LAPORAN TUGAS KECIL II

Penyelesaian Persoalan 15-Puzzle dengan Algoritma Branch and Bound

Laporan dibuat untuk memenuhi salah satu tugas mata kuliah

IF2211 Strategi Algoritma

Disusun oleh:

Dimas Shidqi Parikesit 13520087

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2022

DAFTAR ISI

Algoritma Branch and Bound

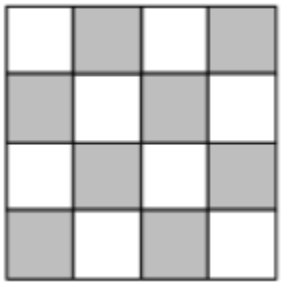
Pada kasus ini digunakan algoritma branch and bound yang berbasis DFS. Algoritma ini terdiri dari beberapa bagian

1. Pembuatan Priority Queue

   Pada program ini, priority queue diimplementasikan sendiri tanpa menggunakan fitur bawaan python. Implementasi priority queue ini dilakukan dengan memodifikasi struktur data List pada python. Pada program dibuat sebuah class PriorityQueue yang memiliki atribut List queue dan Integer count. Kemudian priority queue diimplementasikan pada member function insert, dimana setelah data dimasukkan ke List queue, List tersebut kemudian di sort sesuai prioritas. Prioritas pada priority queue ini adalah nilai cost terendah diikuti nilai jarak dari akar terendah. Selain itu terdapat pula member function pop, isEmpty, length, dan simpulCount.

2. Menghitung jumlah hasil fungsi Kurang(i) tiap ubin+X

   Nilai jumlah Kurang(i) tiap ubin+X ini dapat digunakan untuk mengecek apakah puzzle mungkin untuk diselesaikan atau tidak tanpa mencari kemungkinan yang ada. Kurang(i) didefinisikan sebagai banyaknya ubin bernomor j sedemikian sehingga j < i dan Posisi(j) > Posisi(i), dimana Posisi(i) = posisi ubin bernomor i pada susunan yang diperiksa. Sementara itu X bernilai satu apabila sel kosong terletak pada ubin yang diarsir pada gambar berikut.

   Apabila sel kosong tidak pada ubin yang diarsir, maka X bernilai nol.

   Setelah didapatkan jumlah Kurang(i)+X, maka dapat diambil kesimpulan puzzle dapat diselesaikan apabila nilai nya genap. Apabila ganjil maka ubin tidak dapat diselesaikan.

3. Pembangkitan simpul

   Dari state saat ini, simpul dapat dibangkitkan dengan membuat kemungkinan state apabila puzzle digerakkan ke atas, bawah, kanan, dan kiri ditambah aturan simpul yang baru dibangkitkan ini belum pernah dicek atau dibangkitkan sebelumnya. Dengan tambahan aturan ini dapat dipastikan bahwa puzzle tidak akan kembali ke state yang pernah dialami sebelumnya.

4. Menghitung cost simpul

   Tiap simpul yang akan dibangkitkan ini harus dicari nilai cost function nya. Cost function dapat dihitung dengan menjumlahkan jarak simpul dari akar dengan jumlah ubin yang saat ini belum berada pada posisi akhir. Nilai cost function ini dimasukkan pada sebuah tuple yang sama dengan simpul untuk kemudian dimasukkan ke dalam priority queue.

5. Menjalankan algoritma untuk simpul berikutnya

   Algoritma ini akan terus berjalan sampai priority kosong atau kondisi akhir telah terpenuhi. Penentuan simpul berikutnya yang akan dicek adalah dengan berdasarkan urutan pada priority queue, dimana simpul yang memiliki cost function terkecil akan diprioritaskan.

Kode Program

main.py 2 | prioqueue.py | test.txt | bnb.py

prioqueue.py > PriorityQueue > insert

```python
def takeCost(elem):
    return elem[2],elem[1]


class PriorityQueue(object):
    def __init__(self):
        self.queue = []
        self.count = 0

    def __str__(self):
        return ' '.join([str(i) for i in self.queue])

    # for checking if the queue is empty
    def isEmpty(self):
        return len(self.queue) == 0

    def length(self):
        return len(self.queue)

    def simpulCount(self):
        return self.count

    # for inserting an element in the queue
    def insert(self, data):
        self.queue.append(data)
        self.queue.sort(key=takeCost)
        self.count += 1


    # for popping an element based on Priority
    def pop(self):
        item = self.queue[0]
        del self.queue[0]
        return item
```

main.py 2 | prioqueue.py | test.txt | bnb.py

bnb.py > solve

```python
import prioqueue
import copy
import time

# state adalah tuple dengan urutan (array 15 puzzle, int distFromRoot, int cost, int idx parent di array done)

def kurang(num, state):
    idx = -1
    for i in range(16):
        if state[0][i] == num:
            idx = i

    count = 0
    for i in range(idx, 16):
        if state[0][i] != -1 and state[0][i] < num:
            count += 1

    return count


def sumKurang(state):
    count = 0
    for i in range(1, 17):
        count += kurang(i, state)
    return count


def findX(state):
    shadowed = [1, 3, 4, 6, 9, 11, 12, 14]
    for i in range(1, 16):
        if state[0][i] == 16 and i in shadowed:
            return 1
    return 0


def kurangPlusX(state):
    return sumKurang(state) + findX(state)
```

main.py 2 | prioqueue.py | test.txt | bnb.py

bnb.py > solve

```
    return sumKurang(state) + findX(state)


def solvable(state):
    return kurangPlusX(state) % 2 == 0


def costFuncG(matrix):
    count = 0
    for i in range(16):
        if matrix[i] != i + 1 and matrix[i]!=16:
            count += 1
    return count


def move(state, direction):
    newMatrix = copy.deepcopy(state[0])

    idx = -1
    for i in range(16):
        if newMatrix[i] == 16:
            idx = i

    idxTukar = -1
    if direction == "top":
        if idx < 4:
            return newMatrix
        idxTukar = idx - 4
    elif direction == "bottom":
        if idx > 11:
            return newMatrix
        idxTukar = idx + 4
    elif direction == "left":
        if idx % 4 == 0:
            return newMatrix
        idxTukar = idx - 1
    elif direction == "right":
        if (idx + 1) % 4 == 0:
```

main.py 2 | prioqueue.py | test.txt | bnb.py

bnb.py > solve

```
74            if (idx + 1) % 4 == 0:
75                return newMatrix
76            idxTukar = idx + 1
77
78        newMatrix[idx] = newMatrix[idxTukar]
79        newMatrix[idxTukar] = 16
80
81        return newMatrix
82
83    def solve(initState, path, simpulTime):
84        start = time.perf_counter()
85
86        # Nilai fungsi Kurang(i) tiap ubin awal
87        for i in range(1, 17):
88            print("Kurang",i, kurang(i, initState))
89
90        # Nilai sumKurang(i)+X
91        print()
92        print("sumKurang(i)+X =",kurangPlusX(initState))
93        print()
94
95        prioQueue = prioqueue.PriorityQueue()
96
97        if solvable(initState) == False:
98            print("Jumlah simpul dibangkitkan =", prioQueue.simpulCount())
99            return False
100
101       prioQueue.insert(initState)
102       allPath = []
103       done = []
104       inQueue = []
105
106       while prioQueue.isEmpty()==False:
107           currState = prioQueue.pop()
108           done.append(currState[0])
109           allPath.append(currState)
110
```

