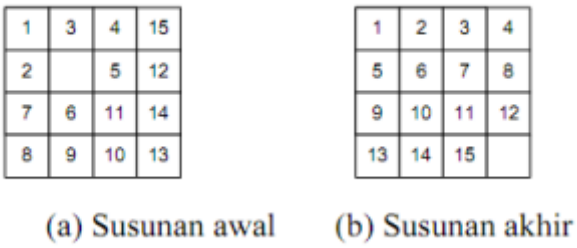
Laporan Tugas Kecil 3 IF2211 Strageti Algoritma
Implementasi Algoritma *Branch and Bounds* untuk menyelesaikan 15 puzzle problem

Muhammad Garebaldhie ER Rahman
K02 / 13520029

A. Branch and Bounds

Branch and Bounds adalah algoritma yang mirip dengan BFS namun ditambah dengan suatu fungsi pembatas agar simpul yang salah tidak akan diperiksa lebih lanjut. Fungsi pembatas yang digunakan dinotasikan c(i) atau cost pada simpul i biasanya didapatkan dari heuristik para ahli di bidangnya sehingga kita sudah mendapatkan fungsi yang dapat meminimalkan serta mengoptimasi penyelesaian.

Algoritma *Branch and Bounds* ini digunakan untuk mengimplementasi pencarian solusi persoalan *15-puzzle* dengan fungsi pembatas berupa julmah ubin yang tidak sesuai dengan posisinya ditambah dengan kedalaman pencarian.



Gambar 1 Susunan puzzle awal dan targer

Kedalaman pencarian perlu dimasukan menjadi parameter dalam penentuan batas karena jika tidak kita tetap akan mendapatkan solusi namun solusi yang didaptkan bukanlah solusi dengan jumlah langkah terpendek.

Setiap konfigurasi puzzle tidak langsung diperiksa namun akan dicek terlebih dahulu melalui fungsi kurang. Fungsi kurang akan mengembalikan jumlah angka yang berada setalah posisi tersebut namun yang kurang dari angka tersebut

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

- KURANG(i) = banyaknya ubin bernomor j sedemikian sehingga $j < i$ dan POSISI(j) > POSISI(i). POSISI(i) = posisi ubin bernomor i pada susunan yang diperiksa.
- KURANG (4) = 1 : terdapat 1 ubin (2)
- Kesimpulan: status tujuan tidak dapat dicapai.

