

LAPORAN TUGAS KECIL 3

Penyelesaian Persoalan 15-Puzzle dengan Algoritma Branch and Bound

Ditujukan untuk memenuhi salah satu tugas kecil mata kuliah IF2211 Strategi Algoritma (Stima) pada Semester II Tahun Akademik 2021/2022

Disusun oleh:

Marcellus Michael Herman Kahari (K3) 13520057



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

2022

A. Algoritma *branch and bound*

Cara kerja algoritma *branch and bound* yang digunakan dalam pembuatan program penyelesaian Puzzle 15 adalah sebagai berikut. Pertama-tama, program akan meminta masukan dari pengguna apakah pengguna ingin menggunakan pembangkit bilangan untuk mengisi nilai-nilai dari matriks atau ingin menggunakan puzzle dari .txt. Puzzle tersebut akan dibaca oleh fungsi `readFile` yang menerima parameter input nama file dan mengembalikan matriks. Hasil matriks kemudian dibaca oleh class `Puzzle` dan dibentuk oleh constructor yang menerima input puzzle.

Class `Puzzle` memiliki beberapa atribut, yaitu `depth` atau kedalaman dari matriks awal menuju matriks tersebut, `cost` yaitu nilai dari `cost(i)`, array `historyRute` yang akan menyimpan urutan command yang akan dieksekusi, `posisiEmptyI` yaitu nilai dari kotak kosong sumbu Y, `posisiEmptyJ` yaitu nilai dari kotak kosong sumbu X. Class `Puzzle` juga memiliki beberapa method getter setter, `moveUp`, `moveDown`, `moveLeft`, `moveRight` yang merupakan method untuk menggeser kotak kosong, `printPuzzle` untuk memprint `Puzzle`, method `addRute` untuk menambahkan rute pada `historyRute`, dan `countCost` untuk menghitung nilai `cost(i)`

Setelah program menerima input dari pengguna, akan dioutputkan terlebih dahulu nilai `KURANG(i)` dan diakhir akan dioutputkan nilai total dari `KURANG(i) + X` dengan `X` adalah posisi dari kotak kosong. Untuk mendapatkan nilai dari `KURANG(i)`, digunakan fungsi `countIBelowJ` dan untuk mendapatkan nilai `X`, didapatkan dari fungsi `valueX` dimana `valueX` akan membaca sebuah file input dari luar, yaitu `valueX.txt` yang berisi nilai jika kotak kosong berada pada indeks tersebut. Jika nilai dari `KURANG(i) + X` ganjil, proses tidak dilanjutkan lagi.

Digunakan sebuah `Prioqueue` untuk menyimpan object dari class `Puzzle`. Pengurutan memasukkan `Prioqueue` didapatkan dari `objectPuzzle.getCost()`. Kemudian, terdapat pula array untuk menghitung jumlah simpul yang telah dibangkitkan. Terdapat sebuah `while` loop dimana jika indeks ke-0 tidak sama dengan puzzle solusi, program akan dilanjutkan hingga terdapat kesamaan. Di dalam `while` loop, dilakukan fungsi `pop(0)` pada `Prioqueue`. Hasil dari `pop(0)` tersebut akan digunakan untuk melakukan `move up`, `down`, `left`, `right`. Untuk setiap langkahnya, akan dilakukan pencopyan terlebih dahulu ke variabel sementara menggunakan `deepcopy`. Setelah dilakukan copy, akan dilakukan apakah indeks posisi kotak kosong memenuhi atau tidak ataupun apakah puzzle tersebut pernah dibangkitkan atau tidak. Jika memenuhi, dilakukan pergeseran sesuai command nya, kemudian pada variabel sementara itu, pada atribut `depth` dilakukan penambahan sejumlah satu yang menggambarkan bahwa telah terjadi satu langkah. Pada variabel sementara itu pula, dilakukan `append` pada `historyRute` menggunakan fungsi `addRute`. Kemudian diappend pada array yang digunakan untuk menghitung jumlah simpul yang dibangkitkan dan diinsert secara `Prioqueue` pada array `Prioqueue`.

