

**PENYELESAIAN PERSOALAN 15-PUZZLE DENGAN
ALGORITMA *BRANCH AND BOUND***

Laporan Tugas Kecil III

**Disusun sebagai syarat kelulusan mata kuliah
IF2211/Strategi Algoritma**

Oleh

HANA FATHIYAH

NIM : 13520047



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

April 2022

DAFTAR ISI

ALGORITMA BRANCH AND BOUND	3
Definisi Branch and Bound	3
Pemanfaatan Algoritma Branch and Bound dalam 15-Puzzle	3
Implementasi Algoritma Branch and Bound pada 15-Puzzle dalam Tugas Kecil Strategi Algoritma ke-3	4
KODE PROGRAM	7
fifteenpuzzlegame.py	7
startgame.py	7
readfile.py	8
convertfile.py	9
reachablechecker.py	10
utility.py	11
output.py	14
node.py	15
nodepuzzle.py	17
movematrix.py	17
bnb.py	20
SCREENSHOT INPUT DAN OUTPUT PROGRAM	26
Kasus File Tidak Ditemukan	26
unsolvable1.txt	27
unsolvable2.txt	28
solvable1.txt	29
solvable2.txt	30
solvable3.txt	34
TABEL KELENGKAPAN KOMPONEN TUGAS	38
LINK REPOSITORY GITHUB	39

BAB I ALGORITMA *BRANCH AND BOUND*

I.1 Definisi *Branch and Bound*

Dilansir dari, [Algoritma Branch and Bound \(itb.ac.id\)](http://itb.ac.id), algoritma *branch and bound* (B&B) digunakan untuk persoalan optimasi, yaitu meminimalkan atau memaksimalkan suatu fungsi objektif yang tidak melanggar batas (*constraints*) persoalan. Dalam hal ini, dapat dikatakan bahwa algoritma *branch and bound* merupakan gabungan dari BFS dan *least cost search*. Pada B&B, setiap simpul diberi nilai $cost\ \hat{c}(i)$ yang merupakan nilai taksiran lintasan termurah ke simpul status tujuan yang melalui status i . Simpul berikutnya yang akan di-*expand* tidak lagi berdasarkan urutan pembangkitannya, tetapi simpul yang memiliki *cost* yang paling kecil (*least cost search*) pada kasus minimasi. Algoritma B&B menerapkan “pemangkasan” pada jalur yang dianggap tidak mengarah pada solusi dengan kriteria nilai simpul tidak lebih baik dari nilai terbaik, ada batasan yang dilanggar, dan solusi pada simpul hanya satu titik.

I.2 Pemanfaatan Algoritma *Branch and Bound* dalam 15-Puzzle

Dilansir dari [Algoritma Branch and Bound \(itb.ac.id\)](http://itb.ac.id), salah satu implementasi algoritma *branch and bound* adalah penyelesaian permainan 15-puzzle. Dalam permainan ini, terdapat state berdasarkan ubin kosong dan 4 buah aksi: *up*, *down*, *left*, dan *right*. Penggunaan algoritma diawali dengan pengecekan *reachable goal*.

Status tujuan hanya dapat dicapai dari status awal jika $\sum_{i=1}^{16} KURANG(i) + X$ bernilai genap. Nilai X bernilai 0 jika penjumlahan kedua koordinat posisi bernilai genap. Nilai X bernilai 1 jika penjumlahan kedua koordinat posisi bernilai ganjil.

Selanjutnya dilakukan perhitungan *cost* dari simpul hidup. *Cost* untuk setiap simpul umumnya berupa taksiran dengan $\hat{c}(i)$ merepresentasikan ongkos untuk simpul i yang merupakan hasil penjumlahan panjang lintasan dari simpul akar ke i dan taksiran panjang lintasan terpendek dari i ke simpul solusi pada upapohon

yang akarnya i . Taksiran panjang lintasan terpendek direpresentasikan dengan jumlah ubin tidak kosong yang tidak terdapat pada susunan akhir. Selanjutnya dibentuk simpul *expand* dan simpul hidup. Simpul *expand* adalah simpul yang sedang dianalisis (elemen indeks pertama simpul hidup yang sudah dianalisis sebelumnya). Simpul hidup merupakan simpul-simpul yang akan dianalisis kemudian dan diurutkan berdasarkan *cost*.

