Tugas Kecil 3 - Strategi Algoritma

Penyelesaian Persoalan 15-Puzzle dengan Algoritma Branch And Bound

Fikri Khoiron Fadhila

13520056

Institut Teknologi Bandung

2022

Daftar Isi

Cara Kerja Program Branch and Bound yang Dibuat

Diberikan suatu masukan matriks yang merepresentasikan posisi awal suatu instansiasi persoalan 15-puzzle sebagai berikut.

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

Akan dilakukan pengecekan apakah puzzle tersebut dapat diselesaikan atau tidak. Untuk menentukannya dapat dilakukan perhitungan sesuai dengan teorema berikut

Status tujuan hanya dapat dicapai dari status awal jika

$$\sum_{i=1}^{16} KURANG(i) + X$$

bernilai Genap.

KURANG(i) adalah banyaknya ubin bernomor j sedemikian sehingga j < i dan POSISI(j) > POSISI(i).
POSISI(i) = posisi ubin bernomor i pada susunan yang diperiksa.
Nilai X adalah nilai dari kolom dan baris dari posisi ubin kosong yang didefiniskan sebagai ubin bernomor 16 kemudian dimodulo dengan 2 (X = (BARIS(16) + KOLOM(16) mod 2).

Berdasarkan masukkan diatas, akan didapatkan data sebagai berikut:

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

Sehingga didapatkan

$$\sum_{i=1}^{16} KURANG(i) + X$$

Bernilai 16.

Karena 16 adalah angka genap maka persoalan dapat diselesaikan.

Selanjutnya, akan dilakukan pembangkitan pohon status pencarian dengan algoritma branch and bound. Pada program, akan digunakan struktur data priority queue untuk menyimpan simpul hidup. Prioritas antrian akan ditentukan oleh nilai cost. Cost pada simpul P didefinisikan sebagai

$$\hat{c}(P) = f(P) + \hat{g}(P)$$

f(P) adalah kedalaman lintasan dari simpul akar ke P sedangkan g(P) adalah taksiran panjang lintasan terpendek dari P ke simpul solusi pada upa pohon dengan akar P. Dengan kata lain, f(P) adalah level dari simpul dan g(P) adalah jumlah ubin tidak kosong yang tidak terdapat pada susunan akhir.

Susunan akhir dari persoalan 15 puzzle diatas adalah

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

Pembangkitan simpul berdasarkan aksi yang dilakukan oleh puzzle. Terdapat empat aksi yang merupakan gerakan memindahkan ubin kosong. Diantaranya:

1. Up, ubin kosong ditukar posisinya dengan ubin di sebelah atasnya.
2. Right, ubin kosong ditukar posisinya dengan ubin di sebelah kanannya.
3. Down, ubin kosong ditukar posisinya dengan ubin di sebelah bawahnya.
4. Left, ubin kosong ditukar posisinya dengan ubin di sebelah kirinya.

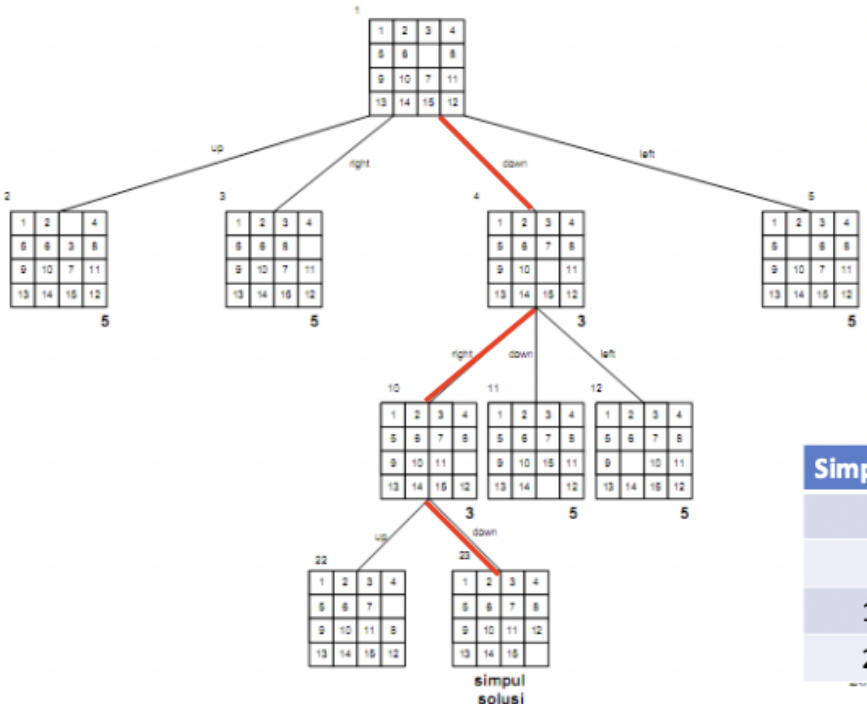
Terdapat batasan atas aksi tersebut sebagai berikut

