for j in range(i+1, 16):
                if(arr[i] > arr[j]):
                    sum+=1
                    kurang[int(arr[i])-1]+=1

        print("Pengecekan nilai fungsi Kurang[i] : ")
        for i in range(16):
            print("Kurang["+str(i+1)+"] =", kurang[i])
        print()
        print("ΣKurang[i] =", sum)
        return sum
    # Mengecek posisi blank apakah di slot diarsir
    def checkX(self):
        (r, c) = self.findBlankIdx()
        x = (r+c) % 2
        print("X =", x)
        return x
    # Mengecek apakah puzzle solvable
    def isSolveable(self):
        Total = self.checkKurang() + self.checkX()

        print("Total (ΣKurang[i] + X) =", Total)
        print()
        return Total % 2 == 0
    # Memindahkan posisi blank ditambah dengan masukan units x dan y
    def move(self, x, y):
        (r, c) = self.findBlankIdx()
        if(r+x>=0 and r+x<4 and c+y>=0 and c+y<4):
            newPuzzle = copy.deepcopy(self)
            newPuzzle.board[r][c], newPuzzle.board[r+x][c+y] = newPuzzle.board[r+x][c+y], newPuzzle.board[r][c]
            return newPuzzle
        else:
            return None
```

```
    # Mencetak board
    def printBoard(self):
        for i in range(4):
            for j in range(4):
                if(self.board[i][j] == 16) :
                    print(" -", end="  ")
                elif(self.board[i][j] < 10) :
                    print("",self.board[i][j], end="  ")
                else :
                    print(self.board[i][j], end="  ")
            print()
```

searchState.py

```
class SearchState: # Menyimpan status dari langkah pencarian yang sedang ditelusuri

    def __init__(self, root, prevState, prevMove, level):
        self.root = root
        self.prevState = prevState
        self.prevMove = prevMove
        self.level = level
```

bnbFunction.py

```
# mengecek nilai g(i), yaitu jumlah slot yang tidak pada tempat seharusnya
def checkFungsiG(puzzle):
    arr = puzzle.board1D()
    value = 0

    for i in range(16):
        if(arr[i] != (i+1)):
            value+=1

    return value
# mengecek apabila telah mencapai solusi
def isSolution(puzzle):
    if(checkFungsiG(puzzle) == 0) :
        return True
    else :
        return False
# menyimpan semua state yang dilalui hingga mencapai solusi
def findSolutionRoute(solutionState):
    solution = []
    curState = solutionState.prevState
    prevState = solutionState
    while(curState != None):
       solution.insert(0,prevState)
       prevState = curState
       curState = curState.prevState

    return solution
```

main.py

```
# Nama  : Rheza Rizqullah Ecaldy
# NIM   : 13520060
# Kelas : K03
```

```
import time
import argparse
from bnbFunction import checkFungsiG
from bnbFunction import isSolution
from bnbFunction import findSolutionRoute
from puzzleBoard import PuzzleBoard
from priorityQueue import PriorityQueue
from searchState import SearchState
# Main Menu
print("SELAMAT DATANG DI 15 PUZZLE SOLVER REZMAN")
print()
print("Terdapat 2 tipe input: ")
print("1. Random generated matrix")
print("2. Input file txt")
print()
choice = int(input("Tipe input pilihan anda: "))
print()
if (choice == 1) :
    root = SearchState(PuzzleBoard("../test/", 1), None, "", 0)
elif (choice == 2) :
    # Input nama file
    filename = input("Masukkan nama file: ")
    print()
    # Menginisiasi startNode/root
    root = SearchState(PuzzleBoard("../test/" + filename, 2), None, "", 0)
else :
    print("Error gan")
    exit()
# mencetak hasil input
print("Puzzle :")
root.root.printBoard()
print()
# mengecek apakah puzzle bisa diselesaikan
if(not root.root.isSolveable()):
    print("Puzzle tidak bisa diselesaikan.")
    exit()
print("Puzzle bisa diselesaikan.")
print()
# Menetapkan fungsi g(i) sebagai fungsi prioritas dari priority queue
costFunction = checkFungsiG
# Menginisiasi priority queue
pq = PriorityQueue(lambda x,y : x.level + costFunction(x.root) <= y.level + costFunction(y.root))
# Memasukkan root puzzle
pq.push(root)
# Menginisiasi jenis moves yang bisa dilakukan
movesUnits = [(-1,0), (0,-1), (1,0), (0,1)]
movesNames = ["Up", "Left", "Down", "Right"]
# Menginisisasi perhitungan node dan state solusi
solutionState = None
nodeCount = 1
# Menghitung waktu mulai
timeStart = time.process_time_ns()
# Algoritma branch and bound untuk mencari solusi puzzle
while(not pq.isEmpty()):