I.3 Implementasi Algoritma *Branch and Bound* pada 15-Puzzle dalam Tugas Kecil Strategi Algoritma ke-3

Implementasi algoritma *branch and bound* pada permainan 15-puzzle dalam tugas kecil strategi algoritma ke-3 ini dituliskan di dalam *file* bnb.py. Ide pengerjaannya dijabarkan di dalam poin-poin berikut ini.

1. Inisialisasi susunan ubin awal
2. Pengecekan *reachable goal* dengan menghitung fungsi $\sum_{i=1}^{16} KURANG(i) + X$. Fungsi-fungsi terkait terdapat secara spesifik di dalam *file* reachablechecker.py.
3. Apabila solusi tidak dapat dicapai, program akan mengeluarkan pesan kesalahan.
4. Apabila solusi dapat dicapai, program akan menjalankan prosedur bnb yang terdapat pada *file* bnb.py
5. Cara kerja prosedur bnb dapat dideskripsikan sebagai berikut.
 - a. Pembentukan objek *live_node* yang merepresentasikan simpul hidup
 - b. Susunan ubin awal dimasukkan ke dalam *live_node*
 - c. Dilakukan iterasi menggunakan *while* selama kondisi terpenuhi, yaitu simpul hidup tidak kosong.

- d. Pembentukan simpul *expand* yang dilambangkan dengan variabel *current_node* dan merupakan simpul hidup indeks pertama setelah diurutkan berdasarkan *cost*
- e. Penghapusan indeks pertama pada simpul hidup karena simpul tersebut telah digunakan sebagai simpul *expand*
- f. Pengecekan apakah simpul *expand* tersebut adalah solusi (solution = *current_node*). Jika iya, lakukan *bound* dengan menggunakan perintah *break*. Jika tidak, lakukan pemindahan posisi
- g. Pemindahan posisi dilakukan searah jarum jam secara berturut-turut, mulai dari *up* (atas), *right* (kanan), *down* (bawah), dan *left* (kiri).
- h. Pemindahan posisi diawali dengan pengecekan apakah puzzle bisa dipindahkan ke posisi tersebut dan apakah posisi sebelumnya berlawanan dengan posisi tujuan. Berlawanan merupakan parameter untuk mencegah *infinite* loop akibat ubin kembali ke tempat semula setelah berpindah.
- i. Apabila kedua parameter tersebut terpenuhi, dibentuk objek *result_node* yang menyimpan node terbaru setelah pemindahan.
- j. Dilakukan juga penambahan pada *live_node* (simpul hidup) dengan menyimpan simpul di posisi yang tepat. Dalam kelas *LiveNode*, terdapat fungsi *add_in* yang berfungsi untuk menempatkan node sesuai dengan urutannya.
- k. Lakukan iterasi secara terus menerus hingga simpul *expand* tersebut adalah solusi (diperoleh solution = *current_node*). Panggil *break* agar iterasi dapat dihentikan.
- l. Pencarian susunan pohon dilakukan melalui objek *solution* dengan melakukan iterasi sampai atribut *parent* bernilai *None*, yakni sampai ditemukan ubin awal kembali. Susunan tersebut di-*reverse* agar terbentuk susunan dari ubin awal menuju ke solusi.

- m. Setelah itu, tahap demi tahap dari matriks awal menuju matriks solusi untuk mencapai solusi dapat diperoleh melalui `array_result`.