1. Apabila ubin kosong terletak dibaris paling atas atau simpul parent melakukan aksi down, akibatnya aksi up tidak bisa dilakukan
2. Apabila ubin kosong terletak dikolom paling kanan atau simpul parent melakukan aksi left, akibatnya aksi right tidak bisa dilakukan
3. Apabila ubin kosong terletak dikolom paling bawah atau simpul parent melakukan aksi up, akibatnya aksi down tidak dapat dilakukan.
4. Apabila ubin kosong terletak dikolom paling kiri atau simpul parent melakukan aksi right, akibatnya aksi left tidak dapat dilakukan.

Algoritma branch and bound diterapkan sebagai berikut.

1. Simpul akar diinisiasi dengan menyimpan status puzzle awal. Simpul akar dimasukkan kedalam priority queue simpul hidup.
2. Kemudian diakukan perulangan pada priority queue simpul hidup. Perulangan akan berhenti ketika antrian kosong atau puzzle sudah terurut.
3. Pada setiap perulangan, satu simpul di dequeue dari antrian, simpul tersebut akan menjadi simpul ekspan. Simpul tersebut merupakan simpul dnegan cost paling kecil.

4. Dilakukan iterasi sebanyak 4 kali pada simpul ekspan tersebut sesuai dengan aksi yang dapat dilakukan. Jika aksi yang diberikan valid, maka akan dakan dibangkitkan simpul baru. Simpul baru tersebut akan menyimpan indormasi simpul ekspan sebagai simpul induknya, status puzzle sesuai dengan aksi yang diberikan.
5. Jika puzzle sudah mencapai kondisi terurut maka perulangan berhenti. Selanjutnya akan dipanggil fungsi yang mencetak langkah penyelesaian puzzle.



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

24

Source Code Program

Agar program menjadi modular, program dipecah menjadi 5 file.