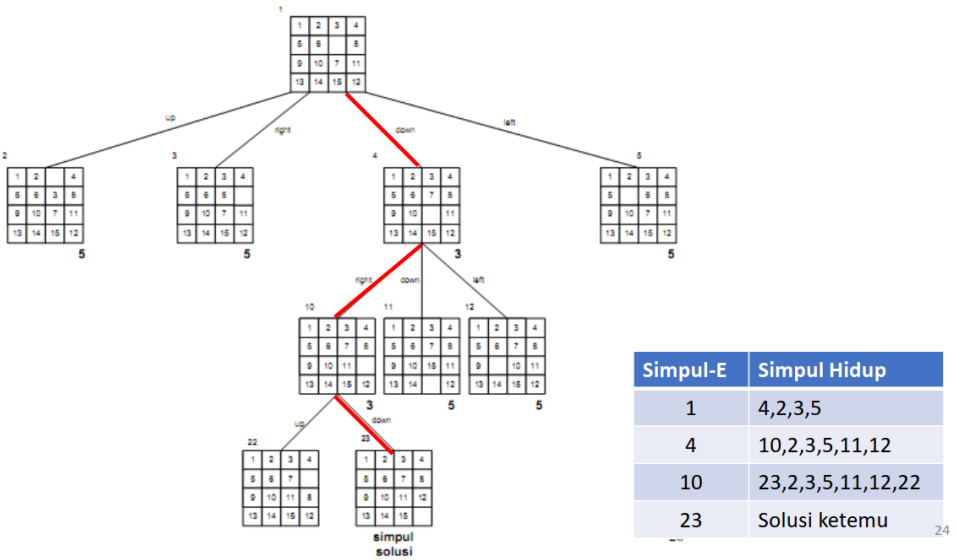
1	3	4	15
2		5	12
7	6	11	14
8	9	10	13

$$\sum_{i=1}^{16} Kurang(i) + X = 37 + 0 = 37$$

Gambar 2 Illustrasi fungsi kurang

Jika fungsi kurang memiliki nilai ganjil maka tidak perlu dilakukan pencarian karena bagaimana pun tidak akan mendapatkan. Heuristik seperti ini yang dibuat oleh para ahli dan digunakan untuk mempermudah pencarian solusi
Setelah cost atau batas dari setiap simpul dihitung kemudian akan dimasukan kedalam priorityQueue untuk diproses. Penggunaan priorityQueue diperlukan untuk melakukan antrian dengan prioritas besar batas/cost dari yang paling kecil hingga paling besar. Pada program ini digunakan sebuah class Node yang menyimpan informasi mengenai state puzzle saat ini mencakup, bentuk puzzle, posisi empty space, nilai cost, nilai kedalaman, serta state dari puzzle tersebut. Class ini juga memiliki fungsi spesial yaitu __lt__ digunakan untuk menentukan simpul mana yang lebih kecil dari yang lain sehingga proses memasukan simpul kedalam antrian untuk di proses.

Progam ini dapat membaca puzzle dari tiga jenis sumber, input file user, file txt serta pembangkitan puzzle secara random. Setelah puzzle terbentuk akan dicari solusi dari puzzle tersebut dan akan dibentuk pohon solusi dari awal hingga mencapai state akhir beserta langkah apa yang harus dilakukan serta nilai cost pada setiap simpulnya



Simpul-E	Simpul Hidup
1	4,2,3,5
4	10,2,3,5,11,12
10	23,2,3,5,11,12,22
23	Solusi ketemu

Gambar 3 Illustrasi ruang pencarian yang dibentuk

B. Source Code Program

Source code terdiri dari tiga folder yaitu main.py, library.py, serta solver.py. Library.py berisi fungsi fungsi apa saja yang akan digunakan untuk proses perhitungan serta pencarian solusi. Pada solver.py berisi algoritma branch and bounds serta bagaimana cara menyelesaikan puzzle. Main.py berisi entry point utama program ini.

main.py

```
import solver

print("Welcome to gare's 15 puzzle solver\n")
while True:
    solver.menu()
    choice = input("> ")
    choice = choice.upper()
    puzzle = None
    if choice == "Q":
        print(
            "\nThank you for using gare's 16-puzzle solver")
        exit(0)
    elif choice == "1":
        puzzle = solver.readPuzzleInput()
    elif choice == "2":
        puzzle = solver.createPuzzle()
    elif choice == "3":
        dir = solver.printDir()
        puzzle = solver.readPuzzle(dir)
    else:
        print("Please input correct option")
        continue
```

```
    solver.solve(puzzle)
```