BAB II KODE PROGRAM

Dalam tugas kecil Strategi Algoritma ke-3 ini, program dibuat dengan dibagi menjadi ke dalam beberapa *file* untuk setiap utilitas yang diperlukan. Secara lebih lengkap, *source code* dapat diakses di [hanafathiyah/Tucil3_13520047 \(github.com\)](https://github.com/hanafathiyah/Tucil3_13520047)

II.1 fifteenpuzzlegame.py

```
import startgame;

if __name__ == '__main__':
    startgame.main()
```

II.2 startgame.py

```
import convertfile
import readfile
import output
import os

def main():
    os.system("cls")

    print("Welcome to The 15-Puzzle Game\n")

    fifteen_puzzle_file_name = input("Input the
15-Puzzle's game file: ")

    fifteen_puzzle_in_file =
readfile.read_file(fifteen_puzzle_file_name)

    if(fifteen_puzzle_in_file == []): # empty, file not
found

        exit()
```

```

        fifteen_puzzle =
convertfile.convert_file_to_int_matrix(fifteen_puzzle_in_f
ile)

        output.game_output(fifteen_puzzle)

```

II.3 readfile.py

```

import sys

def read_file(fifteen_puzzle_file_name):

    file_contents = []

    try:

        with open(fifteen_puzzle_file_name, 'r') as file:

            for row in file:

                row_contents = list(row.strip())

                if len(row_contents) > 0:

                    file_contents.append(row_contents)

    except IOError as e:

        print ("I/O error({0}): {1}".format(e.errno,
e.strerror))

    except:

        print ("Unexpected error: ", sys.exc_info()[0])

    return file_contents

```


II.4 convertfile.py

```
def convert_file_to_int_matrix(fifteen_puzzle_in_file):  
    int_matrix = [[0 for _ in range(4)] for _ in range(4)]  
  
    for i in range(4):  
        idx_col = 0  
        j = 0  
  
        while j < len(fifteen_puzzle_in_file[i]) - 1:  
            if(fifteen_puzzle_in_file[i][j] != " " and  
fifteen_puzzle_in_file[i][j+1] != " "): # 2 digits  
                int_matrix[i][idx_col] =  
int(fifteen_puzzle_in_file[i][j]) * 10 +  
int(fifteen_puzzle_in_file[i][j+1])  
                idx_col += 1;  
                j += 3;  
            elif(fifteen_puzzle_in_file[i][j] != " " and  
fifteen_puzzle_in_file[i][j+1] == " "): # 1 digit  
                int_matrix[i][idx_col] =  
int(fifteen_puzzle_in_file[i][j])  
                idx_col += 1;  
                j += 2;  
  
            if(int_matrix[i][3] == 0):  
                int_matrix[i][3] =  
int(fifteen_puzzle_in_file[i][len(fifteen_puzzle_in_file[i]  
)-1])  
  
        return int_matrix
```

II.5 reachablechecker.py

```
import utility

# find x value

def x_value(fifteen_puzzle): # count x_value

    if((utility.get_empty_cell_idx(fifteen_puzzle)[0] +
utility.get_empty_cell_idx(fifteen_puzzle)[1]) % 2 == 0):
# even

        return 0

    else: # odd

        return 1

def less_than(number, fifteen_puzzle): # count
lower-numbered tiles

    cnt = 0

    for i in
range(utility.get_position_of_number(number,fifteen_puzzle
) + 1, 16):

        if utility.matrix_to_list(fifteen_puzzle)[i] <
number:

            cnt += 1

    return cnt

def sum_of_less_than(fifteen_puzzle): # count sum of
lower-numbered tiles

    sum = 0

    for i in range(1, 17):

        sum += less_than(i, fifteen_puzzle)

    return sum
```

```

def sum_of_less_than_plus_x(fifteen_puzzle): # count sum
of lower-numbered tiles + X

    return sum_of_less_than(fifteen_puzzle) +
x_value(fifteen_puzzle)

def is_reachable(fifteen_puzzle): # check reachable

    if (sum_of_less_than_plus_x(fifteen_puzzle) % 2 == 0):

        return True

    else:

        return False