1. main.py

   File main.py bertanggung jawab sebagai program utama dan menjadi file yang dijalankan pertama kali

```
'''
main.py bertanggung jawab dalam menjalankan program dengan algoritma branch and bound
'''

from Puzzle import Puzzle
from FileManager import FileManager
from PriorityQueue import PriorityQueue
from Node import Node
from Constant import SIZE
from PuzzleGenerator import PuzzleGenerator
import time


'''
megecek apakah puzzle sudah terurut
parameter:
   puzzle: puzzle yang akan dicek
return:
   True jika puzzle sudah terurut
'''
def isFinish(puzzle):
   flat = puzzle.oneLineBoard()
   for i in range(1, (SIZE**2)+1):
       if(flat[i-1] != i):
           return False

   return True

'''
mencetak puzzle solusi langkah yang diambil sampai ke solusi akhir
'''
def generateSolution(solved_state):
   Node_solution = []

   parent = solved_state.parent
   state = solved_state
```

```
    while(parent != None):
        Node_solution.insert(0, state)
        state = parent
        parent = parent.parent
    for i in range(len(Node_solution)):
        Node_solution[i].root.printBoard()
    return Node_solution

'''
mengecek banyak tiles yang tidak sesuai dengan posisinya
parameter:
    puzzle: puzzle yang akan dicek
return:
    banyak tiles yang tidak sesuai dengan posisinya
'''
def numberMisplacedTiles(puzzle):
    result = 0
    flat = puzzle.oneLineBoard()

    for i in range(1, SIZE**2+1):
        if(flat[i-1] != i):
            result+=1

    return result


def main():
    '''
    Tahap Input
    '''
    choose = int(input("1. Random puzzle\n2. Input puzzle (direkomendasikan)\n"))
    if (choose == 1):
        file = PuzzleGenerator()
        root = Node(Puzzle(file.getMatrix()))
    elif(choose == 2):
        filename = input("Masukkan nama file puzzle: contoh: solveable_01.txt\n")
        try:
            fm = FileManager("../test/" + filename)
            root = Node(Puzzle(fm.getMatrix()))
        except:
            print("File tidak ditemukan")
            exit()
    else:
        print("Input salah")
        exit()

    '''
```

```
    Tahap Inisiasi
    '''
    # menginisiasi root
    root.root.printBoard()
    print()

    # mengecek apakah puzzle dapat diselesaikan
    if(not root.root.solveable()):
        print("Puzzle tidak dapat diselesaikan.")
        exit()

    print("Puzzle dapat diselesaikan.")
    print()

    # menginisiasi simpul yang dibuat
    nodeCount = 1

    # cost terendah diprioritaskan dalam priority queue
    costFunction = numberMisplacedTiles
    pq = PriorityQueue(lambda x,y : x.depth + costFunction(x.root) <= y.depth +
costFunction(y.root))

    # menginisiasi priority queue
    pq.push(root)

    # variabel menyimpan state solusi
    solutionState = None

    # variabel menyimpan kemungkinan move di puzzle
    movesUnits = [(-1,0), (0,-1), (1,0), (0,1)]
    movesNames = ["Up", "Left", "Down", "Right"]

    # Start timer
    time_start = time.process_time_ns()


    '''
    Tahap Pencarian Dengan Branch and Bound
    '''
    while(not pq.isEmpty()):
        # mengambil item palign depan dari queue
        current = pq.front()
        pq.pop()

        # jika puzzle sudah terurut maka simpan state lalu berhenti
        if(isFinish(current.root)):
            solutionState = current
            break
```

```
        # memasukkan setiap status yang mungkin ke priority queue
        for i, (dr, dc) in enumerate(movesUnits):
            if(movesNames[(i+2)%4] != current.move):
                # membuat simpul
                result = Node(current.root.move(dr, dc), parent=current,
depth=current.depth+1, move=movesNames[i])

                # Jika move bisa dilakukan, maka tambahkan ke priority queue
                if(result != None and result.root != None):
                    nodeCount += 1
                    pq.push( result )

    # Stop timer
    time_end = time.process_time_ns()

    '''
    Tahap Output
    '''
    # membuat array yang berisi simpul simpul dari pohon ruang status yuang diambil
    solutionArray = generateSolution(solutionState)

    # Total moves yang dilakukan
    print("Total moves:", len(solutionArray))

    # Banyak simpul dibuat
    print(nodeCount,"simpul dibuat")

    # Total waktu yang digunakan
    total_time = time_end - time_start
    print("Total waktu: ", total_time / 1000000, "ms")

if __name__ == "__main__":
    main()
```

2. Filemanager.py

   Bertanggung jawab dalam mengolah input dari file menjadi matrix yang akan diproses

```
'''
Filemanager bertanggung jawab untuk mengonversi input dari file menjadi
matrix
'''
class FileManager:
    '''
    membuat matrix puzzle dari file
    '''
    def __init__(self, path):
        f = open(path, "r")
        temp = f.readlines()
        self.matrix = []
        for item in temp:
            a = item.strip("\n").split(" ")
            self.matrix.append(a)
        for i in range(4) :
            for j in range(4) :
                if self.matrix[i][j] == 'X' :
                    self.matrix[i][j] = 16
                else :
                    self.matrix[i][j] = int(self.matrix[i][j])

    '''
    mengambil matrix puzzle
    return: matrix puzzle
    '''
    def getMatrix(self):
        return self.matri
```