main.py 2 | prioqueue.py | test.txt | bnb.py

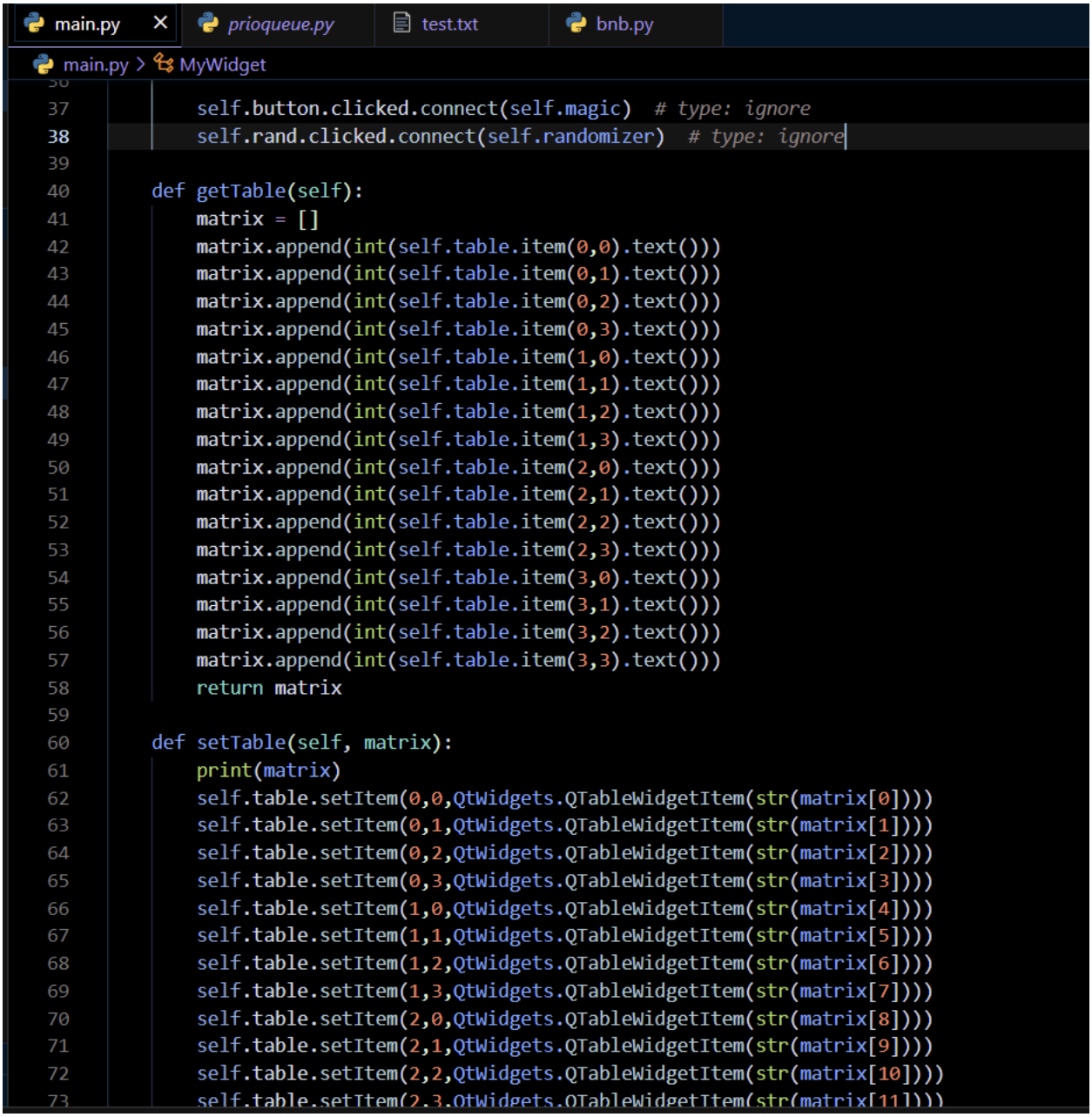
bnb.py > solve

```
111         if solvable(currState):
112             if (currState[1]==currState[2]):
113                 path.insert(0, currState)
114                 while path[0][3]!=-1:
115                     path.insert(0, allPath[path[0][3]])
116 
117                 print("Here")
118                 for i in range(len(path)):
119                     print(path[i][0])
120 
121                 print()
122                 print("Jumlah simpul dibangkitkan =", prioQueue.simpulCount())
123                 print("Waktu =","{:.5f}".format(time.perf_counter()-start), "detik")
124 
125                 simpulTime.append(prioQueue.simpulCount())
126                 simpulTime.append(time.perf_counter()-start)
127                 return True
128 
129             print(currState)
130             print(prioQueue.length())
131             newTop = move(currState, "top")
132             newBot = move(currState, "bottom")
133             newLeft = move(currState, "left")
134             newRight = move(currState, "right")
135 
136             f = currState[1] + 1
137             gTop = costFuncG(newTop) if (newTop not in done) and (newTop not in inQueue) else -1
138             gBot = costFuncG(newBot) if (newBot not in done) and (newBot not in inQueue) else -1
139             gLeft = costFuncG(newLeft) if (newLeft not in done) and (newLeft not in inQueue) else -1
140             gRight = costFuncG(newRight) if (newRight not in done) and (newRight not in inQueue) else -1
141 
142             if gTop != -1:
143                 stateTop = (newTop, f, f + gTop, len(done)-1)
144                 prioQueue.insert(stateTop)
145                 inQueue.append(newTop)
146             if gBot != -1:
147                 stateBot = (newBot, f, f + gBot, len(done)-1)
148                 prioQueue.insert(stateBot)
149                 inQueue.append(newBot)
150             if gLeft -1:
151                 stateLeft = (newLeft, f, f + gLeft,len(done)-1)
152                 prioQueue.insert(stateLeft)
153                 inQueue.append(newLeft)
154             if gRight != -1:
155                 stateRight = (newRight, f, f + gRight,len(done)-1)
156                 prioQueue.insert(stateRight)
157                 inQueue.append(newRight)
158     print("Unknown Error")
```

main.py 2 | prioqueue.py | test.txt | bnb.py

main.py > MyWidget > getTable

```python
from bnb import *
import sys
import os
import random
from PySide6 import (QtCore, QtWidgets, QtGui)

class MyWidget(QtWidgets.QWidget):
    def __init__(self):
        super().__init__()

        self.path = []
        self.isSolvable = False
        self.arrSimpulTime = []
        self.initState = ()

        self.button = QtWidgets.QPushButton("Start!")
        self.text = QtWidgets.QLabel("Masukkan kondisi awal pada tabel. Cell kosong direpresentasikan dengan angka 16. Asumsi input benar.
        Apabila solvable, pencet button start lagi untuk menunjukkan step")
        self.table = QtWidgets.QTableWidget()
        self.status = QtWidgets.QLabel("Status: Not Started")
        self.result = QtWidgets.QTextEdit("")
        self.rand = QtWidgets.QPushButton("Randomize!")
        self.file = QtWidgets.QFileDialog(filter="Text files (*.txt)")

        self.table.setRowCount(4)
        self.table.setColumnCount(4)
        self.result.setReadOnly(True)

        layout = QtWidgets.QVBoxLayout(self)
        layout.addWidget(self.text)
        layout.addWidget(self.button)
        layout.addWidget(self.status)
        layout.addWidget(self.result)
        layout.addWidget(self.table)
        layout.addWidget(self.rand)
        layout.addWidget(self.file)

```


main.py
prioqueue.py
test.txt
bnb.py
main.py > MyWidget
self.button.clicked.connect(self.magic) # type: ignore
self.rand.clicked.connect(self.randomizer) # type: ignore
def getTable(self):
matrix = []
matrix.append(int(self.table.item(0,0).text()))
matrix.append(int(self.table.item(0,1).text()))
matrix.append(int(self.table.item(0,2).text()))
matrix.append(int(self.table.item(0,3).text()))
matrix.append(int(self.table.item(1,0).text()))
matrix.append(int(self.table.item(1,1).text()))
matrix.append(int(self.table.item(1,2).text()))
matrix.append(int(self.table.item(1,3).text()))
matrix.append(int(self.table.item(2,0).text()))
matrix.append(int(self.table.item(2,1).text()))
matrix.append(int(self.table.item(2,2).text()))
matrix.append(int(self.table.item(2,3).text()))
matrix.append(int(self.table.item(3,0).text()))
matrix.append(int(self.table.item(3,1).text()))
matrix.append(int(self.table.item(3,2).text()))
matrix.append(int(self.table.item(3,3).text()))
return matrix
def setTable(self, matrix):
print(matrix)
self.table.setItem(0,0,QtWidgets.QTableWidgetItem(str(matrix[0])))
self.table.setItem(0,1,QtWidgets.QTableWidgetItem(str(matrix[1])))
self.table.setItem(0,2,QtWidgets.QTableWidgetItem(str(matrix[2])))
self.table.setItem(0,3,QtWidgets.QTableWidgetItem(str(matrix[3])))
self.table.setItem(1,0,QtWidgets.QTableWidgetItem(str(matrix[4])))
self.table.setItem(1,1,QtWidgets.QTableWidgetItem(str(matrix[5])))
self.table.setItem(1,2,QtWidgets.QTableWidgetItem(str(matrix[6])))
self.table.setItem(1,3,QtWidgets.QTableWidgetItem(str(matrix[7])))
self.table.setItem(2,0,QtWidgets.QTableWidgetItem(str(matrix[8])))
self.table.setItem(2,1,QtWidgets.QTableWidgetItem(str(matrix[9])))
self.table.setItem(2,2,QtWidgets.QTableWidgetItem(str(matrix[10])))

main.py | prioqueue.py | test.txt | bnb.py

main.py > MyWidget

```
73          self.table.setItem(2,3,QtWidgets.QTableWidgetItem(str(matrix[11])))
74          self.table.setItem(3,0,QtWidgets.QTableWidgetItem(str(matrix[12])))
75          self.table.setItem(3,1,QtWidgets.QTableWidgetItem(str(matrix[13])))
76          self.table.setItem(3,2,QtWidgets.QTableWidgetItem(str(matrix[14])))
77          self.table.setItem(3,3,QtWidgets.QTableWidgetItem(str(matrix[15])))
78
79      def setTextBox(self):
80          for i in range(1,17):
81              self.result.append("Kurang "+str(i)+ " = " +str(kurang(i, self.initState)))
82
83          self.result.append("sumKurang(i)+X = "+ str(kurangPlusX(self.initState)))
84
85      def openFile(self, fileName):
86          matrix = []
87          f = open(fileName, "r")
88          for line in f:
89              matrix.append(int(line.strip('\n').strip()))
90          f.close()
91          return matrix
92
93      @QtCore.Slot()
94      def randomizer(self):
95          matrix = random.sample(range(1,17), 16)
96          self.setTable(matrix)
97          self.initState = (matrix, 0, 0+costFuncG(matrix), -1)
98
99      @QtCore.Slot()
100     def magic(self):
101         if self.isSolvable==False:
102             if self.initState == ():
103                 matrix = []
104                 if self.file.selectedFiles()[0]!=os.getcwd().replace('\\', '/'):
105                     matrix = self.openFile(self.file.selectedFiles()[0])
106                     self.setTable(matrix)
107                 else:
108                     matrix = self.getTable()
109                 self.status.setText("Status: Processing")
```