```

II.6 utility.py

```

import bnb

# change matrix of fifteen puzzle into one dimension list
def matrix_to_list(fifteen_puzzle):

    list_puzzle = [0 for i in range(16)]

    idx = 0

    for i in range(4):

        for j in range(4):

            list_puzzle[idx] = fifteen_puzzle[i][j]

            idx += 1

    return list_puzzle

# get position of number in one dimension list
def get_position_of_number(number, fifteen_puzzle):

    list_puzzle = matrix_to_list(fifteen_puzzle)

```

```

    for i in range(16):
        if (list_puzzle[i] == number):
            return i

# get element from coordinate i,j from 2 dimensions list
def get_element_from_position(i,j,fifteen_puzzle):
    return fifteen_puzzle[i][j]

# get element from coordinate i,j from 2 dimensions list
def set_element_in_position(i,j,fifteen_puzzle,el):
    fifteen_puzzle[i][j] = el

# get coordinate i,j of empty cell
def get_empty_cell_idx(fifteen_puzzle):
    for i in range(4):
        for j in range(4):
            if fifteen_puzzle[i][j] == 16:
                return i,j

# make a copy of matrix
def copy_matrix(fifteen_puzzle):
    result = [[0 for _ in range(4)] for _ in range(4)]
    for i in range(4):
        for j in range(4):
            result[i][j] = fifteen_puzzle[i][j]
    return result

```

```

# print matrix, use "*" for representing 16 as an empty
cell

def print_matrix(fifteen_puzzle):
    for i in range(4):
        for j in range(4):
            if(fifteen_puzzle[i][j] == 16):
                print("*", end = " ")
            else:
                print(fifteen_puzzle[i][j], end = " ")
        print("")

# comparing current matrix with goal state
def is_a_result(fifteen_puzzle):
    i = 0
    same = True
    while i < 16 and same:
        if(matrix_to_list(fifteen_puzzle)[i] != i+1):
            same = False
        else:
            i += 1
    return same

# utility for inserting node into live_node
def is_lower_than(node_x, node_y):
    return node_x.depth + bnb.count_g(node_x.info) <=
node_y.depth + bnb.count_g(node_y.info)

```

II.7 output.py

```
import bnb

import reachablechecker

import utility


def print_all_values_of_less_than(fifteen_puzzle):
    print("i"+"      "+"lower-numbered(i)")

    for i in range(1,17):

print(str(i)+"\t"+str(reachablechecker.less_than(i,fifteen
_puzzle)))


def game_output(fifteen_puzzle):
    print("\n> Initial Arrangement:\n")

    utility.print_matrix(fifteen_puzzle) # 1st output in
project specification

    print("")

    print("> The number of lower-numbered tiles that are
to the right of tile i in the arrangement:\n")

    print_all_values_of_less_than(fifteen_puzzle) # 2nd
output in project specification

    print("")

    print("> Sum of all lower-numbered(i) =",
reachablechecker.sum_of_less_than(fifteen_puzzle))

    print("")

    print("> Value of X =",
reachablechecker.x_value(fifteen_puzzle))

    print("")
```

```

        print("> Sum of all lower-numbered(i) after add by X =
",end = "")

print(reachablechecker.sum_of_less_than_plus_x(fifteen_puz
zle),end = " ") # 3rd output in project specification

if(reachablechecker.sum_of_less_than_plus_x(fifteen_puzzle
) % 2 == 0):

    print("(even)")

    else:

        print("(odd) ")

        if (not
reachablechecker.is_reachable(fifteen_puzzle)): # 4th
output in project specification

            print("\n> Result: The goal state is not reachable
from the initial state.\n")

            exit()

        else:

            print("\n> Result: The goal state is reachable
from the initial state.\n")

            print("> Solution:\n") # 5th output in project
specification

            bnb.procedure_bnb(fifteen_puzzle) # call bnb

```

II.8 node.py

```

# LiveNode = simul hidup (in bahasa)