Library.py

```
import numpy as np
import os
from random import shuffle
from queue import PriorityQueue
from time import sleep, time

direction = ["UP", "RIGHT", "DOWN", "LEFT"]
target = np.reshape([[1, 2, 3, 4], [5, 6, 7, 8], [
                    9, 10, 11, 12], [13, 14, 15, 0]], (4, 4))


class Node:  # Node class
    def __init__(self, parent, puzzle, zeroIdx, cost, level, state):
        self.parent = parent
        self.puzzle = puzzle
        self.zeroIdx = zeroIdx
        self.level = level
        self.cost = cost
        self.state = state

    def __lt__(self, next):
        return self.cost + self.level <= next.cost + next.level


# create next puzzle based on move
def createNextPuzzle(puzzle, zeroIdx, newZeroIdx, costBefore):
    newPuzzle = np.copy(puzzle)
    row1, col1 = zeroIdx
    row2, col2 = newZeroIdx
    before, after = False, False
    if newPuzzle[row2, col2] == target[row2, col2]:
        before = True
    newPuzzle[row1, col1], newPuzzle[row2,
                                     col2] = newPuzzle[row2, col2],
newPuzzle[row1, col1]
    if newPuzzle[row1, col1] == target[row1, col1]:
        after = True

    if before == False and after == True:
        costBefore -= 1

    if before == True and after == False:
        costBefore += 1
```

```
    return newPuzzle, costBefore


def printSolution(root, solution):  # print all possible solution
    if root == None:
        return
    printSolution(root.parent, solution)
    print(

f"================================\n{root.puzzle}\n========================
=========\nMOVE: {root.state}")
    solution.append(root.state)


def zeroIdx(puzzle):  # find zero index
    idx = 0
    for rows in range(4):
        for cols in range(4):
            if puzzle[rows, cols] == 0:
                return (rows, cols)

    return -1


def isValidMove(zeroIdx):  # to check if move is valid
    row, col = zeroIdx
    return (0 <= row < 4) and (0 <= col < 4)


def UP(puzzle, zeroIdx):  # move up
    row, col = zeroIdx
    return (row - 1, col)


def RIGHT(puzzle, zeroIdx):  # move right
    row, col = zeroIdx
    return (row, col + 1)


def DOWN(puzzle, zeroIdx):  # move down
    row, col = zeroIdx
    return (row + 1, col)


def LEFT(puzzle, zeroIdx):  # move left
    row, col = zeroIdx
    return (row, col - 1)


def countCost(puzzle):  # count cost with matching tile
```

```
    cost = 0
    for row in range(4):
        for col in range(4):
            if puzzle[row, col] != target[row, col] and puzzle[row, col]
!= 0:
                cost += 1
    return cost


def kurangNumber(puzzle, rowIdx, colIdx):  # calculate less number in the
puzle
    kurang = 0
    size = len(puzzle)
    target = puzzle[rowIdx, colIdx]
    if target == 0:
        target = 16
    while rowIdx < size:
        while colIdx < size:
            if target > puzzle[rowIdx, colIdx] and puzzle[rowIdx, colIdx]
!= 0:
                kurang += 1
            colIdx += 1
        colIdx = 0
        rowIdx += 1
    return kurang


def kurangFunction(puzzle):  # calculate all less number in the puzzle
    sum = 0
    for row in range(4):
        for col in range(4):
            if puzzle[row, col] == 0 and ((row % 2 == 0 and col % 2 == 1)
or (row % 2 == 1 and col % 2 == 0)):
                sum += 1
            sum += kurangNumber(puzzle, row, col)
    return sum


def cetakKurang(puzzle):  # print all kurang number in the puzzle
    for row in range(4):
        for col in range(4):
            num = kurangNumber(puzzle, row, col)
            if puzzle[row, col] == 0:
                print(f"16\t: {num}")
            else:
                print(f"{puzzle[row, col]}\t: {num}")


def isSolvable(kurangNumber):  # check if solvable or not using kurang
formula
```

```
    return kurangNumber % 2 == 0


def createPuzzle():  # create random puzzle
    puzzle = np.arange(0, 16)
    shuffle(puzzle)
    puzzle = np.reshape(puzzle, (4, 4))
    return puzzle


def printDir():  # print list directory on test folder
    os.chdir("../")
    listdir = os.listdir(f"{os.getcwd()}//test")
    os.chdir(".//src")
    print("Select puzzle file you want to solve")
    for i in range(len(listdir)):
        print(f"{i+1}. {listdir[i]}")
    choice = int(input("> "))
    while(choice < 1 or choice > len(listdir)):
        print("Please select correct number")
        choice = int(input("> "))
    return listdir[choice - 1]


def readPuzzle(fileName):  # read puzzle from txt file
    puzzle = []
    os.chdir("../")
    path = os.getcwd()
    path += f"//test//{fileName}"
    os.chdir(".//src")
    try:
        with open(path) as f:
            lines = f.readlines()
            for line in lines:
                puzzle.append(list(map(int, line.split())))
        return np.reshape(puzzle, (4, 4))
    except:
        print("Wrong file format. Program exiting...")
        exit(0)


def readPuzzleInput():
    print("Input unique number between 1 - 15 (put 0 for empty space)")
    print("The number you choose is 4 per line ex: 1 2 3 4 (seperated by
whitespace)")
    puz = []
    correct = 0
    number = set()
    while correct < 4:
        foundDuplicate = False
```

```
        line = input("> ")
        toAdd = list(map(int, line.split()))

        # check len
        if len(toAdd) != 4:
            print("Please input correct format")
            continue

        # check if there are duplicate number
        for num in toAdd:
            if num in number:
                print(f"{num} is duplicate number")
                foundDuplicate = True
                break

        if foundDuplicate:
            continue
        # insert to set
        for num in toAdd:
            number.add(num)

        if len(toAdd) == 4:
            puz.append(toAdd)
            correct += 1

    print("\n")
    return np.reshape(puz, (4, 4))


def menu():
    print("What do you want to do?")
    print("[1] Read puzzle from input user")
    print("[2] Generate random puzzle")
    print("[3] Read Puzzle from file")
    print("[Q] Exit The Program")
```

