LAPORAN TUGAS KECIL III

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

Laporan dibuat untuk memenuhi salah satu tugas mata kuliah

IF2211 Strategi Algoritma

Disusun oleh:

Fayza Nadia 13520001

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2022

DAFTAR ISI

Algoritma Branch and Bound

Program penyelesaian persoalan 15-*puzzle* merupakan salah satu permasalahan yang dapat diselesaikan dengan algoritma *branch and bound*. Algoritma *branch and bound* merupakan algoritma yang bekerja dengan meminimalkan atau memaksimalkan suatu fungsi objektif dan mematuhi *constraints* persoalan yang diberikan pula.

Dalam menyelesaikan 15-*puzzle*, pertama-tama program akan melakukan perhitungan terhadap rumus

$$\sum_{i=1}^{16} less(i) \; + \; x$$

sesuai dengan teorema. Jika didapatkan hasil berupa angka genap, *puzzle* dinyatakan dapat diselesaikan. Namun, jika didapatkan hasil berupa angka ganjil, *puzzle* dinyatakan tidak dapat diselesaikan. *Puzzle* yang dapat diselesaikan kemudian akan dimasukkan sebagai *root node*. Kemudian, program melakukan *expand node* yang dapat dicapai jika *tile* kosong digeser ke atas, bawah, kiri, ataupun kanan. *Node* yang telah di-*expand* tersebut kemudian akan dimasukkan ke dalam *priority queue* yang telah tersedia.

Algoritma *branch and bound* mengenal *cost* yang digunakan sebagai taksiran dalam menentukan letak *final node*. Perhitungan *cost* dalam permasalahan 15-*puzzle* dilakukan dengan rumus

$$c(i) \; = \; f(i) \; + \; g(i)$$

di mana *f(i)* menyatakan *depth* dari *root node* menuju *node* i dan *g(i)* menyatakan jumlah *tile* tidak kosong yang terletak tidak sesuai dengan posisi seharusnya. *Cost* tersebut menentukan prioritas dari *node* yang akan di-*expand* terlebih dahulu dari *priority queue* yang ada.

Jika sebuah *node* memiliki *cost* yang lebih besar dari *cost final goal puzzle*, *node* tersebut kemudian akan dimatikan atau *pruning*. Selanjutnya, dengan algoritma ini pula, program dapat melakukan *backtracking* terhadap *node* yang telah dikunjungi sebelumnya untuk mengunjungi *child node* lain yang belum di-*expand*.

Program akan berhenti jika ditemukan *node* yang sama dengan *final goal puzzle* yang diinginkan. Setelah itu, program akan mencetak masing-masing bentuk *puzzle* pada setiap *node* dengan melakukan *backtracking* hingga *root node* secara rekursif. Setelah *puzzle* dicetak seluruhnya, program akan mencetak waktu yang dibutuhkan dari algoritma mulai hingga selesai. Selain waktu, program juga mencetak jumlah seluruh *node* yang dibangkitkan dalam pohon ruang status pencarian.

Source Program

Program dibuat menggunakan Bahasa Python dan didekomposisi dalam empat bagian besar.