main.py prioqueue.py test.txt bnb.py

main.py > MyWidget

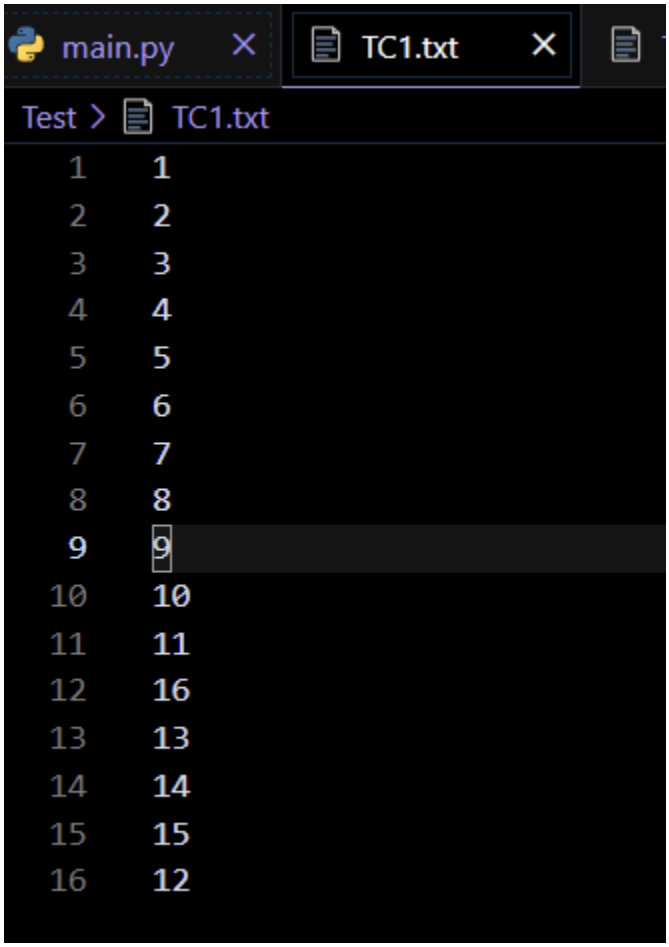
```
                    self.setTable(matrix)
                else:
                    matrix = self.getTable()
                self.status.setText("Status: Processing")
                self.initState = (matrix, 0, 0+costFuncG(matrix), -1)

            self.setTextBox()
            self.isSolvable = solve(self.initState, self.path, self.arrSimpulTime)
            self.status.setText("Status: " + ("Solvable with time "+"{:.5f}".format(self.arrSimpulTime[1]) + " seconds and simpul count " +
            str(self.arrSimpulTime[0])) if self.isSolvable==True else "Unsolvable" )
            if self.isSolvable==True:
                print("Time =", self.arrSimpulTime[1])
                print("Simpul count =", self.arrSimpulTime[0])
                self.result.append("\n Urutan path")
                for i in range(len(self.path)):
                    self.result.append(str(self.path[i][0]))
                    print(self.path[i][0])
                del self.path[0]
        elif len(self.path)>0:
            self.setTable(self.path[0][0])
            del self.path[0]

if __name__ == '__main__':
    app = QtWidgets.QApplication([])

    widget = MyWidget()
    widget.resize(800, 600)
    widget.show()

    sys.exit(app.exec())
```

Screenshot Input - Output Program

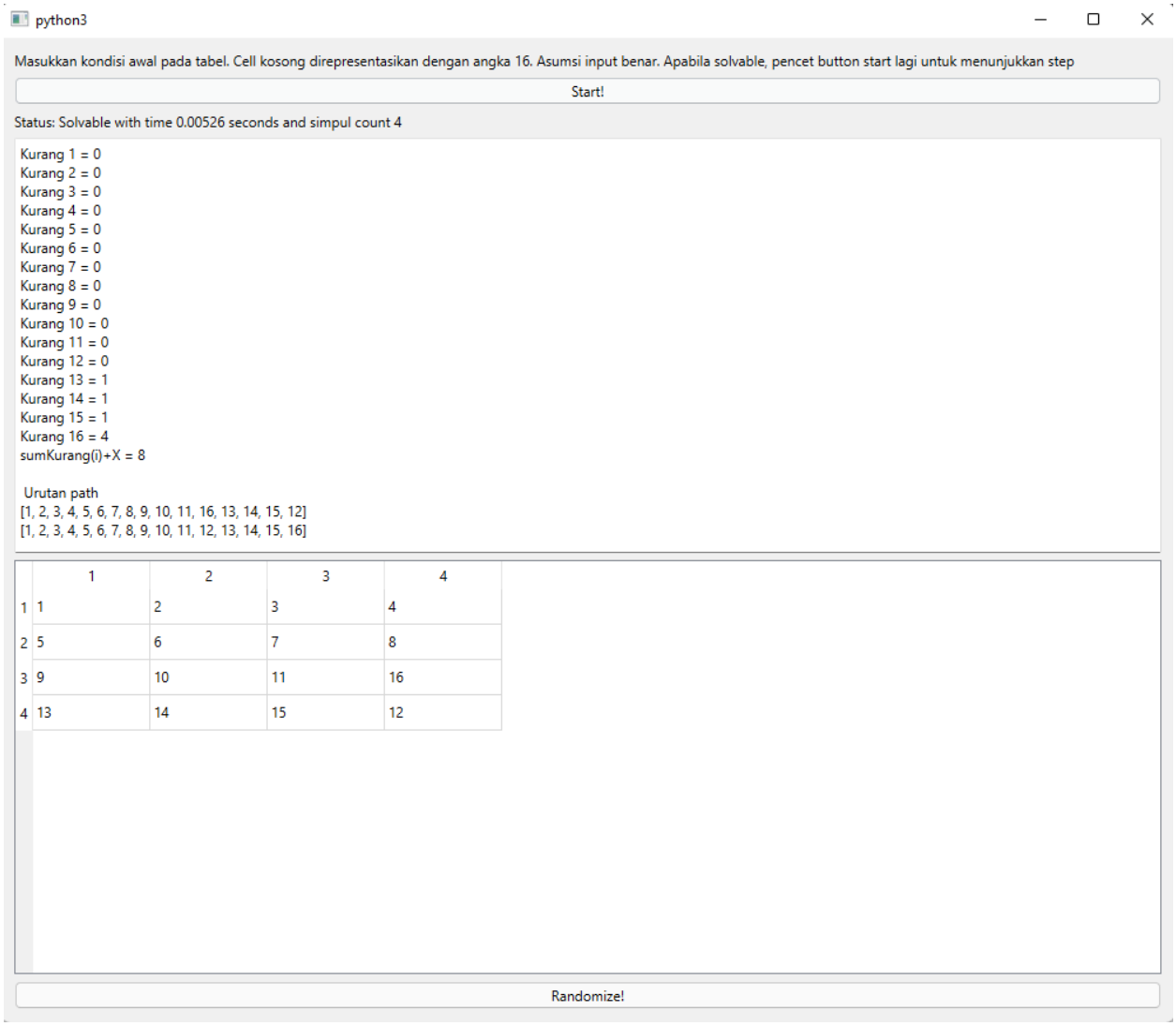
Perhatian: Dalam melakukan input, slot kosong direpresentasikan dengan angka 16

