

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:

Hilda Carissa 13520164

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

2022

DAFTAR ISI

DAFTAR ISI	1
Cara Kerja Program	2
Screenshot Input - Output Program	5
Kode Program	8
Contoh Instansiasi Persoalan 15-Puzzle	14
Link Kode Program	23
Checklist	23
Daftar Referensi	23

Cara Kerja Program

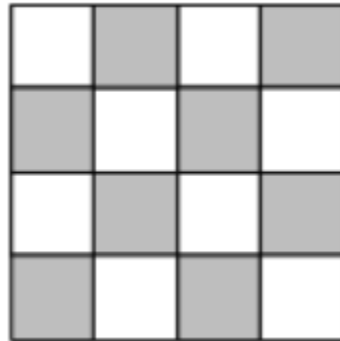
Dalam penyelesaian 15-puzzle, digunakan algoritma *Branch and Bound*. Algoritma ini biasa digunakan untuk persoalan optimasi – meminimalkan atau memaksimalkan suatu fungsi obyektif yang tidak melanggar *constraints* persoalan. Pada algoritma B&B, setiap simpul akan diberikan nilai *cost* yaitu nilai taksiran lintasan termurah ke simpul status tujuan melalui simpul status tertentu. Simpul berikutnya yang akan di-*expand*, yaitu simpul dengan *cost* terkecil (atau terbesar, tergantung persoalan optimasi), dan bukan berdasarkan urutan pembangkitannya. Algoritma ini juga menerapkan pemangkasan pada jalur yang dianggap tidak mengarah pada solusi. Secara umum, simpul yang dipangkas adalah simpul yang memiliki nilai *cost* tidak lebih baik dibanding nilai terbaik sejauh ini dan simpul yang tidak *feasible* karena terdapat *constraints* yang dilanggar.

Pada pembuatan penyelesaian 15-Puzzle, saya menggunakan bahasa Python. Terdapat 3 file yang dibutuhkan dalam menjalankan program utama. Pertama terdapat file `main.py`. File ini digunakan untuk meng-*handle* input, pencarian solusi dan output. Pada input, terdapat 3 pilihan. Pertama dengan memasukkan nama file yang berada di folder test, kedua dengan memasukkan matrix secara manual oleh user, dan ketiga dengan membuat matrix random 16 x 16. Dalam input matrix, untuk *tile* kosong, digunakan angka 16 untuk mempermudah proses pengecekan solusi. Setelah input diterima, akan dibuat Node root yang berisi Id node, parent node, matrix input serta cost. Lalu, matrix input akan di-*print*, kemudian dijalankan fungsi `canBeSolved` yang bertujuan untuk mengetahui apakah matrix persoalan dapat diselesaikan atau tidak. Fungsi ini selain mengembalikan boolean True atau False, juga mencetak hasil fungsi Kurang untuk semua *tile* serta sum untuk semua fungsi dari tiap *tile* + X.

Jika keluaran fungsi itu True, maka node root akan dimasukkan ke dalam heap `activeSet`. Lalu dijalankan loop untuk mengecek, jika kita pop elemen dengan *cost* terkecil dari `activeSet`, apakah elemen tersebut memiliki matrix yang sesuai dengan Goal State / Final State. Jika sesuai, maka akan di-*print* semua matrix dari node parent-nya dan jumlah node yang dibangkitkan. Akan tetapi jika tidak, maka akan dibangkitkan node-node berdasarkan pergeseran *tile* kosong. Tentu saja terdapat beberapa *constraint* seperti apakah *tile* dapat digeser sesuai movement-i (misalkan *tile* berada di paling kanan, maka tidak akan dapat bergeser ke kanan), dan ada juga *constraint* untuk memastikan bahwa pergeseran dari *tile* kosong tidak akan membuat matrix kembali ke state sebelumnya. Namun, jika fungsi `canBeSolved` mengembalikan nilai False, maka akan dikeluarkan pesan bahwa puzzle tidak dapat diselesaikan. Di akhir, dicetak juga execution time yang dibutuhkan dari awal pengecekan matrix hingga ditemukan node dengan final statenya.

Selanjutnya, terdapat file `function.py`. Di dalam file ini, terdapat beberapa fungsi yang berhubungan dengan fungsi Kurang(i). Pertama terdapat fungsi posisi yang bertujuan mengecek int x ada di index berapa di matrix. Index disini berupa integer dari 0 hingga 15. Kemudian terdapat fungsi Kurang yang mengembalikan hasil fungsi Kurang dari integer x yang ada di dalam matrix. Selanjutnya ada fungsi `cekPosisiKosong` yang bertujuan untuk

mengetahui apakah *tile* kosong dalam matrix berada di *tile* yang diarsir seperti pada gambar di bawah ini.