a. Bagian *Main*

   Bagian ini merupakan program utama yang meminta *user* untuk memilih *puzzle* yang ingin diselesaikan.

```python
# Main Program
# Below are listed the functions that are used to run the program.

from solver import *
from puzzle import *
from input import *

# Title
print()
print("8 888888888o   8 8888      88 8888888888',8888' 8888888888',8888' 8 8888         8 8888888888                      ")
print("8 8888    `88. 8 8888      88        ,8',8888'         ,8',8888'  8 8888         8 8888                            ")
print("8 8888     `88 8 8888      88       ,8',8888'         ,8',8888'   8 8888         8 8888                            ")
print("8 8888     ,88 8 8888      88      ,8',8888'         ,8',8888'    8 8888         8 8888                            ")
print("8 8888.   ,88' 8 8888      88     ,8',8888'         ,8',8888'     8 8888         8 888888888888                    ")
print("8 888888888P'  8 8888      88    ,8',8888'         ,8',8888'      8 8888         8 8888                            ")
print("8 8888         8 8888      88   ,8',8888'         ,8',8888'       8 8888         8 8888                            ")
print("8 8888         ` 8888     ,8P  ,8',8888'         ,8',8888'        8 8888         8 8888                            ")
print("8 8888           8888   ,d8P  ,8',8888'         ,8',8888'         8 8888         8 8888                            ")
print("8 8888            `Y88888P'  ,8',8888888888888 ,8',8888888888888  8 888888888888 8 888888888888                    ")
print("                                                                                                                   ")
print("                       d888888o.       ,o888888o.     8 8888 `8.`888b           ,8' 8 8888888888   8 888888888o.   ")
print("                     .`8888:' `88.  . 8888     `88.   8 8888  `8.`888b         ,8'  8 8888         8 8888    `88.  ")
print("                     8.`8888.   Y8 ,8 8888       `8b  8 8888   `8.`888b       ,8'   8 8888         8 8888     `88  ")
print("                     `8.`8888.     88 8888        `8b 8 8888    `8.`888b     ,8'    8 8888         8 8888     ,88  ")
print("                      `8.`8888.    88 8888         88 8 8888     `8.`888b   ,8'     8 888888888888 8 8888.   ,88'  ")
print("                       `8.`8888.   88 8888         88 8 8888      `8.`888b ,8'      8 8888         8 888888888P'   ")
print("                        `8.`8888.  88 8888        ,8P 8 8888       `8.`888b8'       8 8888         8 8888`8b       ")
print("                    8b   `8.`8888. `8 8888       ,8P  8 8888        `8.`888'        8 8888         8 8888 `8b.     ")
print("                    `8b.  ;8.`8888  ` 8888     ,88'   8 8888         `8.`8'         8 8888         8 8888   `8b.   ")
print("                     `Y8888P ,88P'     `8888888P'     8 888888888888  `8.`          8 888888888888 8 8888      `88.")
```

```python
# Menu
print()
print()
print("                        * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *  * *")
print("                        *                                                                *")
print("                        *                      Choose your puzzle:                       *")
print("                        *                      1. Randomize                              *")
print("                        *                      2. Import puzzle                          *")
print("                        *                                                                *")
print("                        * * * * * * * * * * * * * * * * * * * * * * * * * * * *  * * * * *")
print()
puzzleOption = int(input(">> "))
```

```python
# Generate Puzzle
if ((puzzleOption == 1) or (puzzleOption == 2)):
    if (puzzleOption == 1):
        print()
        print("                        * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *  * *")
        print("                        *                         RANDOMIZE PUZZLE                         *")
        print("                        * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *  * *")
        print()
        mat = randomizer()
    elif (puzzleOption == 2):
        print()
        print("                        * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *  * *")
        print("                        *                          IMPORT PUZZLE                           *")
        print("                        * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *  * *")
        print()
        print("Input file name!")
        filename = input(">> ")
        mat = readFile(filename)
    print("                                            INITIAL PUZZLE:")
    printPuzzle(mat)
    print()
    arrLess = lessArray(mat)
    printLess(arrLess)
    print()
    arrLess = np.sum(lessArray(mat))
    x = valueX(mat)
    total = arrLess + x
    print("                                          less(i) + x =", total)
    if (isReachable(mat)):
        print("                                      This puzzle is solveable!")
        print()
        print("                                               SOLUTION")
        solve(mat)

    else:
        print("                                    This puzzle is not solveable!")
else:
    print("Invalid input! Bye~")
```

b. Bagian *Input*

Bagian ini berisi fungsi-fungsi yang digunakan dalam inisiasi *puzzle*. Inisiasi *puzzle* dapat dilakukan baik dengan melakukan *import file* ataupun dengan menggunakan *randomizer*.

```python
# Input
# Below are listed the functions that are used to read the input from the user.

import sys
import numpy as np
import random

from solver import *

def toInteger(arr):
    # Convert the string array to integer array
    for i in range(len(arr)):
        arr[i] = int(arr[i])

    return arr
```

```python
def readFile(filename):
    # Read the file and return the matrix
    file = open(sys.path[0] + '/../test/' + filename, "r")
    numbers = file.read().split()
    file.close()
    numbers = np.array(toInteger(numbers))

    return toMatrix(numbers)

def randomizer():
    # Generate a random 4x4 matrix puzzle
    arr = random.sample(range(1, 17), 16)
    arr = np.array(arr)

    return toMatrix(arr)
```