Solver.py

```
from library import *


def solve(puzzle):  # main function
    print("================================\nStart state of
puzzle\n================================")
    print(puzzle)
    print("================================")
    print("Value of kurang i\n================================")
```

```
    cetakKurang(puzzle)
    kurang = kurangFunction(puzzle)
    print("================================")
    print("Value of kurang function: ", kurang)
    if isSolvable(kurang):
        BranchAndBounds(puzzle)
    else:
        print("The puzzle cannot be solved from this first state\n")


def BranchAndBounds(puzzle):
    solutionPath = []
    nodeCount = 0
    start = time()

    # hashmap of visited state
    visited = {}
    # create priority queue
    pq = PriorityQueue()

    # create root
    root = Node(None, puzzle, zeroIdx(puzzle), countCost(puzzle), 0,
"IDLE")
    # enqueue root
    pq.put(root)
    # do bfs
    visited[root.puzzle.tobytes()] = True
    while(not pq.empty()):
        node = pq.get()
        if node.cost == 0:
            print("================================")
            print("The path solutions are")
            printSolution(node, solutionPath)
            print("================================")
            print(f"Time execution: {time() - start}s")
            print(f"Generated node count: {nodeCount}")
            print(f"Move needed({len(solutionPath)}): {solutionPath}")
            print("================================")

            return
        # generate node
        for dir in direction:
            zeroIndex = 0
            if(dir == "UP"):
                zeroIndex = UP(node.puzzle, node.zeroIdx)
            elif(dir == "RIGHT"):
                zeroIndex = RIGHT(node.puzzle, node.zeroIdx)
            elif(dir == "LEFT"):
                zeroIndex = LEFT(node.puzzle, node.zeroIdx)
            else:
```

```
                zeroIndex = DOWN(node.puzzle, node.zeroIdx)
            if isValidMove(zeroIndex):
                newPuzzle, costAfter = createNextPuzzle(
                    node.puzzle, node.zeroIdx, zeroIndex, node.cost)
                newPuzzleBytes = newPuzzle.tobytes()
                if newPuzzleBytes not in visited:
                    child = Node(node, newPuzzle, zeroIndex,
                                 costAfter, node.level + 1, dir)
                    visited[newPuzzleBytes] = True
                    nodeCount += 1
                    pq.put(child)
```

C. Screenshot IO Program

a. Correct1.txt

Input

```
1 2 3 4
5 6 0 8
9 10 7 11
13 14 15 12
```

Output

```
================================
Start state of puzzle
================================
[[ 1  2  3  4]
 [ 5  6  0  8]
 [ 9 10  7 11]
 [13 14 15 12]]
================================
Value of kurang i
================================
1       : 0
2       : 0
3       : 0
4       : 0
5       : 0
6       : 0
16      : 9
8       : 1
9       : 1
10      : 1
7       : 0
11      : 0
13      : 1
14      : 1
15      : 1
12      : 0
================================
Value of kurang function:  16
```

```
================================
The path solutions are
================================
[[ 1  2  3  4]
 [ 5  6  0  8]
 [ 9 10  7 11]
 [13 14 15 12]]
================================
MOVE: IDLE
================================
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10  0 11]
 [13 14 15 12]]
================================
MOVE: DOWN
================================
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11  0]
 [13 14 15 12]]
================================
MOVE: RIGHT
================================
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]
 [13 14 15  0]]
================================
MOVE: DOWN
================================
Time execution: 0.009139299392700195s
Generated node count: 9
Move needed(4): ['IDLE', 'DOWN', 'RIGHT', 'DOWN']
================================
```