Kemudian terdapat juga fungsi `canBeSolved` yang sebenarnya sudah disebutkan di paragraf sebelum-sebelumnya. Intinya, dalam fungsi ini digunakan fungsi-fungsi lain di dalam file `function.py`. Jika hasil total dari fungsi `Kurang(i)` ditambah `X` yang didapat dari fungsi `cekPosisiKosong` adalah genap, maka dapat disimpulkan bahwa puzzle dapat diselesaikan. Namun jika hasilnya ganjil, maka puzzle tidak dapat diselesaikan. Dalam fungsi ini, hasil fungsi `Kurang(i)` dari masing-masing *tile* juga dicetak, dan hasil total yang ditambah `X` juga dicetak.


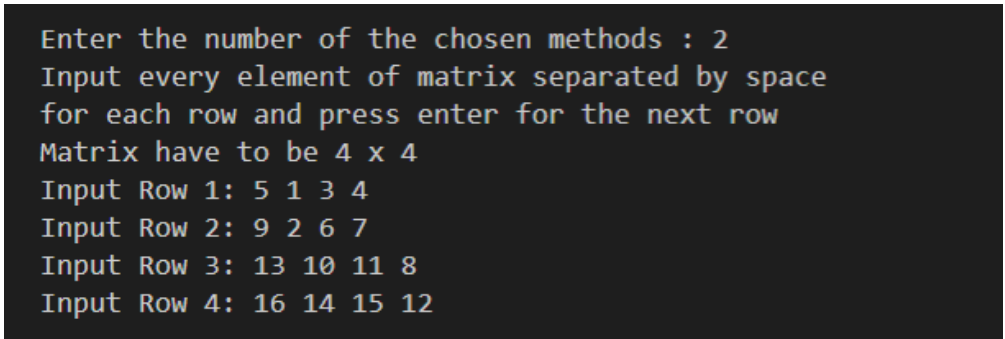
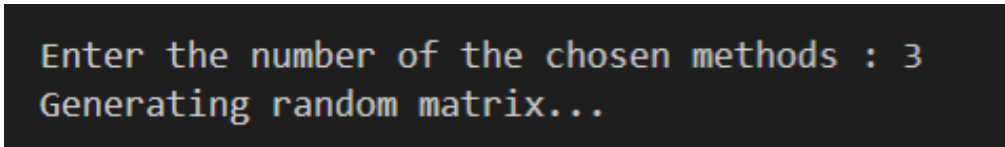
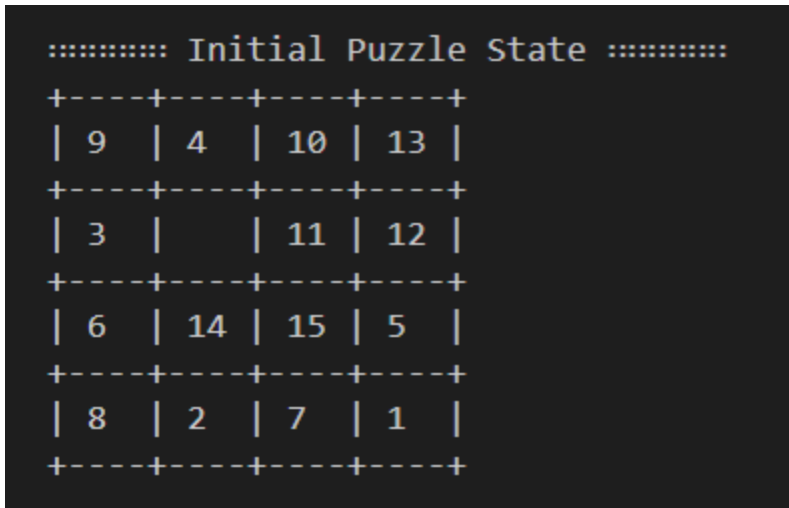
Terakhir, dalam file `node.py`, terdapat beberapa fungsi serta pendeklarasian kelas yang berhubungan dengan node atau simpul. Pertama pendeklarasian kelas `Node` yang menggunakan struktur data `namedTuple` yang tersedia di Python. Terdapat 4 atribut yaitu `nodeKe` yang berisi integer, bisa dibilang sebagai id dari sebuah `Node`. Kemudian terdapat `parent` yaitu list `parent` dari node ini, lalu `matrix` yang berisi 15-Puzzle yang terbaru setelah mengalami pergeseran, dan terakhir ada `cost` yaitu nilai taksiran dari matrix menuju final state.