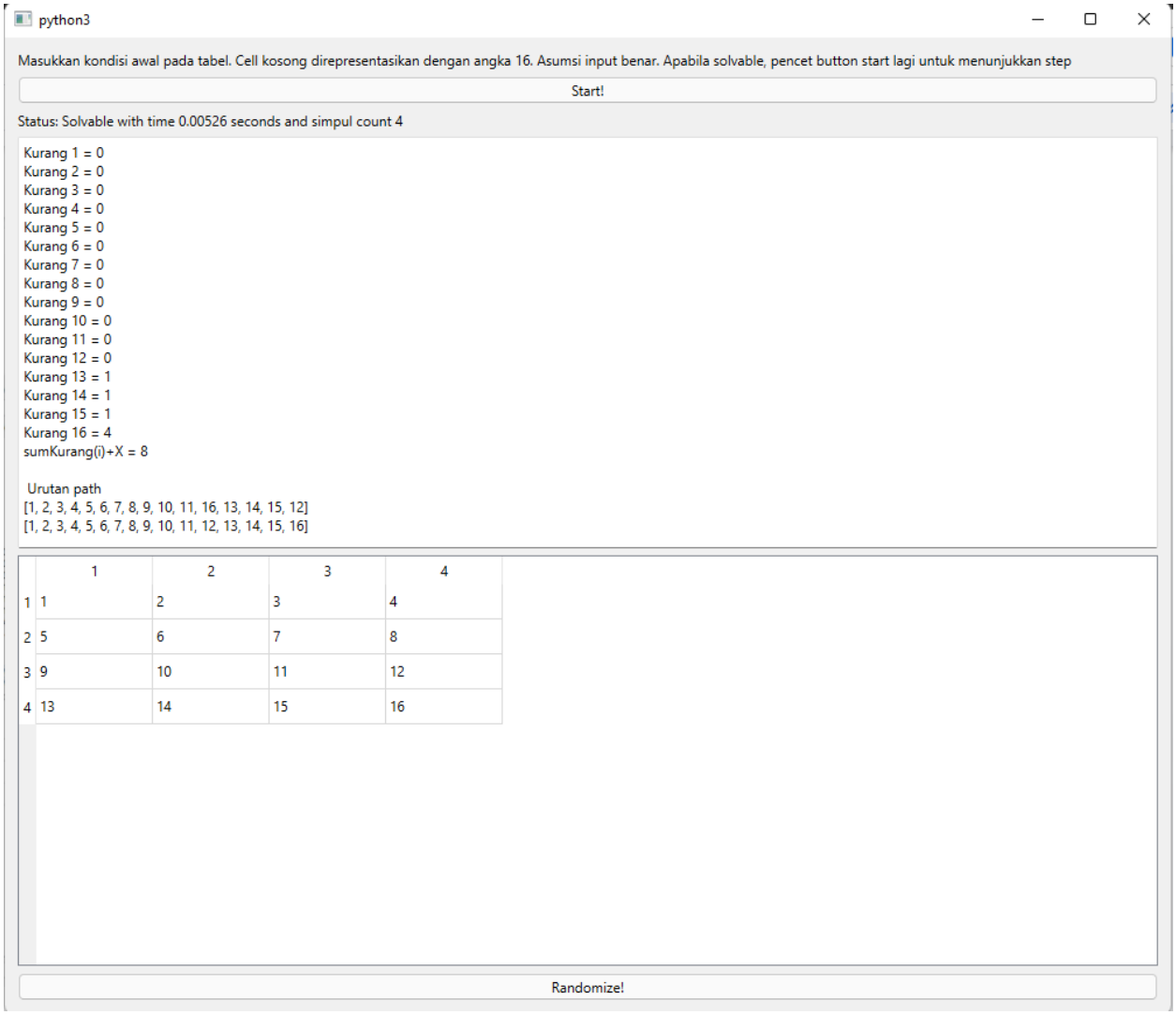
1. **TC 1**

   Input



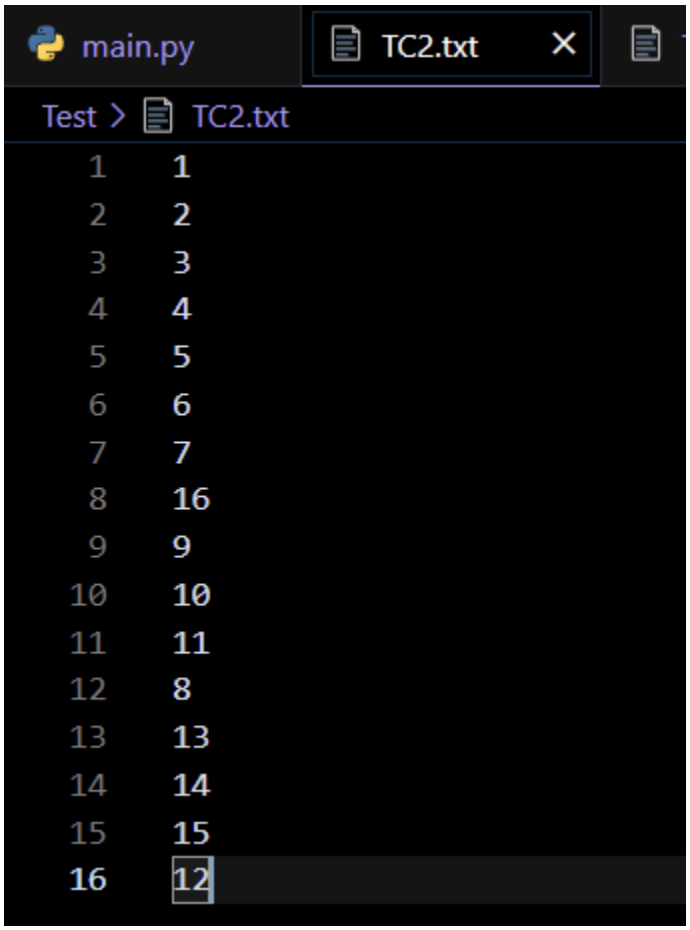
Output

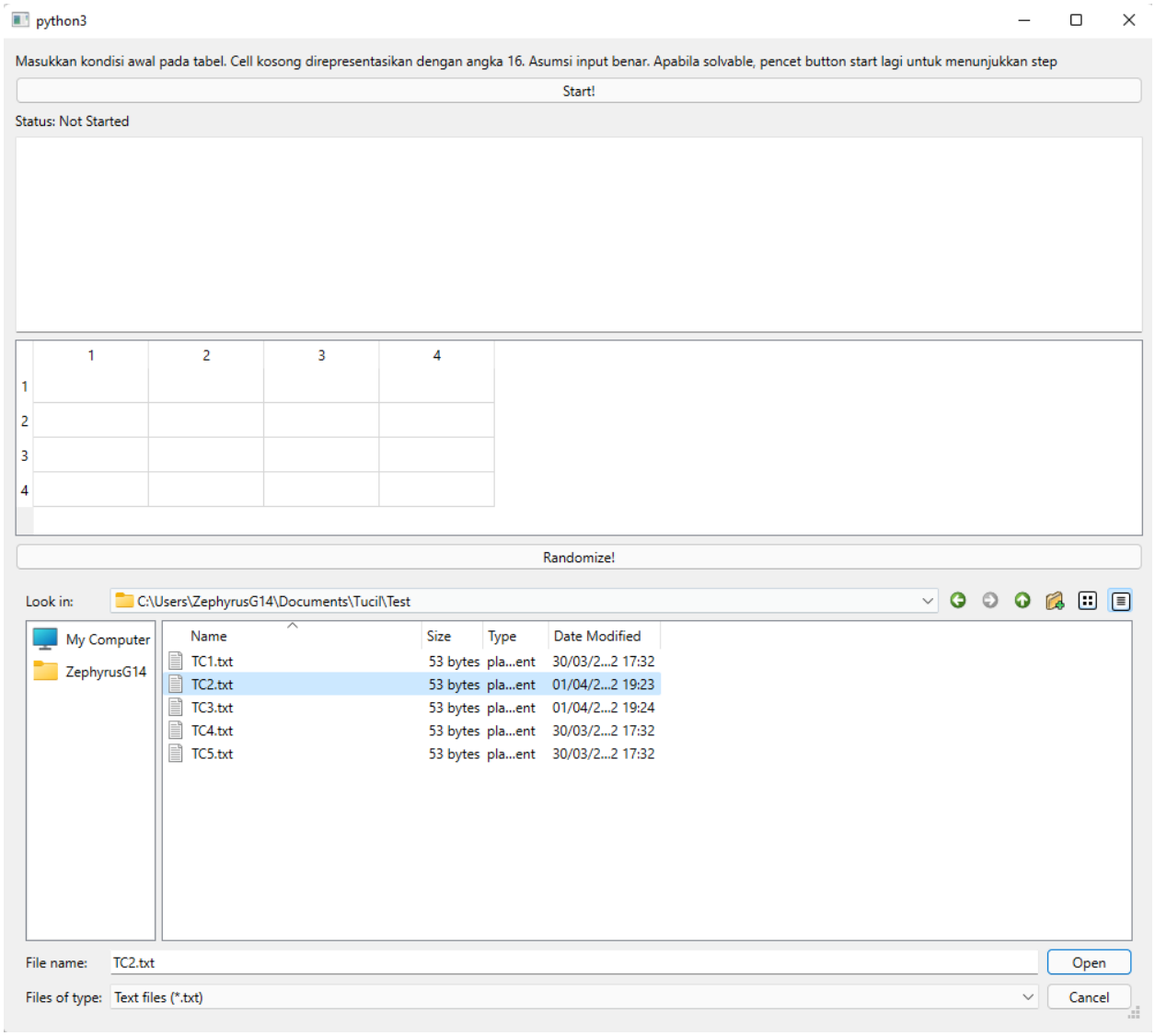

python3
Masukkan kondisi awal pada tabel. Cell kosong direpresentasikan dengan angka 16. Asumsi input benar. Apabila solvable, pencet button start lagi untuk menunjukkan step
Start!
Status: Solvable with time 0.00526 seconds and simpul count 4
Kurang 1 = 0
Kurang 2 = 0
Kurang 3 = 0
Kurang 4 = 0
Kurang 5 = 0
Kurang 6 = 0
Kurang 7 = 0
Kurang 8 = 0
Kurang 9 = 0
Kurang 10 = 0
Kurang 11 = 0
Kurang 12 = 0
Kurang 13 = 1
Kurang 14 = 1
Kurang 15 = 1
Kurang 16 = 4
sumKurang(i)+X = 8
Urutan path
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 16, 13, 14, 15, 12]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]
1 2 3 4
1 1 2 3 4
2 5 6 7 8
3 9 10 11 16
4 13 14 15 12
Randomize!


python3
Masukkan kondisi awal pada tabel. Cell kosong direpresentasikan dengan angka 16. Asumsi input benar. Apabila solvable, pencet button start lagi untuk menunjukkan step
Start!
Status: Solvable with time 0.00526 seconds and simpul count 4
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 9 = 0
Kurang 10 = 0
Kurang 11 = 0
Kurang 12 = 0
Kurang 13 = 1
Kurang 14 = 1
Kurang 15 = 1
Kurang 16 = 4
sumKurang(i)+X = 8
Urutan path
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 16, 13, 14, 15, 12]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]
1 2 3 4
1 1 2 3 4
2 5 6 7 8
3 9 10 11 12
4 13 14 15 16
Randomize!