b. Correct2.txt

Input

```
1 2 3 4
5 6 8 11
9 10 7 12
13 0 14 15
```

Output

```
================================
Start state of puzzle
================================
[[ 1  2  3  4]
 [ 5  6  8 11]
```

```
 [ 9 10  7 12]
 [13  0 14 15]]
================================
Value of kurang i
================================
1       : 0
2       : 0
3       : 0
4       : 0
5       : 0
6       : 0
8       : 1
11      : 3
9       : 1
10      : 1
7       : 0
12      : 0
13      : 0
16      : 2
14      : 0
15      : 0
================================
Value of kurang function:  8
================================
The path solutions are
================================
[[ 1  2  3  4]
 [ 5  6  8 11]
 [ 9 10  7 12]
 [13  0 14 15]]
================================
MOVE: IDLE
================================
[[ 1  2  3  4]
 [ 5  6  8 11]
 [ 9 10  7 12]
 [13 14  0 15]]
================================
MOVE: RIGHT
================================
[[ 1  2  3  4]
 [ 5  6  8 11]
 [ 9 10  7 12]
 [13 14 15  0]]
================================
MOVE: RIGHT
================================
[[ 1  2  3  4]
 [ 5  6  8 11]
 [ 9 10  7  0]
 [13 14 15 12]]
================================
MOVE: UP
================================
```

```
[[ 1  2  3  4]
 [ 5  6  8  0]
 [ 9 10  7 11]
 [13 14 15 12]]
================================
MOVE: UP
================================
[[ 1  2  3  4]
 [ 5  6  0  8]
 [ 9 10  7 11]
 [13 14 15 12]]
================================
MOVE: LEFT
================================
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10  0 11]
 [13 14 15 12]]
================================
MOVE: DOWN
================================
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11  0]
 [13 14 15 12]]
================================
MOVE: RIGHT
================================
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]
 [13 14 15  0]]
================================
MOVE: DOWN
================================
Time execution: 0.0169985294342041s
Generated node count: 37
Move needed(9): ['IDLE', 'RIGHT', 'RIGHT', 'UP', 'UP', 'LEFT', 'DOWN',
'RIGHT', 'DOWN']
================================
```