3. PuzzleGenerator.py

   Bertanggung jawab dalam pembuatan puzzle acak

```
from Constant import SIZE
import numpy as np

'''
membuat matrix puzzle random berukuran 4 x 4 berisi angka 1-16 secara acak
'''
class PuzzleGenerator:
```

```
    def __init__(self):
        self.matrix = []
        temp = np.random.permutation(np.arange(1, 17))
        temp = np.ndarray.tolist(temp)
        self.matrix.append(temp[0:4])
        self.matrix.append(temp[4:8])
        self.matrix.append(temp[8:12])
        self.matrix.append(temp[12:16])

    '''
    mengambil matrix puzzle
    return: matrix puzzle
    '''
    def getMatrix(self):
        return self.matrix
```

4. Puzzle
   Puzzle bertanggung jawab dalam mengolah puzzle yang akan dipecahkan

```
'''
Puzzle.py merepresentasikan puzzle yang akan dicari solusinya.
'''
import copy
from Constant import SIZE

class Puzzle:
    '''
    constructor:
    '''
    def __init__(self, matrix):
        self.board = matrix

    '''
    membuat board dari 2d menjadi 1d
    '''
    def oneLineBoard(self):
        return [item for sublist in self.board for item in sublist]

    '''
    mecari cell yang kosong
    return: tuple (row, col)
    '''
    def findEmpty(self):
        for i,row in enumerate(self.board):
            for j,value in enumerate(row):
                if(value==SIZE**2):
                    return (i,j)
    '''
```

```
    mengecek apakah empty cell berada di bagian arsir atau tidak
    return: 1 jika berada di bagian arsir 0 jika tidak
    '''
    def getParity(self):
        (r, c) = self.findEmpty()
        return (r+c) % 2

    '''
    memindahkan cell kosong ke (r+dr, c+dc)
    parameter:
        dr: row
        dc: column
    return: puzzle baru yang sudah dipindahkan
    '''
    def move(self, dr, dc):
        (r, c) = self.findEmpty()
        if(r+dr>=0 and r+dr<SIZE and c+dc>=0 and c+dc<SIZE):
            movedPuzzle = copy.deepcopy(self)
            movedPuzzle.board[r][c], movedPuzzle.board[r+dr][c+dc] =
movedPuzzle.board[r+dr][c+dc], movedPuzzle.board[r][c]
            return movedPuzzle
        else:
            return None


    '''
    mengecek apakah puzzle solvable atau tidak
    return: True jika solvable, False jika tidak
    '''
    def solveable(self):
        flat = self.oneLineBoard()
        x = self.getParity()

        sum = 0
        for i in range(0,SIZE**2):
            for j in range(i+1,SIZE**2):
                if(flat[i]>flat[j]):
                    sum+=1

        print("Sigma Kurang(i):", sum)
        print("X:", x)
        print("Total:", sum + x, "(genap)" if (sum+x) % 2 == 0 else "(ganjil)")

        return (sum + x) % 2 == 0

    '''
    Mencetak Puzzle ke layar
    '''
```

```
    def printBoard(self):
        for row in self.board:
            for value in row:
                print('%4s' % (value if value!=SIZE**2 else "#"), end="")
            print()
        print("==================")
```

5. Priorityqueue.py

Priorityqueue bertanggung jawab dalam antrian yang digunakan untuk menyimpan simpul hidup

```
'''
pririty queue digunakan untuk membuat antrian dari simpul hidup
'''
class PriorityQueue:
    '''
    membuat priorityFunction
    '''
    def __init__(self, priorityFunction):
        self.queue = []
        self.function = priorityFunction

    '''
    memasukkan elemen ke antrian sesuai dengan fungsi yang didefinisikan
    '''
    def push(self, item):
        pos = 0
        found = False

        while(not found and pos < len(self.queue)):
            if(self.function(item, self.queue[pos])):
                found = True
            else:
                pos+=1

        self.queue.insert(pos, item)

    '''
    melakukan pop elemen dari antrian
    '''
    def pop(self):
        self.queue.pop(0)

    '''
    melihat elemen antrian terdepan
    '''
    def front(self):
        return self.queue[0]
```

```
    '''
    mengecek apakah priority queue kosong
    '''
    def isEmpty(self):
        return len(self.queue) == 0
```