Selanjutnya terdapat 2 fungsi yang berguna pada input. Ada `inputOwnMatrix` yang meng-*handle* pembuatan node root dengan matrix dari input user. Dan ada juga `generateRandomMatrix` yang berguna untuk membuat matrix berisi angka 1 - 16 dengan posisi random. Lalu ada fungsi `printMatrix` yang berguna untuk mencetak matrix dari node yang ada di parameter. Kemudian terdapat fungsi `moveNode` yang bisa dibilang merupakan fungsi utama. Fungsi ini menerima parameter `rootNode`, `movement`, serta `nodeKe`. Pada pemanggilan fungsi ini, akan dibuat variabel untuk mengisi atribut node baru, atribut `nodeKe` akan menggunakan parameter `nodeKe`. Lalu atribut `parent` akan pertama menduplikasi isi list dari `parent rootNode`, lalu menambahkan `rootNode` sebagai `parent`. Sama seperti atribut `parent`, atribut `nMatrix` juga pertama menduplikasi atribut `matrix` dari `rootNode`.

Setelah itu, berdasarkan pergerakannya, 1 untuk atas, 2 untuk kanan, 3 untuk bawah, 4 untuk kiri, akan dilakukan pergeseran-pergeseran sehingga pada akhirnya terdapat matrix baru setelah dilakukan pergeseran itu. Setelah ada matrix baru, dihitung `cost` dari matrix baru tersebut untuk mencapai `finalsState`. Setelah itu, node terbaru akan di `append` ke dalam `parent`, lalu dibuat `Node` baru dengan atribut yang sudah dibuat dan dijelaskan sebelumnya.

Selain itu, masih terdapat 4 fungsi lain. Pertama ada fungsi `isFinalState` untuk mengecek apakah `matrix` di parameter sudah mencapai final state. Lalu ada fungsi `findcost` yang mengembalikan nilai `cost` dari node tersebut. Ada juga fungsi `isPossible` yang akan menerima `matrix` serta `movement` yang diinginkan untuk dicek, lalu kemudian akan mengembalikan `false` apabila `movement` tidak valid. Misalnya jika *tile* kosong berada di paling bawah maka tidak bisa bergeser ke bawah. Terakhir ada fungsi `isGaBalik` yang akan mengembalikan `true` jika `matrix` hasil pergeseran tidak sama dengan `matrix` sebelumnya.

Screenshot Input - Output Program

		Screenshot
Input	File	 <pre> Welcome to : 15-PUZZLE SOLVER Choose input matrix methods : 1. Input File 2. Input Manual 3. Randomize Matrix Enter the number of the chosen methods : 1 Enter file name in folder test ex. 'matrix.txt' (without the apostrophe) : test1.txt </pre>
	Manual	 <pre> Enter the number of the chosen methods : 2 Input every element of matrix separated by space for each row and press enter for the next row Matrix have to be 4 x 4 Input Row 1: 5 1 3 4 Input Row 2: 9 2 6 7 Input Row 3: 13 10 11 8 Input Row 4: 16 14 15 12 </pre>
	Random	 <pre> Enter the number of the chosen methods : 3 Generating random matrix... </pre>
	State Awal	 <pre> ::::::::: Initial Puzzle State :::::::::: +---+---+---+---+ 9 4 10 13 +---+---+---+---+ 3 11 12 +---+---+---+---+ 6 14 15 5 +---+---+---+---+ 8 2 7 1 +---+---+---+---+ </pre>

Output	Fungsi Kurang	<pre> : Values of "Kurang" : • Kurang(1) = 0 • Kurang(2) = 0 • Kurang(3) = 1 • Kurang(4) = 1 • Kurang(5) = 4 • Kurang(6) = 0 • Kurang(7) = 0 • Kurang(8) = 0 • Kurang(9) = 4 • Kurang(10) = 1 • Kurang(11) = 1 • Kurang(12) = 0 • Kurang(13) = 4 • Kurang(14) = 1 • Kurang(15) = 1 • Kurang(16) = 3 Sum Of Kurang(i) + X = 22 Puzzle is solvable! Finding solution... </pre>	<pre> : Values of "Kurang" : • Kurang(1) = 0 • Kurang(2) = 1 • Kurang(3) = 2 • Kurang(4) = 3 • Kurang(5) = 2 • Kurang(6) = 3 • Kurang(7) = 1 • Kurang(8) = 3 • Kurang(9) = 8 • Kurang(10) = 7 • Kurang(11) = 6 • Kurang(12) = 6 • Kurang(13) = 9 • Kurang(14) = 5 • Kurang(15) = 5 • Kurang(16) = 10 Sum Of Kurang(i) + X = 71 Puzzle is not solvable! Execution Time : 0.01095128 Seconds Press any key to exit... </pre>
	Solusi	<pre> The solution is found! Step 0 +---+---+---+---+ 5 1 3 4 +---+---+---+---+ 9 2 6 7 +---+---+---+---+ 13 10 11 8 +---+---+---+---+ 14 15 12 +---+---+---+---+ Step 1 +---+---+---+---+ 5 1 3 4 +---+---+---+---+ 9 2 6 7 +---+---+---+---+ 10 11 8 +---+---+---+---+ 13 14 15 12 +---+---+---+---+ </pre>	<pre> Step 8 +---+---+---+---+ 1 2 3 4 +---+---+---+---+ 5 6 7 8 +---+---+---+---+ 9 10 11 +---+---+---+---+ 13 14 15 12 +---+---+---+---+ Step 9 +---+---+---+---+ 1 2 3 4 +---+---+---+---+ 5 6 7 8 +---+---+---+---+ 9 10 11 12 +---+---+---+---+ 13 14 15 +---+---+---+---+ </pre>