c. Correct3.txt

Input

```
0 1 2 3
5 6 7 8
4 9 10 11
13 14 15 12
```

Output

```
================================
Start state of puzzle
================================
[[ 0  1  2  3]
 [ 5  6  7  8]
 [ 4  9 10 11]
 [13 14 15 12]]
================================
Value of kurang i
================================
16      : 15
1       : 0
2       : 0
3       : 0
5       : 1
6       : 1
7       : 1
8       : 1
4       : 0
9       : 0
10      : 0
11      : 0
13      : 1
14      : 1
15      : 1
12      : 0
================================
Value of kurang function:  22
================================
The path solutions are
================================
[[ 0  1  2  3]
 [ 5  6  7  8]
 [ 4  9 10 11]
 [13 14 15 12]]
================================
MOVE: IDLE
================================
[[ 5  1  2  3]
 [ 0  6  7  8]
 [ 4  9 10 11]
 [13 14 15 12]]
================================
MOVE: DOWN
================================
[[ 5  1  2  3]
 [ 4  6  7  8]
 [ 0  9 10 11]
 [13 14 15 12]]
================================
MOVE: DOWN
================================
[[ 5  1  2  3]
 [ 4  6  7  8]
```

```
 [ 9  0 10 11]
 [13 14 15 12]]
================================
MOVE: RIGHT
================================
[[ 5  1  2  3]
 [ 4  0  7  8]
 [ 9  6 10 11]
 [13 14 15 12]]
================================
MOVE: UP
================================
[[ 5  1  2  3]
 [ 0  4  7  8]
 [ 9  6 10 11]
 [13 14 15 12]]
================================
MOVE: LEFT
================================
[[ 0  1  2  3]
 [ 5  4  7  8]
 [ 9  6 10 11]
 [13 14 15 12]]
================================
MOVE: UP
================================
[[ 1  0  2  3]
 [ 5  4  7  8]
 [ 9  6 10 11]
 [13 14 15 12]]
================================
MOVE: RIGHT
================================
[[ 1  2  0  3]
 [ 5  4  7  8]
 [ 9  6 10 11]
 [13 14 15 12]]
================================
MOVE: RIGHT
================================
[[ 1  2  7  3]
 [ 5  4  0  8]
 [ 9  6 10 11]
 [13 14 15 12]]
================================
MOVE: DOWN
================================
[[ 1  2  7  3]
 [ 5  0  4  8]
 [ 9  6 10 11]
 [13 14 15 12]]
================================
MOVE: LEFT
================================
```

```
[[ 1  2  7  3]
 [ 5  6  4  8]
 [ 9  0 10 11]
 [13 14 15 12]]
================================
MOVE: DOWN
================================
[[ 1  2  7  3]
 [ 5  6  4  8]
 [ 9 10  0 11]
 [13 14 15 12]]
================================
MOVE: RIGHT
================================
[[ 1  2  7  3]
 [ 5  6  4  8]
 [ 9 10 11  0]
 [13 14 15 12]]
================================
MOVE: RIGHT
================================
[[ 1  2  7  3]
 [ 5  6  4  0]
 [ 9 10 11  8]
 [13 14 15 12]]
================================
MOVE: UP
================================
[[ 1  2  7  3]
 [ 5  6  0  4]
 [ 9 10 11  8]
 [13 14 15 12]]
================================
MOVE: LEFT
================================
[[ 1  2  0  3]
 [ 5  6  7  4]
 [ 9 10 11  8]
 [13 14 15 12]]
================================
MOVE: UP
================================
[[ 1  2  3  0]
 [ 5  6  7  4]
 [ 9 10 11  8]
 [13 14 15 12]]
================================
MOVE: RIGHT
================================
[[ 1  2  3  4]
 [ 5  6  7  0]
 [ 9 10 11  8]
 [13 14 15 12]]
================================
```

```
MOVE: DOWN
================================
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11  0]
 [13 14 15 12]]
================================
MOVE: DOWN
================================
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]
 [13 14 15  0]]
================================
MOVE: DOWN
================================
Time execution: 0.15086054801940918s
Generated node count: 5962
Move needed(21): ['IDLE', 'DOWN', 'DOWN', 'RIGHT', 'UP',
'LEFT', 'UP', 'RIGHT', 'RIGHT', 'DOWN', 'LEFT', 'DOWN', 'RIGHT', 'RIGHT',
'UP', 'LEFT', 'UP', 'RIGHT', 'DOWN', 'DOWN', 'DOWN']
================================
```