c. Bagian *Puzzle*

Bagian ini merupakan bagian yang berisi *user-defined class* queue dengan *node* yang berisi atribut-atribut diperlukan dalam penyelesaian *puzzle*. Pada bagian ini juga terdapat pemanggilan fungsi utama yang diperlukan untuk menyelesaikan *puzzle*.

```python
# Puzzle
# Below are listed the functions used for the data structure.

import time

from heapq import heappush, heappop
from solver import *
from copy import deepcopy

# DATA STRUCTURE
class queue:
    # A queue is a data structure that stores a list of elements in a particular order.
    def __init__(self):
        # Initialize the queue
        self.heap = []

    def isEmpty(self):
        # Check if the queue is empty
        if not (self.heap):
            return True
        else:
            return False

    def push(self, node):
        # Push the node to the queue
        heappush(self.heap, node)

    def pop(self):
        # Pop the node from the queue
        # Popped node is the node with the lowest value
        return heappop(self.heap)
```

```python
class node:
    # A node is a data structure that stores the information of a node in the tree.
    def __init__(self, mat, parent, point, depth, cost, direction):
        # Initialize the node
        self.mat = mat
        self.parent = parent
        self.point = point
        self.depth = depth
        self.cost = cost
        self.direction = direction

    def __lt__(self, other):
        # Override the less than operator
        return (self.depth + self.cost < other.depth + other.cost)
```

```python
# PUZZLE
def printPuzzle(mat):
    # Print the puzzle
    print("                                          +----+----+----+----+")
    for i in range(4):
        print("                                              |", end="")
        for j in range(4):
            if (mat[i, j] < 10):
                print("", mat[i, j], " |", end="")
            elif (mat[i, j] == 16):
                print("    |", end="")
            else:
                print("", mat[i, j], "|", end="")
        print()
        print("                                              +----+----+----+----+")

def printPath(root):
    # Print the path from the root to the leaf
    if (root == None):
        # Base case
        return
    else:
        # Recursive case
        printPath(root.parent)
        if (root.depth != 0):
            print("                                                    STEP", root.depth, ":", root.direction)
            printPuzzle(root.mat)
        print()

def isExist(matrices, mat):
    # Check if the matrix is already in the list
    for i in range(len(matrices)):
        if (np.array_equal(matrices[i], mat)):
            return True
    return False
```

```
def solve(mat):
    # Solve the puzzle
    timeStart = time.time()
    nodeCount = 0
    matrices = []
    q = queue()
    point = idxPoint(mat, 16)
    root = node(mat, None, point, 0, countWrongTile(mat), None)
    q.push(root)
    matrices.append(mat)
    nodeCount += 1
    while not (q.isEmpty()):
        current = q.pop()
        if (current.cost == 0):
            timeEnd = time.time()
            timeElapsed = timeEnd - timeStart
            printPath(current)
            print("                                                Elapsed Time: " + "%.6f" % timeElapsed + " s")
            print("                                                  Node Count:", nodeCount, "nodes")
            return
```

```
        else:
            if (isMoveable(current.mat, 'UP') and current.direction != 'DOWN'):
                newPoint = deepcopy(current.point)
                newPoint[0] = current.point[0] - 1
                matChild = swap(current.mat, current.point, newPoint)
                if not (isExist(matrices, matChild)):
                    nodeCount += 1
                    child = node(matChild, current, newPoint, current.depth + 1, countWrongTile(matChild), 'UP')
                    q.push(child)
                    matrices.append(matChild)
            if (isMoveable(current.mat, 'DOWN') and current.direction != 'UP'):
                newPoint = deepcopy(current.point)
                newPoint[0] = current.point[0] + 1
                matChild = swap(current.mat, current.point, newPoint)
                if not (isExist(matrices, matChild)):
                    nodeCount += 1
                    child = node(matChild, current, newPoint, current.depth + 1, countWrongTile(matChild), 'DOWN')
                    q.push(child)
                    matrices.append(matChild)
            if (isMoveable(current.mat, 'LEFT') and current.direction != 'RIGHT'):
                newPoint = deepcopy(current.point)
                newPoint[1] = current.point[1] - 1
                matChild = swap(current.mat, current.point, newPoint)
                if not (isExist(matrices, matChild)):
                    nodeCount += 1
                    child = node(matChild, current, newPoint, current.depth + 1, countWrongTile(matChild), 'LEFT')
                    q.push(child)
                    matrices.append(matChild)
            if (isMoveable(current.mat, 'RIGHT') and current.direction != 'LEFT'):
                newPoint = deepcopy(current.point)
                newPoint[1] = current.point[1] + 1
                matChild = swap(current.mat, current.point, newPoint)
                if not (isExist(matrices, matChild)):
                    nodeCount += 1
                    child = node(matChild, current, newPoint, current.depth + 1, countWrongTile(matChild), 'RIGHT')
                    q.push(child)
                    matrices.append(matChild)
```

d. Bagian *Solver*

Bagian ini merupakan bagian yang berisi fungsi-fungsi yang digunakan dalam manipulasi matriks *puzzle* serta perhitungan rumus.

```python
# Solver
# Below are listed the functions that are used to solve the puzzle.

import numpy as np

from copy import deepcopy

def toArray(mat):
    # Convert the matrix to array
    arr = np.zeros(16, np.int8)
    idx = 0
    for i in range(4):
        for j in range(4):
            arr[idx] = mat[i, j]
            idx += 1

    return arr

def toMatrix(arr):
    # Convert the array to matrix
    mat = arr.reshape(4,4)

    return mat
```

```python
def lessArray(mat):
    # Count the misplaced smaller value tiles and return the values in an array
    arrLess = np.zeros(16, np.int8)
    arr = toArray(mat)
    for i in range(arr.size):
        if (i != arr.size):
            for j in range(i + 1, arr.size):
                if (arr[j] < arr[i]):
                    arrLess[arr[i] - 1] += 1

    return arrLess

def printLess(arrLess):
    # Print the array of less(i) array
    print("                                                +----+---------+")
    print("                                                | i  | less(i) |")
    print("                                                +----+---------+")
    for i in range(arrLess.size):
        if (i < 9):
            print("                                                        |", i + 1, " |", end="")
        else:
            print("                                                        |", i + 1, "|", end="")
        if (arrLess[i] < 10):
            print("   ", arrLess[i], "    |")
        else:
            print("  ", arrLess[i], "    |")
    print("                                                +----+---------+")
```

```python
def idxPoint(mat, val):
    # Return the index of the value in the matrix
    idx = np.where(mat == val)
    idx = np.array(list(zip(idx[0], idx[1])))

    return idx[0]

def valueX(mat):
    # Return the value of X used in counting the reachability of the puzzle based on the initial location of the blank tile
    valueMatrix = np.matrix([[0, 1, 0, 1], [1, 0, 1, 0], [0, 1, 0, 1], [1, 0, 1, 0]])
    idx = idxPoint(mat, 16)

    return valueMatrix[idx[0], idx[1]]

def isReachable(mat):
    # Check if the puzzle is reachable
    # Return true if sum of less(i) + X is even
    # Return false if sum of less(i) + X is odd
    arrLess = np.sum(lessArray(mat))
    x = valueX(mat)
    total = arrLess + x
    if (total % 2 == 0):
        return True
    else:
        return False
```