Code Program

```
main.py

1  from node import *
2  from functions import *
3  from queue import *
4  import time
5  import heapq
6
7  # node : nodeke, parent, matrix, cost, movementlist
8  # awal program
9  print("Welcome to :")
10 print()
11 print("15-Puzzle Solver")
12 print("1. Input File")
13 print("2. Input Manual")
14 print("3. Randomize Matrix")
15 print()
16 print("Choose input matrix methods : ")
17 print()
18 print("1. Input File")
19 print("2. Input Manual")
20 print("3. Randomize Matrix")
21 print()
22 print()
23 methods = int(input("Enter the number of the chosen methods : "))
24 # ngecek input
25 while(methods != 1 and methods !=2 and methods!=3):
26     print("Invalid input, try again")
27     methods = int(input("Enter the number of the chosen methods : "))
28 # input untuk yang file
29 if(methods == 1):
30     namaFile = str(input("Enter file name in folder test\nex. 'matrix.txt' (without the apostrophe) : "))
31     with open("./test/" + namaFile) as f:
32         lines = f.readlines()
33         matrix = [[int(x) for x in line.split()]for line in lines]
34         cost = findcost(matrix, 0)
35         root = Node(1, [], matrix, cost)
36 # input matrix sendiri
37 elif(methods == 2):
38     root = inputOwnMatrix()
39 else:
40     print("Generating random matrix...")
41     matrix = generateRandomMatrix()
42     cost = findcost(matrix, 0)
43     root = Node(1, [], matrix, cost)
```

Gambar 1 Input 15-Puzzle

```

#solving
allNode = []
counterNode = 1
print()
print(":::::::::: Initial Puzzle State ::::::::::")
printMatrix(root)
print("\n:::::::::: Values of \"Kurang\" ::::::::::")
start_time = time.time()
if(canBeSolved(root.matrix)):
    print()
    print("Puzzle is solvable! \n")
    print("Finding solution...\n")
    time.sleep(0.5)
    activeSet = []
    heapq.heappush(activeSet, (root.cost, root))
    allNode.append(root)
    while(len(activeSet)!= 0):
        check = heapq.heappop(activeSet)
        # get the node only
        checking = check[1]
        if(isFinalState(checking)):
            print("The solution is found!")
            counter = 0
            # print semua solusi
            for nodes in (checking.parent):
                print("::::::::::")
                print("Step "+str(counter))
                printMatrix(allNode[nodes-1])
                print("::::::::::")
                counter+=1
            break
        else:
            # ngecek untuk setiap movement di node itu
            for i in range(1,5):
                #kalau movementnya ga bikin balik ke state sebelumnya, dan bisa pindah(ga nabrak pinggir")
                if(isGaBalik(checking, allNode) and isPossible(checking.matrix, i)):
                    #hitung jumlah simpul
                    counterNode += 1
                    #buat simpul baru yang udah pindah
                    newNode = moveNode(checking, i, counterNode)
                    #masukkin simpul baru ke array semua node
                    allNode.append(newNode)
                    #masukkin simpul baru ke queue
                    heapq.heappush(activeSet, (newNode.cost, newNode))
                    del newNode
            print("::::::::::")
            print("Sum of Nodes\n(including the root node) : ", counterNode)
    else:
        print("Puzzle is not solvable!")
        print("::::::::::")
print("Execution Time : %.10s Seconds" % (time.time() - start_time))
print("::::::::::")
input("Press any key to exit...")