d. Correct4.txt

Input

```
2 3 4 0
1 5 8 11
9 6 10 12
13 14 7 15
```

Output

```
================================
Start state of puzzle
================================
[[ 2  3  4  0]
 [ 1  5  8 11]
 [ 9  6 10 12]
 [13 14  7 15]]
================================
Value of kurang i
================================
2       : 1
3       : 1
4       : 1
16      : 12
1       : 0
5       : 0
8       : 2
11      : 4
```

```
9       : 2
6       : 0
10      : 1
12      : 1
13      : 1
14      : 1
7       : 0
15      : 0
================================
Value of kurang function:  28
================================
The path solutions are
================================
[[ 2  3  4  0]
 [ 1  5  8 11]
 [ 9  6 10 12]
 [13 14  7 15]]
================================
MOVE: IDLE
================================
[[ 2  3  0  4]
 [ 1  5  8 11]
 [ 9  6 10 12]
 [13 14  7 15]]
================================
MOVE: LEFT
================================
[[ 2  0  3  4]
 [ 1  5  8 11]
 [ 9  6 10 12]
 [13 14  7 15]]
================================
MOVE: LEFT
================================
[[ 0  2  3  4]
 [ 1  5  8 11]
 [ 9  6 10 12]
 [13 14  7 15]]
================================
MOVE: LEFT
================================
[[ 1  2  3  4]
 [ 0  5  8 11]
 [ 9  6 10 12]
 [13 14  7 15]]
================================
MOVE: DOWN
================================
[[ 1  2  3  4]
 [ 5  0  8 11]
 [ 9  6 10 12]
 [13 14  7 15]]
================================
MOVE: RIGHT
```

```
==============================
[[ 1  2  3  4]
 [ 5  6  8 11]
 [ 9  0 10 12]
 [13 14  7 15]]
==============================
MOVE: DOWN
==============================
[[ 1  2  3  4]
 [ 5  6  8 11]
 [ 9 10  0 12]
 [13 14  7 15]]
==============================
MOVE: RIGHT
==============================
[[ 1  2  3  4]
 [ 5  6  8 11]
 [ 9 10  7 12]
 [13 14  0 15]]
==============================
MOVE: DOWN
==============================
[[ 1  2  3  4]
 [ 5  6  8 11]
 [ 9 10  7 12]
 [13 14 15  0]]
==============================
MOVE: RIGHT
==============================
[[ 1  2  3  4]
 [ 5  6  8 11]
 [ 9 10  7  0]
 [13 14 15 12]]
==============================
MOVE: UP
==============================
[[ 1  2  3  4]
 [ 5  6  8  0]
 [ 9 10  7 11]
 [13 14 15 12]]
==============================
MOVE: UP
==============================
[[ 1  2  3  4]
 [ 5  6  0  8]
 [ 9 10  7 11]
 [13 14 15 12]]
==============================
MOVE: LEFT
==============================
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10  0 11]
 [13 14 15 12]]
```

```
================================
MOVE: DOWN
================================
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11  0]
 [13 14 15 12]]
================================
MOVE: RIGHT
================================
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]
 [13 14 15  0]]
================================
MOVE: DOWN
================================
Time execution: 0.03756833076477051s
Generated node count: 253
Move needed(16): ['IDLE', 'LEFT', 'LEFT', 'LEFT', 'DOWN', 'RIGHT',
'DOWN', 'RIGHT', 'DOWN', 'RIGHT', 'UP', 'UP', 'LEFT', 'DOWN', 'RIGHT',
'DOWN']
================================
```