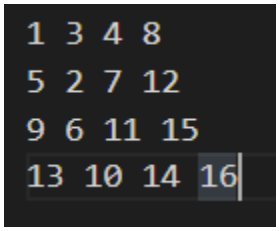
```python
def countWrongTile(mat):
    # Count the number of misplaced tiles
    matGoal = np.matrix([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16]])
    count = 0
    for i in range(4):
        for j in range(4):
            if ((mat[i, j] != matGoal[i, j]) and (mat[i, j] != 16)):
                count += 1

    return count

def inRange(idx):
    # Check if the index is in puzzle range
    if (idx[0] >= 0 and idx[0] < 4 and idx[1] >= 0 and idx[1] < 4):
        return True
    else:
        return False
```

```python
def isMoveable(mat, direction):
    # Check if the move is valid
    idx = idxPoint(mat, 16)
    if (direction == 'UP'):
        idx[0] -= 1
    elif (direction == 'DOWN'):
        idx[0] += 1
    elif (direction == 'LEFT'):
        idx[1] -= 1
    elif (direction == 'RIGHT'):
        idx[1] += 1
    if (inRange(idx)):
        return True
    else:
        return False

def swap(mat, idx1, idx2):
    # Swap the two tiles
    matCopy = deepcopy(mat)
    temp = matCopy[idx1[0], idx1[1]]
    matCopy[idx1[0], idx1[1]] = matCopy[idx2[0], idx2[1]]
    matCopy[idx2[0], idx2[1]] = temp

    return matCopy
```

Screenshot Input dan Output

a. Test Case Puzzle Berhasil 1

Input



Gambar 1
Puzzle Berhasil 1

Output

INITIAL PUZZLE:

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

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

less(i) + x = 18
This puzzle is solveable!