    # Menetapkan elemen pertama pada priority queue menjadi state yang ditelusuri
    current = pq.first()
    pq.pop()
    # Mengecek apakah state yang ditelusuri sudah merupakan solusi
    if(isSolution(current.root)):
        solutionState = current
        break
    # Menelusuri semua gerakan yang mungkin
    for i, (dr, dc) in enumerate(movesUnits):
        # Mengecek apakah gerakan kebalikan dari gerakan sebelumnya
        if(movesNames[(i+2)%4] != current.prevMove):
            # Membangkitkan node baru
```

```
            nextNode = SearchState(current.root.move(dr, dc), prevState=current, prevMove=movesNames[i], level=current.le
vel+1)
            # Mengecek apabila node selanjutnya valid
            if(nextNode != None and nextNode.root != None):
                nodeCount += 1
                pq.push(nextNode)
# Menyimpan rute solusi
solution = findSolutionRoute(solutionState)
# Menghitung waktu selesai
timeEnd = time.process_time_ns()
# Mencetak langkah solusi
print("Solusi puzzle :")
for index, state in enumerate(solution):
    print("-----","Langkah", str(index+1) + ":", state.prevMove , "-----")
    state.root.printBoard()
    print()
# Mencetak waktu running program
time = timeEnd - timeStart
print("Waktu program berjalan :",time / 1000000, "ms")
# Mencetak total gerakan
print("Total langkah :", len(solution))
# Mencetak jumlah simpul yang dibangkitkan
print("Jumlah simpul dibangkitkan :",nodeCount)
```

C. Test Case

solveable1.txt
1 2 3 4
5 6 16 8
9 10 7 11
13 14 15 12

solveable2.txt
1 2 4 7
5 6 16 3
9 11 12 8
13 10 14 15

solveable3.txt
1 2 3 4
5 6 7 8
11 12 15 14
10 9 13 16

unsolveable1.txt
11 7 5 15
14 16 3 8
12 6 1 9

2 10 4 13

unsolveable2.txt
1 2 14 3
5 6 4 7
15 10 11 16
9 13 8 12

D. Input dan Output

Nomor	Input/Output
1	D:\KULIAH\Sem 4\Stima\Tucil\Tucil 3 15 puzzle\src>main.py SELAMAT DATANG DI 15 PUZZLE SOLVER REZMAN Terdapat 2 tipe input: 1. Random generated matrix 2. Input file txt Tipe input pilihan anda: 2 Masukkan nama file: solveable1.txt Puzzle : 1 2 3 4 5 6 - 8 9 10 7 11 13 14 15 12 Pengecekan nilai fungsi Kurang[i] : Kurang[1] = 0 Kurang[2] = 0 Kurang[3] = 0 Kurang[4] = 0 Kurang[5] = 0 Kurang[6] = 0 Kurang[7] = 0 Kurang[8] = 1 Kurang[9] = 1 Kurang[10] = 1