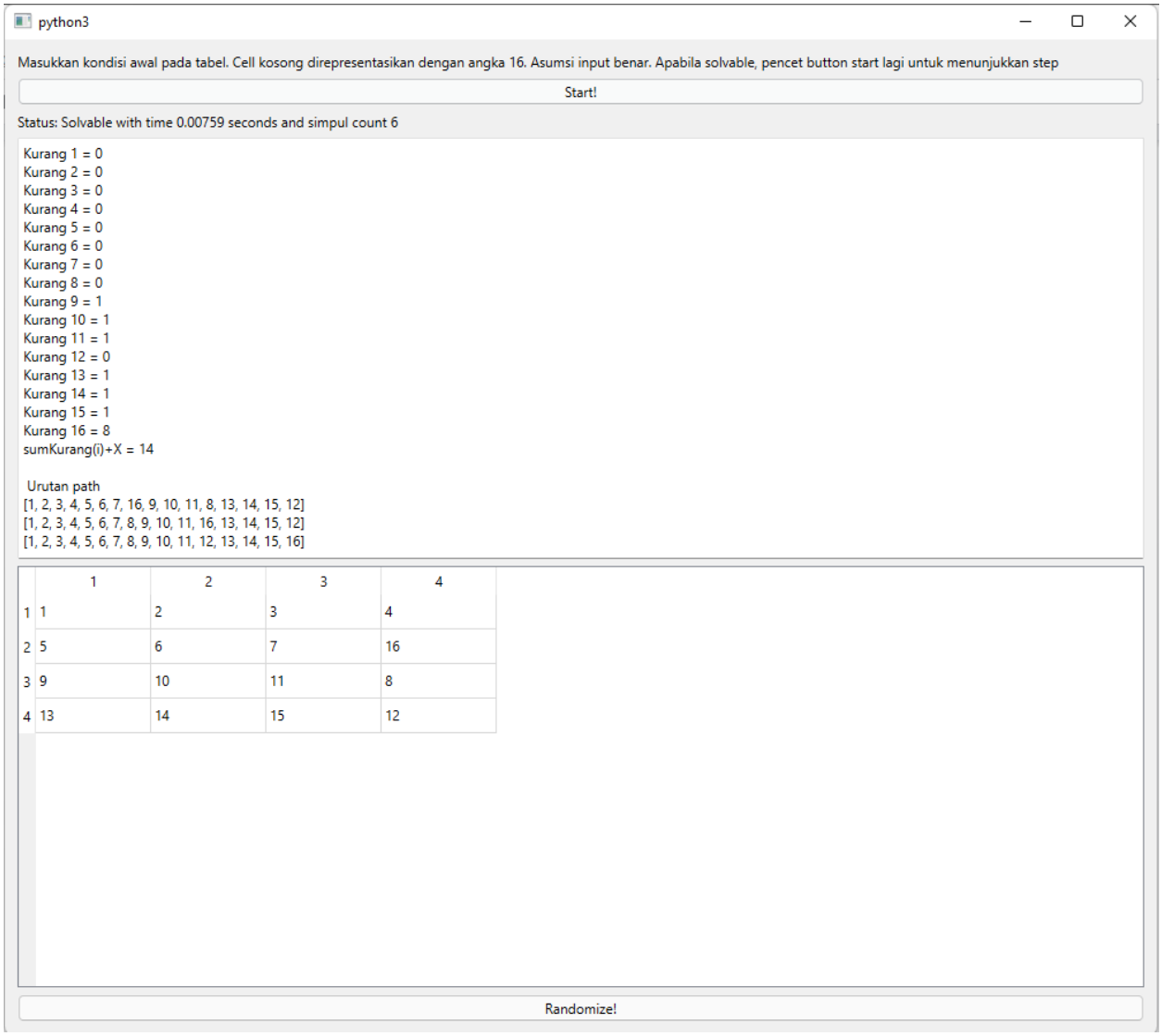
2. TC 2

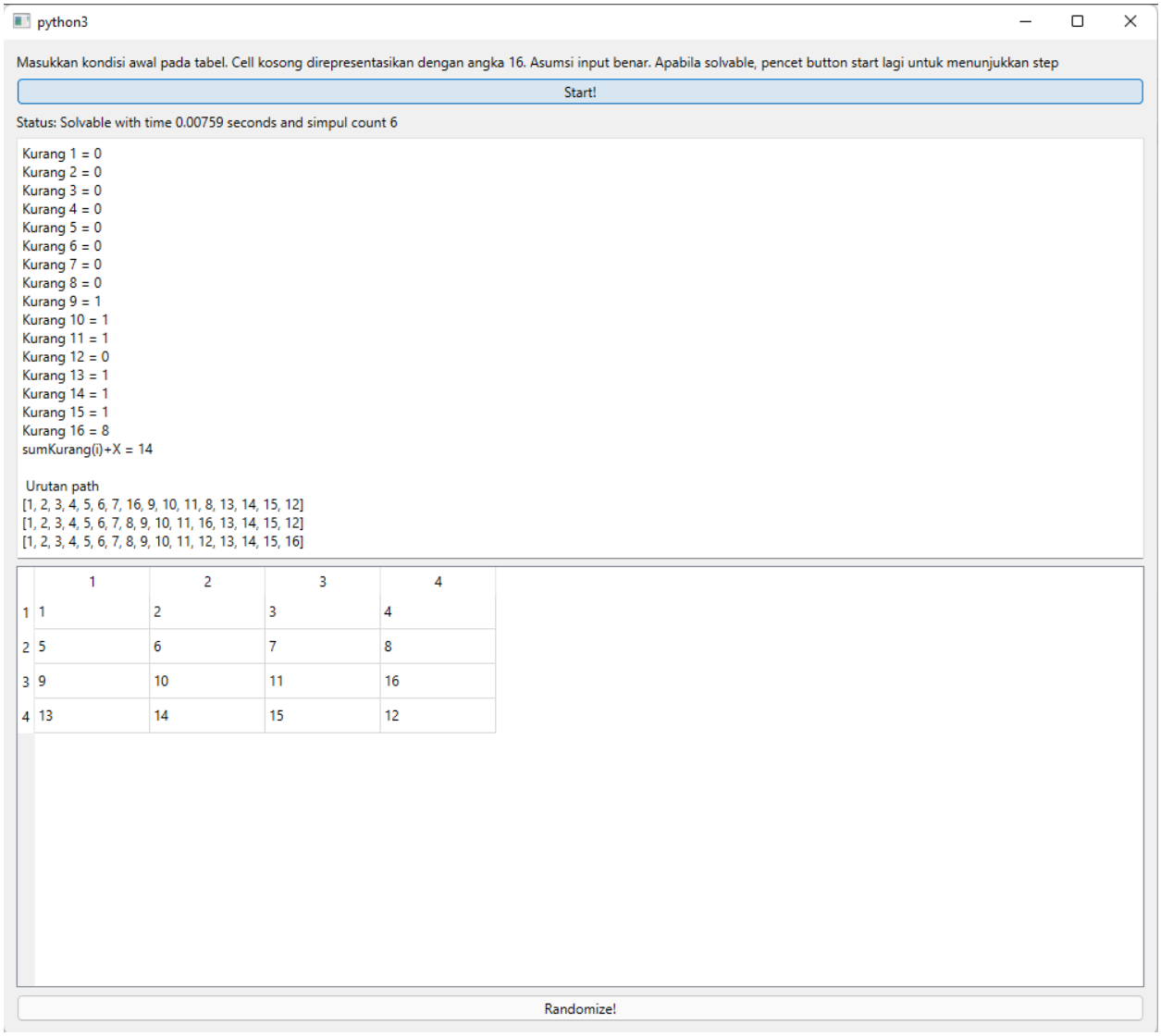
Input

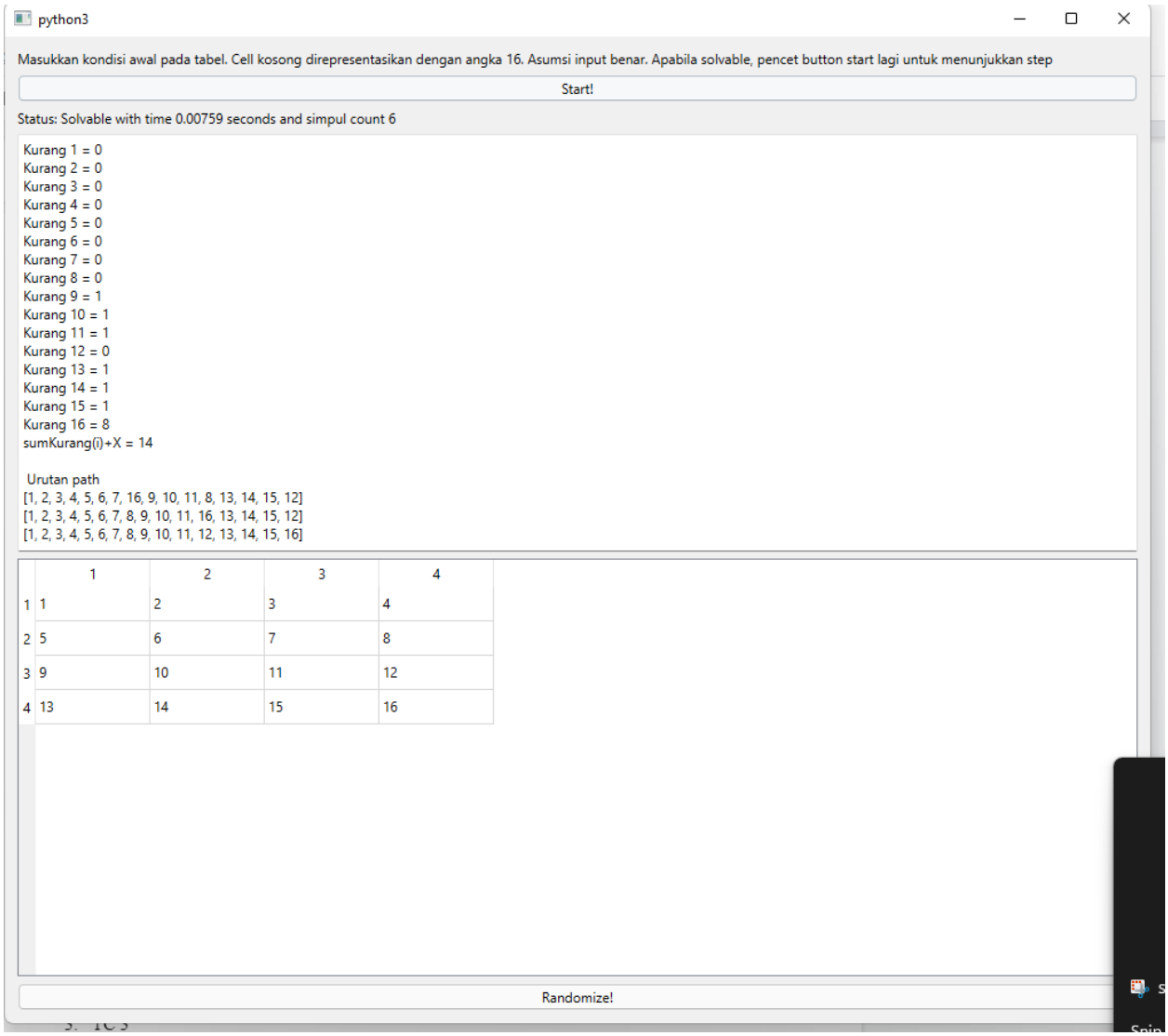

main.py
TC2.txt
Test > TC2.txt
1 1
2 2
3 3
4 4
5 5
6 6
7 7
8 16
9 9
10 10
11 11
12 8
13 13
14 14
15 15
16 12


python3
Masukkan kondisi awal pada tabel. Cell kosong direpresentasikan dengan angka 16. Asumsi input benar. Apabila solvable, pencet button start lagi untuk menunjukkan step
Start!
Status: Not Started
1 2 3 4
1
2
3
4
Randomize!
Look in:
C:\Users\ZephyrusG14\Documents\Tucil\Test
My Computer
ZephyrusG14
Name Size Type Date Modified
TC1.txt 53 bytes pla...ent 30/03/2...2 17:32
TC2.txt 53 bytes pla...ent 01/04/2...2 19:23
TC3.txt 53 bytes pla...ent 01/04/2...2 19:24
TC4.txt 53 bytes pla...ent 30/03/2...2 17:32
TC5.txt 53 bytes pla...ent 30/03/2...2 17:32
File name:
TC2.txt
Open
Files of type:
Text files (*.txt)
Cancel