6. Node.py

```
Node digunakan untuk merepresentasikan simpul dalam pohon ruang status
'''
Node digunakan untuk merepresentasikan pohon ruang status dalam branch and bound
'''
class Node:
    def __init__(self, root, parent = None, depth=0, move=""):
        self.root = root
        self.parent = parent
        self.move = move
        self.depth = depth
```

7. Constant.py
   Constant.py digunakan untuk menyimpan konstanta

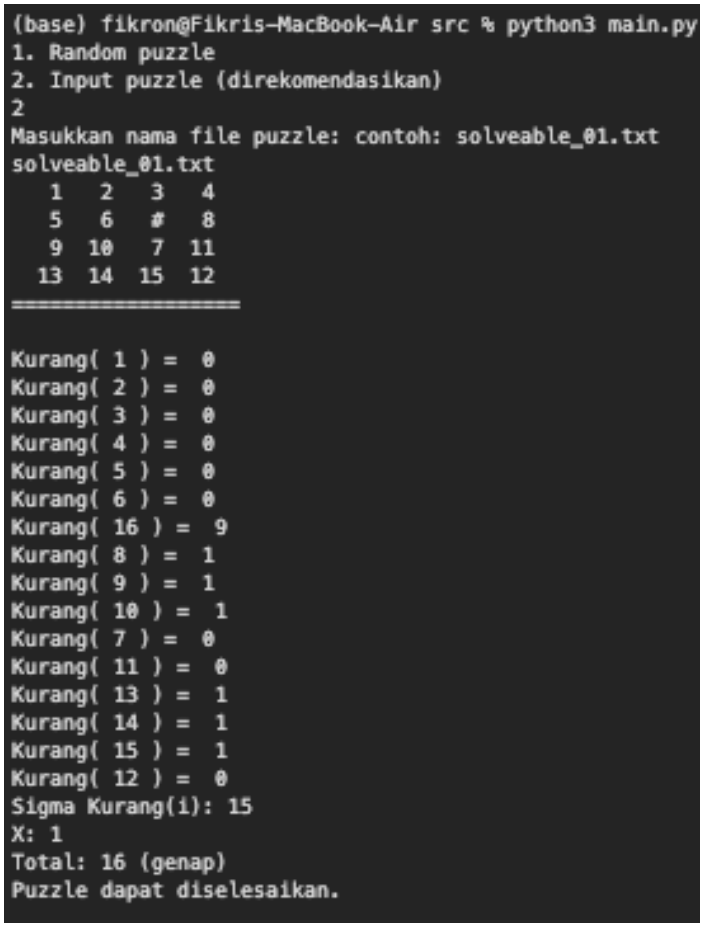
```
'''
constant mendefinisikan SIZE Puzzle
'''
SIZE = 4
```

Screenshot Input dan Output

Untuk menguji kebenaran program, diberikan 5 buah instansiasi persoalan 15-puzzle dengan 2 kasus tidak dapat diselesaikan dan 3 kasus yang dapat diselesaikan