	Kurang[11] = 0 Kurang[12] = 0 Kurang[13] = 1 Kurang[14] = 1 Kurang[15] = 1 Kurang[16] = 9 ΣKurang[i] = 15 X = 1 Total (ΣKurang[i] + X) = 16 Puzzle bisa diselesaikan. Solusi puzzle : ----- Langkah 1: Down ----- 1 2 3 4 5 6 7 8 9 10 - 11 13 14 15 12 ----- Langkah 2: Right ----- 1 2 3 4 5 6 7 8 9 10 11 - 13 14 15 12 ----- Langkah 3: Down ----- 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 - Waktu program berjalan : 0.0 ms Total langkah : 3 Jumlah simpul dibangkitkan : 10
2	D:\KULIAH\Sem 4\Stima\Tucil\Tucil 3 15 puzzle\src>main.py SELAMAT DATANG DI 15 PUZZLE SOLVER REZMAN Terdapat 2 tipe input:

	1. Random generated matrix 2. Input file txt Tipe input pilihan anda: 2 Masukkan nama file: solveable2.txt Puzzle : 1 2 4 7 5 6 - 3 9 11 12 8 13 10 14 15 Pengecekan nilai fungsi Kurang[i] : Kurang[1] = 0 Kurang[2] = 0 Kurang[3] = 0 Kurang[4] = 1 Kurang[5] = 1 Kurang[6] = 1 Kurang[7] = 3 Kurang[8] = 0 Kurang[9] = 1 Kurang[10] = 0 Kurang[11] = 2 Kurang[12] = 2 Kurang[13] = 1 Kurang[14] = 0 Kurang[15] = 0 Kurang[16] = 9 ΣKurang[i] = 21 X = 1 Total (ΣKurang[i] + X) = 22 Puzzle bisa diselesaikan. Solusi puzzle : ----- Langkah 1: Right ----- 1 2 4 7

	5 6 3 - 9 11 12 8 13 10 14 15 ----- Langkah 2: Up ----- 1 2 4 - 5 6 3 7 9 11 12 8 13 10 14 15 ----- Langkah 3: Left ----- 1 2 - 4 5 6 3 7 9 11 12 8 13 10 14 15 ----- Langkah 4: Down ----- 1 2 3 4 5 6 - 7 9 11 12 8 13 10 14 15 ----- Langkah 5: Right ----- 1 2 3 4 5 6 7 - 9 11 12 8 13 10 14 15 ----- Langkah 6: Down ----- 1 2 3 4 5 6 7 8 9 11 12 - 13 10 14 15 ----- Langkah 7: Left ----- 1 2 3 4 5 6 7 8 9 11 - 12 13 10 14 15

	----- Langkah 8: Left ----- 1 2 3 4 5 6 7 8 9 - 11 12 13 10 14 15 ----- Langkah 9: Down ----- 1 2 3 4 5 6 7 8 9 10 11 12 13 - 14 15 ----- Langkah 10: Right ----- 1 2 3 4 5 6 7 8 9 10 11 12 13 14 - 15 ----- Langkah 11: Right ----- 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 - Waktu program berjalan : 0.0 ms Total langkah : 11 Jumlah simpul dibangkitkan : 70
3	D:\KULIAH\Sem 4\Stima\Tucil\Tucil 3 15 puzzle\src>main.py SELAMAT DATANG DI 15 PUZZLE SOLVER REZMAN Terdapat 2 tipe input: 1. Random generated matrix 2. Input file txt Tipe input pilihan anda: 2 Masukkan nama file: solveable3.txt Puzzle :

	1 2 3 4 5 6 7 8 11 12 15 14 10 9 13 - Pengecekan nilai fungsi Kurang[i] : Kurang[1] = 0 Kurang[2] = 0 Kurang[3] = 0 Kurang[4] = 0 Kurang[5] = 0 Kurang[6] = 0 Kurang[7] = 0 Kurang[8] = 0 Kurang[9] = 0 Kurang[10] = 1 Kurang[11] = 2 Kurang[12] = 2 Kurang[13] = 0 Kurang[14] = 3 Kurang[15] = 4 Kurang[16] = 0 ΣKurang[i] = 12 X = 0 Total (ΣKurang[i] + X) = 12 Puzzle bisa diselesaikan. Solusi puzzle : ----- Langkah 1: Up ----- 1 2 3 4 5 6 7 8 11 12 15 - 10 9 13 14 ----- Langkah 2: Left ----- 1 2 3 4 5 6 7 8 11 12 - 15