```

Gambar 2 dan 3 Print puzzle, start time, cek apakah puzzle dapat di solve, mencari solution puzzle

node.py

```
from typing import NamedTuple
import copy
from sympy import root
from functions import *

You, 3 hours ago | 1 author (You)
class Node(NamedTuple):
    nodeKe: int
    parent: list
    matrix: list[list]
    cost: int

# input matrix dari user
def inputOwnMatrix():
    print("Input every element of matrix separated by space\nfor each row and press enter for the next row\nMatrix have to be 4 x 4")
    newMatrix = []
    for i in range(4):
        line = input("Input Row "+str(i+1)+" : ")
        newRow = [int(x) for x in line.split()]
        newMatrix.append(newRow)
    cost = findcost(newMatrix, 0)
    root = Node(1, [], newMatrix, cost)
    return root

# print matrix
def printMatrix(Node):
    for i in range(4):
        print("+---+---+---+---+")
        for j in range(4):
            print("| ", end="")
            if(Node.matrix[i][j] < 10):
                print(Node.matrix[i][j], end=' ')
            elif(Node.matrix[i][j] == 16):
                print(' ', end=' ')
            else:
                print(Node.matrix[i][j], end=' ')
            if(j==3):
                print("|", end="")
        print()
    print("+---+---+---+---+")
```

Gambar 4 class Node, input matrix sendiri, print matrix

```

# membangun node baru setelah tile kosong di geser
def moveNode(rootNode, movement, nodeKe):
    # movement
    # 1 - atas
    # 2 - kanan
    # 3 - bawah
    # 4 - kiri
    isTraded = False
    #parent dari root ditambahkan ke node
    parent = copy.deepcopy(rootNode.parent)
    if(rootNode.nodeKe not in parent):
        parent.append(rootNode.nodeKe)
    #matrix dari root ditambahkan ke node
    nMatrix = copy.deepcopy(rootNode.matrix)
    #tile 16 dipindah ke atas
    if(movement==1):
        for i in range(4):
            for j in range(4):
                if(nMatrix[i][j] == 16):
                    # trade tile 16 dengan atasnya
                    temp = nMatrix[i-1][j]
                    nMatrix[i-1][j] = nMatrix[i][j]
                    nMatrix[i][j] = temp
                    # cari cost new matrix
                    cost = findcost(nMatrix, len(parent))
                    parent.append(nodeKe)
                    # make new node
                    newNode = Node(nodeKe, parent, nMatrix, cost)
                    isTraded = True
                    break
            if(isTraded):
                break

```

```

#tile 16 dipindah ke kanan
elif(movement==2):
    for i in range(4):
        for j in range(4):
            if(nMatrix[i][j] == 16):
                # trade tile 16 dengan kanannya
                temp = nMatrix[i][j+1]
                nMatrix[i][j+1] = nMatrix[i][j]
                nMatrix[i][j] = temp
                # cari cost new matrix
                cost = findcost(nMatrix, len(parent))
                parent.append(nodeKe)
                # make new node
                newNode = Node(nodeKe, parent, nMatrix, cost)
                isTraded = True
                break
            if(isTraded):
                break
elif(movement==3):
    #tile 16 dipindah ke bawah
    for i in range(4):
        for j in range(4):
            if(nMatrix[i][j] == 16):
                # trade tile 16 dengan bawahnya
                temp = nMatrix[i+1][j]
                nMatrix[i+1][j] = nMatrix[i][j]
                nMatrix[i][j] = temp
                # cari cost new matrix
                cost = findcost(nMatrix, len(parent))
                parent.append(nodeKe)
                # make new node
                newNode = Node(nodeKe, parent, nMatrix, cost)
                isTraded = True
                break
            if(isTraded):
                break

```

```

elif(movement==4):
    #tile 16 dipindah ke kiri
    for i in range(4):
        for j in range(4):
            if(nMatrix[i][j] == 16):
                # trade tile 16 dengan kirinya
                temp = nMatrix[i][j-1]
                nMatrix[i][j-1] = nMatrix[i][j]
                nMatrix[i][j] = temp
                # cari cost new matrix
                cost = findcost(nMatrix, len(parent))
                parent.append(nodeKe)
                # make new node
                newNode = Node(nodeKe, parent, nMatrix, cost)
                isTraded = True
                break
            if(isTraded):
                break
    else:
        print("Invalid Movement Input")
    return newNode

# mengecek apakah matrix sudah di goal state
def isFinalState(finalNode):
    for i in range(4):
        for j in range(4):
            if(finalNode.matrix[i][j] != 4*i+j+1):
                return False
    return True

# mencari cost dari matrix
def findcost(matrix, jarak):
    cost = jarak
    for i in range(4):
        for j in range(4):
            if(matrix[i][j] != 4*i+j+1 and matrix[i][j] != 16):
                cost+=1
    return cost