Pada akhirnya, jika telah ditemukan puzzle yang sama dengan puzzle solusi, object class `Puzzle` yang sama tersebut dicek pada `historyRute` nya dan dibaca. Jika pada

historyRute berisi command 'up', akan dilakukan moveUp pada object tersebut kemudian dicetak. Begitu seterusnya hingga seluruh historyRute pada object tersebut telah dibaca semua.

Prinsip algoritma Branch and Bound terletak pada Prioqueue tersebut. Jika terdapat nilai cost itu paling kecil, maka anak-anaknya akan dibangkitkan. Akan tetapi, jika nilai cost tersebut tidak paling kecil, anak-anaknya tidak akan dibangkitkan. Jika sudah ditemukan solusinya, simpul-simpul yang memiliki cost lebih besar dari solusi kemudian akan dimatikan.

B. Source Program

1. Test Gagal (1)

Input :

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

Output :

```
Posisi KURANG(i) untuk setiap i:
0 , kurang( 0 ) bernilai 0
1 , kurang( 1 ) bernilai 1
2 , kurang( 2 ) bernilai 1
3 , kurang( 3 ) bernilai 11
4 , kurang( 4 ) bernilai 0
5 , kurang( 5 ) bernilai 1
6 , kurang( 6 ) bernilai 0
7 , kurang( 7 ) bernilai 5
8 , kurang( 8 ) bernilai 0
9 , kurang( 9 ) bernilai 1
10 , kurang( 10 ) bernilai 2
11 , kurang( 11 ) bernilai 3
12 , kurang( 12 ) bernilai 0
13 , kurang( 13 ) bernilai 0
14 , kurang( 14 ) bernilai 0
15 , kurang( 15 ) bernilai 0
Total KURANG(i) + X adalah 25
Tidak bisa diselesaikan
```

2. Test Gagal (2)

Input :

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

Output :

```
Posisi KURANG(i) untuk setiap i:
0 , kurang( 0 ) bernilai 2
1 , kurang( 1 ) bernilai 7
2 , kurang( 2 ) bernilai 0
3 , kurang( 3 ) bernilai 11
4 , kurang( 4 ) bernilai 10
5 , kurang( 5 ) bernilai 7
6 , kurang( 6 ) bernilai 1
7 , kurang( 7 ) bernilai 2
8 , kurang( 8 ) bernilai 6
9 , kurang( 9 ) bernilai 6
10 , kurang( 10 ) bernilai 4
11 , kurang( 11 ) bernilai 4
12 , kurang( 12 ) bernilai 0
13 , kurang( 13 ) bernilai 1
14 , kurang( 14 ) bernilai 1
15 , kurang( 15 ) bernilai 0
Total KURANG(i) + X adalah 63
Tidak bisa diselesaikan
```

3. Test Sukses (1)

Input :

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

Output :

Posisi KURANG(i) untuk setiap i:

```
0 , kurang( 0 ) bernilai 0
1 , kurang( 1 ) bernilai 0
2 , kurang( 2 ) bernilai 0
3 , kurang( 3 ) bernilai 0
4 , kurang( 4 ) bernilai 0
5 , kurang( 5 ) bernilai 0
6 , kurang( 6 ) bernilai 9
7 , kurang( 7 ) bernilai 5
8 , kurang( 8 ) bernilai 2
9 , kurang( 9 ) bernilai 2
10 , kurang( 10 ) bernilai 1
11 , kurang( 11 ) bernilai 0
12 , kurang( 12 ) bernilai 1
13 , kurang( 13 ) bernilai 1
14 , kurang( 14 ) bernilai 0
15 , kurang( 15 ) bernilai 0
Total KURANG(i) + X adalah 22
```

Posisi puzzle 15 awal:

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

Langkah ke- 1  adalah
*-----*
| 1  2  3  4 |
*-----*
| 5  6  8 12 |
*-----*
| 9 10      7 |
*-----*
| 13 14 11 15 |
*-----*
```

Langkah ke- 2 adalah

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

Langkah ke- 3 adalah

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

Langkah ke- 4 adalah

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

Langkah ke- 5 adalah

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

Langkah ke- 6 adalah

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

Langkah ke- 7 adalah

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

Banyak simpul yang dibuat adalah 41
waktu eksekusi program adalah 0.002 sekon

4. Test Sukses (2)

Input :

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

Output :