```

```

class LiveNode:

    def __init__(self, function):

        self.liveNode = []

        self.function = function


    def is_empty(self):

        return len(self.liveNode) == 0


    def get_first(self):

        return self.liveNode[0]


    def add_in(self, new_node):

        i = 0

        lower_than = False

        while (not lower_than and i < len(self.liveNode)):

            if(self.function(new_node, self.liveNode[i])):

                lower_than = True

            else:

                i += 1

        self.liveNode.insert(i, new_node)


    def delete_first(self):

        self.liveNode.pop(0)

```


II.9 nodepuzzle.py

```
class NodePuzzle:

    def __init__(self, info, parent = None, depth = 0,
move = ""):

        self.info = info

        self.parent = parent

        self.move = move

        self.depth = depth
```

II.10 movematrix.py

```
import utility

def move_up(fifteen_puzzle):

    fp_move = utility.copy_matrix(fifteen_puzzle)

    origin_i = utility.get_empty_cell_idx(fp_move)[0]
    origin_j = utility.get_empty_cell_idx(fp_move)[1]

    result_i = origin_i - 1 # move up = i - 1
    result_j = origin_j

    tmp =
utility.get_element_from_position(origin_i,origin_j,fp_mov
e)

    # swap

    utility.set_element_in_position(origin_i, origin_j,
fp_move,
utility.get_element_from_position(result_i,result_j,fp_mov
e))

    utility.set_element_in_position(result_i, result_j,
fp_move, tmp)
```

```

        return fp_move

def move_down(fifteen_puzzle):

    fp_move = utility.copy_matrix(fifteen_puzzle)

    origin_i = utility.get_empty_cell_idx(fp_move)[0]
    origin_j = utility.get_empty_cell_idx(fp_move)[1]

    result_i = origin_i + 1 # move down = i + 1
    result_j = origin_j

    tmp =
utility.get_element_from_position(origin_i,origin_j,fp_mov
e)

    # swap
    utility.set_element_in_position(origin_i, origin_j,
fp_move,
utility.get_element_from_position(result_i,result_j,fp_mov
e))

    utility.set_element_in_position(result_i, result_j,
fp_move, tmp)

    return fp_move

def move_left(fifteen_puzzle):

    fp_move = utility.copy_matrix(fifteen_puzzle)

    origin_i = utility.get_empty_cell_idx(fp_move)[0]
    origin_j = utility.get_empty_cell_idx(fp_move)[1]

    result_i = origin_i
    result_j = origin_j - 1 # move left = j - 1

    tmp =
utility.get_element_from_position(origin_i,origin_j,fp mov
e)

```

```

        # swap
        utility.set_element_in_position(origin_i, origin_j,
fp_move,
utility.get_element_from_position(result_i,result_j,fp_mov
e))

        utility.set_element_in_position(result_i, result_j,
fp_move, tmp)

        return fp_move

def move_right(fifteen_puzzle):

    fp_move = utility.copy_matrix(fifteen_puzzle)

    origin_i = utility.get_empty_cell_idx(fp_move)[0]
    origin_j = utility.get_empty_cell_idx(fp_move)[1]
    result_i = origin_i
    result_j = origin_j + 1 # move right = j + 1

    tmp =
utility.get_element_from_position(origin_i,origin_j,fp_mov
e)

    # swap
    utility.set_element_in_position(origin_i, origin_j,
fp_move,
utility.get_element_from_position(result_i,result_j,fp mov
e))

    utility.set_element_in_position(result_i, result_j,
fp_move, tmp)

    return fp_move

def is_enable_to_move_up(fifteen_puzzle):

    return utility.get_empty_cell_idx(fifteen_puzzle)[0] >
0

```

```

def is_enable_to_move_down(fifteen_puzzle):
    return utility.get_empty_cell_idx(fifteen_puzzle)[0] <
3

def is_enable_to_move_right(fifteen_puzzle):
    return utility.get_empty_cell_idx(fifteen_puzzle)[1] <
3

def is_enable_to_move_left(fifteen_puzzle):
    return utility.get_empty_cell_idx(fifteen_puzzle)[0] >
0