```

Gambar 5, 6, 7 Penggeseran Node, pengecekan final state matrix, pencarian cost

```

# mengecek apakah tile dapat digerakkan sesuai dengan movement
def isPossible(matrix, movement):
    for i in range(4):
        for j in range(4):
            if(matrix[i][j] == 16):
                if(movement==1 and i == 0):
                    return False
                elif(movement==2 and j==3):
                    return False
                elif(movement==3 and i==3):
                    return False
                elif(movement==4 and j==0):
                    return False
                elif(movement < 5 and movement > 0):
                    return True

# mengecek apakah matrix setelah dipindahkan tidak akan sama dengan state sebelumnya
def isGaBalik(node, allNode):
    if(node.nodeKe == 1):
        return True
    idxparent = len(node.parent) - 2
    for i in range(4):
        for j in range(4):
            if(node.matrix[i][j] != allNode[idxparent].matrix[i][j]):
                return True
    return False

```

Gambar 8 Pengecekan penggeseran tile dan state matrix sama atau tidak dengan state sebelumnya

functions.py

```
import numpy as np

# Check posisi x di matrix
def posisi(x, matrix):
    index = 0
    for i in range(4):
        for j in range(4):
            if matrix[i][j] == x:
                index = (4*i)+j
    return index

# Fungsi kurang
def kurang(x, matrix):
    hasil = 0
    for i in range(4):
        for j in range(4):
            if(matrix[i][j] < x and posisi(matrix[i][j], matrix) > posisi(x, matrix)):
                hasil += 1
    return hasil

# Cek apakah posisi kosong ada di tile yang diarsir
def cekPosisiKosong(matrix):
    finalPos = 0
    shadedArea = [1, 3, 4, 6, 9, 11, 12, 14]
    for i in range(4):
        for j in range(4):
            if (matrix[i][j] == 16 and ((4*i+j) in shadedArea)):
                finalPos = 1
    return finalPos
```

Gambar 9 cek posisi, fungsi kurang dan cek posisi tile kosong

```
# Cek apakah matrix dapat di solve
def canBeSolved(matrix):
    kurangi = 0
    # display fungsi kurang untuk semua tile
    for i in range(1, 17):
        kurangi += kurang(i, matrix)
        print("• Kurang("+str(i)+") = ", kurang(i, matrix))
    kurangi += cekPosisiKosong(matrix)
    print("\nSum Of Kurang(i) + X = ", kurangi)

    # bila genap, bisa di solve
    if(kurangi%2 == 0):
        return True
    return False

# Generate a random 4x4 matrix
def generateRandomMatrix():
    matrix = np.arange(1, 17)
    np.random.shuffle(matrix)

    matrix = np.reshape(matrix, (4,4))
    return matrix.tolist()
```

Gambar 10 cek apakah matrix dapat diselesaikan, generate matrix random

Contoh Instansiasi Persoalan 15-Puzzle

1. Test 1 - Solvable

Input Matrix (in file test1.txt)
<pre>6 5 2 4 9 1 3 8 10 16 7 15 13 14 12 11</pre>
Output
<pre>Welcome to : 15-PUZZLE SOLVER Choose input matrix methods : 1. Input File 2. Input Manual 3. Randomize Matrix Enter the number of the chosen methods : 1 Enter file name in folder test ex. 'matrix.txt' (without the apostrophe) : test1.txt :~::~: Initial Puzzle State ~::~: +---+---+---+---+ 6 5 2 4 +---+---+---+---+ 9 1 3 8 +---+---+---+---+ 10 7 15 +---+---+---+---+ 13 14 12 11 +---+---+---+---+ :~::~: Values of "Kurang" ~::~: • Kurang(1) = 0 • Kurang(2) = 1 • Kurang(3) = 0 • Kurang(4) = 2 • Kurang(5) = 4 • Kurang(6) = 5 • Kurang(7) = 0 • Kurang(8) = 1 • Kurang(9) = 4 • Kurang(10) = 1 • Kurang(11) = 0 • Kurang(12) = 1 • Kurang(13) = 2 • Kurang(14) = 2 • Kurang(15) = 4 • Kurang(16) = 6 Sum Of Kurang(i) + X = 34</pre>

Puzzle is solvable!

Finding solution...

The solution is found!

Step 0

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

Step 1

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

Step 2

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

Step 3

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

Step 4

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

Step 5

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

Step 6

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

Step 7

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

Step 8

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


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

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

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

```

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

```

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

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

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

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

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

```
Press any key to exit...
```