```

Posisi KURANG(i) untuk setiap i:
0 , kurang( 0 ) bernilai 0
1 , kurang( 1 ) bernilai 0
2 , kurang( 2 ) bernilai 0
3 , kurang( 3 ) bernilai 0
4 , kurang( 4 ) bernilai 0
5 , kurang( 5 ) bernilai 0
6 , kurang( 6 ) bernilai 9
7 , kurang( 7 ) bernilai 1
8 , kurang( 8 ) bernilai 1
9 , kurang( 9 ) bernilai 1
10 , kurang( 10 ) bernilai 0
11 , kurang( 11 ) bernilai 0
12 , kurang( 12 ) bernilai 1
13 , kurang( 13 ) bernilai 1
14 , kurang( 14 ) bernilai 1
15 , kurang( 15 ) bernilai 0
Total KURANG(i) + X adalah 16

Posisi puzzle 15 awal:
*-*-*-*
| 1 2 3 4 |
*-*-*-*
| 5 6 8 |
*-*-*-*
| 9 10 7 11 |
*-*-*-*
| 13 14 15 12 |
*-*-*-*

```

```
Langkah ke- 1 adalah
*-----*
| 1   2   3   4 |
*-----*
| 5   6   7   8 |
*-----*
| 9  10      11 |
*-----*
| 13  14   15  12 |
*-----*

Langkah ke- 2 adalah
*-----*
| 1   2   3   4 |
*-----*
| 5   6   7   8 |
*-----*
| 9  10   11 |
*-----*
| 13  14   15  12 |
*-----*

Langkah ke- 3 adalah
*-----*
| 1   2   3   4 |
*-----*
| 5   6   7   8 |
*-----*
| 9  10   11  12 |
*-----*
| 13  14   15 |
*-----*
```

5. Test Sukses (3)

Input :

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

Output :

```
Posisi KURANG(i) untuk setiap i:
0 , kurang( 0 ) bernilai 15
1 , kurang( 1 ) bernilai 1
2 , kurang( 2 ) bernilai 1
3 , kurang( 3 ) bernilai 1
4 , kurang( 4 ) bernilai 0
5 , kurang( 5 ) bernilai 1
6 , kurang( 6 ) bernilai 1
7 , kurang( 7 ) bernilai 1
8 , kurang( 8 ) bernilai 0
9 , kurang( 9 ) bernilai 1
10 , kurang( 10 ) bernilai 1
11 , kurang( 11 ) bernilai 1
12 , kurang( 12 ) bernilai 0
13 , kurang( 13 ) bernilai 0
14 , kurang( 14 ) bernilai 0
15 , kurang( 15 ) bernilai 0
Total KURANG(i) + X adalah 24
```

Posisi puzzle 15 awal:

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

Langkah ke- 5 adalah

```
*-*-*-*-*
| 1 2 3 4 |
*-*-*-*-*
| 5 6 7 8 |
*-*-*-*-*
| 9 10 11 12 |
*-*-*-*-*
| 13 14 15 |
*-*-*-*-*
Langkah ke- 6 adalah
*-*-*-*-*
| 1 2 3 4 |
*-*-*-*-*
| 5 6 7 8 |
*-*-*-*-*
| 9 10 11 12 |
*-*-*-*-*
| 13 14 15 |
*-*-*-*-*
```

Banyak simpul yang dibuat adalah 12
Waktu eksekusi program adalah 0.0 sekon

Langkah ke- 1 adalah

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

Langkah ke- 2 adalah

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

Langkah ke- 3 adalah

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

Langkah ke- 4 adalah

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

C. Checklist

Poin	Ya	Tidak
1. Program berhasil dikompilasi	Ya	
2. Program berhasil running	Ya	
3. Program dapat menerima input dan menuliskan output.	Ya	
4. Luaran sudah benar untuk semua data uji	Ya	
5. Bonus dibuat		Tidak

D. Source Program

Source code program dituliskan dapat bahasa Python

File Puzzle.py yang berisi class Puzzle