STEP 1 : UP

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

STEP 2 : UP

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

STEP 3 : UP

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

STEP 4 : LEFT

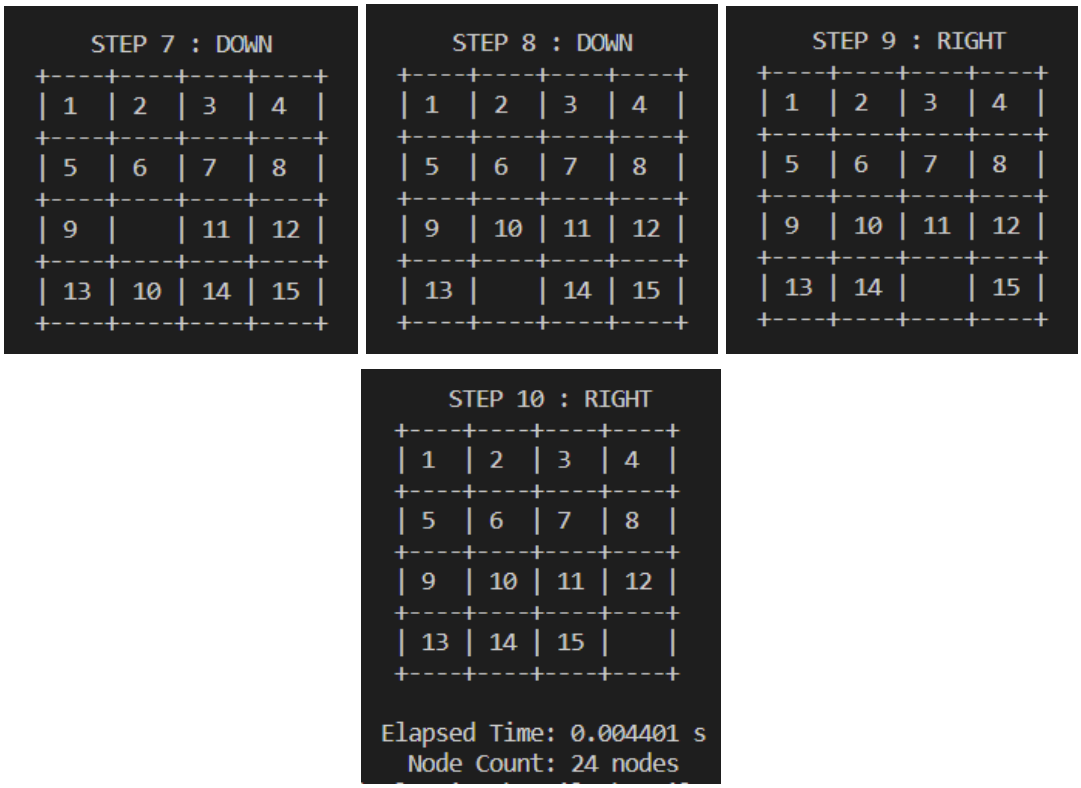
1	3		4
5	2	7	8
9	6	11	12
13	10	14	15

STEP 5 : LEFT

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

STEP 6 : DOWN

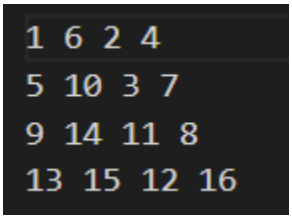
1	2	3	4
5		7	8
9	6	11	12
13	10	14	15



Gambar 2
Output Puzzle Berhasil 1

b. Test Case Puzzle Berhasil 2

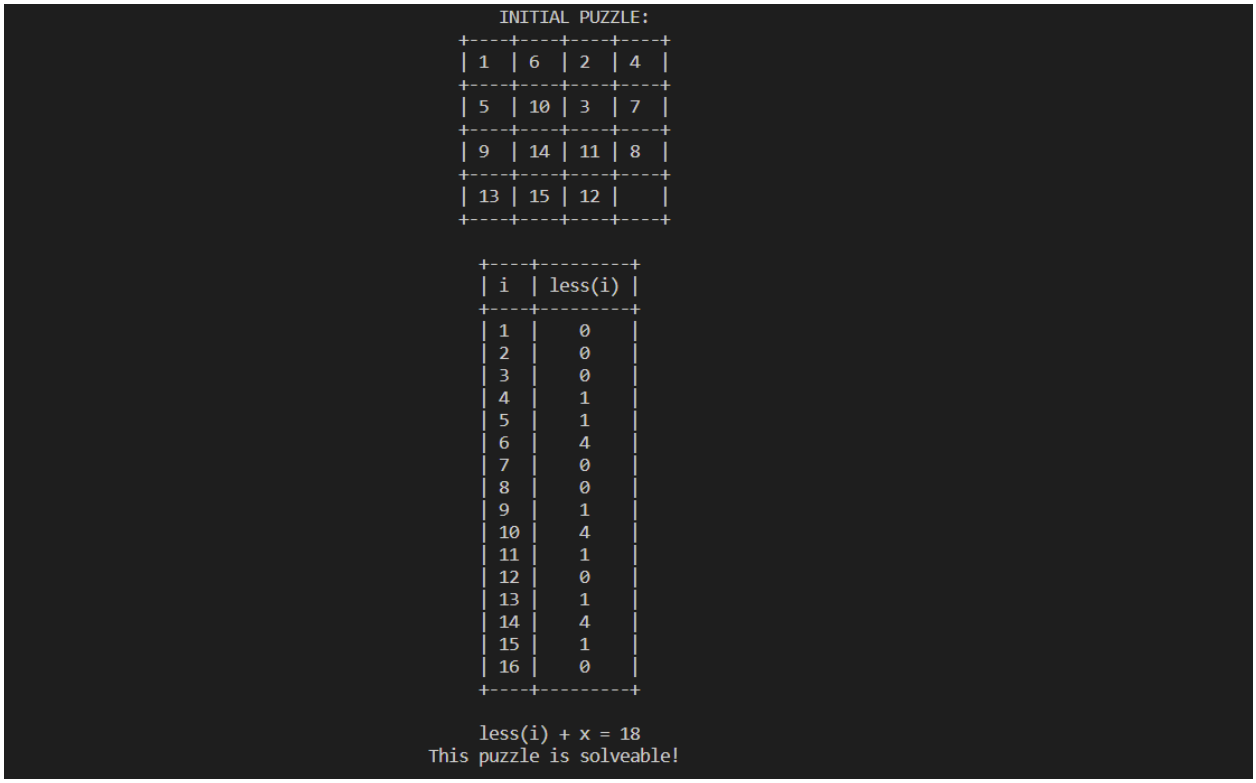
Input

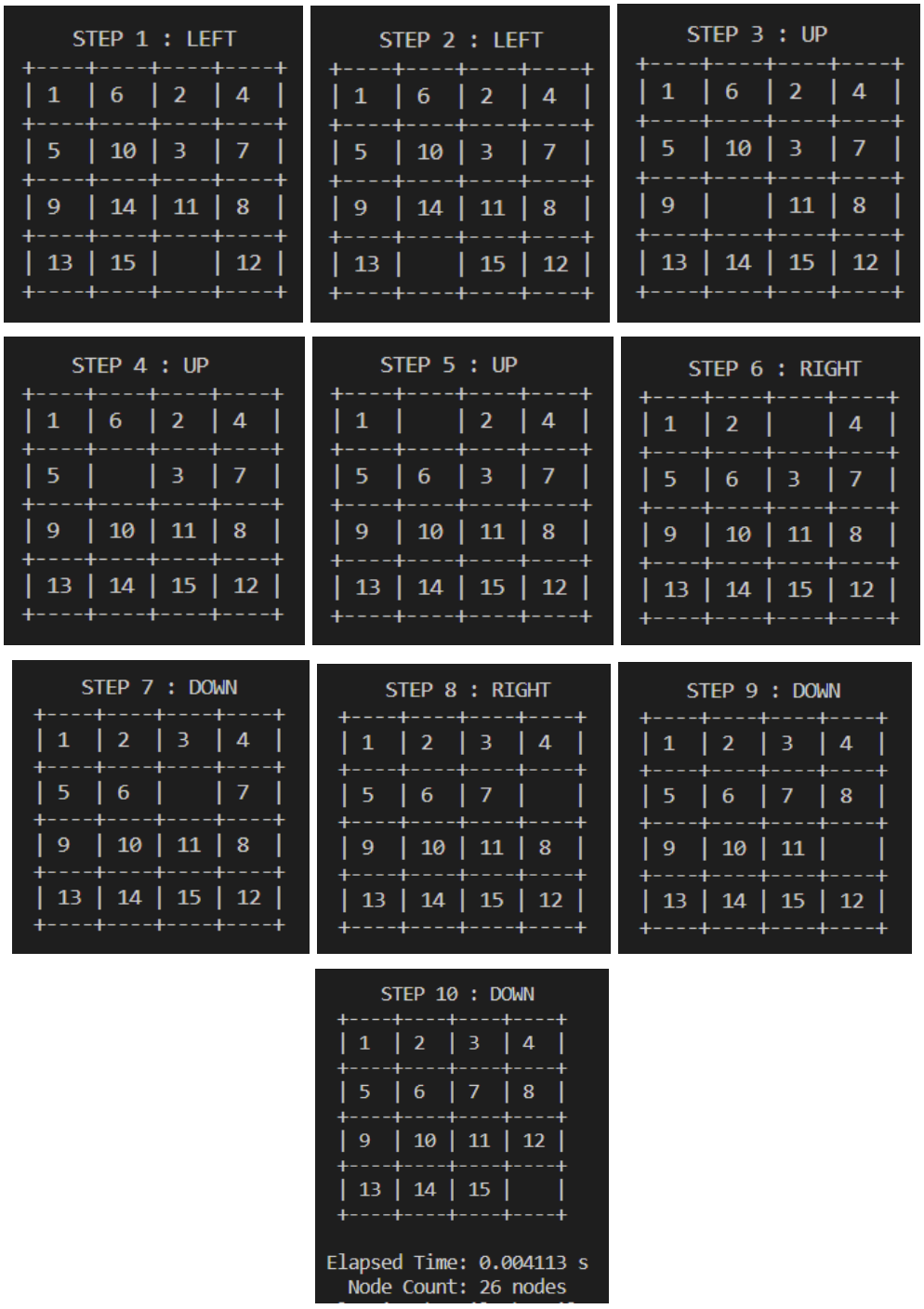


Gambar 3
Puzzle Berhasil 2

Output


8 888888888o 8 8888 88 8888888888',8888' 8888888888',8888' 8 8888 8 8888888888
8 8888 `88. 8 8888 88 ,8',8888' ,8',8888' 8 8888 8 8888
8 8888 `88 8 8888 88 ,8',8888' ,8',8888' 8 8888 8 8888
8 8888 ,88 8 8888 88 ,8',8888' ,8',8888' 8 8888 8 8888
8 8888. ,88' 8 8888 88 ,8',8888' ,8',8888' 8 8888 8 888888888888
8 888888888P' 8 8888 88 ,8',8888' ,8',8888' 8 8888 8 8888