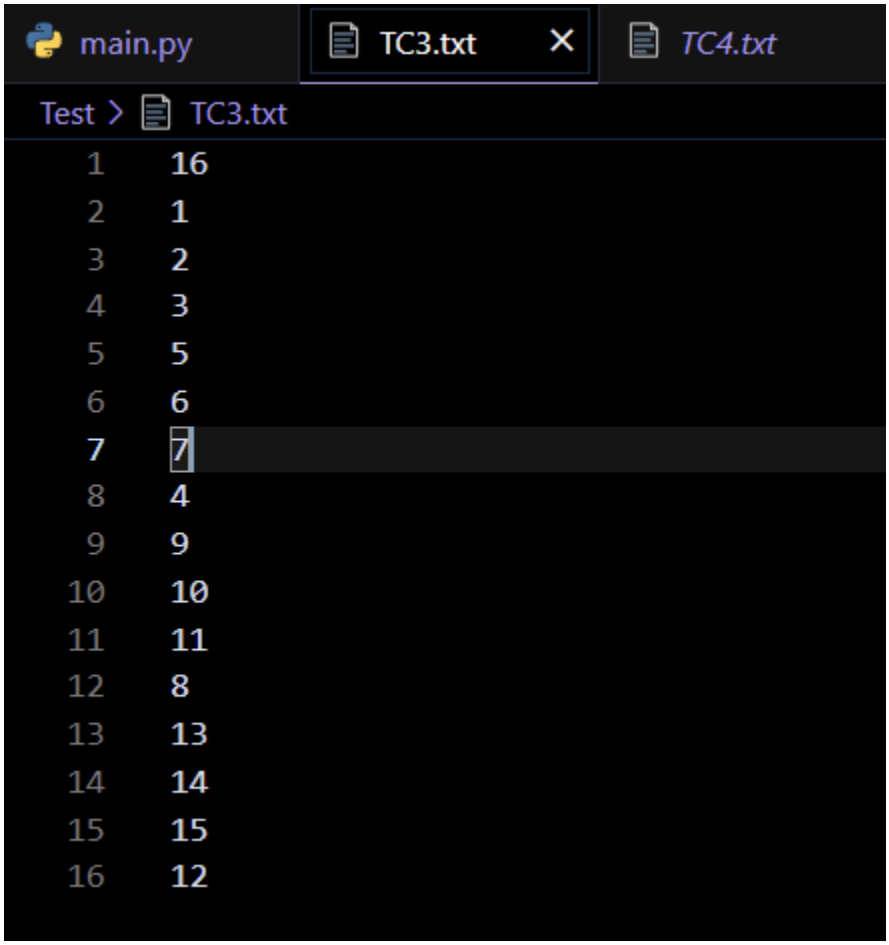
Output


python3
Masukkan kondisi awal pada tabel. Cell kosong direpresentasikan dengan angka 16. Asumsi input benar. Apabila solvable, pencet button start lagi untuk menunjukkan step
Start!
Status: Solvable with time 0.00759 seconds and simpul count 6
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 9 = 1
Kurang 10 = 1
Kurang 11 = 1
Kurang 12 = 0
Kurang 13 = 1
Kurang 14 = 1
Kurang 15 = 1
Kurang 16 = 8
sumKurang(i)+X = 14
Urutan path
[1, 2, 3, 4, 5, 6, 7, 16, 9, 10, 11, 8, 13, 14, 15, 12]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 16, 13, 14, 15, 12]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]
1 2 3 4
1 1 2 3 4
2 5 6 7 16
3 9 10 11 8
4 13 14 15 12
Randomize!


python3
Masukkan kondisi awal pada tabel. Cell kosong direpresentasikan dengan angka 16. Asumsi input benar. Apabila solvable, pencet button start lagi untuk menunjukkan step
Start!
Status: Solvable with time 0.00759 seconds and simpul count 6
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 9 = 1
Kurang 10 = 1
Kurang 11 = 1
Kurang 12 = 0
Kurang 13 = 1
Kurang 14 = 1
Kurang 15 = 1
Kurang 16 = 8
sumKurang(i)+X = 14
Urutan path
[1, 2, 3, 4, 5, 6, 7, 16, 9, 10, 11, 8, 13, 14, 15, 12]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 16, 13, 14, 15, 12]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]
1 2 3 4
1 1 2 3 4
2 5 6 7 8
3 9 10 11 16
4 13 14 15 12
Randomize!

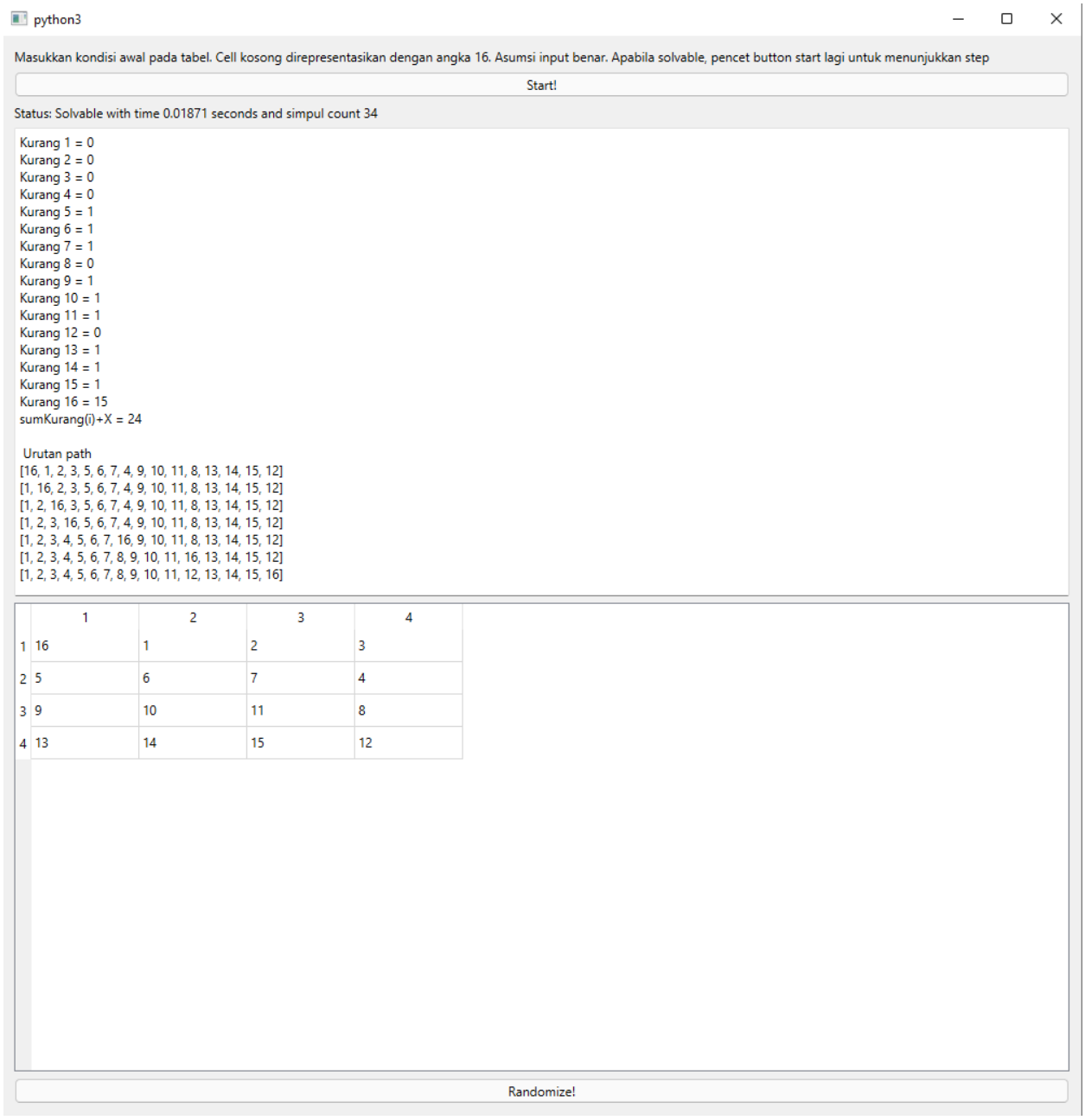

python3
Masukkan kondisi awal pada tabel. Cell kosong direpresentasikan dengan angka 16. Asumsi input benar. Apabila solvable, pencet button start lagi untuk menunjukkan step
Start!
Status: Solvable with time 0.00759 seconds and simpul count 6
Kurang 1 = 0
Kurang 2 = 0
Kurang 3 = 0
Kurang 4 = 0
Kurang 5 = 0
Kurang 6 = 0
Kurang 7 = 0
Kurang 8 = 0
Kurang 9 = 1
Kurang 10 = 1
Kurang 11 = 1
Kurang 12 = 0
Kurang 13 = 1
Kurang 14 = 1
Kurang 15 = 1
Kurang 16 = 8
sumKurang(i)+X = 14
Urutan path
[1, 2, 3, 4, 5, 6, 7, 16, 9, 10, 11, 8, 13, 14, 15, 12]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 16, 13, 14, 15, 12]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]
1 2 3 4
1 1 2 3 4
2 5 6 7 8
3 9 10 11 12
4 13 14 15 16
Randomize!