```
from helper import *

class Puzzle:
    # Daftar atribut
    depth = 0
    historyRute = ['blank']
    posisiEmptyI = 0
    posisiEmptyJ = 0

    # constructor
    def __init__(self, puzzle):
        self.puzzle = puzzle
        for i in range(4):
            for j in range(4):
                if(int(puzzle[i][j]) == 16):
                    self.setPosisiEmptyI(i)
                    self.setPosisiEmptyJ(j)
                    break

    def __lt__(self, other):
        return True

    # Getter
    def getPosisiEmptyI(self):
        return self.posisiEmptyI

    def getPosisiEmptyJ(self):
        return self.posisiEmptyJ
```



```

def getPuzzle(self):
    return self.puzzle

def getDepth(self):
    return self.depth

def getHistoryRute(self):
    return self.historyRute

# Setter
def setDepth(self, depth):
    self.depth += depth

def setPosisiEmptyI(self, posisiI):
    self.posisiEmptyI = posisiI

def setPosisiEmptyJ(self, posisiJ):
    self.posisiEmptyJ = posisiJ

# Method
def moveUp(self):
    if(self.posisiEmptyI != 0):
        self.puzzle[self.posisiEmptyI][self.posisiEmptyJ],
self.puzzle[self.posisiEmptyI-1][self.posisiEmptyJ] =
self.puzzle[self.posisiEmptyI-1][self.posisiEmptyJ],
self.puzzle[self.posisiEmptyI][self.posisiEmptyJ]
        self.posisiEmptyI -= 1

def moveDown(self):
    if(self.posisiEmptyI != 3):
        self.puzzle[self.posisiEmptyI][self.posisiEmptyJ],
self.puzzle[self.posisiEmptyI+1][self.posisiEmptyJ] =
self.puzzle[self.posisiEmptyI+1][self.posisiEmptyJ],
self.puzzle[self.posisiEmptyI][self.posisiEmptyJ]
        self.posisiEmptyI += 1

def moveLeft(self):
    if(self.posisiEmptyJ != 0):
        self.puzzle[self.posisiEmptyI][self.posisiEmptyJ],
self.puzzle[self.posisiEmptyI][self.posisiEmptyJ-1] =
self.puzzle[self.posisiEmptyI][self.posisiEmptyJ-1],
self.puzzle[self.posisiEmptyI][self.posisiEmptyJ]
        self.posisiEmptyJ -= 1

```

```

def moveRight(self):
    if(self.posisiEmptyJ != 3):
        self.puzzle[self.posisiEmptyI][self.posisiEmptyJ],
self.puzzle[self.posisiEmptyI][self.posisiEmptyJ+1] =
self.puzzle[self.posisiEmptyI][self.posisiEmptyJ+1],
self.puzzle[self.posisiEmptyI][self.posisiEmptyJ]
        self.posisiEmptyJ += 1

def printPuzzle(self):
    print("*-----*-----*-----*-----*")
    for i in range(4):
        print("|", end="")
        for j in range(4):
            if(self.puzzle[i][j] == 16):
                print('  ', end=' ')
            else:
                if(self.puzzle[i][j] < 10):
                    print(' ' + str(self.puzzle[i][j]), end=' ')
                else:
                    print(' ' + str(self.puzzle[i][j]), end=' ')
        print("|")
    print("*-----*-----*-----*-----*")

def addRute(self, rute):
    self.historyRute.append(rute)

def countCost(self, puzzle_solution):
    return countDifferent(self.puzzle, puzzle_solution) + self.depth

```

File Helper.py yang berisi fungsi-fungsi bantu

```

from copy import deepcopy
import random

def readFile(file):
    lines = []
    with open(file) as f:
        lines = f.readlines()
    count = 0
    puzzle = [[0 for i in range(4)] for j in range(4)]
    for line in lines:
        line = line.strip()
        count_temp = 0
        word = line.split(' ')

```