8 8888 8 8888 88 ,8',8888' ,8',8888' 8 8888 8 8888
8 8888 ` 8888 ,8P ,8',8888' ,8',8888' 8 8888 8 8888
8 8888 8888 ,d8P ,8',8888' ,8',8888' 8 8888 8 8888
8 8888 `Y88888P' ,8',88888888888 ,8',88888888888 8 88888888888 8 88888888888
d8888888o. ,o8888888o. 8 8888 `8.`888b ,8' 8 8888888888 8 8888888888o.
.`8888:' `88. . 8888 `88. 8 8888 `8.`888b ,8' 8 8888 8 8888 `88.
8.`8888. Y8 ,8 8888 `8b 8 8888 `8.`888b ,8' 8 8888 8 8888 `88
`8.`8888. 88 8888 `8b 8 8888 `8.`888b ,8' 8 8888 8 8888 ,88
`8.`8888. 88 8888 88 8 8888 `8.`888b ,8' 8 888888888888 8 8888. ,88'
`8.`8888. 88 8888 88 8 8888 `8.`888b ,8' 8 8888 8 888888888P'
`8.`8888. 88 8888 ,8P 8 8888 `8.`888b8' 8 8888 8 8888`8b
8b `8.`8888. `8 8888 ,8P 8 8888 `8.`888' 8 8888 8 8888 `8b.
`8b. ;8.`8888 ` 8888 ,88' 8 8888 `8.`8' 8 8888 8 8888 `8b.
`Y8888P ,88P' `8888888P' 8 888888888888 `8.` 8 888888888888 8 8888 `88.
* *
*
*
Choose your puzzle:
1. Randomize
2. Import puzzle
*
*
* *
>> 2
* *
*
IMPORT PUZZLE
*
* *
Input file name!
>> success2.txt