e. Correct5.txt

Input

```
1 2 3 4
5 7 10 8
11 9 6 0
13 14 15 12
```

Output

```
================================
Start state of puzzle
================================
[[ 1  2  3  4]
 [ 5  7 10  8]
 [11  9  6  0]
 [13 14 15 12]]
================================
Value of kurang i
================================
1       : 0
2       : 0
3       : 0
4       : 0
5       : 0
7       : 1
10      : 3
```

```
8       : 1
11      : 2
9       : 1
6       : 0
16      : 4
13      : 1
14      : 1
15      : 1
12      : 0
================================
Value of kurang function:  16
================================
The path solutions are
================================
[[ 1  2  3  4]
 [ 5  7 10  8]
 [11  9  6  0]
 [13 14 15 12]]
================================
MOVE: IDLE
================================
[[ 1  2  3  4]
 [ 5  7 10  8]
 [11  9  0  6]
 [13 14 15 12]]
================================
MOVE: LEFT
================================
[[ 1  2  3  4]
 [ 5  7 10  8]
 [11  0  9  6]
 [13 14 15 12]]
================================
MOVE: LEFT
================================
[[ 1  2  3  4]
 [ 5  7 10  8]
 [ 0 11  9  6]
 [13 14 15 12]]
================================
MOVE: LEFT
================================
[[ 1  2  3  4]
 [ 5  7 10  8]
 [13 11  9  6]
 [ 0 14 15 12]]
================================
MOVE: DOWN
================================
[[ 1  2  3  4]
 [ 5  7 10  8]
 [13 11  9  6]
 [14  0 15 12]]
================================
```

```
MOVE: RIGHT
================================
[[ 1  2  3  4]
 [ 5  7 10  8]
 [13  0  9  6]
 [14 11 15 12]]
================================
MOVE: UP
================================
[[ 1  2  3  4]
 [ 5  7 10  8]
 [13  9  0  6]
 [14 11 15 12]]
================================
MOVE: RIGHT
================================
[[ 1  2  3  4]
 [ 5  7 10  8]
 [13  9  6  0]
 [14 11 15 12]]
================================
MOVE: RIGHT
================================
[[ 1  2  3  4]
 [ 5  7 10  8]
 [13  9  6 12]
 [14 11 15  0]]
================================
MOVE: DOWN
================================
[[ 1  2  3  4]
 [ 5  7 10  8]
 [13  9  6 12]
 [14 11  0 15]]
================================
MOVE: LEFT
================================
[[ 1  2  3  4]
 [ 5  7 10  8]
 [13  9  6 12]
 [14  0 11 15]]
================================
MOVE: LEFT
================================
[[ 1  2  3  4]
 [ 5  7 10  8]
 [13  9  6 12]
 [ 0 14 11 15]]
================================
MOVE: LEFT
================================
[[ 1  2  3  4]
 [ 5  7 10  8]
 [ 0  9  6 12]
```

```
 [13 14 11 15]]
==============================
MOVE: UP
==============================
[[ 1  2  3  4]
 [ 5  7 10  8]
 [ 9  0  6 12]
 [13 14 11 15]]
==============================
MOVE: RIGHT
==============================
[[ 1  2  3  4]
 [ 5  7 10  8]
 [ 9  6  0 12]
 [13 14 11 15]]
==============================
MOVE: RIGHT
==============================
[[ 1  2  3  4]
 [ 5  7  0  8]
 [ 9  6 10 12]
 [13 14 11 15]]
==============================
MOVE: UP
==============================
[[ 1  2  3  4]
 [ 5  0  7  8]
 [ 9  6 10 12]
 [13 14 11 15]]
==============================
MOVE: LEFT
==============================
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9  0 10 12]
 [13 14 11 15]]
==============================
MOVE: DOWN
==============================
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10  0 12]
 [13 14 11 15]]
==============================
MOVE: RIGHT
==============================
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]
 [13 14  0 15]]
==============================
MOVE: DOWN
==============================
[[ 1  2  3  4]
```

```
 [ 5  6  7  8]
 [ 9 10 11 12]
 [13 14 15  0]]
================================
MOVE: RIGHT
================================
Time execution: 0.8100292682647705s
Generated node count: 39107
Move needed(22): ['IDLE', 'LEFT', 'LEFT', 'LEFT', 'DOWN', 'RIGHT', 'UP',
'RIGHT', 'RIGHT', 'DOWN', 'LEFT', 'LEFT', 'LEFT', 'UP', 'RIGHT', 'RIGHT',
'UP', 'LEFT', 'DOWN', 'RIGHT', 'DOWN', 'RIGHT']
================================
```

f. false1.txt

Input

```
1 3 4 15
2 0 5 12
7 6 11 14
8 9 10 13
```

Output

```
================================
Start state of puzzle
================================
[[ 1  3  4 15]
 [ 2  0  5 12]
 [ 7  6 11 14]
 [ 8  9 10 13]]
================================
Value of kurang i
================================
1       : 0
3       : 1
4       : 1
15      : 11
2       : 0
16      : 10
5       : 0
12      : 6
7       : 1
6       : 0
11      : 3
14      : 4
8       : 0
9       : 0
10      : 0
13      : 0
================================
Value of kurang function:  37
```

```
The puzzle cannot be solved from this first state
```

g. false2.txt

Input

```
11 10  8  5
3  9  1 13
14 12 15  7
2  6  4  0
```

Output

```
================================
Start state of puzzle
================================
[[11 10  8  5]
 [ 3  9  1 13]
 [14 12 15  7]
 [ 2  6  4  0]]
================================
Value of kurang i
================================
11      : 10
10      : 9
8       : 7
5       : 4
3       : 2
9       : 5
1       : 0
13      : 5
14      : 5
12      : 4
15      : 4
7       : 3
2       : 0
6       : 1
4       : 0
16      : 0
================================
Value of kurang function:  59
The puzzle cannot be solved from this first state
```

D. Extra

Poin	Ya	Tidak
Program berhasil dikompilasi	✔	
Program berhasil running	✔	

Program dapat menerima input dan menuliskan output	✔	
Luaran sudah benar untuk semua data uji	✔	
Bonus dibuat		✔

Alamat Github:
https://github.com/lloveNooodles/15-puzzle-solver.git