```

        for w in word:
            puzzle[count][count_temp] = int(w)
            count_temp += 1
        count += 1
    return puzzle

def countDifferent(puzzle, puzzle_solution):
    count = 0
    for i in range(4):
        for j in range(4):
            if (puzzle[i][j] != puzzle_solution[i][j]):
                count += 1
    return count

def valueX(puzzle):
    puzzle_16 = readFile('src/valueX.txt')
    for i in range(4):
        for j in range(4):
            if puzzle[i][j] == 16:
                return int(puzzle_16[i][j]), i, j

def countIBelowJ(puzzle, tampilkan):
    count = 0
    for i in range(4):
        for j in range(4):
            counter_tampilkan = 0
            for k in range(4):
                for l in range(4):
                    if (int(puzzle[i][j]) > int(puzzle[k][l])) and ((j + 4 * i) <
(k * 4 + l)):
                        count += 1
                        counter_tampilkan += 1
            if(tampilkan):
                print(4*i + j, ", kurang(", 4*i + j,") bernilai ",
counter_tampilkan)
    return count

def copyPuzzle(puzzle, puzzle_temp):
    for i in range(4):
        for j in range(4):
            puzzle_temp[i][j] = puzzle[i][j]
    return puzzle_temp

def generatePuzzle():
    puzzle = [[0 for i in range(4)] for i in range(4)]

```

```

pembangkit_acak = random.sample(range(1, 17), 16)
for i in range(4):
    for j in range(4):
        puzzle[i][j] = pembangkit_acak[4*i + j]
return puzzle

def functionMove(temp, tujuan, puzzle_saver, puzzle_solution, rute,
puzzle_count_node):
    temp.historyRute = deepcopy(rute)
    temp.setDepth(1)
    temp.addRute(tujuan)
    cost = temp.countCost(puzzle_solution)
    puzzle_saver.put((cost,temp))
    puzzle_count_node.append(temp)

```

File main.py yang berisi program utama untuk dijalankan

```

from helper import *
from puzzle import *
import time
from queue import PriorityQueue

print("""
  _ _ _ _ _
 / _ | _ _ | _ _ _ \ _ _ _ _ _
 | | | _ _ _ _ | _ _ ) | _ _ _ _ _
 | | _ _ \ _ _ | _ _ / | | | _ _ / _ _
 | | _ _ ) | _ _ | | | _ _ / _ _ / _ _
 | _ _ _ / _ _ | _ _ \ _ _ / _ _ | _ _ \ _ _
""")
print()
print("Selamat datang di permainan puzzle 15")
print("Kami tidak pernah meragukan pemain meski permintaannya aneh-aneh")
print("Kami akan membantu pemain untuk menyelesaikan puzzle ini")
print()
print("Anda ingin memasukkan puzzle anda sendiri dari file.txt atau ingin membuat puzzle dari pembangkit bilangan acak? (Masukkan nomornya saja)")
print("1. Masukkan puzzle dari file.txt")
print("2. Buat puzzle dari pembangkit bilangan acak")
tipe = int(input("Masukkan nomor: "))
if(tipe == 1):
    file = input("Masukkan nama file yang hendak diselesaikan yang terdapat pada folder test (tidak perlu memasukkan path test/): ")
    puzzle = Puzzle(readFile('test/' + file))

```

```

elif(tipe == 2):
    print()
    print("Puzzle yang akan diselesaikan adalah puzzle ini:")
    puzzle = Puzzle(generatePuzzle()) # Akan membangkitkan puzzle acak
    puzzle.printPuzzle()
    print()
else:
    print("Masukkan nomor yang valid")
    exit()

# Membaca file solusi
puzzle_solution = readFile('src\solution.txt')

# Menghitung nilai KURANG(i)
valueX_temp, i, j = valueX(puzzle.getPuzzle())
print("Posisi KURANG(i) untuk setiap i:")
total_I_below_J = countIBelowJ(puzzle.getPuzzle(), True)
print("Total KURANG(i) + X adalah ", total_I_below_J+ valueX_temp)