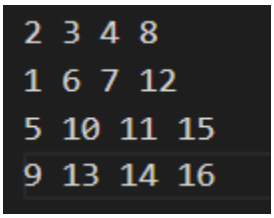

INITIAL PUZZLE:
1	6	2	4
5	10	3	7
9	14	11	8
13	15	12	
i	less(i)
1	0
2	0
3	0
4	1
5	1
6	4
7	0
8	0
9	1
10	4
11	1
12	0
13	1
14	4
15	1
16	0
less(i) + x = 18
This puzzle is solveable!



Gambar 4
Output Puzzle Berhasil 2

c. Test Case Puzzle Berhasil 3

Input



Gambar 5
Puzzle Berhasil 3

Output

INITIAL PUZZLE:

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

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

less(i) + x = 18
This puzzle is solveable!

STEP 1 : UP

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

STEP 2 : UP

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

STEP 3 : UP

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

STEP 4 : LEFT

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

STEP 5 : LEFT

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

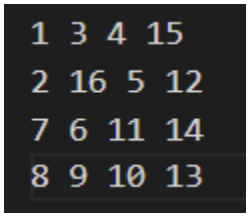
STEP 6 : LEFT

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

Gambar 6
Output Puzzle Berhasil 3

d. Test Case Puzzle Gagal 1

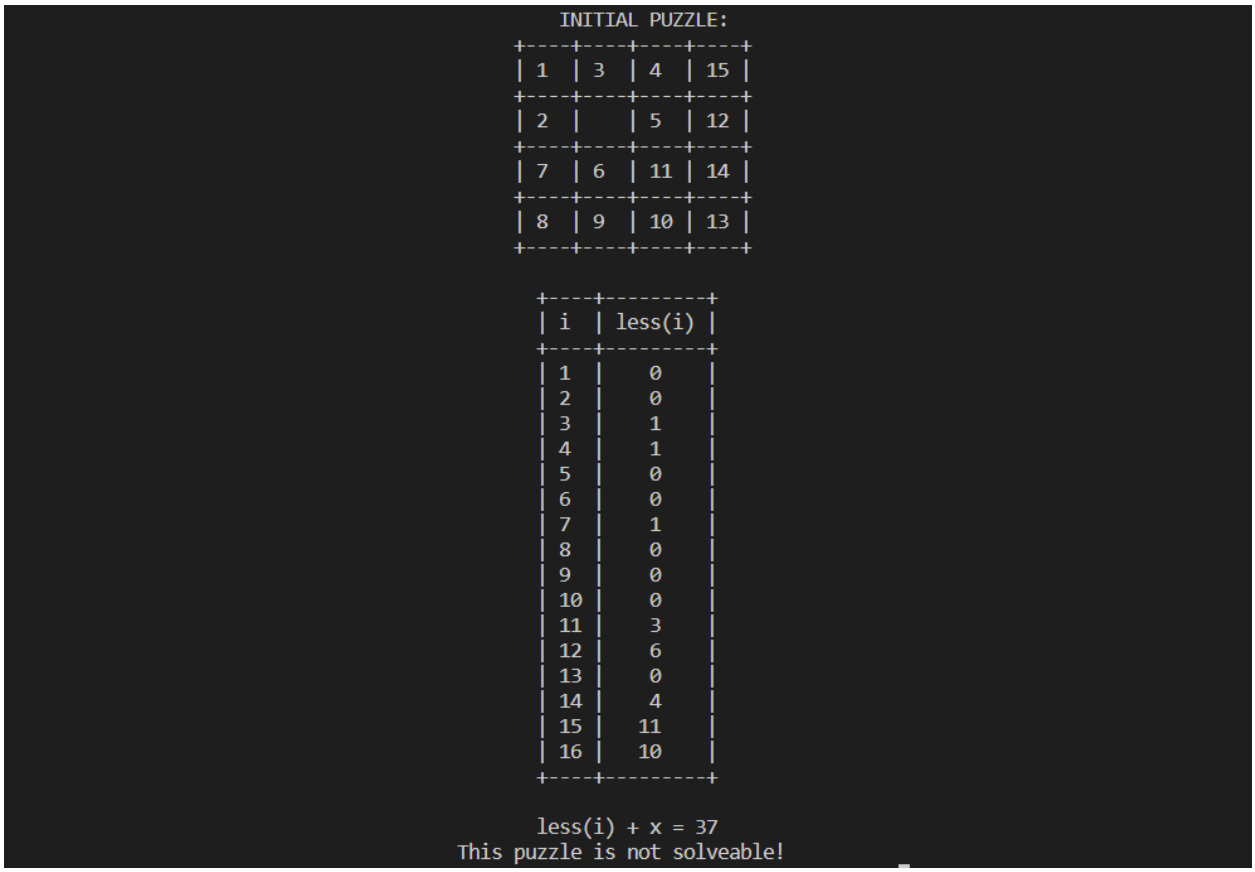
Input



Gambar 7
Puzzle Gagal 1

Output

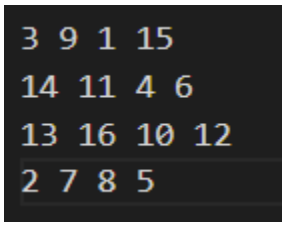

Choose your puzzle:
1. Randomize
2. Import puzzle
>> 2
IMPORT PUZZLE
Input file name!
>> failed1.txt



Gambar 8
Output Puzzle Gagal 1

e. Test Case Puzzle Gagal 2

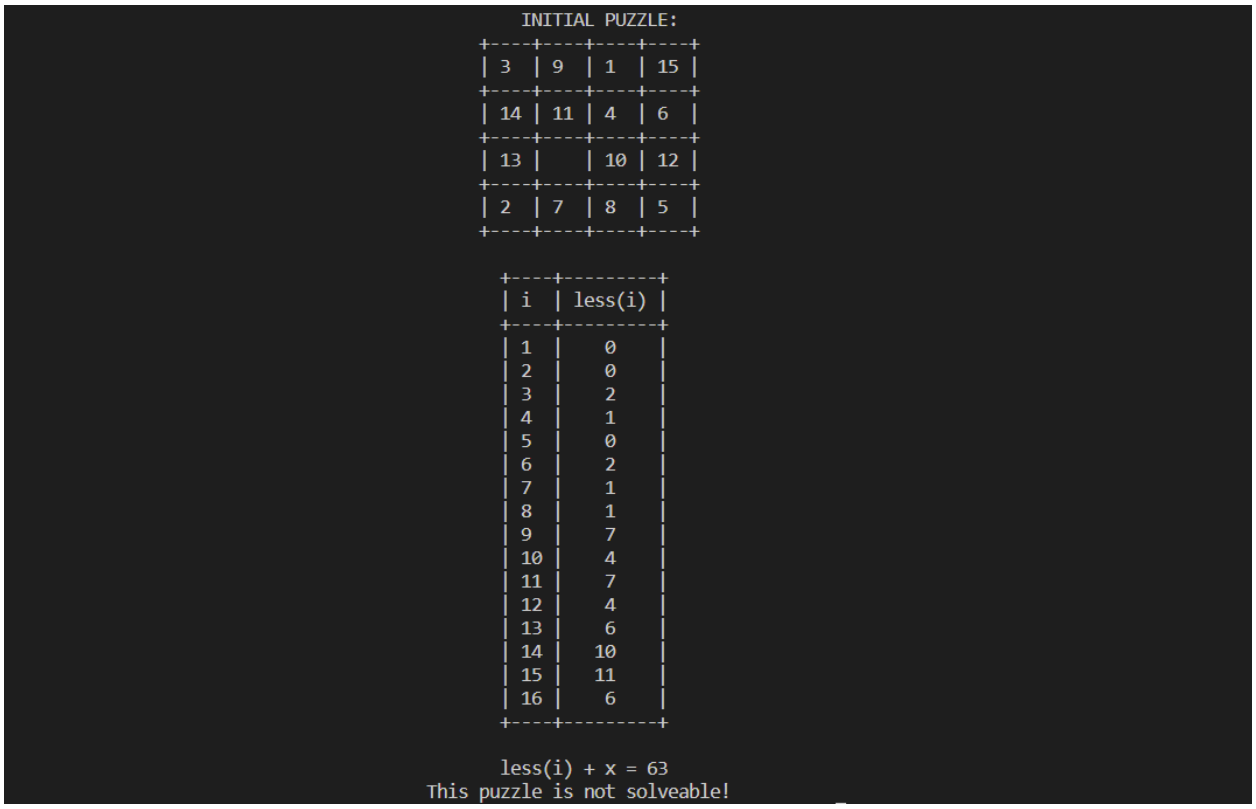
Input



Gambar 9
Puzzle Gagal 2

Output


Choose your puzzle:
1. Randomize
2. Import puzzle
>> 2
IMPORT PUZZLE
Input file name!
>> failed2.txt



Gambar 10
Output Puzzle Gagal 2

Link Kode Program

https://github.com/fayzanadia/Tucil3_13520001

Checklist

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		✔

Daftar Referensi

https://informatika.stei.itb.ac.id/~rinaldi.munir/

https://numpy.org/doc/