	10 9 13 14 ----- Langkah 3: Left ----- 1 2 3 4 5 6 7 8 11 - 12 15 10 9 13 14 ----- Langkah 4: Left ----- 1 2 3 4 5 6 7 8 - 11 12 15 10 9 13 14 ----- Langkah 5: Down ----- 1 2 3 4 5 6 7 8 10 11 12 15 - 9 13 14 ----- Langkah 6: Right ----- 1 2 3 4 5 6 7 8 10 11 12 15 9 - 13 14 ----- Langkah 7: Right ----- 1 2 3 4 5 6 7 8 10 11 12 15 9 13 - 14 ----- Langkah 8: Right ----- 1 2 3 4 5 6 7 8 10 11 12 15 9 13 14 - ----- Langkah 9: Up ----- 1 2 3 4

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

	----- Langkah 16: Right ----- 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 - Waktu program berjalan : 3062.5 ms Total langkah : 16 Jumlah simpul dibangkitkan : 1199
4	D:\KULIAH\Sem 4\Stima\Tucil\Tucil 3 15 puzzle\src>python main.py SELAMAT DATANG DI 15 PUZZLE SOLVER REZMAN Terdapat 2 tipe input: 1. Random generated matrix 2. Input file txt Tipe input pilihan anda: 2 Masukkan nama file: unsolveable1.txt Puzzle : 11 7 5 15 14 - 3 8 12 6 1 9 2 10 4 13 Pengecekan nilai fungsi Kurang[i] : Kurang[1] = 0 Kurang[2] = 0 Kurang[3] = 2 Kurang[4] = 0 Kurang[5] = 4 Kurang[6] = 3 Kurang[7] = 6 Kurang[8] = 4 Kurang[9] = 2 Kurang[10] = 1 Kurang[11] = 10 Kurang[12] = 6

	Kurang[13] = 0 Kurang[14] = 10 Kurang[15] = 11 Kurang[16] = 10 ΣKurang[i] = 69 X = 0 Total (ΣKurang[i] + X) = 69 Puzzle tidak bisa diselesaikan.
5	D:\KULIAH\Sem 4\Stima\Tucil\Tucil 3 15 puzzle\src>python main.py SELAMAT DATANG DI 15 PUZZLE SOLVER REZMAN Terdapat 2 tipe input: 1. Random generated matrix 2. Input file txt Tipe input pilihan anda: 2 Masukkan nama file: unsolveable2.txt Puzzle : 1 2 14 3 5 6 4 7 15 10 11 - 9 13 8 12 Pengecekan nilai fungsi Kurang[i] : Kurang[1] = 0 Kurang[2] = 0 Kurang[3] = 0 Kurang[4] = 0 Kurang[5] = 1 Kurang[6] = 1 Kurang[7] = 0 Kurang[8] = 0 Kurang[9] = 1 Kurang[10] = 2 Kurang[11] = 2

	Kurang[12] = 0 Kurang[13] = 2 Kurang[14] = 11 Kurang[15] = 6 Kurang[16] = 4 ΣKurang[i] = 30 X = 1 Total (ΣKurang[i] + X) = 31 Puzzle tidak bisa diselesaikan.

E. Link github

EdgarAllanPoo/Tucil3_13520060: Program solver 15 puzzle dengan menggunakan algoritma branch and bound oleh Rheza Rizqullah Ecaldy/13520060 (github.com)

F. Tabel Check List

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		√