2. Test 2 - Solvable

Input Matrix (in file test2.txt)																
<table><tr><td>5</td><td>1</td><td>3</td><td>4</td></tr><tr><td>9</td><td>2</td><td>6</td><td>7</td></tr><tr><td>13</td><td>10</td><td>11</td><td>8</td></tr><tr><td>16</td><td>14</td><td>15</td><td>12</td></tr></table>	5	1	3	4	9	2	6	7	13	10	11	8	16	14	15	12
5	1	3	4													
9	2	6	7													
13	10	11	8													
16	14	15	12													
Output																
<pre>Welcome to : 15-PUZZLE SOLVER Choose input matrix methods : 1. Input File 2. Input Manual 3. Randomize Matrix Enter the number of the chosen methods : 1 Enter file name in folder test ex. 'matrix.txt' (without the apostrophe) : test2.txt ::: Initial Puzzle State ::: +---+---+---+---+ 5 1 3 4 +---+---+---+---+ 9 2 6 7 +---+---+---+---+ 13 10 11 8 +---+---+---+---+ 14 15 12 +---+---+---+---+ ::: Values of "Kurang" ::: • Kurang(1) = 0 • Kurang(2) = 0 • Kurang(3) = 1 • Kurang(4) = 1 • Kurang(5) = 4 • Kurang(6) = 0 • Kurang(7) = 0 • Kurang(8) = 0 • Kurang(9) = 4 • Kurang(10) = 1 • Kurang(11) = 1 • Kurang(12) = 0 • Kurang(13) = 4 • Kurang(14) = 1 • Kurang(15) = 1 • Kurang(16) = 3 Sum Of Kurang(i) + X = 22 Puzzle is solvable!</pre>																

The solution is found!

Step 0

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

Step 1

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

Step 2

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

Step 3

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

Step 4

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

Step 5

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

Step 6

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

Step 7

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

Step 8

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

Step 9

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

Sum of Nodes

(including the root node) : 28

Execution Time : 0.55882883 Seconds

Press any key to exit...

3. Test 3 - Solvable

Input Matrix (in file test3.txt)	
	<pre>2 3 7 4 1 6 11 8 5 10 12 15 9 13 14 16</pre>
Output	
<pre>Welcome to : 15-PUZZLE SOLVER Choose input matrix methods : 1. Input File 2. Input Manual 3. Randomize Matrix Enter the number of the chosen methods : 1 Enter file name in folder test ex. 'matrix.txt' (without the apostrophe) : test3.txt ::: Initial Puzzle State ::: +-----+ 2 3 7 4 +-----+ 1 6 11 8 +-----+ 5 10 12 15 +-----+ 9 13 14 +-----+ ::: Values of "Kurang" ::: • Kurang(1) = 0 • Kurang(2) = 1 • Kurang(3) = 1 • Kurang(4) = 1 • Kurang(5) = 0 • Kurang(6) = 1 • Kurang(7) = 4 • Kurang(8) = 1 • Kurang(9) = 0 • Kurang(10) = 1 • Kurang(11) = 4 • Kurang(12) = 1 • Kurang(13) = 0 • Kurang(14) = 0 • Kurang(15) = 3 • Kurang(16) = 0 Sum Of Kurang(i) + X = 18 Puzzle is solvable!</pre>	

```
Finding solution...

The solution is found!
=====
Step 0
+---+---+---+---+
| 2 | 3 | 7 | 4 |
+---+---+---+---+
| 1 | 6 | 11 | 8 |
+---+---+---+---+
| 5 | 10 | 12 | 15 |
+---+---+---+---+
| 9 | 13 | 14 |   |
+---+---+---+---+
=====
=====
Step 1
+---+---+---+---+
| 2 | 3 | 7 | 4 |
+---+---+---+---+
| 1 | 6 | 11 | 8 |
+---+---+---+---+
| 5 | 10 | 12 |   |
+---+---+---+---+
| 9 | 13 | 14 | 15 |
+---+---+---+---+
=====
=====
Step 2
+---+---+---+---+
| 2 | 3 | 7 | 4 |
+---+---+---+---+
| 1 | 6 | 11 | 8 |
+---+---+---+---+
| 5 | 10 |   | 12 |
+---+---+---+---+
| 9 | 13 | 14 | 15 |
+---+---+---+---+
=====
=====
Step 3
+---+---+---+---+
| 2 | 3 | 7 | 4 |
+---+---+---+---+
| 1 | 6 |   | 8 |
+---+---+---+---+
| 5 | 10 | 11 | 12 |
+---+---+---+---+
| 9 | 13 | 14 | 15 |
+---+---+---+---+
=====
=====
Step 4
+---+---+---+---+
| 2 | 3 |   | 4 |
+---+---+---+---+
| 1 | 6 | 7 | 8 |
+---+---+---+---+
| 5 | 10 | 11 | 12 |
+---+---+---+---+
| 9 | 13 | 14 | 15 |
+---+---+---+---+
=====
=====
Step 5
+---+---+---+---+
| 2 |   | 3 | 4 |
+---+---+---+---+
| 1 | 6 | 7 | 8 |
+---+---+---+---+
| 5 | 10 | 11 | 12 |
+---+---+---+---+
| 9 | 13 | 14 | 15 |
+---+---+---+---+
=====
=====
Step 6
+---+---+---+---+
|   | 2 | 3 | 4 |
+---+---+---+---+
| 1 | 6 | 7 | 8 |
+---+---+---+---+
| 5 | 10 | 11 | 12 |
+---+---+---+---+
| 9 | 13 | 14 | 15 |
+---+---+---+---+
=====
=====
Step 7
+---+---+---+---+
| 1 | 2 | 3 | 4 |
+---+---+---+---+
|   | 6 | 7 | 8 |
+---+---+---+---+
| 5 | 10 | 11 | 12 |
+---+---+---+---+
| 9 | 13 | 14 | 15 |
+---+---+---+---+
=====
=====
Step 8
+---+---+---+---+
| 1 | 2 | 3 | 4 |
+---+---+---+---+
| 5 | 6 | 7 | 8 |
+---+---+---+---+
|   | 10 | 11 | 12 |
+---+---+---+---+
| 9 | 13 | 14 | 15 |
+---+---+---+---+
=====
=====
Step 9
+---+---+---+---+
+---+---+---+---+
| 5 | 6 | 7 | 8 |
+---+---+---+---+
| 9 | 10 | 11 | 12 |
+---+---+---+---+
| 13 | 14 | 15 |   |
+---+---+---+---+
=====
=====
Sum of Nodes
(including the root node) : 39
Execution Time : 0.56383466 Seconds
=====
Press any key to exit...
=====
```