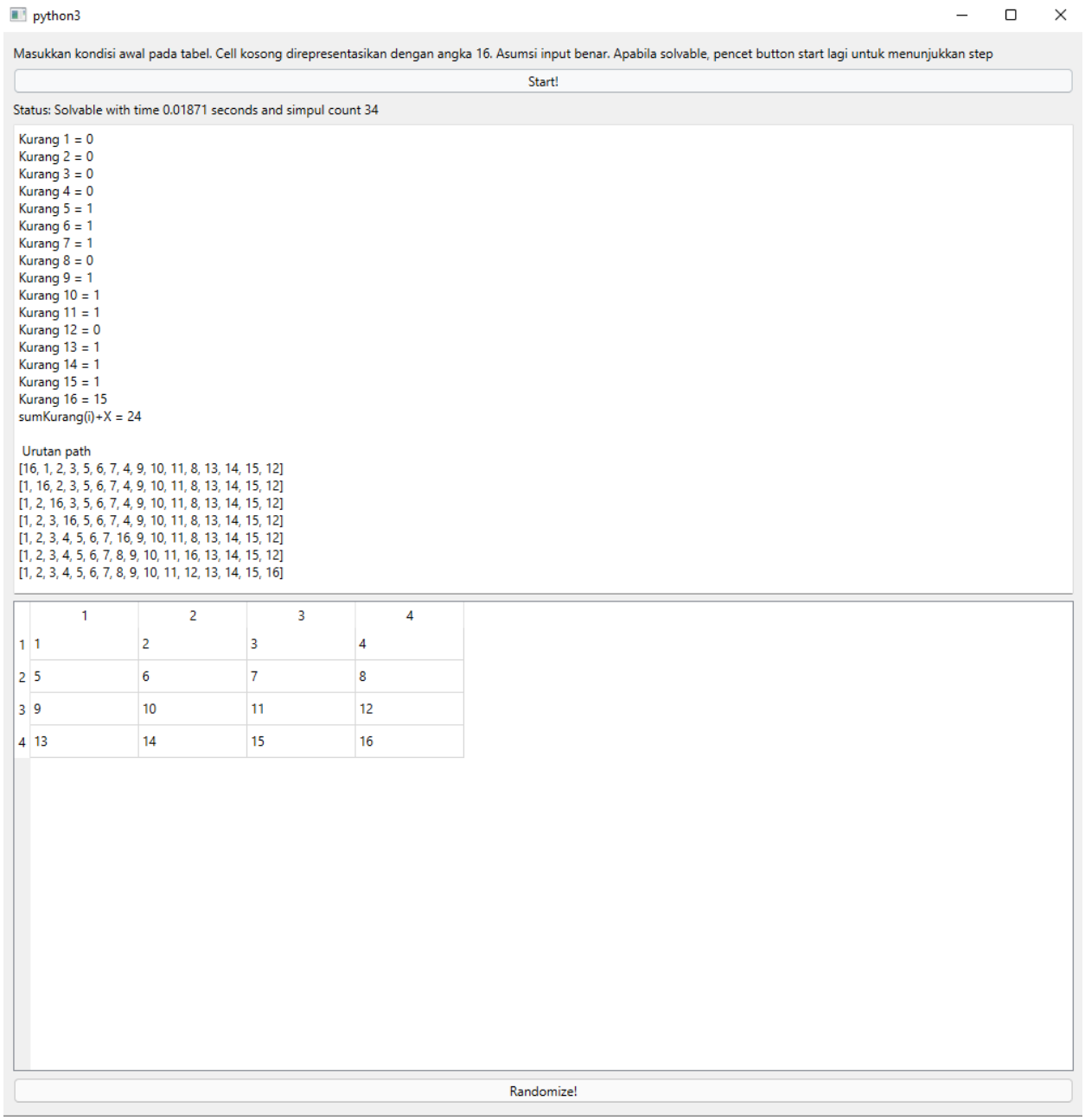
3. **TC 3**



Output

State awal

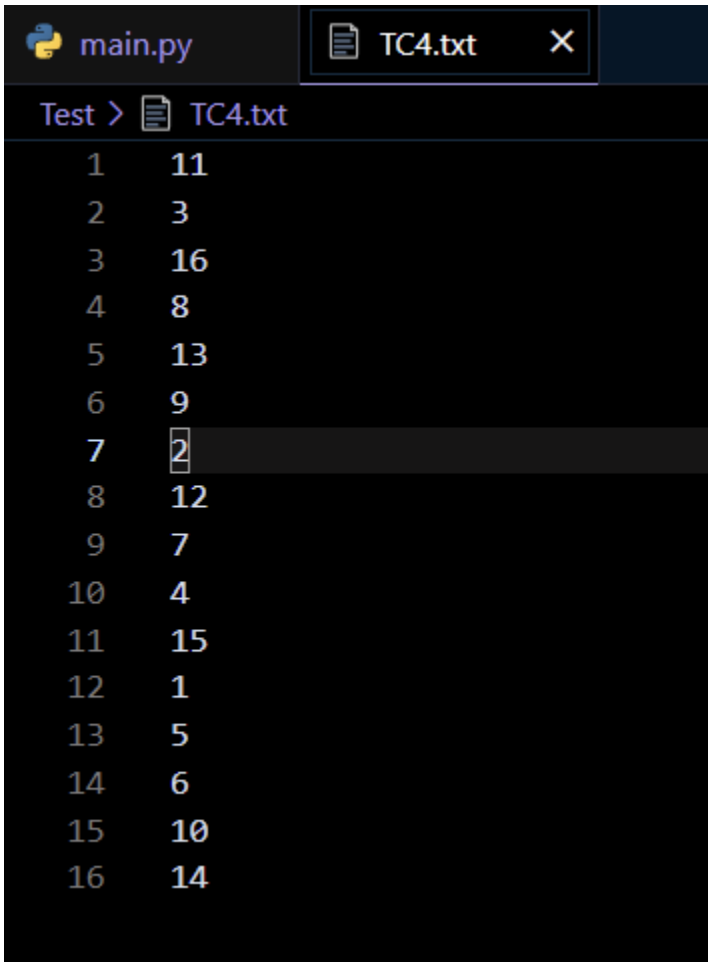

python3
Masukkan kondisi awal pada tabel. Cell kosong direpresentasikan dengan angka 16. Asumsi input benar. Apabila solvable, pencet button start lagi untuk menunjukkan step
Start!
Status: Solvable with time 0.01871 seconds and simpul count 34
Kurang 1 = 0
Kurang 2 = 0
Kurang 3 = 0
Kurang 4 = 0
Kurang 5 = 1
Kurang 6 = 1
Kurang 7 = 1
Kurang 8 = 0
Kurang 9 = 1
Kurang 10 = 1
Kurang 11 = 1
Kurang 12 = 0
Kurang 13 = 1
Kurang 14 = 1
Kurang 15 = 1
Kurang 16 = 15
sumKurang(i)+X = 24
Urutan path
[16, 1, 2, 3, 5, 6, 7, 4, 9, 10, 11, 8, 13, 14, 15, 12]
[1, 16, 2, 3, 5, 6, 7, 4, 9, 10, 11, 8, 13, 14, 15, 12]
[1, 2, 16, 3, 5, 6, 7, 4, 9, 10, 11, 8, 13, 14, 15, 12]
[1, 2, 3, 16, 5, 6, 7, 4, 9, 10, 11, 8, 13, 14, 15, 12]
[1, 2, 3, 4, 5, 6, 7, 16, 9, 10, 11, 8, 13, 14, 15, 12]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 16, 13, 14, 15, 12]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]
1 2 3 4
1 16 1 2 3
2 5 6 7 4
3 9 10 11 8
4 13 14 15 12
Randomize!



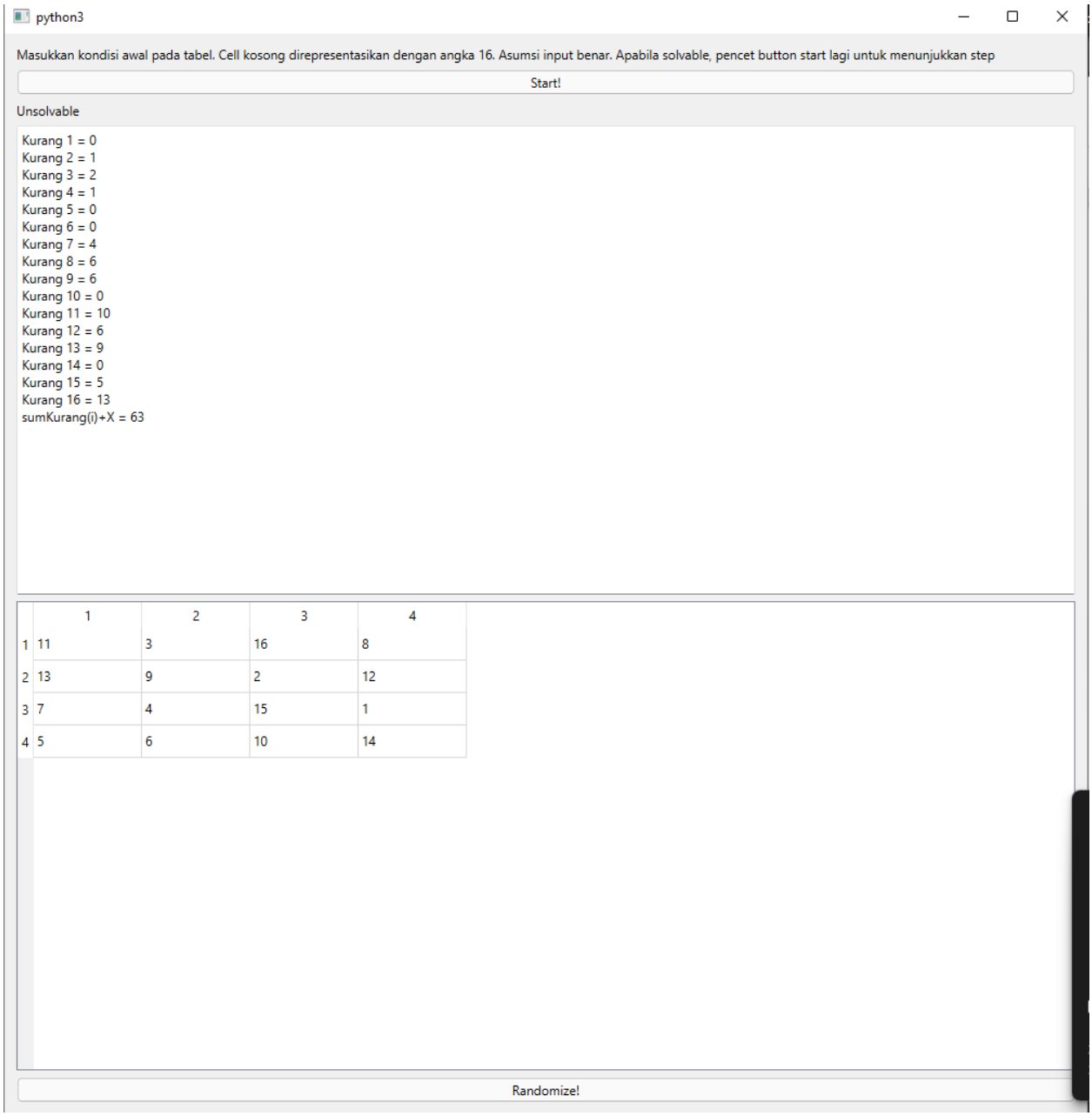
Mohon maaf tidak di screenshot semua karena cukup banyak