```

II.11 bnb.py

```

import time

import utility

import node

import nodepuzzle

import movematrix

def count_g(fifteen_puzzle): # count g(i)

    cnt = 0

    for i in range(16):

        if(utility.matrix_to_list(fifteen_puzzle)[i] !=
i+1 and utility.matrix_to_list(fifteen_puzzle)[i] != 16):

            cnt += 1

    return cnt

def opposite_node(move_direction): # find opposite_node

```

```

    if (move_direction == 'left'):
        return 'right'

    if (move_direction == 'right'):
        return 'left'

    if (move_direction == 'up'):
        return 'down'

    if (move_direction == 'down'):
        return 'up'

def get_solution(solution_found): # get step by step of
solution from goal node

    solution = []

    parent = solution_found.parent
    previous = solution_found

    while(parent != None):
        solution.insert(0,previous) # reverse add
        previous = parent
        parent = parent.parent

    return solution

def procedure_bnb(fifteen_puzzle): # bnb procedure

    # set initial node

    initial_node = nodepuzzle.NodePuzzle(fifteen_puzzle)

```

```

print("0: Initial Arrangement")

utility.print_matrix(fifteen_puzzle)

print()

cnt_node = 1

# create an object with function lower_than
live_node = node.LiveNode(utility.is_lower_than)

# initialization
solution = None

# add initial_node to array live_node
live_node.add_in(initial_node)

# start time
time_begin = time.process_time_ns()

# iteration
while(not live_node.is_empty()):

    # current_node = simpul expand (in bahasa)
    current_node = live_node.get_first()

    # delete first element in array live_node, because
    # it has moved to current_node
    live_node.delete_first()

```

```

        # comparing with goal node

        if (utility.is_a_result(current_node.info)):

            solution = current_node

            break # bound

    # moving

    if

(movematrix.is_enable_to_move_up(current_node.info) and
current_node.move != 'down'):

        result_node =
nodepuzzle.NodePuzzle(movematrix.move_up(current_node.info
),
parent=current_node,depth=current_node.depth+1,move='up')

        cnt_node += 1

        live_node.add_in(result_node)

    if

(movematrix.is_enable_to_move_right(current_node.info) and
current_node.move != 'left'):

        result_node =
nodepuzzle.NodePuzzle(movematrix.move_right(current_node.i
nfo),
parent=current_node,depth=current_node.depth+1,move='right
')

        cnt_node += 1

        live_node.add_in(result_node)

    if

(movematrix.is_enable_to_move_down(current_node.info) and
current_node.move != 'up'):

```

```

        result_node =
nodepuzzle.NodePuzzle(movematrix.move_down(current_node.in
fo),
parent=current_node,depth=current_node.depth+1,move='down'
)

        cnt_node += 1

        live_node.add_in(result_node)

    if
(movematrix.is_enable_to_move_left(current_node.info) and
current_node.move != 'right'):

        result_node =
nodepuzzle.NodePuzzle(movematrix.move_left(current_node.in
fo),
parent=current_node,depth=current_node.depth+1,move='left'
)

        cnt_node += 1

        live_node.add_in(result_node)

    # array_result representing step by step to get
solution from initial node

    array_result = get_solution(solution)

    # end time

    time_end = time.process_time_ns()

    # display all steps

    for i in range(len(array_result)):

        print(str(i+1)+": Move "+"<
"+str(array_result[i].move).title()+" >")