1. Persoalan Kasus Dapat Diselesaikan 1

   Input: solveable_01.txt



2. Persoalan kasus dapat diselesaikan 2

   Input: solveable_02.txt

   1 2 12 3

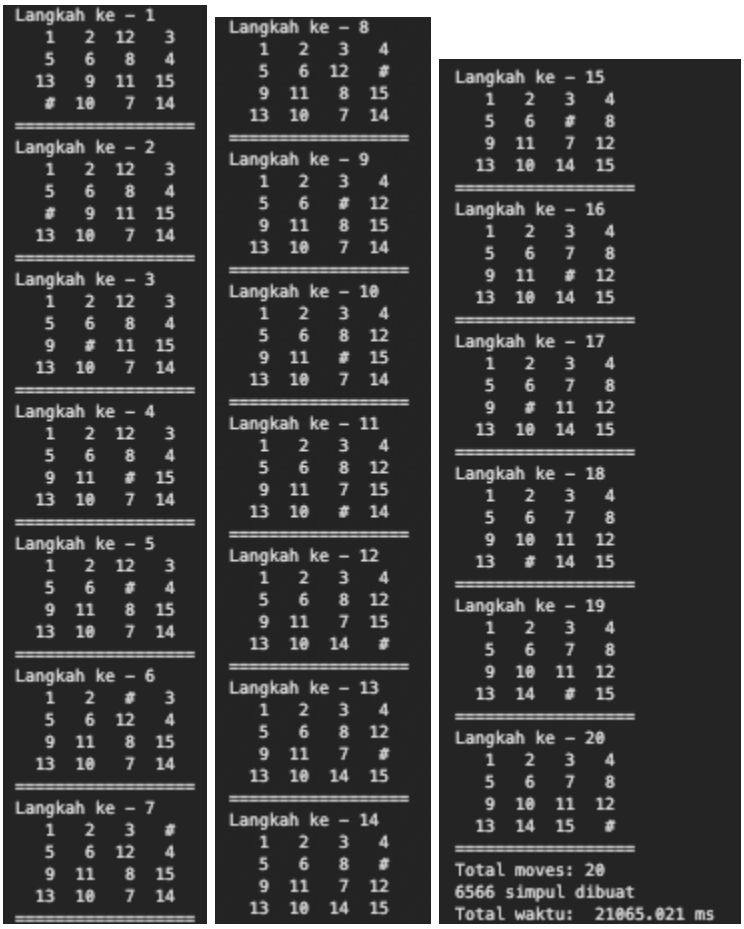
   5 6 8 4

   13 9 11 15

   10 16 7 14

```
solveable_02.txt
   1   2  12   3
   5   6   8   4
  13   9  11  15
  10   #   7  14
====================

Kurang( 1 ) =  0
Kurang( 2 ) =  0
Kurang( 12 ) =  9
Kurang( 3 ) =  0
Kurang( 5 ) =  1
Kurang( 6 ) =  1
Kurang( 8 ) =  2
Kurang( 4 ) =  0
Kurang( 13 ) =  4
Kurang( 9 ) =  1
Kurang( 11 ) =  2
Kurang( 15 ) =  3
Kurang( 10 ) =  1
Kurang( 16 ) =  2
Kurang( 7 ) =  0
Kurang( 14 ) =  0
Sigma Kurang(i): 26
X: 0
Total: 26 (genap)
Puzzle dapat diselesaikan.
```