4. TC 4

Input

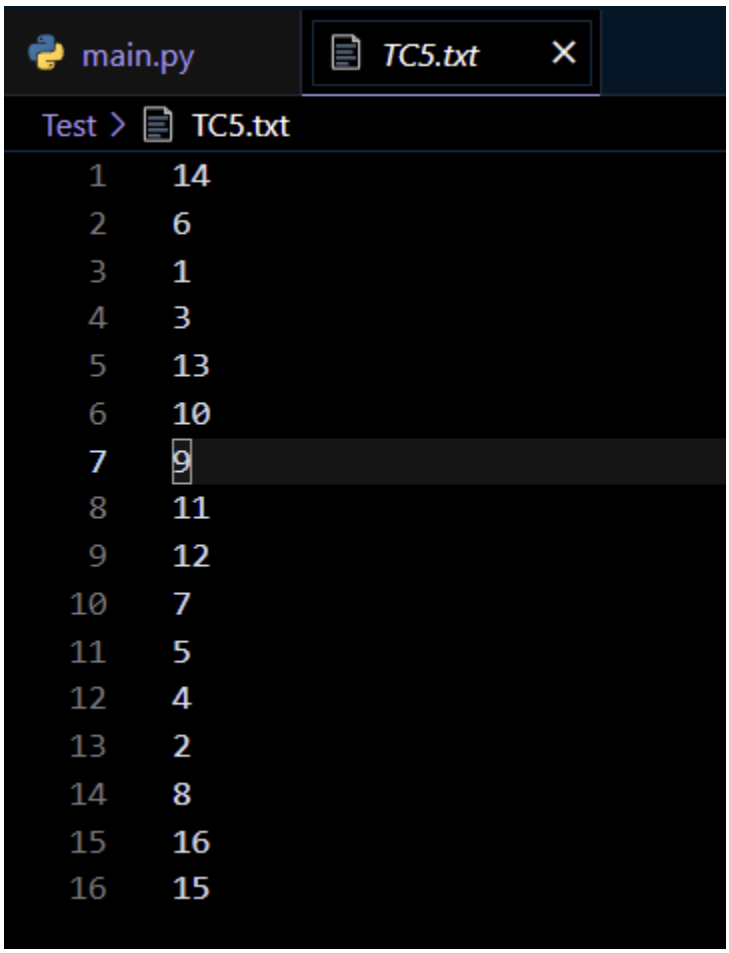

main.py
TC4.txt
Test > TC4.txt
11
3
16
8
13
9
2
12
7
4
15
1
5
6
10
14

Output

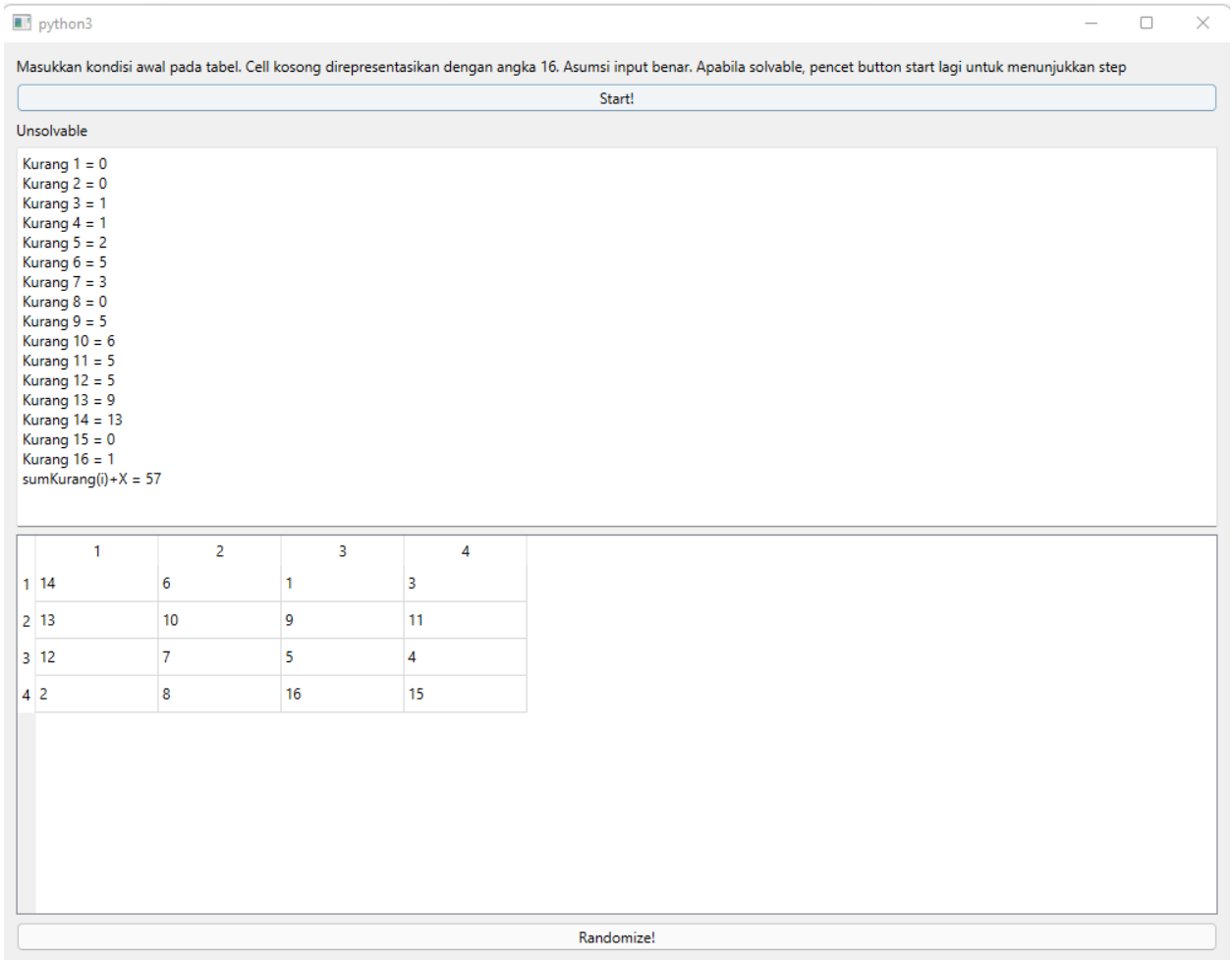

python3
Masukkan kondisi awal pada tabel. Cell kosong direpresentasikan dengan angka 16. Asumsi input benar. Apabila solvable, pencet button start lagi untuk menunjukkan step
Start!
Unsolvable
Kurang 1 = 0
Kurang 2 = 1
Kurang 3 = 2
Kurang 4 = 1
Kurang 5 = 0
Kurang 6 = 0
Kurang 7 = 4
Kurang 8 = 6
Kurang 9 = 6
Kurang 10 = 0
Kurang 11 = 10
Kurang 12 = 6
Kurang 13 = 9
Kurang 14 = 0
Kurang 15 = 5
Kurang 16 = 13
sumKurang(i)+X = 63
1 2 3 4
1 11 3 16 8
2 13 9 2 12
3 7 4 15 1
4 5 6 10 14
Randomize!

5. **TC 5**

Input


main.py
TC5.txt
Test > TC5.txt
1 14
2 6
3 1
4 3
5 13
6 10
7 9
8 11
9 12
10 7
11 5
12 4
13 2
14 8
15 16
16 15

Output


python3
Masukkan kondisi awal pada tabel. Cell kosong direpresentasikan dengan angka 16. Asumsi input benar. Apabila solvable, pencet button start lagi untuk menunjukkan step
Start!
Unsolvable
Kurang 1 = 0
Kurang 2 = 0
Kurang 3 = 1
Kurang 4 = 1
Kurang 5 = 2
Kurang 6 = 5
Kurang 7 = 3
Kurang 8 = 0
Kurang 9 = 5
Kurang 10 = 6
Kurang 11 = 5
Kurang 12 = 5
Kurang 13 = 9
Kurang 14 = 13
Kurang 15 = 0
Kurang 16 = 1
sumKurang(i)+X = 57
1 2 3 4
1 14 6 1 3
2 13 10 9 11
3 12 7 5 4
4 2 8 16 15
Randomize!

Link Kode Program

Repository Github : https://github.com/dParikesit/TucilStima3/

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

1. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Branch-and-Bound-2021-Bagian1.pdf