```



```

        utility.print_matrix(array_result[i].info)

    print()

    # display all moves

    print("All moves:", end = " ")

    for i in range(len(array_result)):

        print(str(array_result[i].move).title()[0], end =
" ")

    print()

    print("Total moves:", len(array_result))

    print("Time spent:", (time_end - time_begin)/10000000,
"ms") # 6th output in project specification

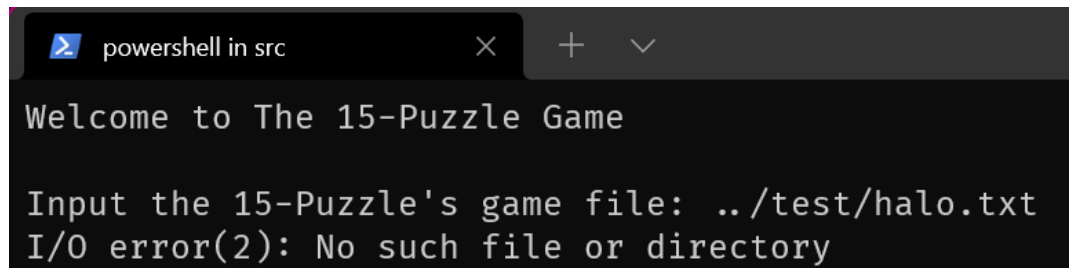
    print("Total nodes:", cnt_node) # 7th output in
project specification

```

BAB III

SCREENSHOT INPUT DAN OUTPUT PROGRAM

III.1 Kasus *File* Tidak Ditemukan



```
powershell in src  
Welcome to The 15-Puzzle Game  
Input the 15-Puzzle's game file: ../test/halo.txt  
I/O error(2): No such file or directory
```

III.2 unsolvable1.txt

```
powershell in src
Welcome to The 15-Puzzle Game

Input the 15-Puzzle's game file: ../test/unsolvable1.txt

> Initial Arrangement:

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

> The number of lower-numbered tiles that are to the right of tile i in the arrangement:

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

> Sum of all lower-numbered(i) = 43

> Value of X = 0

> Sum of all lower-numbered(i) after add by X = 43 (odd)

> Result: The goal state is not reachable from the initial state.
```

III.3 unsolvable2.txt

```
powershell in src × + ∨  
Welcome to The 15-Puzzle Game  
Input the 15-Puzzle's game file: ../test/unsolvable2.txt  
> Initial Arrangement:  
1 2 3 4  
5 6 7 8  
9 10 11 15  
13 14 12 *  
> The number of lower-numbered tiles that are to the right of tile i in the arrangement:  


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

  
> Sum of all lower-numbered(i) = 5  
> Value of X = 0  
> Sum of all lower-numbered(i) after add by X = 5 (odd)  
> Result: The goal state is not reachable from the initial state.
```

III.4 solveable1.txt

```
powershell in src
Welcome to The 15-Puzzle Game

Input the 15-Puzzle's game file: ../test/solvable1.txt

> Initial Arrangement:

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

> The number of lower-numbered tiles that are to the right of tile i in the arrangement:

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

> Sum of all lower-numbered(i) = 15

> Value of X = 1

> Sum of all lower-numbered(i) after add by X = 16 (even)

> Result: The goal state is reachable from the initial state.

> Solution:
```

```
> Solution:

0: Initial Arrangement
1 2 3 4
5 6 * 8
9 10 7 11
13 14 15 12

1: Move < Down >
1 2 3 4
5 6 7 8
9 10 * 11
13 14 15 12

2: Move < Right >
1 2 3 4
5 6 7 8
9 10 11 *
13 14 15 12

3: Move < Down >
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 *

All moves: D R D
Total moves: 3
Time spent: 0.0 ms
Total nodes: 10
```

III.5 solveable2.txt

```
powershell in src
Welcome to The 15-Puzzle Game

Input the 15-Puzzle's game file: ../test/solvable2.txt

> Initial Arrangement:

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

> The number of lower-numbered tiles that are to the right of tile i in the arrangement:

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

> Sum of all lower-numbered(i) = 22

> Value of X = 0

> Sum of all lower-numbered(i) after add by X = 22 (even)

> Result: The goal state is reachable from the initial state.

> Solution:
```

> Solution:

0: Initial Arrangement

1 6 2 3

9 5 8 4

* 10 7 11

13 14 15 12

1: Move < Up >

1 6 2 3

* 5 8 4

9 10 7 11

13 14 15 12

2: Move < Right >

1 6 2 3

5 * 8 4

9 10 7 11

13 14 15 12

3: Move < Up >

1 * 2 3

5 6 8 4

9 10 7 11

13 14 15 12

4: Move < Right >

1 2 * 3

5 6 8 4

9 10 7 11

13 14 15 12

5: Move < Right >

1 2 3 *

5 6 8 4

9 10 7 11

13 14 15 12

6: Move < Down >

1 2 3 4

5 6 8 *

9 10 7 11

13 14 15 12

7: Move < Left >

1 2 3 4

5 6 * 8

9 10 7 11

13 14 15 12

8: Move < Down >

1 2 3 4

5 6 7 8

9 10 * 11

13 14 15 12

9: Move < Right >

1 2 3 4

5 6 7 8

9 10 11 *

13 14 15 12

10: Move < Down >

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 *

All moves: U R U R R D L D R D

Total moves: 10

Time spent: 1.5625 ms

Total nodes: 25

III.6 solvable3.txt

```
powershell in src × + ∨  
Welcome to The 15-Puzzle Game  
Input the 15-Puzzle's game file: ../test/solvable3.txt  
> Initial Arrangement:  
5 1 2 3  
9 6 8 4  
10 7 * 11  
13 14 15 12  
  
> The number of lower-numbered tiles that are to the right of tile i in the arrangement:  


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

  
> Sum of all lower-numbered(i) = 20  
> Value of X = 0  
> Sum of all lower-numbered(i) after add by X = 20 (even)  
> Result: The goal state is reachable from the initial state.  
> Solution:
```

> Solution:

0: Initial Arrangement

5 1 2 3

9 6 8 4

10 7 * 11

13 14 15 12

1: Move < Left >

5 1 2 3

9 6 8 4

10 * 7 11

13 14 15 12

2: Move < Left >

5 1 2 3

9 6 8 4

* 10 7 11

13 14 15 12

3: Move < Up >

5 1 2 3

* 6 8 4

9 10 7 11

13 14 15 12

4: Move < Up >

* 1 2 3

5 6 8 4

9 10 7 11

13 14 15 12

5: Move < Right >

1 * 2 3

5 6 8 4

9 10 7 11

13 14 15 12

```
powershell in src

6: Move < Right >
1 2 * 3
5 6 8 4
9 10 7 11
13 14 15 12

7: Move < Right >
1 2 3 *
5 6 8 4
9 10 7 11
13 14 15 12

8: Move < Down >
1 2 3 4
5 6 8 *
9 10 7 11
13 14 15 12

9: Move < Left >
1 2 3 4
5 6 * 8
9 10 7 11
13 14 15 12

10: Move < Down >
1 2 3 4
5 6 7 8
9 10 * 11
13 14 15 12

11: Move < Right >
1 2 3 4
5 6 7 8
9 10 11 *
13 14 15 12
```

```
12: Move < Down >
```

```
1 2 3 4
```

```
5 6 7 8
```

```
9 10 11 12
```

```
13 14 15 *
```

```
All moves: L L U U R R R D L D R D
```

```
Total moves: 12
```

```
Time spent: 1.5625 ms
```

```
Total nodes: 38
```

TABEL KELENGKAPAN KOMPONEN TUGAS

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

DAFTAR TEST CASE

1. unsolvable1.txt

3 6 2 11

9 8 4 16

1 10 15 12

7 13 5 14

2. unsolvable2.txt

1 2 3 4

5 6 7 8

9 10 11 15

13 14 12 16

3. solvable1.txt

1 2 3 4

5 6 16 8

9 10 7 11

13 14 15 12

4. solvable2.txt

1 6 2 3

9 5 8 4

16 10 7 11

13 14 15 12

5. solvable3.txt

5 1 2 3

9 6 8 4

10 7 16 11

13 14 15 12

LINK REPOSITORY GITHUB

https://github.com/hanafathiyah/Tucil3_13520047