3. Pesoalan kasus dapat diselesaikan 3

   Input: solveable_03.txt

   1 2 3 4

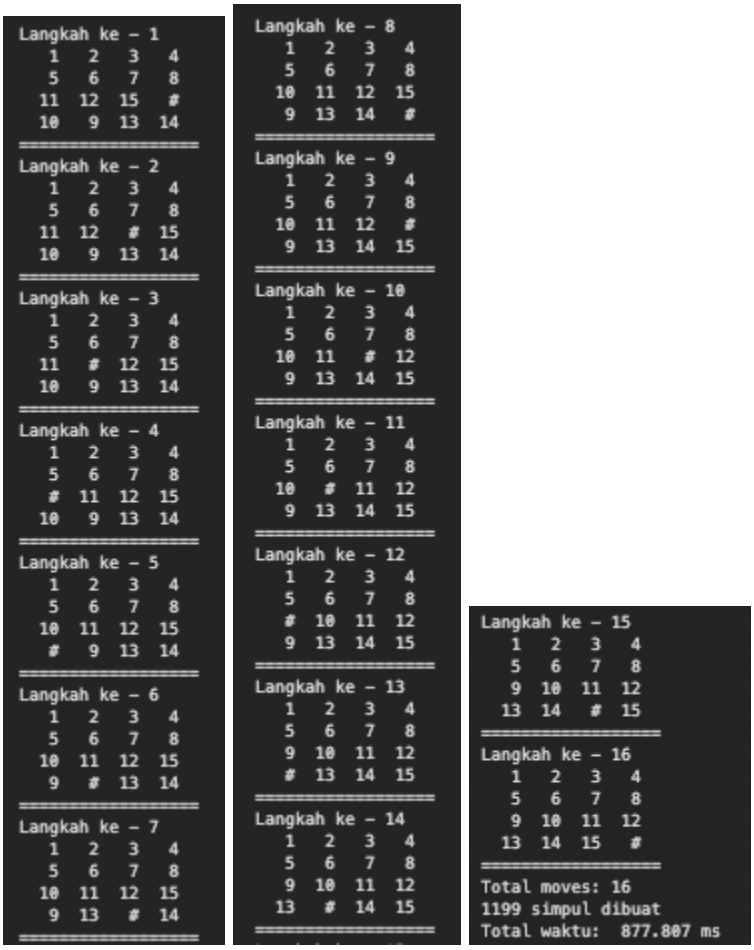
   5 6 7 8

   11 12 15 14

   10 9 13 16

```
1. Random puzzle
2. Input puzzle (direkomendasikan)
2
Masukkan nama file puzzle: contoh: solveable_01.txt
solveable_03.txt
   1   2   3   4
   5   6   7   8
  11  12  15  14
  10   9  13   #
====================

Kurang( 1 ) =  0
Kurang( 2 ) =  0
Kurang( 3 ) =  0
Kurang( 4 ) =  0
Kurang( 5 ) =  0
Kurang( 6 ) =  0
Kurang( 7 ) =  0
Kurang( 8 ) =  0
Kurang( 11 ) =  2
Kurang( 12 ) =  2
Kurang( 15 ) =  4
Kurang( 14 ) =  3
Kurang( 10 ) =  1
Kurang( 9 ) =  0
Kurang( 13 ) =  0
Kurang( 16 ) =  0
Sigma Kurang(i): 12
X: 0
Total: 12 (genap)
Puzzle dapat diselesaikan.
```



4. Persoalan kasus tidak dapat diselesaikan 1

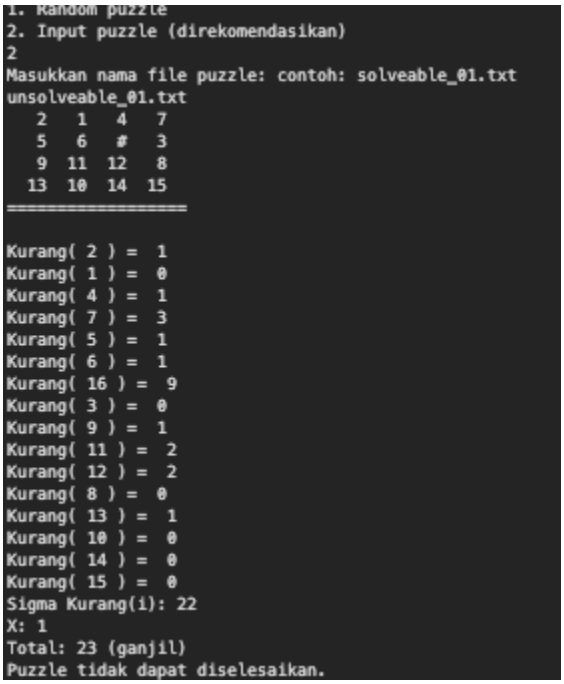
   Input: unsolveable_01.txt

   2 1 4 7

   5 6 16 3

   9 11 12 8

   13 10 14 15



5. Persoalan kasus tidak dapat diselesaikan 2

   Input: unsolveable_02.txt

   1 2 14 3

   5 6 4 7

   15 10 11 16

   9 13 8 12

```
1. Random puzzle
2. Input puzzle (direkomendasikan)
2
Masukkan nama file puzzle: contoh: solveable_01.txt
unsolveable_02.txt
   1   2  14   3
   5   6   4   7
  15  10  11   #
   9  13   8  12
====================

Kurang( 1 ) =  0
Kurang( 2 ) =  0
Kurang( 14 ) =  11
Kurang( 3 ) =  0
Kurang( 5 ) =  1
Kurang( 6 ) =  1
Kurang( 4 ) =  0
Kurang( 7 ) =  0
Kurang( 15 ) =  6
Kurang( 10 ) =  2
Kurang( 11 ) =  2
Kurang( 16 ) =  4
Kurang( 9 ) =  1
Kurang( 13 ) =  2
Kurang( 8 ) =  0
Kurang( 12 ) =  0
Sigma Kurang(i): 30
X: 1
Total: 31 (ganjil)
Puzzle tidak dapat diselesaikan.
```

Alamat Repository Program

https://github.com/fikrikhoironn/15-puzzle-solver

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		