4. Test 4 - Unsolvble

Input Matrix (in file test4.txt)
<pre>7 9 3 14 4 10 6 8 15 16 13 2 1 5 11 12</pre>
Output
<pre>15-PUZZLE SOLVER Choose input matrix methods : 1. Input File 2. Input Manual 3. Randomize Matrix Enter the number of the chosen methods : 1 Enter file name in folder test ex. 'matrix.txt' (without the apostrophe) : test4.txt ::: Initial Puzzle State ::: +-----+ 7 9 3 14 +-----+ 4 10 6 8 +-----+ 15 13 2 +-----+ 1 5 11 12 +-----+ ::: Values of "Kurang" ::: • Kurang(1) = 0 • Kurang(2) = 1 • Kurang(3) = 2 • Kurang(4) = 2 • Kurang(5) = 0 • Kurang(6) = 3 • Kurang(7) = 6 • Kurang(8) = 3 • Kurang(9) = 7 • Kurang(10) = 5 • Kurang(11) = 0 • Kurang(12) = 0 • Kurang(13) = 5 • Kurang(14) = 10 • Kurang(15) = 6 • Kurang(16) = 6 Sum Of Kurang(i) + X = 57 Puzzle is not solvable! ::: Execution Time : 0.00613427 Seconds ::: Press any key to exit...█</pre>

5. Test 5 - Unsolvable

Input Matrix (in file test5.txt)
<pre>6 11 7 13 8 5 9 16 3 4 14 15 2 1 12 10</pre>
Output
<pre>15-PUZZLE SOLVER Choose input matrix methods : 1. Input File 2. Input Manual 3. Randomize Matrix Enter the number of the chosen methods : 1 Enter file name in folder test ex. 'matrix.txt' (without the apostrophe) : test5.txt ::: Initial Puzzle State ::: +---+---+---+---+ 6 11 7 13 +---+---+---+---+ 8 5 9 +---+---+---+---+ 3 4 14 15 +---+---+---+---+ 2 1 12 10 +---+---+---+---+ ::: Values of "Kurang" ::: • Kurang(1) = 0 • Kurang(2) = 1 • Kurang(3) = 2 • Kurang(4) = 2 • Kurang(5) = 4 • Kurang(6) = 5 • Kurang(7) = 5 • Kurang(8) = 5 • Kurang(9) = 4 • Kurang(10) = 0 • Kurang(11) = 9 • Kurang(12) = 1 • Kurang(13) = 9 • Kurang(14) = 4 • Kurang(15) = 4 • Kurang(16) = 8 Sum Of Kurang(i) + X = 63 Puzzle is not solvable! ::: Execution Time : 0.00599980 Seconds ::: Press any key to exit...</pre>

Link Kode Program

https://github.com/hcarissa/Tucil3_13520164

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

Algoritma Branch and Bound. Munir, R. 2022.

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Branch-and-Bound-2021-Bagian1.pdf> . Diakses pada 30 April 2022.

Pemanfaatan Algoritma A* dalam Penyelesaian 15 Puzzle Game. Pradanika, E. 2020.

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2019-2020/Makalah/stima2020k3-002.pdf> . Diakses pada 31 April 2022