# Mengecek apakah puzzle bisa diselesaikan atau tidak
if(total_I_below_J + valueX_temp % 2 == 1):
    print("Tidak bisa diselesaikan")
else:
    puzzle_saver = PriorityQueue() # Menyimpan puzzle yang sudah dikunjungi dan
    bertindak sebagai prioqueue
    puzzle_count_node = [] # Menyimpan jumlah node yang dibangkitkan untuk puzzle
    yang sudah dikunjungi

    puzzle_saver.put((0,puzzle)) # Inisialisasi pertama kali
    puzzle_count_node.append(puzzle) # Inisialisasi pertama kali

    startTime = time.time() # Waktu menghitung dimulai
    while(countDifferent(puzzle_saver.queue[0][1].getPuzzle(), puzzle_solution)
    != 0): # Jika masih terdapat perbedaan, lakukan while loop
        currentPuzzle = puzzle_saver.get()[1] # Menghapus elemen pertama dari
        priqueue

        # Membangkitkan anak dari parent dengan memindahkan kotak kosong ke atas
        if(currentPuzzle.getPosisiEmptyI() != 0 and
        currentPuzzle.getHistoryRute()[-1] != 'down'):
            up = deepcopy(currentPuzzle)
            up.moveUp()
            functionMove(up, 'up', puzzle_saver, puzzle_solution,
            currentPuzzle.historyRute, puzzle_count_node)

```

```

        # Membangkitkan anak dari parent dengan memindahkan kotak kosong ke bawah
        if(currentPuzzle.getPosisiEmptyI() != 3 and
currentPuzzle.getHistoryRute()[-1] != 'up'):
            down = deepcopy(currentPuzzle)
            down.moveDown()
            functionMove(down, 'down', puzzle_saver, puzzle_solution,
currentPuzzle.historyRute, puzzle_count_node)

        # Membangkitkan anak dari parent dengan memindahkan kotak kosong ke kiri
        if(currentPuzzle.getPosisiEmptyJ() != 0 and
currentPuzzle.getHistoryRute()[-1] != 'right'):
            left = deepcopy(currentPuzzle)
            left.moveLeft()
            functionMove(left, 'left', puzzle_saver, puzzle_solution,
currentPuzzle.historyRute, puzzle_count_node)

        # Membangkitkan anak dari parent dengan memindahkan kotak kosong ke kanan
        if(currentPuzzle.getPosisiEmptyJ() != 3 and
currentPuzzle.getHistoryRute()[-1] != 'left'):
            right = deepcopy(currentPuzzle)
            right.moveRight()
            functionMove(right, 'right', puzzle_saver, puzzle_solution,
currentPuzzle.historyRute, puzzle_count_node)

    # Waktu menghitung selesai
    endTime = time.time()
    executionTime = (endTime-startTime)*1.00
    index = 1

    # Melakukan print hasil
    print()
    print("Posisi puzzle 15 awal:")
    puzzle.printPuzzle()
    # Digunakan history rute sebagai penyimpan command yang akan dilakukan oleh
puzzle
    # Hasil dari historyRute pada indeks 0 akan dijamin jika dilakukan secara
berurutan
    # akan menghasilkan puzzle tujuan
    for i in puzzle_saver.queue[0][1].historyRute:
        if(i == 'blank'):
            continue
        print("Langkah ke-", index, " adalah ")
        if(i == 'up'):
            puzzle.moveUp()
            puzzle.printPuzzle()

```

```

elif(i == 'down'):
    puzzle.moveDown()
    puzzle.printPuzzle()
elif(i == 'left'):
    puzzle.moveLeft()
    puzzle.printPuzzle()
elif(i == 'right'):
    puzzle.moveRight()
    puzzle.printPuzzle()
index += 1

print("Banyak simpul yang dibuat adalah " + str(len(puzzle_count_node)) + " # Diambil dari panjang puzzle_count_node")
print("Waktu eksekusi program adalah " + str(round((executionTime),4)) + " sekon")

```

E. Alamat Github

https://github.com/pandora-1/Tucil3_13520057